Two-Dimensional Input-Revolving Automata

S. James Immanuel[1(✉)], D.G. Thomas[1], Henning Fernau[2], Robinson Thamburaj[1], and Atulya K. Nagar[3]

[1] Department of Mathematics, Madras Christian College, Tambaram, Chennai, India
james_imch@yahoo.co.in, dgthomasmcc@yahoo.com, robin.mcc@gmail.com

[2] Fachbereich 4 – Abteilung Informatik, Universität Trier, 54286 Trier, Germany
Fernau@uni-trier.de

[3] Department of Mathematics and Computer Science, Liverpool Hope University, Liverpool, UK
nagara@hope.ac.uk

Abstract. A new type of two-dimensional automaton for accepting two-dimensional languages, called as two-dimensional input-revolving automaton is introduced in this paper. It is an extension of input-revolving automaton for string languages. We bring out all the variants of this automaton which are based on the various types of column-revolving operations considered here. We compare the families of array languages accepted by the variants of these automata along with the well known families of Siromoney matrix languages. We discuss some of the closure properties of the new families of array languages and give an application in steganography.

Keywords: Picture languages · Extended finite automaton · Rectangular arrays · Input-revolving

1 Introduction

Two-dimensional languages, also called picture languages, are sets of two-dimensional arrays of symbols chosen from a finite alphabet. Their application and importance can be seen in image processing, pattern recognition, character recognition and also in studies concerning cellular automaton and other models of computing. Due to the structure handling ability of the syntactic models, syntactic techniques of generation of digital picture arrays have become one of the major areas of theoretical studies in picture analysis and pattern recognition. Various grammars and automata have already been proposed for the generation and recognition of rectangular picture languages [4,8–11,13,14,16].

Automata play a vital role in the theory of picture recognizability. The generalization of finite state automata to two-dimensional languages can be attributed to Blum and Hewitt [2] who introduced the notion of 4-way automata. Array automata acting on scenes (two-dimensional tapes) are defined in [5]. Motivated by certain floor designs called "Kolam" patterns, Siromoney et al. [12] introduced Siromoney Matrix Grammars (SMG), a simple and elegant grammar model for generating rectangular arrays. Several variations of Siromoney matrix grammars have already been added to the literature, for instance [15,17].

In this paper, we extend the concept of input-revolving automata systematically investigated in [1], with several shift operations, namely left-revolving, right-revolving and circular-interchanging operations to two dimensional languages. Input-revolving automata work similar to ordinary finite automata except that they can make three special shift operations as mentioned. Two-dimensional input-revolving automata work on an input array row by row and every row computation is same as that of input-revolving automata for string languages. Here we introduce column shift transitions, namely left column-revolving, right-column revolving and circular column-interchanging transitions which shifts an entire column. We study all the variants of these automata, both deterministic and non-deterministic, based on the various types of column-revolving operations considered. We compare various classes of languages accepted by these automata with the families of Siromoney matrix languages [12].

2 Preliminaries

In this section we recall some notions related to formal language theory and array grammars (see [7,12]). Let Σ be a finite alphabet, Σ^* is the set of words over Σ including the empty word λ. $\Sigma^+ = \Sigma^* - \{\lambda\}$. For $w \in \Sigma^*$ and $a \in \Sigma$, $|w|_a$ denotes the number of occurrences of a in w. An array consists of finitely many symbols from Σ that are arranged as rows and columns in some particular order and is written in the form, $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$ or in short $A = [a_{ij}]_{m \times n}$, for all $a_{ij} \in \Sigma$, $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. The set of all arrays over Σ is denoted by Σ^{**} which also includes the empty array Λ (zero rows and zero columns). $\Sigma^{++} = \Sigma^{**} - \{\Lambda\}$. For $a \in \Sigma$, $|A|_a$ denotes the number of occurrences of a in A. The column concatenation of $A = \begin{bmatrix} a_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} \end{bmatrix}$, and $B = \begin{bmatrix} b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nq} \end{bmatrix}$, defined only when $m = n$, is given by $A \oplus B = \begin{bmatrix} a_{11} & \cdots & a_{1p} & b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} & b_{n1} & \cdots & b_{nq} \end{bmatrix}$. As $1 \times n$-dimensional arrays can be easily interpreted as words of length n (and vice versa), we will then write their column catenation by juxtaposition (as usual). Similarly, the row concatenation, defined only when $p = q$, is given by $A \ominus B = \begin{bmatrix} a_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} \\ b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nq} \end{bmatrix}$. The empty array acts as the identity for column and row catenation of arrays of arbitrary dimensions. If $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$, then the transpose of A is $A^T = \begin{bmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{bmatrix}$.

Given a picture $A \in \Sigma^{**}$ of size (m, n), we define $\hat{A}$ as the picture of size $(m, n+1)$ obtained by adjoining a special symbol $\# \notin \Sigma$ to the right end of the input array. $|A|_c$ is the number of columns in picture A and $|A|_r$ is the number of rows in picture A. If every element in an $m \times n$ array is a, then we write this array as a_m^n.

Definition 1. *A* Context-sensitive matrix grammar (CSMG) (Context-free matrix grammar (CFMG), Right-linear matrix grammar (RLMG)) *is defined by a 7-tuple $G = (V_h, V_v, \Sigma_I, \Sigma, S, R_h, R_v)$, where: V_h is a finite set of* horizontal nonterminals*; V_v is a finite set of* vertical nonterminals*; $\Sigma_I \subseteq V_v$ is a finite set of* intermediates*; Σ is a finite set of* terminals*; $S \in V_h$ is a* starting symbol*; R_h is a finite set of* horizontal context-sensitive (context-free, right-linear) rules*; R_v is a finite set of* vertical right-linear rules.

There are two phases of derivation of the Siromoney Matrix Grammars. In the first phase, a horizontal string of intermediate symbols is generated by means of any type of Chomsky grammar rules in R_h. During the second phase treating each intermediate as a start symbol, vertical generation of the actual picture is done parallely, by applying R_v. Parallel application ensures that the terminating rules are all applied simultaneously in every column and that the column grows only in downward direction. The language generated by a CSMG (CFMG, RLG) is called a CSML (CFML, RML). For more information, we can refer to [12]. We denote the family of Siromoney matrix languages by $\mathfrak{L}(X)$, where $X \in \{CSML, CFML, RML\}$.

3 Two-Dimensional Input-Revolving Finite Automata

In this section we introduce the two-dimensional input-revolving finite automaton and explain its working with an example.

Definition 2. *A two-dimensional non-deterministic column revolving finite automaton, a 2-NCRFA for short, is a 6-tuple $M = (Q, \Sigma, \delta, \Delta, q_0, F)$, where Q is a finite set of states, Σ is an input alphabet, $Q \cap \Sigma = \emptyset$, $\delta \subseteq Q \times (\Sigma \cup \{\lambda, \#\}) \times 2^Q$ is the finite set of transition rules, $\#$ is a special symbol not in $Q \cup \Sigma$, $\Delta \subseteq Q \times (\Sigma \cup \{\lambda\}) \times 2^Q$ is the finite set of column revolving rules, $q_0 \in Q$ is the initial state, and F is the set of final states.*

M is said to be λ-free if δ is a mapping from $Q \times (\Sigma \cup \{\#\})$ to 2^Q and Δ is a mapping from $Q \times \Sigma$ to 2^Q.

Members of δ, Δ are referred to as the rules of M and instead of $(p, y, q) \in \delta$ (or Δ), we write $py \to q \in \delta$ (or Δ). Based on the different interpretations of the mapping Δ, we formally distinguish the different operations on the input array and hence categorize various types of 2-NCRFA. For this, we consider configurations of this automaton for some input array $A = \begin{array}{c} X \\ uv \\ Z \end{array}$, of size $m \times n$ with $X, Z, u, v \in \Sigma^{**}$, uv is $1 \times n$ array, to be of the form $\begin{array}{c} X \\ uqv \\ Z \end{array}$ where $q \in Q$. This *means that q is the current state of the finite control with tape head reading the first element in v. (X, u) and (v, Z) are therefore the read and unread part of the input array, respectively. The transition of a configuration to next configuration can be induced by either δ or Δ.*

1. *Let $a \in \Sigma \cup \{\lambda\}$ and $A = \begin{array}{c} X \\ uav \\ Z \end{array}$ be an $m \times n$ array in Σ^{**} with $|X|_c = |uav|_c = |Z|_c$. If $qa \to p$ is in δ then, $\begin{array}{c} \hat{X} \\ uqav\# \\ \hat{Z} \end{array} \vdash_M \begin{array}{c} \hat{X} \\ uapv\# \\ \hat{Z} \end{array}$. These transitions are called as the* normal transitions. *Let $A = \begin{array}{c} X \\ u \\ Z \end{array}$ be a $m \times n$ array in Σ^{**} with $|X|_c = |u|_c = |Z|_c, u$ is a $1 \times n$ array. If $q\# \to p$ is in δ then, $\begin{array}{c} \hat{X} \\ u\ q\# \\ \hat{Z} \end{array} \vdash_M \begin{array}{c} \hat{X}' \\ pu'\# \\ \hat{Z}' \end{array}$, where, $X' = \begin{array}{c} X \\ u \end{array}, Z = \begin{array}{c} u' \\ Z' \end{array}$ and u' is a $1 \times n$ array.*
2. *Column revolving operations are performed by applying the rules from Δ. For $a \in \Sigma \cup \{\lambda\}, b \in \Sigma$, a $m \times n$ array, $A = \begin{array}{cccc} X_1 & X_2 & X_3 & X_4 \\ u & a & v & b \\ Z_1 & Z_2 & Z_3 & Z_4 \end{array}$ in Σ^{**} with $|X_1|_c = |u|_c = |Z_1|_c, |X_3|_c = |v|_c = |Z_3|_c, |X_2|_c = |Z_2|_c = |X_4|_c = |Z_4|_c = 1$ and a rule $qa \to p$ in Δ,*
   (a) a left column-revolving transition is defined by

$$\boxed{\begin{array}{cccccc} X_1 & & X_2 & X_3 & \mathbf{X_4} & \# \\ u & q & a & v & \mathbf{b} & \# \\ Z_1 & & Z_2 & Z_3 & \mathbf{Z_4} & \# \end{array}} \vdash_M \boxed{\begin{array}{cccccc} X_1 & & \mathbf{X_4} & X_2 & X_3 & \# \\ u & q & \mathbf{b} & a & v & \# \\ Z_1 & & \mathbf{Z_4} & Z_2 & Z_3 & \# \end{array}}$$

   (b) a right column-revolving transition is defined by

$$\boxed{\begin{array}{cccccc} X_1 & & \mathbf{X_2} & X_3 & X_4 & \# \\ u & q & \mathbf{a} & v & b & \# \\ Z_1 & & \mathbf{Z_2} & Z_3 & Z_4 & \# \end{array}} \vdash_M \boxed{\begin{array}{cccccc} X_1 & & X_3 & X_4 & \mathbf{X_2} & \# \\ u & q & v & b & \mathbf{a} & \# \\ Z_1 & & Z_3 & Z_4 & \mathbf{Z_2} & \# \end{array}}$$

   (c) a circular column-interchanging transition is defined by

$$\boxed{\begin{array}{cccccc} X_1 & & \mathbf{X_2} & X_3 & \mathbf{X_4} & \# \\ u & q & \mathbf{a} & v & \mathbf{b} & \# \\ Z_1 & & \mathbf{Z_2} & Z_3 & \mathbf{Z_4} & \# \end{array}} \vdash_M \boxed{\begin{array}{cccccc} X_1 & & \mathbf{X_4} & X_3 & \mathbf{X_2} & \# \\ u & q & \mathbf{b} & v & \mathbf{a} & \# \\ Z_1 & & \mathbf{Z_4} & Z_3 & \mathbf{Z_2} & \# \end{array}}$$

   If $a = \lambda$, then the column-revolving operation is carried out irrespective of what symbol is being read by the finite state control. If 'a' is being read by the state q at the end of some row of the input array then there is no column-revolving involved and simply the state will be changed to p.

Whenever there is a choice between a normal transition or a column-revolving transition, the next move is non-deterministically chosen by the automaton. A deterministic 2-dimensional column revolving automaton, a 2-DCRFA for short is defined in the same way as 2-NCRFA except that $\delta \subseteq Q \times (\Sigma \cup \{\lambda, \#\}) \times Q$, $\Delta \subseteq Q \times (\Sigma \cup \{\lambda\}) \times Q$ and for which there is at most one choice for any possible configuration.

The reflexive transitive closure of $\vdash_M$ is denoted by, $\vdash_M^*$.

Based on the column-revolving operation involved, a 2-NCRFA is referred to as two-dimensional non-deterministic left-column revolving (2-NLCRA), right column-revolving (2-NRCRA) or circular column-interchanging finite automaton (2-NCIRA). The corresponding deterministic variants are referred to as 2-DLCRA, 2-DRCRA and 2-DCIRA. We can also consider bi-column-revolving automaton, where both left column-revolving and right column-revolving transitions are involved. We can formally define this two-dimensional non-deterministic bi-column-revolving automaton or in short 2-NBCRA to be a 7-tuple, $M = (Q, \Sigma, \delta, \Delta_l, \Delta_r, q_0, F)$, where $M = (Q, \Sigma, \delta, \Delta_l, q_0, F)$ is a 2-NLCRA and $M = (Q, \Sigma, \delta, \Delta_r, q_0, F)$ is a 2-NRCRA. Whenever we refer to an automaton as two-dimensional input-revolving finite automaton it is either left column-revolving, right column-revolving, bi-column-revolving or circular column interchanging.

We define the language accepted by a two-dimensional input-revolving finite automaton M to be,

$$L(M) = \left\{ A = \begin{array}{c} A_1 \\ \vdots \\ A_m \end{array} \in \Sigma^{**} \;\middle|\; \begin{array}{c} q_0 A_1 \\ \vdots \\ A_m \end{array} \vdash_M^* \begin{array}{c} A_1 \\ \vdots \\ A_m \\ q\lambda \end{array} \text{ or simply } \begin{array}{c} q_0 A_1 \\ \vdots \\ A_m \end{array} \vdash_M^* q \text{ with } q \in F \right\}.$$

We denote the family of languages accepted by devices of type X by $\mathfrak{L}(X)$.

Example 1. Let, $M = (\{q_0, q_1, q_2\}, \{0, 1, 2\}, \delta, \Delta, q_0, \{q_0\})$ be a 2-DCRFA, where $\delta = \{q_0 0 \to q_1, q_1 1 \to q_2, q_2 2 \to q_0, q_0 \# \to q_0\}$ and $\Delta = \{q_0 1 \to q_0, q_0 2 \to q_0, q_1 0 \to q_1, q_1 2 \to q_1, q_2 0 \to q_2, q_2 1 \to q_2\}$.

The automaton M accepts the following language regarded as either 2-DLCRA or 2-DRCRA,

$$L(M) = \left\{ X = \begin{array}{c} X_1 \\ \vdots \\ X_m \end{array} \in \{0,1,2\}^{**} \;\middle|\; \begin{array}{c} \text{each } X_i\text{'s are of size } 1 \times n \text{ and} \\ |X_i|_0 = |X_i|_1 = |X_i|_2, \forall \quad i = 1, 2, \ldots, m \end{array} \right\}$$

Working:

The automaton works on $\hat{X}$ if X is given as the input array. Starting from state q_0 and the tape head on the first element of the first row in the input X, the automaton M tries to read 0, 1 and 2 sequentially beginning with 0. It uses the δ transitions to store the current missing symbol in its finite control in order to search for it. Being in a search state, all non-matching symbol are column-wise shifted by the transitions from Δ. If the automaton reads #, it means that a complete row is read. Now the automaton uses the rule $q_0\# \rightarrow q_0$ in δ and moves to the first element of the very next row. This computational process is repeated for the subsequent rows until the entire array is read. If after reading the entire row, the automaton is in state q_0, then the input array is accepted or else it is rejected. Thus accepted array has equal number of 0s, 1s and 2s in each of its rows.

Example 2. The language $L = \{(\bullet^n X \bullet^n)_m \mid n, m > 0\}$ is accepted by a 2-DLCRA, $M = (Q, \{\bullet, X\}, \delta, \Delta, q_0, \{q_4\})$, where $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, $\delta = \{q_1\bullet \rightarrow q_2, q_2\bullet \rightarrow q_0, q_0X \rightarrow q_3, q_3\# \rightarrow q_4, q_4\bullet \rightarrow q_5, q_5\bullet \rightarrow q_4, q_4X \rightarrow q_3\}$ and $\Delta = \{q_0\bullet \rightarrow q_1\}$. The working of this automaton for an input array $\begin{array}{c} \bullet\bullet X\bullet\bullet \\ \bullet\bullet X\bullet\bullet \\ \bullet\bullet X\bullet\bullet \\ \bullet\bullet X\bullet\bullet \end{array}$ is given as follows (Successive configurations are listed one below the other. The configuration at the end of a column is succeeded by the configuration at the top of the next column. The last configuration at the last column of this page is succeeded by the first configuration on the first column of the next page):

$$\begin{array}{r} q_0\,\bullet\bullet X\bullet\bullet\,\# \\ \bullet\bullet X\bullet\bullet\,\# \\ \bullet\bullet X\bullet\bullet\,\# \\ \bullet\bullet X\bullet\bullet\,\# \end{array} \qquad \begin{array}{r} \bullet\bullet\bullet\, q_2 \bullet X\,\# \\ \bullet\bullet\bullet \quad \bullet X\,\# \\ \bullet\bullet\bullet \quad \bullet X\,\# \\ \bullet\bullet\bullet \quad \bullet X\,\# \end{array} \qquad \begin{array}{r} \bullet\bullet \quad \bullet\bullet X\,\# \\ \bullet\bullet\, q_4 \bullet\bullet X\,\# \\ \bullet\bullet \quad \bullet\bullet X\,\# \\ \bullet\bullet \quad \bullet\bullet X\,\# \end{array} \qquad \begin{array}{r} \bullet \quad \bullet\bullet\bullet X\,\# \\ \bullet \quad \bullet\bullet\bullet X\,\# \\ \bullet\, q_5 \bullet\bullet\bullet X\,\# \\ \bullet \quad \bullet\bullet\bullet X\,\# \end{array}$$

$$\vdash \qquad\qquad \vdash \qquad\qquad \vdash \qquad\qquad \vdash$$

$$\begin{array}{r} q_1\,\bullet\bullet\bullet X\bullet\,\# \\ \bullet\bullet\bullet X\bullet\,\# \\ \bullet\bullet\bullet X\bullet\,\# \\ \bullet\bullet\bullet X\bullet\,\# \end{array} \qquad \begin{array}{r} \bullet\bullet\bullet\bullet\, q_0\, X\,\# \\ \bullet\bullet\bullet\bullet \quad X\,\# \\ \bullet\bullet\bullet\bullet \quad X\,\# \\ \bullet\bullet\bullet\bullet \quad X\,\# \end{array} \qquad \begin{array}{r} \bullet\bullet\bullet \quad \bullet X\,\# \\ \bullet\bullet\bullet\, q_5 \bullet X\,\# \\ \bullet\bullet\bullet \quad \bullet X\,\# \\ \bullet\bullet\bullet \quad \bullet X\,\# \end{array} \qquad \begin{array}{r} \bullet\bullet \quad \bullet\bullet X\,\# \\ \bullet\bullet \quad \bullet\bullet X\,\# \\ \bullet\bullet\, q_4 \bullet\bullet X\,\# \\ \bullet\bullet \quad \bullet\bullet X\,\# \end{array}$$

$$\vdash \qquad\qquad \vdash \qquad\qquad \vdash \qquad\qquad \vdash$$

$$\begin{array}{r} \bullet\, q_2 \bullet\bullet X\bullet\,\# \\ \bullet \quad \bullet\bullet X\bullet\,\# \\ \bullet \quad \bullet\bullet X\bullet\,\# \\ \bullet \quad \bullet\bullet X\bullet\,\# \end{array} \qquad \begin{array}{r} \bullet\bullet\bullet\bullet X\, q_3\,\# \\ \bullet\bullet\bullet\bullet X \quad \# \\ \bullet\bullet\bullet\bullet X \quad \# \\ \bullet\bullet\bullet\bullet X \quad \# \end{array} \qquad \begin{array}{r} \bullet\bullet\bullet\bullet \quad X\,\# \\ \bullet\bullet\bullet\bullet\, q_4\, X\,\# \\ \bullet\bullet\bullet\bullet \quad X\,\# \\ \bullet\bullet\bullet\bullet \quad X\,\# \end{array} \qquad \begin{array}{r} \bullet\bullet\bullet \quad \bullet X\,\# \\ \bullet\bullet\bullet \quad \bullet X\,\# \\ \bullet\bullet\bullet\, q_5 \bullet X\,\# \\ \bullet\bullet\bullet \quad \bullet X\,\# \end{array}$$

$$\vdash \qquad\qquad \vdash \qquad\qquad \vdash \qquad\qquad \vdash$$

$$\begin{array}{r} \bullet\bullet\, q_0 \bullet X\bullet\,\# \\ \bullet\bullet \quad \bullet X\bullet\,\# \\ \bullet\bullet \quad \bullet X\bullet\,\# \\ \bullet\bullet \quad \bullet X\bullet\,\# \end{array} \qquad \begin{array}{r} \bullet\bullet\bullet\bullet X\,\# \\ q_4\,\bullet\bullet\bullet\bullet X\,\# \\ \bullet\bullet\bullet\bullet X\,\# \\ \bullet\bullet\bullet\bullet X\,\# \end{array} \qquad \begin{array}{r} \bullet\bullet\bullet\bullet X \quad \# \\ \bullet\bullet\bullet\bullet X\, q_3\,\# \\ \bullet\bullet\bullet\bullet X \quad \# \\ \bullet\bullet\bullet\bullet X \quad \# \end{array} \qquad \begin{array}{r} \bullet\bullet\bullet\bullet \quad X\,\# \\ \bullet\bullet\bullet\bullet \quad X\,\# \\ \bullet\bullet\bullet\bullet\, q_4\, X\,\# \\ \bullet\bullet\bullet\bullet \quad X\,\# \end{array}$$

$$\vdash \qquad\qquad \vdash \qquad\qquad \vdash \qquad\qquad \vdash$$

$$\begin{array}{r} \bullet\bullet\, q_1 \bullet\bullet X\,\# \\ \bullet\bullet \quad \bullet\bullet X\,\# \\ \bullet\bullet \quad \bullet\bullet X\,\# \\ \bullet\bullet \quad \bullet\bullet X\,\# \end{array} \qquad \begin{array}{r} \bullet \quad \bullet\bullet\bullet X\,\# \\ \bullet\, q_5 \bullet\bullet\bullet X\,\# \\ \bullet \quad \bullet\bullet\bullet X\,\# \\ \bullet \quad \bullet\bullet\bullet X\,\# \end{array} \qquad \begin{array}{r} \bullet\bullet\bullet\bullet X\,\# \\ \bullet\bullet\bullet\bullet X\,\# \\ q_4\,\bullet\bullet\bullet\bullet X\,\# \\ \bullet\bullet\bullet\bullet X\,\# \end{array} \qquad \begin{array}{r} \bullet\bullet\bullet\bullet X \quad \# \\ \bullet\bullet\bullet\bullet X \quad \# \\ \bullet\bullet\bullet\bullet X\, q_3\,\# \\ \bullet\bullet\bullet\bullet X \quad \# \end{array}$$

$$\vdash \qquad\qquad \vdash \qquad\qquad \vdash \qquad\qquad \vdash$$

$$\begin{array}{r} \bullet\bullet\bullet\bullet\, X\ \# \\ \bullet\bullet\bullet\bullet\, X\ \# \\ \bullet\bullet\bullet\bullet\, X\ \# \\ q_4\ \bullet\bullet\bullet\bullet\, X\ \# \end{array} \qquad \begin{array}{l} \bullet\bullet\quad\ \bullet\bullet\, X\ \# \\ \bullet\bullet\quad\ \bullet\bullet\, X\ \# \\ \bullet\bullet\quad\ \bullet\bullet\, X\ \# \\ \bullet\bullet\ q_4\ \bullet\bullet\, X\ \# \end{array} \qquad \begin{array}{l} \bullet\bullet\bullet\bullet\quad\ X\ \# \\ \bullet\bullet\bullet\bullet\quad\ X\ \# \\ \bullet\bullet\bullet\bullet\quad\ X\ \# \\ \bullet\bullet\bullet\bullet\ q_4\ X\ \# \end{array} \qquad\qquad q_4$$

$$\vdash \qquad\qquad\qquad \vdash \qquad\qquad\qquad \vdash$$

$$\begin{array}{l} \bullet\quad\ \bullet\bullet\bullet\, X\ \# \\ \bullet\quad\ \bullet\bullet\bullet\, X\ \# \\ \bullet\quad\ \bullet\bullet\bullet\, X\ \# \\ \bullet\ q_5\ \bullet\bullet\bullet\, X\ \# \end{array} \qquad \begin{array}{l} \bullet\bullet\bullet\quad\ \bullet\, X\ \# \\ \bullet\bullet\bullet\quad\ \bullet\, X\ \# \\ \bullet\bullet\bullet\quad\ \bullet\, X\ \# \\ \bullet\bullet\bullet\ q_5\ \bullet\, X\ \# \end{array} \qquad \begin{array}{l} \bullet\bullet\bullet\bullet\, X\quad\ \# \\ \bullet\bullet\bullet\bullet\, X\quad\ \# \\ \bullet\bullet\bullet\bullet\, X\quad\ \# \\ \bullet\bullet\bullet\bullet\, X\ q_3\ \# \end{array}$$

$$\vdash \qquad\qquad\qquad \vdash \qquad\qquad\qquad \vdash$$

The definition of two-dimensional column revolving finite automaton allows λ-transitions of both δ and Δ. The λ moves do not increase the computational power of any such automaton. (It is so, because, for every non-deterministic column-revolving automaton involving λ transitions in normal rules or column-revolving rules, an equivalent λ-free non-deterministic column-revolving automaton can be easily constructed (same for the deterministic variants)).

4 Comparison of the Variants

In this section we consider two-dimensional deterministic and non-deterministic variants of the column-revolving finite automata in a detailed way. In particular, we give the comparison among the different column-revolving rules and it turns out that the non-deterministic variants are better than the deterministic ones.

Theorem 1. *The language $L = \{(\bullet^n X \bullet^n)_m \mid n, m > 0\}$ is not accepted by any 2-NRCRA.*

Proof. We assume that the language L is accepted by some 2-NRCRA, $M = (Q, \Sigma, \delta, \Delta, q_0, F)$ with $|Q| = t$. Let us consider an array A with $|A|_c > 2t$, which is given as an input to the automaton M. The automaton begins its computation starting from the first element of the first row. During its computation on the first row some state, say p, appears at least twice, because of Pigeon Hole Principle. Let us consider that the first appearance of state p is reached after i ordinary moves and j right column-revolving moves with $0 \leq j < t$. Therefore we have,

$$\begin{array}{l} q_0 \bullet \bullet^{n-1}\ X \bullet^{n}\ \# \\ (\ \ \bullet \bullet^{n-1}\ X \bullet^{n}\ \#)_{m-1} \end{array} \vdash_M^* \begin{array}{l} \bullet^{i}\ p \bullet \bullet^{n-1-i-j}\ X \bullet^{n+j}\ \# \\ (\bullet^{i}\quad \bullet \bullet^{n-1-i-j}\ X \bullet^{n+j}\ \#)_{m-1} \end{array}$$

Let us consider that the second appearance of state p is reached after k ordinary moves and l right column-revolving moves with $1 \leq l < t-j$. Therefore we have,

$$\begin{array}{l} \bullet^{i}\ p \bullet \bullet^{n-1-i-j}\ X \bullet^{n+j}\ \# \\ (\bullet^{i}\quad \bullet \bullet^{n-1-i-j}\ X \bullet^{n+j}\ \#)_{m-1} \end{array} \vdash_M^* \begin{array}{l} \bullet^{i+k}\ p \bullet \bullet^{n-1-i-j-k-l}\ X \bullet^{n+j+l}\ \# \\ (\bullet^{i+k}\quad \bullet \bullet^{n-1-i-j-k-l}\ X \bullet^{n+j+l}\ \#)_{m-1} \end{array}$$

Since we have assumed that the array A is accepted, the computation reaches an accepting state, say q_f, such that,

$$\begin{array}{l} \bullet^{i+k}\ p \bullet \bullet^{n-1-i-j-k-l}\ X\ \bullet^{n+j+l}\ \# \\ (\bullet^{i+k}\quad \bullet \bullet^{n-1-i-j-k-l}\ X\ \bullet^{n+j+l}\ \#)_{m-1} \end{array} \vdash_M^* q_f$$

When we consider the input array as $A' = (\bullet^{n-k-l} X \bullet^{n+l})_m$, the automaton accepts this input too,

$$\begin{array}{l} q_0 \bullet \bullet^{n-1-k-l}\ X\ \bullet^{n+l}\ \# \\ (\ \bullet \bullet^{n-1-k-l}\ X\ \bullet^{n+l}\ \#)_{m-1} \end{array} \vdash_M^* \begin{array}{l} \bullet^{i}\ p \bullet \bullet^{n-1-i-j-k-l}\ X\ \bullet^{n+j+l}\ \# \\ (\bullet^{i}\quad \bullet \bullet^{n-1-i-j-k-l}\ X\ \bullet^{n+j+l}\ \#)_{m-1} \end{array} \vdash_M^* q_f$$

But the array A' is not a member of the language L. This is a contradiction to the initial assumption. Hence the theorem is proved. □

Theorem 2. *Let $L = \{\{0,1\}^{**} | \exists$ at least one row in which number of 0s and 1s are not equal$\}$. This language is not accepted by any 2-DBCRA but is accepted by some 2-NRCRA.*

Proof. We first prove that the language L is not accepted by any 2-DBCRA, by the method of contradiction. Let us assume that the language is accepted by some 2-DBCRA, $M = (Q, \Sigma, \delta, \Delta, q_0, F)$ with $|Q| = t$. Let $A = (0^n 1^{2n-j} 0^n)_m$ with $n, m, j > 2$, $j < n$, $j \neq 0$ and $n > t$ be the input given to the automaton. Based on pigeon hole principle, there exists a state, say p, which appears at least twice during the computation on first row. Let the first occurrence of state p be reached after: i normal transition, l_1 left column-revolving transition and r_1 right column-revolving transition,

$$\begin{array}{l} q_0\ 0\ 0^{n-1}\ 1^{2n-j}\ 0^n\ \# \\ (\ 0\ 0^{n-1}\ 1^{2n-j}\ 0^n\ \#)_{m-1} \end{array} \vdash_M^* \begin{array}{l} 0^i\ p\ 0\ 0^{n-1-i+l_1-r_1}\ 1^{2n-j}\ 0^{n-l_1+r_1}\ \# \\ (0^i\quad 0\ 0^{n-1-i+l_1-r_1}\ 1^{2n-j}\ 0^{n-l_1+r_1}\ \#)_{m-1} \end{array}$$

Let the second occurrence of state p be reached after: j normal transition, l_2 left column-revolving transition and r_2 right column-revolving transition.

$$\begin{array}{l} 0^i\ p\ 0\ 0^{n-1-i+l_1-r_1}\ 1^{2n-j}\ 0^{n-l_1+r_1}\ \# \\ (0^i\quad 0\ 0^{n-1-i+l_1-r_1}\ 1^{2n-j}\ 0^{n-l_1+r_1}\ \#)_{m-1} \end{array} \vdash_M^*$$
$$\begin{array}{l} 0^{i+j}\ p\ 0\ 0^{n-1-i-j+l_1-r_1+l_2-r_2}\ 1^{2n-j}\ 0^{n-l_1+r_1-l_2+r_2}\ \# \\ (0^{i+j}\quad 0\ 0^{n-1-i-j+l_1-r_1+l_2-r_2}\ 1^{2n-j}\ 0^{n-l_1+r_1-l_2+r_2}\ \#)_{m-1} \end{array}$$

Since the considered array A is in the language L, the computation reaches the accepting state, say q_f i.e.,

$$\begin{array}{l} 0^{i+j}\ p\ 0\ 0^{n-1-i-j+l_1-r_1+l_2-r_2}\ 1^{2n-j}\ 0^{n-l_1+r_1-l_2+r_2}\ \# \\ (0^{i+j}\quad 0\ 0^{n-1-i-j+l_1-r_1+l_2-r_2}\ 1^{2n-j}\ 0^{n-l_1+r_1-l_2+r_2}\ \#)_{m-1} \end{array} \vdash_M^* q_f$$

Let us now consider the input array $A' = (0^{n-j+l_2-r_2} 1^{2n-j} 0^{n-l_2+r_2})_m$. Due to the deterministic behavior, the computation on the input array A' is,

$$\begin{array}{l} q_0\ 0\ 0^{n-1-j+l_2-r_2}\ 1^{2n-j}\ 0^{n-l_2+r_2}\ \# \\ (\ 0\ 0^{n-1-j+l_2-r_2}\ 1^{2n-j}\ 0^{n-l_2+r_2}\ \#)_{m-1} \end{array} \vdash_M^*$$
$$\begin{array}{l} 0^i\ p\ 0\ 0^{n-1-i-j+l_1-r_1+l_2-r_2}\ 1^{2n-j}\ 0^{n-l_1+r_1-l_2+r_2}\ \# \\ (0^i\quad 0\ 0^{n-1-i-j+l_1-r_1+l_2-r_2}\ 1^{2n-j}\ 0^{n-l_1+r_1-l_2+r_2}\ \#)_{m-1} \end{array} \vdash_M^* q_f$$

But this is a contradiction because A' is not in L. Now we prove that there is a 2-NRCRA, M', that accepts the language L. M' works on each row by reading a 0 and a 1 alternatively. This part is similar to the working of the automaton considered in Example 1. The input array is rejected if M' finds that the numbers are equal in each and every row. If there is at least one row in which the numbers are not equal then M' accepts the input array. Also, during the computation on each and every row, M' guesses whether there are only 0's or 1's remaining on present row and if that is the case then it simply reads the remaining elements on the row and the guess is verified. Hence, M' can accept the input array for correct guess on at least one row or else reject it. Hence the proof. $\square$

Theorem 3. *There is a language L not accepted by any 2-NLCRA but accepted by some 2-DRCRA.*

Proof. To prove this theorem, we consider the following language $L =$

$$\left\{ Y \oplus X \;\; | \; Y = (0^{n+1}1)_m, X = \begin{array}{c} X_1 \\ \vdots \\ X_m \end{array}, X_i \in \{0,1\}^+, n + |X_i|_0 = 1 + |X_i|_1, \forall\ n, m \geq 0, i = 1, \dots, m \right\}$$

Let us assume that L is accepted by some 2-NLCRA, $M = (Q, \Sigma, \delta, \Delta, q_0, F)$ with $|Q| = t$. Let us consider the array $A = (0^{n+1}1^{2n}0^n)_m$ with $n, m > 0$ and $n > t$ to be the input given to the automaton. Based on assumption, there is an accepting computation such that a state, say p appears at least twice during the computation on first row. Let the first occurrence of state p be reached after i normal transition and j left column-revolving transition and its second appearance be reached after k normal transition and l left column-revolving transition with $0 \leq i, j, k, l \leq n$ and $k + l > 0$. Hence we have,

$$\begin{array}{l} q_0\, 0\, 0^n\ 1^{2n}\ 0^n\ \# \\ (\ \ 0\, 0^n\ 1^{2n}\ 0^n\ \#)_{m-1} \end{array} \vdash_M^* \begin{array}{l} 0^i\ p\ 0\, 0^{n-i+j}\ 1^{2n}\ 0^{n-j}\ \# \\ (0^i\ \ \ 0\, 0^{n-i+j}\ 1^{2n}\ 0^{n-j}\ \#)_{m-1} \end{array} \vdash_M^+$$

$$\begin{array}{l} 0^{i+k}\ p\ 0\, 0^{n-i+j-k+l}\ 1^{2n}\ 0^{n-j-l}\ \# \\ (0^{i+k}\ \ \ 0\, 0^{n-i+j-k+l}\ 1^{2n}\ 0^{n-j-l}\ \#)_{m-1} \end{array} \vdash_M^* q_f$$

Let us consider the computation of M for the input array $A' = (0^{n+1-k+l}1^{2n}0^{n-l})$,

$$\begin{array}{l} q_0\, 0\, 0^{n-k+l}\ 1^{2n}\ 0^{n-l}\ \# \\ (\ 0\, 0^{n-k+l}\ 1^{2n}\ 0^{n-l}\ \#)_{m-1} \end{array} \vdash_M^* \begin{array}{l} 0^i\ p\ 0\, 0^{n-i+j-k+l}\ 1^{2n}\ 0^{n-j-l}\ \# \\ (0^i\ \ \ 0\, 0^{n-i+j-k+l}\ 1^{2n}\ 0^{n-j-l}\ \#)_{m-1} \end{array} \vdash_M^* q_f$$

But this implies that $A' \in L$. Since, $n + \frac{l-k}{2} + n - l = 2n - \frac{l+k}{2} \neq 2n$ implies $A' \notin L$, we arrive at a contradiction.

The 2-DRCRA, $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, \Delta, q_0, \{q_4\})$, where $\delta = \{q_0 0 \rightarrow q_1, q_1 1 \rightarrow q_2, q_2 0 \rightarrow q_3, q_3 1 \rightarrow q_2, q_3\# \rightarrow q_4, q_4 0 \rightarrow q_3\}$ and $\Delta = \{q_1 0 \rightarrow q_1, q_2 1 \rightarrow q_2, q_3 0 \rightarrow q_3\}$ accepts the language L.

Theorem 4. *(i) The family $\mathfrak{L}$(2-DLCRA) is properly included in $\mathfrak{L}$(2-NLCRA)*
(ii) The family $\mathfrak{L}$(2-DRCRA) is properly included in $\mathfrak{L}$(2-NRCRA)

(iii) The family $\mathfrak{L}$(2-DBCRA) is properly included in $\mathfrak{L}$(2-NBCRA)
(iv) The family $\mathfrak{L}$(2-DRCRA) is properly included in $\mathfrak{L}$(2-DBCRA)
(v) The family $\mathfrak{L}$(2-NRCRA) is properly included in $\mathfrak{L}$(2-NBCRA)
(vi) The family $\mathfrak{L}$(2-DLCRA) is properly included in $\mathfrak{L}$(2-DBCRA)
(vii) The family $\mathfrak{L}$(2-NLCRA) is properly included in $\mathfrak{L}$(2-NBCRA)

Proof. All inclusions are trivial. From Theorem 1, $\exists$ a language not accepted by any 2-NRCRA and hence also by any 2-DRCRA. But, as seen in Example 2 it is accepted by some 2-DLCRA and hence by some 2-DBCRA and 2-NBCRA. This proves (iv) and (v). By Theorem 2, $\exists$ a language not accepted by any 2-DBCRA and hence by any 2-DLCRA, 2-DRCRA but is accepted by 2-NRCRA, and hence by some 2-NBCRA. This proves (ii) and (iii). By Theorem 3 $\exists$ a language not accepted by any 2-NLCRA and hence by any 2-DLCRA but is accepted by some 2-DRCRA and hence by 2-DBCRA and 2-NBCRA. This proves (vi) and (vii).

To prove strict inclusion of (i), we consider the language L from Example 2. Let $L' = L \cup (L \oplus L'')$, where $L'' = \{Y_m \oplus \{\bullet, Y\}^{**}\}$. Clearly L' is accepted by some 2-NLCRA. But L' is not accepted by any 2-DLCRA. We prove this by contradiction. Let us assume that there is a 2-DLCRA, $M = (Q, \Sigma, \delta, \Delta, q_0, F)$ with $|Q| = t$. We consider an input array $A = (\bullet^{2n} X \bullet^{2n} Y^n \bullet^{2n})_m \in L'$ with $n > t$. During the computation on the first row some state, say $p \in Q$ appears at least twice due to our choice of n, let the first appearance be after i normal moves and j left column-revolving moves and the second appearance be after k normal moves and l left column-revolving moves, with $0 \le i, j, k, l \le n$, $k + l > 0$. Since M is deterministic we have,

$$\begin{array}{l} q_0 \bullet \bullet^{2n-1} X \bullet^{2n} Y^n \bullet^{2n} \# \\ (\ \bullet \bullet^{2n-1} X \bullet^{2n} Y^n \bullet^{2n} \#)_{m-1} \end{array} \vdash_M^* \begin{array}{l} \bullet^i\, p \bullet \bullet^{2n-1-i+j} X \bullet^{2n} Y^n \bullet^{2n-j} \# \\ (\bullet^i \quad \bullet \bullet^{2n-1-i+j} X \bullet^{2n} Y^n \bullet^{2n-j} \#)_{m-1} \end{array}$$
$$\vdash_M^+ \begin{array}{l} \bullet^{i+k}\, p \bullet \bullet^{2n-1-i+j-k+l} X \bullet^{2n} Y^n \bullet^{2n-j-l} \# \\ (\bullet^{i+k} \quad \bullet \bullet^{2n-1-i+j-k+l} X \bullet^{2n} Y^n \bullet^{2n-j-l} \#)_{m-1} \end{array} \vdash_M^* q_f$$

We find that there exists an accepting configuration for the input array $A' = (\bullet^{2n-k+l} X \bullet^{2n} Y^n \bullet^{2n-l})_m$ such that,

$$\begin{array}{l} q_0 \bullet \bullet^{2n-1-k+l} X \bullet^{2n} Y^n \bullet^{2n-l} \# \\ (\ \bullet \bullet^{2n-1-k+l} X \bullet^{2n} Y^n \bullet^{2n-l} \#)_{m-1} \end{array} \vdash_M^*$$
$$\begin{array}{l} \bullet^i\, p \bullet \bullet^{2n-1-i+j-k+l} X \bullet^{2n} Y^n \bullet^{2n-j-l} \# \\ (\bullet^i \quad \bullet \bullet^{2n-1-i+j-k+l} X \bullet^{2n} Y^n \bullet^{2n-j-l} \#)_{m-1} \end{array} \vdash_M^* q_f$$

But this implies that $k = l$. Since $l + k > 0$, we derive at $l > 0$. Since, M is deterministic, the accepting computation for the input array $A'' = (\bullet^{2n} X \bullet^{2n})_m$ is,

$$\begin{array}{l} q_0 \bullet \bullet^{2n-1} X \bullet^{2n} \# \\ (\ \bullet \bullet^{2n-1} X \bullet^{2n} \#)_{m-1} \end{array} \vdash_M^* \begin{array}{l} \bullet^i\, p \bullet \bullet^{2n-1-i+j} X \bullet^{2n-j} \# \\ (\bullet^i \quad \bullet \bullet^{2n-1-i+j} X \bullet^{2n-j} \#)_{m-1} \end{array} \vdash_M^+$$
$$\begin{array}{l} \bullet^{i+k}\, p \bullet \bullet^{2n-1-i+j-k+l} X \bullet^{2n-j-l} \# \\ (\bullet^{i+k} \quad \bullet \bullet^{2n-1-i+j-k+l} X \bullet^{2n-j-l} \#)_{m-1} \end{array} \vdash_M^* q'_f$$

where $q'_f \in Q$. We can also obtain an accepting configuration for the input array $A''' = (\bullet^{2n-k+l} X \bullet^{2n-l})_m$ as follows,

$$\begin{array}{l} q_0 \bullet \bullet^{2n-1-k+l}\ X\ \bullet^{2n-l}\ \# \\ \ (\bullet \bullet^{2n-1-k+l}\ X\ \bullet^{2n-l}\ \#)_{m-1} \end{array} \vdash^*_M \begin{array}{l} \bullet^i\ p \bullet \bullet^{2n-1-i+j-k+l}\ X\ \bullet^{2n-j-l}\ \# \\ (\bullet^i\ \ \ \bullet \bullet^{2n-1-i+j-k+l}\ X\ \bullet^{2n-j-l}\ \#)_{m-1} \end{array} \vdash^*_M q'_f$$

Since we have $l = k$ and $l > 0$, the array $A''' \notin L'$, which is a contradiction. Hence this proves (i). □

Theorem 5. *(i) The family $\mathfrak{L}$(2-DLCRA) is incomparable with $\mathfrak{L}$(2-DRCRA)*
(ii) The family $\mathfrak{L}$(2-DLCRA) is incomparable with $\mathfrak{L}$(2-NRCRA)
(iii) The family $\mathfrak{L}$(2-NLCRA) is incomparable with $\mathfrak{L}$(2-NRCRA)
(iv) The family $\mathfrak{L}$(2-NLCRA) is incomparable with $\mathfrak{L}$(2-DRCRA)
(v) The family $\mathfrak{L}$(2-NLCRA) is incomparable with $\mathfrak{L}$(2-DBCRA)
(vi) The family $\mathfrak{L}$(2-NRCRA) is incomparable with $\mathfrak{L}$(2-DBCRA)

Proof. Follows from the proof of Theorem 4. □

5 Comparison with Siromoney Matrix Languages

In this section we give the comparison of the family of languages accepted by all the variants of 2-NCRFA with the family of Siromoney matrix languages. Here we consider RML^T to be the transposition of RML.

Theorem 6. *$\mathfrak{L}$(RML^T) = $\mathfrak{L}$(2-DCIRA) = $\mathfrak{L}$(2-NCIRA)*

Proof. The transposition of any regular matrix language is accepted by a returning finite automaton (RFA), as can be seen in [3], which can be viewed as a 2-DCIRA where $\Delta = \emptyset$ and which is also a particular 2-NCIRA. Hence it remains to prove that any 2-NCIRA accepts a regular matrix language.

While we consider a 2-NCIRA, during the application of Δ rules, the automaton performs a circular-column-interchanging operation and it interchanges the column of the currently read symbol of the input array with the last column. So, once the last letter of the current reading row is known, a simulating automaton can remember the current last symbol of the row in its finite control to act correctly. Also we see that, initially the last symbol in the current row of the input can be guessed and finally, it can also be verified. Thus during the acceptance of any input array it can be easily seen that every row as well as column is regular. This simply corresponds with the definition of right-linear matrix grammars, where the vertical string of intermediate symbols are generated by using right-linear grammar rules followed by horizontal generation in parallel using right-linear grammar rules, yielding the regular matrix language. Hence any 2-NCIRA accepts a regular matrix language. □

Theorem 7. *1. $\mathfrak{L}$(2-NLCRA) and $\mathfrak{L}$(CFML) are incomparable.*
2. $\mathfrak{L}$(2-NRCRA) and $\mathfrak{L}$(CFML) are incomparable.

Proof. To Prove: $\mathfrak{L}$(2-NLCRA) − $\mathfrak{L}$(CFML) $\neq \emptyset$ and $\mathfrak{L}$(2-NRCRA) − $\mathfrak{L}$(CFML) $\neq \emptyset$. Consider the language,

$$L = \left\{ X = \begin{array}{c} X_1 \\ \vdots \\ X_m \end{array} \in \{0,1,2\}^{**} \middle| \begin{array}{c} \text{each } X_i\text{'s are of size } 1 \times n \text{ and} \\ |X_i|_0 = |X_i|_1 = |X_i|_2, \forall \quad i = 1,2,\ldots,m \end{array} \right\}$$

This language can be generated by any 2-NLCRA and 2-NRCRA, as can be seen in Example 1. But $L \notin \mathfrak{L}$CFML. Hence, $\mathfrak{L}$(2-NLCRA) − $\mathfrak{L}$CFML $\neq \emptyset$ and $\mathfrak{L}$(2-NRCRA) − $\mathfrak{L}$(CFML) $\neq \emptyset$.

To prove: $\mathfrak{L}$(CFML) − $\mathfrak{L}$(2-NLCRA) $\neq \emptyset$ and $\mathfrak{L}$(CFML) − $\mathfrak{L}$(2-NRCRA) $\neq \emptyset$. Consider the language, $L' = \left\{ \begin{array}{c} X^m\ Y^m\ X\ Y^n\ X^n \\ (X^m\ \bullet^m\ X\ \bullet^n\ X^n)_i \\ X^m\ Y^m\ X\ Y^n\ X^n \end{array} \middle| \ n,m,i \geq 1 \right\}$. $L' \in \mathfrak{L}$(CFML), as can be seen in [12], the vertical string of intermediate symbols can be generated by using right-linear grammar rules followed vertical generation in parallel using context-free grammar rules.

However, this language cannot be generated by any 2-NBCRA. We prove this by the method of contradiction. Let us assume that the language is accepted by some 2-NBCRA, $M = (Q, \Sigma, \delta, \Delta, q_0, F)$ with $|Q| = t$ and no (useless) loops. Let $A = \begin{array}{c} X^{n+1}\ Y^{n+1}\ X\ Y^n\ X^n \\ (X^{n+1}\ \bullet^{n+1}\ X\ \bullet^n\ X^n)_i \\ X^{n+1}\ Y^{n+1}\ X\ Y^n\ X^n \end{array}$ with $n > 2$, $i > 0$ and $n > t$ be the input given to the automaton. By the pigeon hole principle, there exists a state, say p which appears at least twice during the computation on first row. Let the first occurrence of state p be reached in r steps which involves j normal transition, l_1 left column-revolving transition and r_1 right column-revolving transition,

$$\begin{array}{c} q_0\ X\ X^n\ Y^{n+1}\ X\ Y^n\ X^n\ \# \\ (\ X\ X^n\ \bullet^{n+1}\ X\ \bullet^n\ X^n\ \#)_i \\ X\ X^n\ Y^{n+1}\ X\ Y^n\ X^n\ \# \end{array} \vdash_M^* \begin{array}{c} X^j\ p\ X\ X^{n-j+l_1-r_1}\ Y^{n+1}\ X\ Y^n\ X^{n-l_1+r_1}\ \# \\ (X^j\ \ X\ X^{n-j+l_1-r_1}\ \bullet^{n+1}\ X\ \bullet^n\ X^{n-l_1+r_1}\ \#)_i \\ X^j\ \ X\ X^{n-j+l_1-r_1}\ Y^{n+1}\ X\ Y^n\ X^{n-l_1+r_1}\ \# \end{array}$$

Let the second occurrence of state p be reached in s steps which involves k normal transition, l_2 left column-revolving transition and r_2 right column-revolving transition,

$$\begin{array}{c} X^j\ p\ X\ X^{n-j+l_1-r_1}\ Y^{n+1}\ X\ Y^n\ X^{n-l_1+r_1}\ \# \\ (X^j\ \ X\ X^{n-j+l_1-r_1}\ \bullet^{n+1}\ X\ \bullet^n\ X^{n-l_1+r_1}\ \#)_i \\ X^j\ \ X\ X^{n-j+l_1-r_1}\ Y^{n+1}\ X\ Y^n\ X^{n-l_1+r_1}\ \# \end{array} \vdash_M^*$$

$$\begin{array}{c} X^{j+k}\ p\ X\ X^{n-j-k+l_1-r_1+l_2-r_2}\ Y^{n+1}\ X\ Y^n\ X^{n-l_1+r_1-l_2+r_2}\ \# \\ (X^{j+k}\ \ X\ X^{n-j-k+l_1-r_1+l_2-r_2}\ \bullet^{n+1}\ X\ \bullet^n\ X^{n-l_1+r_1-l_2+r_2}\ \#)_i \\ X^{j+k}\ \ X\ X^{n-j-k+l_1-r_1+l_2-r_2}\ Y^{n+1}\ X\ Y^n\ X^{n-l_1+r_1-l_2-r_2}\ \# \end{array}$$

Since the considered array A is in the language L, the computation reaches the accepting state, say $q_f \in Q$ i.e.,

$$\begin{array}{c} X^{j+k}\ p\ X\ X^{n-j-k+l_1-r_1+l_2-r_2}\ Y^{n+1}\ X\ Y^n\ X^{n-l_1+r_1-l_2+r_2}\ \# \\ (X^{j+k}\ \ X\ X^{n-j-k+l_1-r_1+l_2-r_2}\ \bullet^{n+1}\ X\ \bullet^n\ X^{n-l_1+r_1-l_2+r_2}\ \#)_i \\ X^{j+k}\ \ X\ X^{n-j-k+l_1-r_1+l_2-r_2}\ Y^{n+1}\ X\ Y^n\ X^{n-l_1+r_1-l_2-r_2}\ \# \end{array} \vdash_M^* \ q_f$$

We find that $l_2 - r_2 = 0$. Otherwise, the computation

$$\begin{array}{l} q_0\ X\ X^{n-k+l_2-r_2}\ Y^{n+1}\ X\ Y^n\ X^{n-l_2+r_2}\ \# \\ (\ X\ X^{n-k+l_2-r_1}\ \bullet^{n+1}\ \ X\ \bullet^n\ \ X^{n-l_2+r_2}\ \#)_i\ \vdash_M^* \\ \ \ X\ X^{n-j+l_1-r_1}\ Y^{n+1}\ X\ Y^n\ X^{n-l_1+r_1}\ \# \end{array}$$

$$\begin{array}{l} X^j\ p\ X\ X^{n-j-k+l_1-r_1+l_2-r_2}\ Y^{n+1}\ X\ Y^n\ X^{n-l_1+r_1-l_2+r_2}\ \# \\ (X^j\ \ \ X\ X^{n-j-k+l_1-r_1+l_2-r_2}\ \bullet^{n+1}\ \ X\ \bullet^n\ \ X^{n-l_1+r_1-l_2+r_2}\ \#)_i\ \vdash_M^*\ q_f \\ X^j\ \ \ X\ X^{n-j-k+l_1-r_1+l_2-r_2}\ Y^{n+1}\ X\ Y^n\ X^{n-l_1+r_1-l_2-r_2}\ \# \end{array}$$

accepts the array $A' = \begin{array}{c} X^{n+1-k+l_2-r_2}\ Y^{n+1}\ X\ Y^n\ \ X^{n-l_2+r_2} \\ (X^{n+1-k+l_2-r_2}\ \ \bullet^{n+1}\ \ X\ \ \bullet^n\ \ X^{n-l_2+r_2})_i \\ X^{n+1-k+l_2-r_2}\ Y^{n+1}\ X\ Y^n\ \ X^{n-l_2+r_2} \end{array} \notin L'$. For same reason we also find that $l_2 - r_2 - k = 0$ which implies $k = 0$. But this is a contradiction to our assumption because it follows either that $r = s$ or M loops. This implies L' is not accepted by any 2-NBCRA and hence by any 2-NLCRA and 2-NRCRA. Hence, $\mathfrak{L}$(CFML) − $\mathfrak{L}$(2-NLCRA) $\neq \emptyset$ and $\mathfrak{L}$(CFML) − $\mathfrak{L}$(2-NRCRA) $\neq \emptyset$. □

6 Closure Properties

In this section we discuss the closure properties of the family of languages accepted by any 2-DCIRA, 2-DLCRA, 2-DRCRA and 2-DBCRA.

Theorem 8. *$\mathfrak{L}$(2-DCIRA) is closed under complementation and union.*

Proof. From [14], RML is closed under complementation and union. By Theorem 6, this theorem is proved. □

Theorem 9. *$\mathfrak{L}$(2-DLCRA), $\mathfrak{L}$(2-DRCRA) and $\mathfrak{L}$(2-DBCRA) are not closed under complementation.*

Proof. Consider the language,

$$L = \left\{ X = \begin{array}{c} X_1 \\ \vdots \\ X_m \end{array} \in \{0,1\}^{**} \,\middle|\, \begin{array}{c} \text{each } X_i\text{'s are of size } 1 \times n \text{ and} \\ |X_i|_0 = |X_i|_1, \forall \quad i = 1, 2, \ldots, m \end{array} \right\}$$

By Theorem 2, the language $\bar{L} = \{\{0,1\}^{**} | \exists$ at least one row in which no. of 0s and 1s are not equal$\}$ is not accepted by any 2-DBCRA and hence by any 2-DLCRA and 2-DRCRA. But by Example 1, we see that the language L is accepted by some 2-DBCRA and hence by some 2-DLCRA and 2-DRCRA. □

Theorem 10. *$\mathfrak{L}$(2-DLCRA), $\mathfrak{L}$(2-DRCRA) and $\mathfrak{L}$(2-DBCRA) are not closed under union.*

Proof. Consider the languages,

$$L_1 = \left\{ X = \begin{matrix} X_1 \\ \vdots \\ X_m \end{matrix} \in \{0,1\}^{**} \,\middle|\, \begin{matrix} \text{each } X_i\text{'s are of size } 1 \times n \text{ and} \\ |X_i|_0 = |X_i|_1, \forall \quad i = 1, 2, \dots, m \end{matrix} \right\}$$

and $L_2 = \{0\}^{**}$. The language L_1 can be accepted by a 2-DBCRA, as can be seen in Example 1. L_2 can be accepted by a 2-DBCRA, $M = (\{q_0\}, \{0\}, \{q_0 0 \rightarrow q_0, q_0\# \rightarrow q_0\}, \emptyset, \emptyset, q_0, \{q_0\})$. By the same reasoning as in the proof of Theorem 9 we can show that $L_1 \cup L_2$ is not accepted by any 2-DBCRA and hence by any 2-DLCRA and 2-DRCRA. □

Theorem 11. *$\mathfrak{L}$(2-DCIRA) is closed under column catenation, but not under row catenation.*

Proof. From [14], RML is closed under column catenation, but not under row catenation. By Theorem 6, this theorem is proved. □

Theorem 12. *$\mathfrak{L}$(2-DLCRA), $\mathfrak{L}$(2-DRCRA) and $\mathfrak{L}$(2-DBCRA) are neither closed under column catenation nor row catenation.*

Proof. Consider the languages, $L_1 = \left\{ \begin{matrix} X^m \, Y^m \;\; X \\ (X^m \;\; .^m \;\; X)_i \\ X^m \; Y^m \; X \end{matrix} \,\middle|\, m, i \geq 1 \right\}$ and $L_2 = \left\{ \begin{matrix} Y^n \;\; X^n \\ (.^n \;\; X^n)_i \\ Y^n \;\; X^n \end{matrix} \,\middle|\, n, i \geq 1 \right\}$. Both the languages L_1 and L_2 belong to $\mathfrak{L}$(2-DLCRA).

But the column catenation, $L_1 \oplus L_2 = \left\{ \begin{matrix} X^m \, Y^m \; X \, Y^n \;\; X^n \\ (X^m \;\; .^m \;\; X \;\; .^n \;\; X^n)_i \\ X^m \; Y^m \; X \, Y^n \; X^n \end{matrix} \,\middle|\, n, m, i \geq 1 \right\}$ does not belong to $\mathfrak{L}$(2-DBCRA), as seen in proof of Theorem 7. This proves that $\mathfrak{L}$(2-DLCRA) and $\mathfrak{L}$(2-DBCRA) are not closed under column catenation.

If we consider the language,

$$L_3 = \left\{ X = \begin{matrix} X_1 \\ \vdots \\ X_m \end{matrix} \in \{0,1\}^{**} \,\middle|\, \begin{matrix} \text{each } X_i\text{'s are of size } 1 \times n \text{ and} \\ |X_i|_0 = |X_i|_1, \forall \quad i = 1, 2, \dots, m \end{matrix} \right\}$$

belonging to $\mathfrak{L}$(2-DLCRA) and also $\mathfrak{L}$(2-DRCRA), it is clear that $L_3 \ominus L_2$ does not even belong to $\mathfrak{L}$(2-DBCRA), because the columns shuffling involved during the simulation of the automaton accepting L_2 makes the simulation of the automaton accepting L_3 impossible. This proves that $\mathfrak{L}$(2-DLCRA), $\mathfrak{L}$(2-DRCRA) and $\mathfrak{L}$(2-DBCRA) is not closed under row catenation.

Now we consider the language,

$$L_4 = \left\{ X = \begin{matrix} 2X_1 \\ \vdots \\ 2X_m \end{matrix} \in \{0,1,2\}^{**} \,\middle|\, \begin{matrix} \text{each } X_i\text{'s are of size } 1 \times n \text{ and} \\ |X_i|_0 = |X_i|_1, \forall \quad i = 1, 2, \dots, m \end{matrix} \right\}$$

Clearly, the languages L_3 and L_4 belong to 2-DRCRA, but the language $L_1 \varobar L_2$ does not belong to 2-DRCRA (can be proved by a similar argument as in Theorem 1). Hence $\mathfrak{L}$(2-DRCRA) are not closed under column catenation.

Theorem 13. *$\mathfrak{L}$(2-DCIRA) is closed under column Kleene star, but not under row Kleene star.*

Proof. The proof follows from Theorem 11. □

Theorem 14. *$\mathfrak{L}$(2-DLCRA), $\mathfrak{L}$(2-DRCRA) and $\mathfrak{L}$(2-DBCRA) are neither closed under column Kleene star nor row Kleene star.*

Proof. The proof follows from Theorem 12. □

Theorem 15. *$\mathfrak{L}$(2-DCRFA) is not closed under quarter turn and transpose.*

Proof. For $\mathfrak{L}$(2-DCIRA) the properties are true, from [14] and Theorem 6.

Consider the language,

$L(M) = \left\{ X = \begin{matrix} X_1 \\ \vdots \\ X_m \end{matrix} \in \{0,1\}^{**} \middle| \begin{matrix} X_i = a_{i1}a_{i2}\dots a_{in} \\ |X_i|_0 = |X_i|_1, \forall \quad i = 1,2,\dots,m \end{matrix} \right\}$. This language can be generated by some 2-DBCRA, and hence by some 2-DLCRA and 2-DRCRA as can be seen in Example 1. Let us represent the quarter turn of this language by L^{QT} and we have,

$$L^{QT} = \left\{ X = X_m \dots X_1 \in \{0,1\}^{**} |\ X_i = \begin{matrix} a_{i1} \\ \vdots \\ a_{in} \end{matrix}, |X_i|_0 = |X_i|_1, \forall\ i = 1,2,\dots,m \right\}.$$

This language cannot be generated by any 2-DCRFA, because no such automaton can accept the input X from L^{QT} with the condition $|X_i|_0 = |X_i|_1, \forall\ i = 1,2,\dots,k$. This proves that $\mathfrak{L}$(2-DBCRA) is not closed under quarter turn.

Let us represent the transpose of the language L by L^T and we have, $L^T = \left\{ X = X_1 \dots X_m\ \in \{0,1\}^{**} |\ X_i = \begin{matrix} a_{i1} \\ \vdots \\ a_{in} \end{matrix}, |X_i|_0 = |X_i|_1, \forall\ i = 1,2,\dots,m \right\}$. By the same argument as discussed for L^{QT}, we see that there is no 2-DCRFA to accept L^T. This proves that $\mathfrak{L}$(2-DBCRA) is not closed under transpose. □

Theorem 16. *$\mathfrak{L}$(2-DCRFA) is closed under half turn, reflection on rightmost vertical and reflection on base.*

Proof. For $\mathfrak{L}$(2-DCIRA) the property is true, from [14] and Theorem 6. We shall give the proof for $\mathfrak{L}$(2-DRCRA) and reflection about the rightmost vertical, proof being similar for the rest of the cases. Let $M = (Q, \Sigma, \delta, \Delta, q_0, F)$ be a 2-DRCRA that accepts the language $L = L(M)$. We now construct a 2-DRCRA, M' that accepts the language obtained by taking reflection on the rightmost vertical of L represented by L^{RB}. $M' = (Q \cup \{q_0'\}, \Sigma, \delta', \Delta, q_0', F')$, where $\delta' = \delta \cup \{q_0' \to q \mid p\# \to q \in \delta\} \cup \{p \to q_0 \mid p \in Q\} \cup \{q \to p \mid q \in F, p \in Q\}$ and $F' = \{q \mid q_0 \to q_1, q_1 \to q_2, \dots, q_{n-1} \to q_n, q_n\# \to q \in \delta'\}$. Clearly, $L(M') = L^{RB}$. □

7 Application

One of the applications of this newly constructed automata is in the field of steganography [6]. Steganography is the art and science of hiding information by embedding messages within others. This hidden information can be plain text, cipher text, or even images. Steganographic communications hide often in plain sight, whereas encrypted communications in Cryptography are very obvious of the fact that they are sending secrets.

Here we consider hiding text messages behind some pictures which can be retrieved only by giving them as input in the newly constructed automata and arriving at the final configuration. For example, let us consider the 2-DRCRA automaton from Example 1 and an array $\begin{array}{cccccc} 2&2&1&0&1&0\\ 2&1&0&1&0&2\\ 0&1&2&1&0&2 \end{array}$. Let $\begin{array}{cccccc} T&\square&\square&S&H&I\\ I&\square&\square&A&S&\square\\ E&S&T&E&C&R \end{array}$ be the scrambled hidden message behind the array A. Now to get the proper hidden message $\begin{array}{cccccc} \square&T&H&I&S&\square\\ \square&I&S&\square&A&\square\\ S&E&C&R&E&T \end{array}$, the array A is given as the input to the automaton and an accepting configuration has to be reached. The working of this automaton on array A along with the hidden message is as follows:

$$\begin{array}{rccccccc} q_0&2&2&1&0&1&0&\#\\ &2&1&0&1&0&2&\#\\ &0&1&2&1&0&2&\# \end{array} \qquad \begin{array}{cccccc} I&T&H&S&\square&\square\\ \square&I&S&A&\square&\square\\ R&E&C&E&S&T \end{array}$$

$$\vdash^*$$

$$\begin{array}{rccccccc} &0&1&2&0&1&2&\#\\ q_0&1&0&2&2&0&1&\#\\ &1&0&0&2&2&1&\# \end{array} \qquad \begin{array}{cccccc} S&\square&I&\square&H&T\\ A&\square&\square&\square&S&I\\ E&S&R&T&C&E \end{array}$$

$$\vdash^*$$

$$\begin{array}{rccccccc} &1&2&2&1&0&0&\#\\ &0&1&2&0&1&2&\#\\ q_0&0&1&0&2&1&2&\# \end{array} \qquad \begin{array}{cccccc} \square&T&I&H&S&\square\\ \square&I&\square&S&A&\square\\ S&E&R&C&E&T \end{array}$$

$$\vdash^*$$

$$q_0 \qquad \begin{array}{cccccc} &T&H&I&S&\\ &I&S&&A&\\ S&E&C&R&E&T \end{array}$$

8 Conclusion

It is worth examining the decidability results (non-emptiness, membership, finiteness, etc.) of these automata and explore their further applications in character recognition and floor designs. Construction of two-dimensional automaton which involves both column-revolving and row-revolving can also be studied.

References

1. Bensch, S., Bordihn, H., Holzer, M., Kutrib, M.: On input-revolving deterministic and nondeterministic finite automata. Inf. Comput. **207**, 1140–1155 (2009)
2. Blum, M., Hewitt, C.: Automata on a 2-dimensional tape. In: SWAT 1967, 8th Annual Symposium on Switching and Automata Theory, pp. 155–160 (1967)
3. Fernau, H., Paramasivan, M., Schmid, M.L., Thomas, D.G.: Scanning pictures the boustrophedon way. In: Barneva, R.P., Bhattacharya, B.B., Brimkov, V.E. (eds.) IWCIA 2015. LNCS, vol. 9448, pp. 202–216. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26145-4sps15
4. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3, pp. 215–267. Springer, Heidelberg (1997)
5. Kamala, K., Siromoney, R.: Array automata and operations on array languages. Int. J. Comput. Appl. **4**(1–4), 3–30 (1974)
6. Kipper, G.: Investigator's Guide to Steganography. Auerbach Publications, Boca Raton (2003)
7. Meduna, A.: Automata and Languages: Theory and Applications. Springer, Heidelberg (2000)
8. Pradella, M., Cherubini, A., Crespi-Reghizzi, S.: A unifying approach to picture grammars. Inf. Comput. **209**, 1246–1267 (2011)
9. Rosenfeld, A.: Picture Languages: Formal Models for Picture Recognition. Acadamic Press, Cambridge (1979)
10. Rosenfeld, A., Siromoney, R.: Picture languages - a survey. Lang. Des. **1**(3), 229–245 (1993)
11. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, vols. 1–3. Springer, Heidelberg (1997)
12. Siromoney, G., Siromoney, R., Krithivasan, K.: Abstract families of matrices and picture languages. Comput. Graph. Image Proc. **1**, 284–307 (1972)
13. Siromoney, R.: Advances in array languages. In: Ehrig, H., Nagl, M., Rozenberg, G., Rosenfeld, A. (eds.) Graph Grammars 1986. LNCS, vol. 291, pp. 549–563. Springer, Heidelberg (1987). doi:10.1007/3-540-18771-5_75
14. Siromoney, G., Siromoney, R., Krithivasan, K.: Picture languages with array rewriting rules. Inf. Control **22**, 447–470 (1973)
15. Siromoney, R., Subramanian, K.G., Rangarajan, K.: Parallel/sequential rectangular arrays with tables. Int. J. Comput. Math. **6A**, 143–158 (1977)
16. Wang, P.S.P.: Array Grammars, Patterns and Recognizers. World Scientific Series in Computer Science, vol. 18. World Scientific Publishing, Singapore (1989)
17. Wang, P.S.P.: Sequential/parallel matrix array languages. J. Cybern. **5**, 19–36 (1975)