Math. Appl. **5** (2016), 105–122
DOI: 10.13164/ma.2016.08

TWO-DIMENSIONAL JUMPING FINITE AUTOMATA

S. JAMES IMMANUEL AND D. G. THOMAS

Abstract. In this paper, we extend a newly introduced concept called the jumping finite automata for accepting string languages to two-dimensional jumping finite automata for accepting two-dimensional languages. We discuss some of the basic properties of these automata and compare the family of languages accepted by these automata with the family of Siromoney matrix languages and also recognizable picture languages (REC). We also discuss some of the closure properties of these automata along with some of their decidability properties.

1. Introduction

Two-dimensional languages are sets of pictures or two-dimensional arrays of symbols chosen from a finite alphabet. They play important roles in image processing, pattern recognition, character recognition and also in studies concerning cellular automata and other models of computing. A number of rectangular picture generating and recognizing mechanisms such as grammars and automata have been proposed in the literature [10–13, 16, 17, 19]. Motivated by certain floor designs called "Kolam" patterns, Siromoney et al. [15] introduced Siromoney Matrix Grammars (SMG), a simple and elegant grammar model for generating rectangular arrays. This is one of the earliest picture models and has been widely investigated for its theoretical properties and applications. Several variations of Siromoney matrix grammars have been made in the literature [18, 20]. A particular case is the Siromoney Array Grammars (AG) where array rewriting rules are deployed in the derivation of pictures. Derivations are restricted by the conditions for row and column catenation.

The isometric array grammars, introduced by A. Rosenfeld [11], impose the constraint that the left and right parts of a rewriting rule must be of same size; this condition overcomes the inherent problem of "shearing" which pops up while substituting a sub-array in a host array. In [20], Wang presents a new system for generating two-dimensional patterns using matrix grammars and array grammars. This model corresponds to a sequential/parallel system. In [21], Wang introduces parallel context-free array languages and studies the relationship between parallel isometric array languages and sequential isometric array languages. Tiling system [4, 5] is a notion of recognizability of a set of pictures belonging to the family REC of recognizable picture languages. In this system, recognizability is defined

MSC (2010): primary 68Q45.
Keywords: picture languages, jumping automata, rectangular arrays.

by "projection of local properties". The class REC is regarded as the generalization of the class of regular (one-dimensional) languages because it unifies several approaches to define the two-dimensional analogue of regular languages via finite automata, grammars, logic and regular expressions.

An important role in the theory of picture recognizability is played by automata. The generalization of a finite state automaton to two-dimensional languages can be attributed to M.Blum and C. Hewitt [3]. They have introduced the notion of 4-way automata. Array automata acting on scenes (two-dimensional tapes) are defined in [8]. An interesting model of automaton to recognize REC class of picture languages is the two-dimensional online tessellation acceptor [6]. Recently, in [9], Wang automaton, a model of automaton for rectangular picture language recognition, has been introduced, which combines the features of both online tessellation acceptors and 4-way automata but differs in the aspect of having many scanning strategies to visit the input picture.

The motivation for this paper is based on Jumping Finite Automata [1] for string languages. Most of the classical computer science methods developed in the previous century is for continuous information processing and hence their formal models, such as finite automata work strictly continuously from left to right in a symbol-by-symbol way. But lately we have been able to see computational methods frequently processing information in a discontinuous way. Within a particular running process, a typical computational step may be performed somewhere in the middle of information while the very next computational step is executed far away from it; therefore, before the next step is carried out, the process has to jump over a large portion of the information to the desired position of execution. This gives rise to the idea of adapting classical formal models in a discontinuous way. In this way, the study in [1] introduces new versions of finite automata, referred to as jumping finite automata which have always had a central role in computer science as a whole presenting a new, attractive, significant and up-to-date topic of modern automata theory.

This paper is a major improvement to and development of [7] that has recently been introduced in the literature. Here the concept of Jumping Finite Automata [1] is extended to two-dimensional languages. New computational models based on two-dimensional languages are always of great interest. The jumping finite automata work like classical finite automata except that they read input words discontinuously – that is, after reading a symbol, they can jump over some symbols with the words and continue their computation from there. A two-dimensional jumping finite automaton works on an input array row by row and every row computation is similar to that of a jumping finite automaton working on strings except that the symbols that are read are replaced by dummy symbols. Here we also introduce another jump called row jump performed by the automaton after every symbol in that row is replaced by a dummy symbol. The paper also establishes some results concerning two-dimensional row jumping finite automata in terms of commonly investigated areas of automata theory, such as closure properties and decidability properties. We also compare the class of languages accepted by these automata with the class of Siromoney matrix languages [15] and also the class of recognizable picture languages [4, 5]. We conclude the paper with a brief remark.

2. Preliminaries

In this section we recall some notions related to formal language theory and array grammars (refer to [2, 14]). Let Σ be a finite alphabet, card(Σ) denotes the cardinality of Σ. Σ^* is the set of words over Σ including the empty word λ. $\Sigma^+ = \Sigma^* - \{\lambda\}$. An array consists of finitely many symbols from Σ that are arranged as rows and columns in some particular order. A rectangular $m \times n$ array A is written in the form, $A = \begin{matrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{matrix}$ 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 $\wedge$(zero rows and zero columns). $\Sigma^{++} = \Sigma^{**} - \{\wedge\}$. For $A \in \Sigma^{**}, alph(A)$ denotes the set of symbols occurring in the array$A$. For $a \in \Sigma$, $|A|_a$ denotes the number of occurrences of a in A. $|A|_c$ is the number of columns in picture A and $|A|_r$ is the number of rows in picture A. Given a picture $A \in \Sigma^{**}$ of size (m, n), we define $\hat{A}$ as the picture of size $(m, n+2)$ obtained by adjoining a special symbol $\$ \notin \Sigma$ at the left end and $\# \notin \Sigma$ at the right end of each row. The transpose of A is $A^T = \begin{matrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{matrix}$. The quarter turn of A is $A^{QT} = \begin{matrix} a_{m1} & \cdots & a_{11} \\ \vdots & \ddots & \vdots \\ a_{mn} & \cdots & a_{1n} \end{matrix}$. The half turn of A is $A^{HT} = \begin{matrix} a_{mn} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{11} \end{matrix}$. The reflection on base of A is $A^{RB} = \begin{matrix} a_{m1} & \cdots & a_{mn} \\ \vdots & \ddots & \vdots \\ a_{11} & \cdots & a_{1n} \end{matrix}$. The reflection on the rightmost vertical of A is $A^{RR} = \begin{matrix} a_{1n} & \cdots & a_{11} \\ \vdots & \ddots & \vdots \\ a_{mn} & \cdots & a_{m1} \end{matrix}$.

Definition 2.1. Let A $= \begin{matrix} a_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} \end{matrix}$, B $= \begin{matrix} b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nq} \end{matrix}$. The column catenation A$\oplus$B $= \begin{matrix} a_{11} & \cdots & a_{1p} & b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} & b_{n1} & \cdots & b_{nq} \end{matrix}$ is defined only when $m = n$.

Similarly, the row catenation A$\ominus$B $= \begin{matrix} a_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} \\ b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nq} \end{matrix}$ is defined only when $p = q$.

Definition 2.2. The column shuffle operation on arrays denoted by $\sqcup\!\sqcup_c$, is defined recursively by, $P \sqcup\!\sqcup_c Q = (A \oplus X \sqcup\!\sqcup_c B \oplus Y) = A(X \sqcup\!\sqcup_c B \oplus Y) \cup B(A \oplus X \sqcup\!\sqcup_c Y)$, where $P = A \oplus X$ and $Q = B \oplus Y, P, Q \in \Sigma^{**}$, A is the first column of the array P and B is the first column of array Q. The operation is defined only when the number of rows in P and the number of rows in Q are equal. If A is empty, then $X = P$. Likewise, if B is empty, then $Y = Q$. Also $P \sqcup\!\sqcup_c \wedge = \wedge \sqcup\!\sqcup_c P = P$.

Similarly, we can define a row shuffle operation.

Definition 2.3. 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 non-terminals*; V_v is a finite set of *vertical non-terminals*; $\Sigma_I \subseteq V_v$ 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 the 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, regarding 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 the downward direction. The language generated by a CSMG(CFMG, RLG) is called a CSML(CFML, RML). For more information, we can refer to [15]. We denote the family of Siromoney matrix languages by $\mathfrak{L}(X)$, where $X \in$ {CSML,CFML,RML}.

Definition 2.4. A two-dimensional language is "Tiling Recognizable" (REC) if it can be obtained as a projection of a local picture language.

We can refer to [4] and [5] for further details about REC languages.

Definition 2.5. Let $A = \begin{matrix} A_1 \\ \vdots \\ A_m \end{matrix}$, where $A_i = a_{i1}a_{i2}\ldots a_{in}$, $a_{ij} \in \Sigma$ for all $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$, for some $m, n \geq 0$. The set of all row-wise column permutations of A, $rwcolperm(A)$, is defined as $rwcolperm(A) = \left\{ \begin{matrix} B_1 \\ \vdots \\ B_m \end{matrix} \middle| B_i = (b_{i1}b_{i2}\ldots b_{in}) \in \{alph(A_i)\}^* \ \forall \ i = 1, 2, \ldots, m, \text{ and } (b_{i1}, b_{i2}, \ldots, b_{in}) \right.$ is a permutation of $\left. (a_{i1}, a_{i2}, \ldots, a_{in}) \right\}$. For $L \subseteq \Sigma^{**}$, we define $rwcolperm(L) = \cup_{A \in L} rwcolperm(A)$. The set of all row permutations of A, $rowperm(A)$ is defined as $rowperm(A) = \left\{ \begin{matrix} B_1 \\ \vdots \\ B_m \end{matrix} \middle| B_i \in \{alph(A)\}^* \ \forall \ i = 1, 2, \ldots, m, \text{ and} \right.$

$(B_1, B_2, \ldots, B_m)$ is a permutation of $(A_1, A_2, \ldots, A_m)$ $\Big\}$. For $L \subseteq \Sigma^{**}$, we define $rowperm(L) = \cup_{A \in L} rowperm(A)$.

3. Two-dimensional jumping finite automaton

In this section we introduce a two-dimensional general row jumping finite automaton and explain its working with an example.

Definition 3.1. A two-dimensional general row jumping finite automaton, a 2-GRJFA for short, is an octuple $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$, where,

— Q is the finite set of states;
— $Q' \subset Q$ is a finite set of check and row jump states;
— Σ is an input alphabet;
— $R_1 \subseteq (Q \times \Sigma^* \times Q)$;
— $R_2 \subseteq (Q \times \$ \times Q') \cup \{r \times \boxtimes \times r | r \in Q'\}, \$, \boxtimes \notin \Sigma$;
— $R_3 \subseteq Q' \times \# \times Q, \# \notin \Sigma$;
— $s \in Q$ is the initial state;
— $F \subseteq Q$ is the set of final states.

The elements of $R = R_1 \cup R_2 \cup R_3$ are the rules of M and, instead of $(p, y, q) \in R$, we write $py \to q \in R$. The rules in R_1 perform normal jumping operation, denoted by $\curvearrowright$, the rules in R_2 perform normal transitions from left to right, denoted by $\to$ and the rules in R_3 perform row jumping operation, denoted by $\rightsquigarrow$. The normal jumping operator is defined as follows: Let $A_{m \times n} \in \Sigma^{**}$ be an input array of M. Then, $\hat{A}$ is considered in the two-dimensional tape and each of its rows is regarded as a string. Initially, let xyz with $x, y, z \in \Sigma^*$ be any row of A. Let $py \to q \in R_1$ with $y \in \Sigma^*$, $y' \in \Sigma^*$, $x, z, x', z', x'_1, x'_2, z'_1, z'_2 \in (\Sigma \cup \{\boxtimes\})^*$ with $|\boxtimes|_c = 1$, such that either $xz = x'_1 x'_2 y' z'$ with $x' = x'_1 \boxtimes^{|y|_c} x'_2$ or $xz = x' y' z'_1 z'_2$ with $z' = z'_1 \boxtimes^{|y|_c} z'_2$ and $|xyz|_c = |x'y'z'|_c > 0$. For all y', M makes a jump either left or right from $xpyz$ to $x'qy'z'$ using the rule $py \to q \in R_1$, written as $xpyz \curvearrowright x'qy'z'$, where y is replaced by $\boxtimes^{|y|_c}$ and is either in x' or z'. Here $\boxtimes$ helps to maintain the length of the row, so that the initial rectangular form of the array A is maintained. If there is some y' to which no rule in R_1 can be applied, M halts without accepting the input array A. The normal jumping operator is continually applied until all the symbols in the current row are completely read and replaced with $\boxtimes$s. When there is no y' implying that $|x|_c = |x|_{\boxtimes}$, $|z|_c = |z|_{\boxtimes}$ and M is in state p reading the symbol y, M moves to state q replacing the symbol y with $\boxtimes^{|y|_c}$ and jumps to \$ at the left end of the row. Now a rule $q\$ \to r \in R_2$ is applied which allows M to enter check and row jump state $r \in Q'$ on reading \$. Now a rule $r\boxtimes \to r \in R_2$ is continually applied to check whether every symbol in the current row has been read and replaced with $\boxtimes$. After the check, M reads the symbol $\#$. Now M performs another jump $\rightsquigarrow$ called the row jump. The row jumping operator $\rightsquigarrow$ is defined as follows: Let $X, Z, X', Z' \in \Sigma^{**}$, $u', v' \in \Sigma^*$, $u \in \{\boxtimes\}^*$, $r\# \to p \in R$, for some $p \in Q$; M makes a row jump either up or down from $\begin{matrix} \hat{X} \\ \$ur\# \\ \hat{Z} \end{matrix}$ to $\begin{matrix} \hat{X'} \\ \$u'pv'\# \\ \hat{Z'} \end{matrix}$ denoted by, $\begin{matrix} \hat{X} \\ \$ur\# \\ \hat{Z} \end{matrix} \rightsquigarrow \begin{matrix} \hat{X'} \\ \$u'pv'\# \\ \hat{Z'} \end{matrix}$ with $X \ominus Z = X' \ominus uv \ominus Z'$,

$|X|_c = |Z|_c = |X'|_c = |u'v'|_c = |Z'|_c$, $u = \boxtimes^{|X|_c}$. $|X|_r + |Z|_r = |X'|_r + |Z'|_r + 1$, $u'v'$ is any row of either X or Z. The row $u = \boxtimes^{|X|_c}$ is removed and Z shifts one row up after the row jump is performed.

A successful computation on an array A by a 2-GRJFA is as follows:

The automaton can start its computation from any symbol on any row of the input array A of size $m \times n$ depending on the symbol that the start state reads from the given rule R_1. Based on R_1, the automaton starts its computation on a row using the binary jumping relation $\curvearrowright$ and it is continually applied until all the symbols in the entire row have been read and replaced with $\boxtimes$. Using the rules in R_2, the automaton checks whether every symbol in the current row is replaced with $\boxtimes$. After the check, the automaton reads # and uses a rule in R_3 to performs the row jump $\wr$ deleting the current row and shifting the remaining array below it one row up. Now for further computation, the array of size $(m-1) \times n$ is considered and the binary jumping relation $\curvearrowright$ is applied as before on some other row of the input array A. The above process is repeated recursively row by row until all the elements of the array are replaced by $\boxtimes$s and the machine enters the final state. In this case, A is accepted.

Combining the normal jumping operator $\curvearrowright$, the normal transition $\rightarrow$ and the row jumping operator $\wr$, we define a complete row jump operator $\wr\!\!\curvearrowright$ as follows:

Definition 3.2. Let $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ be a 2-GRJFA. Let $U \in \Sigma^{**}$, with $U = \begin{matrix} u_1 \\ \vdots \\ u_m \end{matrix}$ and each $u_i = u_{i1}u_{i2}\dots u_{in}$ where $u_{if} \in \Sigma$, $\forall\ i = 1, 2, \dots, m$ and $f = 1, 2, \dots, n$. Let us consider some $u_j = u_{j1}u_{j2}\dots u_{j(r-1)}u_{jr}\dots u_{jn}$ and $u_k = u_{k1}u_{k2}\dots u_{k(t-1)}u_{kt}\dots u_{kn}$ with $j \neq k$ and $j, k = 1, 2, \dots, m$ along with some rules $s_{jg}u_{jr}\dots u_{jh} \rightarrow s_{jl} \in R_1, s_{j(n+1)}\$ \rightarrow s_{j(n+2)}, s_{j(n+2)}\boxtimes \rightarrow s_{j(n+2)} \in R_2, s_{j(n+2)}\# \rightarrow s_{kg} \in R_3$ where $s_{jg}, s_{j(n+1)}, s_{kg} \in Q, s_{j(n+2)} \in Q'\ \forall\ g, r, h, l = 1, 2, \dots, n$, $g \neq l$ and $r \leq h$. We define $\wr\!\!\curvearrowright$ as,

$\$u_{j1}u_{j2}\dots u_{j(r-1)}s_{jg}u_{jr}\dots u_{jn}\# \quad \wr\!\!\curvearrowright \quad \$u_{k1}u_{k2}\dots u_{k(t-1)}s_{kg}u_{kt}\dots u_{kn}\# = \$u_{j1}u_{j2}\dots u_{j(r-1)}s_{jg}u_{jr}\dots u_{jn}\# \curvearrowright^* s_{j(n+1)}\$\boxtimes\boxtimes\dots\boxtimes\# \rightarrow \$s_{j(n+2)}\boxtimes\boxtimes\dots\boxtimes\# \rightarrow^* \$\boxtimes\boxtimes\dots\boxtimes s_{j(n+2)}\# \wr u_{k1}u_{k2}\dots u_{k(t-1)}s_{kg}u_{kt}\dots u_{kn}\#$.

Let $\wr\!\!\curvearrowright^*$ denote the transitive-reflexive closure of $\wr\!\!\curvearrowright$. The language accepted by M, denoted by $L(M)$ is,

$$L(M) = \left\{ U = \begin{matrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{matrix} \in \Sigma^{**} \;\middle|\; usv \begin{matrix} u_1 \\ \vdots \\ \wr\!\!\curvearrowright^* \\ \vdots \\ u_m \end{matrix} f, f \in F, \text{ for some } u_k = uv, k = 1, 2, \dots, m \right\}.$$

Definition 3.3. A transition graph for $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$, a 2-GRJFA denoted as $\Delta(M)$ is defined as a labeled directed multi-graph $(Q, R = R_1 \cup R_2 \cup R_3)$, where $py \rightarrow q \in R$ with $p, q \in Q, y \in \Sigma^* \cup \{\boxtimes, \#\}$ implies that there is an edge from p to q labeled by y. If there is a path from s to q in $\Delta(M)$, then state q is reachable; q is terminating if there is a path from q to $f \in F$. We write $pY \rightsquigarrow q$, if there is a path from p to q, where for all $i = 1, 2, \dots, m$, $j = 1, 2, \dots, n$, $q_{ij} \in Q$,

with $p = q_{11}, q = q_{(m+1)n}$, $Y = \begin{matrix} Y_1 \\ \vdots \\ Y_m \end{matrix}$, with $Y_i = y_{i1}y_{i2}\dots y_{in}$ and $q_{ij}y_{ij} \to q_{i(j+1)} \in R_1, q_{i(n+1)}\$ \to q_{i(n+2)}, q_{i(n+2)}\boxtimes \to q_{i(n+2)} \in R_2, q_{i(n+2)}\# \to q_{(i+1)1} \in R_3$.

Example 3.4. Let us consider an example for a 2-GRJFA,
$M = (\{s, p\}, \{p\}, \{0, 1\}, \{s01 \to s\}, \{s\$ \to p, p\boxtimes \to p\}, \{p\# \to s\}, s, \{s\})$
$L(M) = \{(01)^m_n | m, n \geq 1\}$.

Working: Consider the array, $A = \begin{array}{|c|c|c|c|c|c|}\hline 0&1&0&1&0&1\\\hline 0&1&0&1&0&1\\\hline 0&1&0&1&0&1\\\hline 0&1&0&1&0&1\\\hline\end{array}$. The automaton can start its computation from any symbol on any row depending on the symbol that the start state reads from the given rule. Say that the automaton starts its computation from the third symbol of the second row, the successive computations carried out by the automaton to reach a successful computation are as follows:

\$	0	1	0	1	0	1	#
\$	0	1	s0	1	0	1	#
\$	0	1	0	1	0	1	#
\$	0	1	0	1	0	1	#

$\curvearrowright$

\$	0	1	0	1	0	1	#
\$	0	1	$\boxtimes$	$\boxtimes$	s0	1	#
\$	0	1	0	1	0	1	#
\$	0	1	0	1	0	1	#

$\curvearrowright$

\$	0	1	0	1	0	1	#
\$	s0	1	$\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$	#
\$	0	1	0	1	0	1	#
\$	0	1	0	1	0	1	#

$\curvearrowright$

s\$	$\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$	#
\$	0	1	0	1	0	1	#
\$	0	1	0	1	0	1	#

$\to$

\$	0	1	0	1	0	1	#
\$	$p\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$	#
\$	0	1	0	1	0	1	#
\$	0	1	0	1	0	1	#

$\to^*$

\$	0	1	0	1	0	1	#
\$	$\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$	p#
\$	0	1	0	1	0	1	#
\$	0	1	0	1	0	1	#

$\curvearrowleft$

\$	0	1	0	1	0	1	#
\$	0	1	0	1	0	1	#
\$	0	1	0	1	s0	1	#

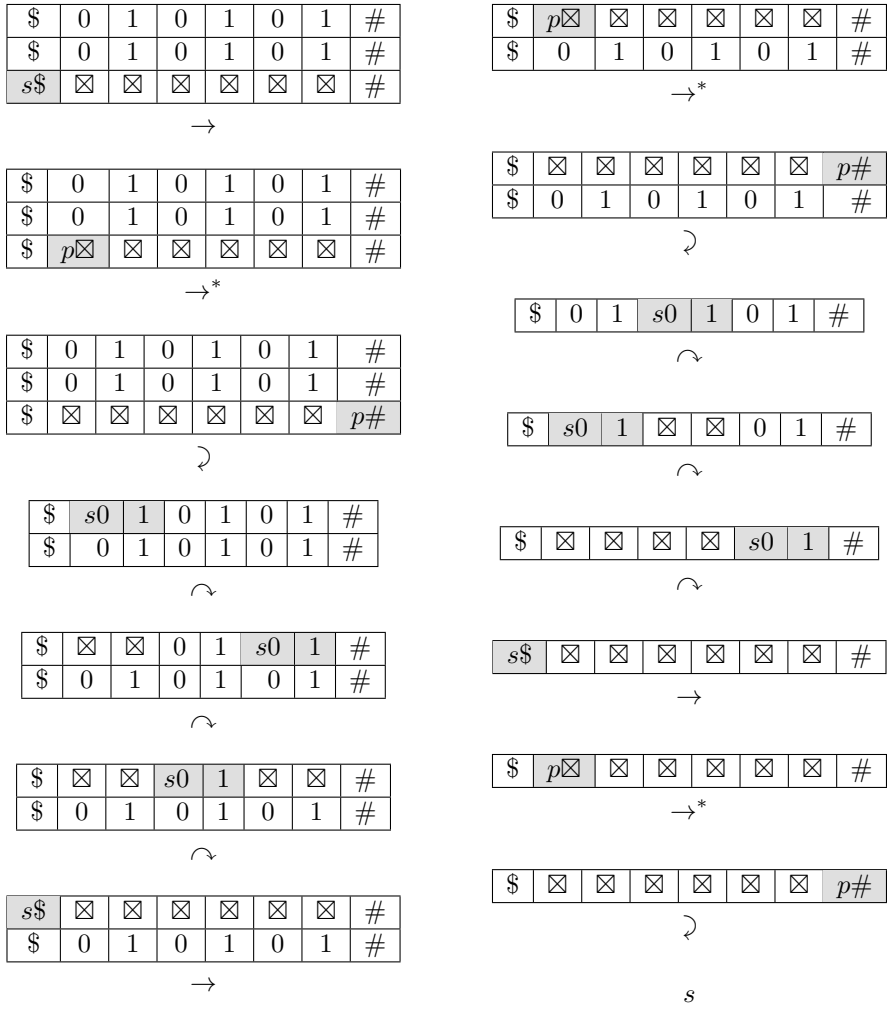
$\curvearrowright$

\$	0	1	0	1	0	1	#
\$	0	1	0	1	0	1	#
\$	s0	1	0	1	$\boxtimes$	$\boxtimes$	#

$\curvearrowright$

\$	0	1	0	1	0	1	#
\$	0	1	0	1	0	1	#
\$	$\boxtimes$	$\boxtimes$	s0	1	$\boxtimes$	$\boxtimes$	#

$\curvearrowright$

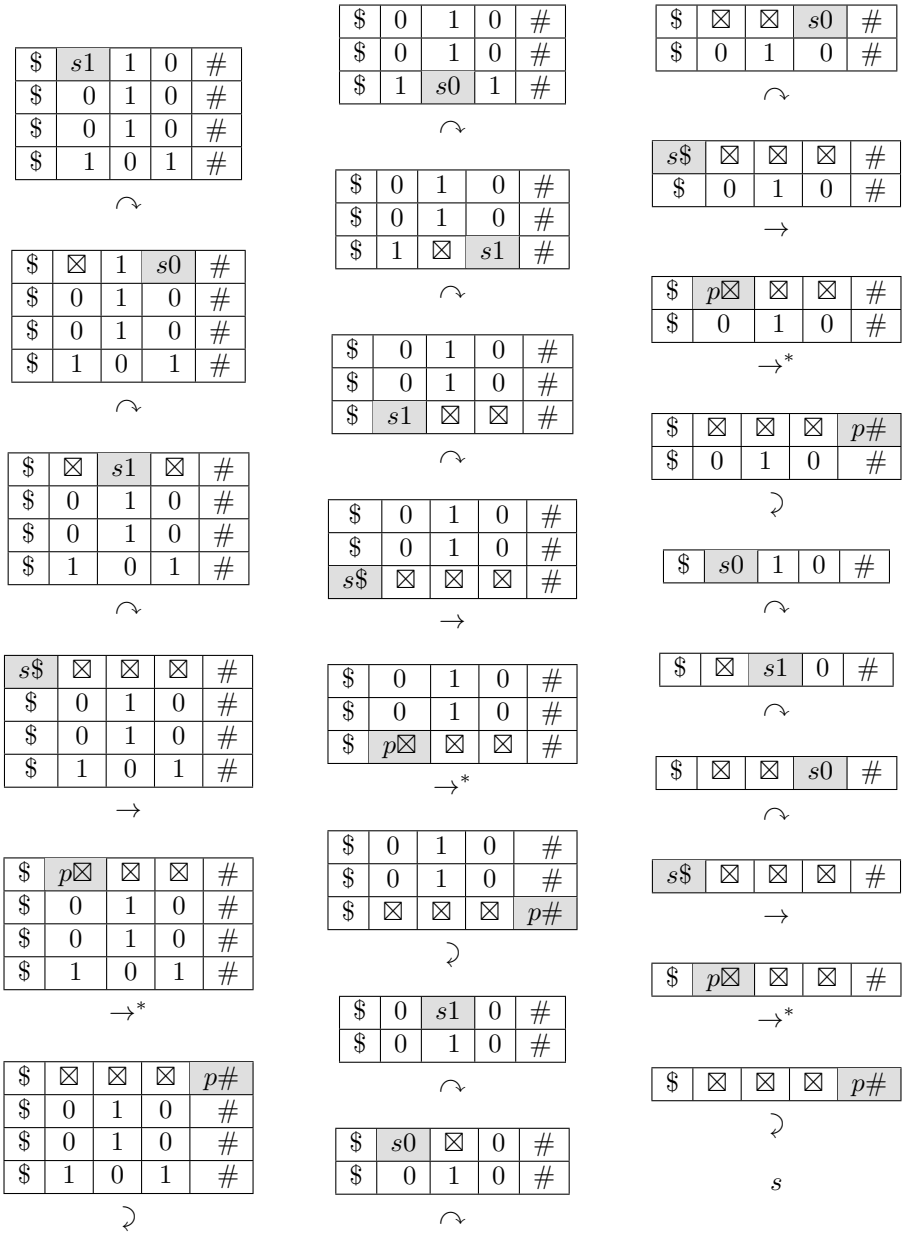


Here, we observe from the working of M that it can start its computation from any row of the given input array A, but once it starts computing on a row, it has to completely read and replace every symbol in that row with ⊠ which is followed by a check that every symbol in that row is replaced with ⊠ and only then it is allowed to perform the row jumping operation to any other row.

Definition 3.5. Let $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ be a 2-GRJFA. M is an ϵ-free 2-GRJFA if $py \to q \in R_1$ implies that $|y| \geq 1$. M is of degree n, where $n \geq 0$, if $py \to q \in R_1$ implies that $|y| \leq n$. M is a two-dimensional row jumping finite automaton, a 2-RJFA for short, if its degree is 1.

Example 3.6. Example for 2-RJFA
$M = (\{s, p\}, \{p\}, \{0, 1\}, \{s0 \to s, s1 \to s\}, \{s\$ \to p, p⊠ \to p\}, \{p\# \to s\}, s, \{s\})$
$L(M) = \{0, 1\}^{**}$

Working: Consider the input array, $A = \begin{array}{|c|c|c|}\hline 1 & 1 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array}$. The automaton can start its computation from any symbol of any row depending on the symbol that the start state reads from the given rule. If it starts from the first symbol of the first row, the successive computations are as follows:



Example 3.7. An example of a language not accepted by any 2-GRJFA is

$$L = \left\{ \begin{matrix} X \\ X \end{matrix} \begin{matrix} . \\ X \end{matrix}, \begin{matrix} X & . \\ X & . \\ X & X \end{matrix}, \begin{matrix} X & . & . \\ X & . & . \\ X & X & X \end{matrix}, \begin{matrix} X & . & . & . \\ X & . & . & . \\ X & X & X & X \end{matrix}, \ldots \right\}$$

i.e., the token L of all sizes and of all proportions.

4. Some basic properties

In this section we give some basic properties of two-dimensional general row jumping finite automata.

Theorem 4.1. *$\mathfrak{L}$(2-RJFA) $\subset$ $\mathfrak{L}$(2-GRJFA).*

Proof. $\mathfrak{L}$(2-RJFA)$\subseteq$ $\mathfrak{L}$(2-GRJFA) follows directly from the definition of 2-RJFA. Also $\mathfrak{L}$(2-GRJFA) $-$ $\mathfrak{L}$(2-RJFA) $\neq \emptyset$, because the language accepted by the 2-GRJFA in example 3.4 is not accepted by any 2-RJFA. □

Theorem 4.2. *Let $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ be a 2-GRJFA such that $card(\Sigma) = 1$. Then $L(M)$ is RML.*

Proof. The proof is straightforward. □

Theorem 4.3. *Let L be an arbitrary array language. Then, $L \in$2-RJFA only if $L = rowperm(L)$ and $L = rwcolperm(L)$.*

Proof. Let $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ be a 2-RJFA. Let $A \in L(M)$. We prove that $rwcolperm(A) \subseteq L(M)$. If $A = \wedge$, then $rwcolperm(\wedge) = \wedge \in L(M)$.

We assume that $A \neq \wedge$. Let $A = \begin{matrix} A_1 \\ \vdots \\ A_m \end{matrix}$, where $A_i = a_{i1}a_{i2}\ldots a_{in}$, $a_{ij} \in \Sigma$ for all $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$, for some $m, n \geq 0$. Since $A \in L(M)$, we consider $\hat{A}$ and we have, $sa_{ij_1} \to s_{ij_1}, s_{ij_1}a_{ij_2} \to s_{ij_2}, \ldots, s_{ij_{n-1}}a_{ij_n} \to s_{ij_n} \in R_1, s_{ij_n}\$ \to s_{ij_{n+1}}, s_{ij_{n+1}}\boxtimes \to s_{ij_{n+1}} \in R_2, s_{ij_{n+1}}\# \to s_{(i+1)j_1} \in R_3$ for all $i = 1, 2, \ldots, m$ where, $s_l \in Q$, for all $l \in \{ij_1, ij_2, \ldots, ij_n, ij_{n+1}\} \cup \{(m+1)n_1\}$. Here $(ij_1, ij_2, \ldots, ij_n)$ is a permutation of $(i1, i2, \ldots, in)$ and $s_{(m+1)n_1} \in F$. However, this implies that $\begin{matrix} A_{1k} \\ \vdots \\ A_{mk} \end{matrix} \in L(M)$, with $A_{ik} = a_{ik_1}a_{ik_2}\ldots a_{ik_n}$ where $(ik_1, ik_2, \ldots, ik_n)$ is a permutation of $(i1, i2, \ldots, in)$ for all $i = 1, 2, \ldots, m$. Hence, $rwcolperm(A) \subseteq L(M)$.

Now to prove that $rowperm(A) \subseteq L(M)$, we consider the row jump rules,

$$s_{1j_{n+1}}\# \to s_{2j_1}, s_{2j_{n+1}}\# \to s_{3j_1}, \ldots, s_{mj_{n+1}}\# \to s_{(m+1)j_1} \in R_3.$$

Here $(1j_{n+1}, 2j_{n+1}, \ldots, mj_{n+1})$ is a permutation of $(1, 2, \ldots, m)$ and $s_{(m+1)n_1} \in F$. However, this implies that $\begin{matrix} A_{k_1} \\ \vdots \\ A_{k_m} \end{matrix} \in L(M)$, where $(k_1, k_2, \ldots, k_m)$ is permutation of $(1, 2, \ldots, m)$, so $rowperm(A) \subseteq L(M)$. □

5. Comparison with Siromoney Matrix Languages

In this section we give a comparison of the family of languages accepted by 2-GRJFA with Siromoney matrix languages. Here we consider the derivation of a Siromoney matrix grammar, $G = (V_h, V_v, \Sigma_I, \Sigma, S, R_h, R_v)$ in a modified form. Here in the first phase, a vertical 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, horizontal generation of the actual picture is done parallely by applying R_v. So, we get the transpositions of original Siromoney Matrix Languages.

Theorem 5.1. *$\mathfrak{L}(RML^T)$ and $\mathfrak{L}$(2-GRJFA) are incomparable.*

Proof. To prove: $\mathfrak{L}$(2-GRJFA) $-$ $\mathfrak{L}(\mathrm{RML}^T) \neq \emptyset$

consider the language,

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

This language can be generated by a 2-GRJFA,

$M = (\{s, r, p\}, \{p\}, \{0,1\}, \{s0 \to r, r1 \to s\}, \{s\$ \to p, p\boxtimes \to p\}, \{p\# \to s\}, s, \{s\})$

But clearly $L(M) \notin \mathfrak{L}(\mathrm{RML}^T)$. Hence, $\mathfrak{L}$(2-GRJFA) $-$ $\mathfrak{L}(\mathrm{RML}^T) \neq \emptyset$

To prove: $\mathfrak{L}(\mathrm{RML}^T) - \mathfrak{L}$(2-GRJFA) $\neq \emptyset$

consider the language, $L = \left\{ \begin{matrix} X^{n+1} \\ ((.)^n X)_m \end{matrix} \middle| n, m \geq 1 \right\}$. $L \in \mathfrak{L}(\mathrm{RML}^T)$ because the language $\left\{ \begin{matrix} (X(.)^n)_m \\ X^{n+1} \end{matrix} \middle| n, m \geq 1 \right\}$ is an RML, as can be seen in [15]. However, this language cannot be generated by any 2-GRJFA. By theorem 4.3, if there is a 2-GRJFA, $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ such that $L(M) = L$, and, if $A = \begin{matrix} X & X & X \\ . & . & X \\ . & . & X \end{matrix}$ with $A \in L$, then M also accepts 3×3 arrays like,

$$\begin{matrix} . & X & . \\ X & X & X \\ X & . & . \end{matrix}, \quad \begin{matrix} X & X & X \\ . & X & . \\ . & X & . \end{matrix}, \quad \begin{matrix} . & X & . \\ X & . & . \\ X & X & X \end{matrix}, \ldots$$

But these arrays do not belong to the language L. Hence, there is no 2-GRJFA that accepts the language L.

Hence, $\mathfrak{L}(\mathrm{RML}^T) - \mathfrak{L}$(2-GRJFA) $\neq \emptyset$. □

Theorem 5.2. *$\mathfrak{L}(CFML^T)$ and $\mathfrak{L}$(2-GRJFA) are incomparable.*

Proof. To Prove: $\mathfrak{L}$(2-GRJFA) $-$ $\mathfrak{L}(\mathrm{CFML}^T) \neq \emptyset$ consider the language

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

This language can be generated by a 2-GRJFA, $M = (\{s, r, t, p\}, \{p\}, \{0,1,2\}, \{s0 \to r, r1 \to t, t2 \to s\}, \{s\$ \to p, s\boxtimes \to p\}, \{p\# \to s\}, s, \{s\})$.

But $L(M) \notin \mathfrak{L}(\mathrm{CFML}^T)$. Hence, $\mathfrak{L}$(2-GRJFA) $-$ $\mathfrak{L}(\mathrm{CFML}^T) \neq \emptyset$.

To prove: $\mathfrak{L}(\text{CFML}^T) - \mathfrak{L}(\text{2-GRJFA}) \neq \emptyset$

consider the language, $L = \left\{ \begin{matrix} (X(.)^n X)_m \\ X^{n+2} \\ (X(.)^n X)_m \end{matrix} \middle| n, m \geq 1 \right\}$. $L \in \mathfrak{L}(\text{CFML}^T)$ because the language $\left\{ \begin{matrix} X^{2n+1} \\ ((.)^n X(.)^n)_m \\ X^{2n+1} \end{matrix} \middle| n, m \geq 1 \right\}$ is a CFML, as can be seen in [15]. However, this language cannot be generated by any 2-GRJFA. By theorem 4.3, if there is a 2-GRJFA, $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ such that $L(M) = L$, and if $A = \begin{matrix} X & . & . & . & X \\ X & . & . & . & X \\ X & X & X & X & X \\ X & . & . & . & X \\ X & . & . & . & X \end{matrix}$ with $A \in L$, then M also accepts 5×5 arrays like, $\begin{matrix} . & X & . & . & X \\ X & . & . & X & . \\ . & X & X & . & . \\ X & . & . & . & X \\ X & X & X & X & X \end{matrix}$, $\begin{matrix} X & . & X & . & . \\ X & X & X & X & X \\ . & . & X & . & X \\ . & X & . & X & . \\ . & . & X & . & . \end{matrix}$, $\begin{matrix} X & X & X & X & X \\ . & X & . & . & . \\ . & X & X & . & . \\ X & . & . & X & . \\ . & X & . & X & . \end{matrix}$, ... But these arrays do not belong to the language L. Hence, there is no 2-GRJFA that accepts the language L.

Hence, $\mathfrak{L}(\text{CFML}^T) - \mathfrak{L}(\text{2-GRJFA}) \neq \emptyset$. □

Theorem 5.3. *$\mathfrak{L}(CSML^T)$ and $\mathfrak{L}$(2-GRJFA) are incomparable.*

Proof. The proof uses the same argument as the proofs of Theorems 5.1 and 5.2 and is, hence, omitted. □

6. Comparison with recognizable picture languages

In this section we give the comparison of the family of languages accepted by 2-GRJFA with recognizable picture languages.

Theorem 6.1. *REC and $\mathfrak{L}$(2-GRJFA) are incomparable.*

Proof. To prove $\mathfrak{L}(\text{2-GRJFA}) - REC \neq \emptyset$, consider the language,

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

This language can be generated by a 2-GRJFA, $M = (\{s, r, t, p\}, \{p\}, \{0, 1, 2\}, \{s0 \to r, r1 \to t, t2 \to s\}, \{s\$ \to p, s\boxtimes \to p\}, \{p\# \to s\}, s, \{s\})$.

Clearly $L(M) \notin REC$, we can refer to [4]. Hence, $\mathfrak{L}(\text{2-GRJFA}) - REC \neq \emptyset$.

To Prove $REC - \mathfrak{L}(\text{2-GRJFA}) \neq \emptyset$, let $\Sigma = \{a\}$ be a one-letter alphabet and let L be the language of squares over Σ, that is, $L = \{(a^n)_m | n = m\}$. This language is in REC, as can be seen in [4]. However, this language cannot be generated by 2-GRJFA. We prove this by contradiction. Assume that there is a 2-GRJFA, $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ such that $L(M) = L$. Let $A =$

$\begin{matrix} a & a & a \\ a & a & a \\ a & a & a \end{matrix}$. Since $A \in L$, when accepting the 3×3 array A, using the rules $\{sa \to s\} \in R_1, \{s\$ \to p, p\boxtimes \to p\} \in R_2, \{p\# \to s\} \in R_3$, M also accepts non-square arrays like $\begin{matrix} a & a & a \\ a & a & a \end{matrix}$, $\begin{matrix} a & a \\ a & a \\ a & a \end{matrix}$,... which do not belong to the language L. Hence, we arrive at a contradiction with the assumption that $L(M) = L$. Therefore, there is no 2-GRJFA that accepts the language L.

Hence, $REC - \mathfrak{L}(\text{2-GRJFA}) \neq \emptyset$. □

7. Closure properties

In this section we discuss some of the closure properties of the family of languages accepted by 2-GRJFA.

Theorem 7.1. *Both $\mathfrak{L}$(2-GRJFA) and $\mathfrak{L}$(2-RJFA) are closed under union.*

Proof. Let us consider 2-GRJFAs, $M_1 = (Q_1, Q_1', \Sigma_1, R_1, R_2, R_3, s_1, F_1)$ and $M_2 = (Q_2, Q_2', \Sigma_2, R_1', R_2', R_3', s_2, F_2)$. Let $L(M_1)$ and $L(M_2)$ be the language generated by M_1 and M_2, respectively. Without loss of generality, we assume that $Q_1 \cap Q_2 = \emptyset$ and $s \notin (Q_1 \cup Q_2)$.

Define the 2-GRJFA, $M = (Q_1 \cup Q_2 \cup \{s\}, Q_1' \cup Q_2', \Sigma_1 \cup \Sigma_2, R_1 \cup R_1' \cup \{s \to s_1, s \to s_2\}, R_2 \cup R_2', R_3 \cup R_3', s, F_1 \cup F_2)$

Clearly, we can see that, $L(M) = L(M_1) \cup L(M_2)$.

If both M_1 and M_2 are 2-RJFA, then M is also a 2-RJFA. □

Theorem 7.2. *Neither $\mathfrak{L}$(2-GRJFA) nor $\mathfrak{L}$(2-RJFA) is closed under row catenation and column catenation.*

Proof. Consider the language $L_1 = \{a\}^{**}$ and $L_2 = \{b\}^{**}$. Clearly L_1, L_2 can be generated by 2-RJFAs $M_1 = (\{s_1, p_1\}, \{p_1\}, \{a\}, \{s_1 a \to s_1\}, \{s_1\$ \to p_1, p_1\boxtimes \to p_1\}, \{p_1\# \to s_1\}, s_1, \{s_1\})$ and $M_2 = (\{s_2, p_2\}, \{p_2\}, \{b\}, \{s_2 b \to s_2\}, \{s_2\$ \to p_2, p_2\boxtimes \to p_2\}, \{p_2\# \to s_2\}, s_2, \{s_2\})$, respectively.

Let $L_C = L_1 \oplus L_2 = \{(a_m^i)(b_m^j) | i, j, m \geq 0\}$. This language cannot be generated by any 2-RJFAs or 2-GRJFAs since the language does not satisfy the condition $L_C = rwcolperm(L_C)$ given in theorem 4.3. Therefore, $L_C \notin \mathfrak{L}$(2-RJFA) and $L_C \notin \mathfrak{L}$(2-GRJFA).

Hence, $\mathfrak{L}$(2-RJFA) and $\mathfrak{L}$(2-GRJFA) are not closed under column catenation.

Let $L_R = L_1 \ominus L_2 = \left\{ \begin{matrix} (a_m^i) \\ (b_m^j) \end{matrix} \middle| i, j, m \geq 0 \right\}$. This language cannot be generated by any 2-RJFAs or 2-GRJFAs since the language does not satisfy the condition $L_R = rowperm(L_R)$ given in theorem 4.3. Therefore, $L_R \notin \mathfrak{L}$(2-RJFA) and $L_R \notin \mathfrak{L}$(2-GRJFA).

Hence, $\mathfrak{L}$(2-RJFA) and $\mathfrak{L}$(2-GRJFA) are not closed under row catenation. □

Theorem 7.3. *$\mathfrak{L}$(2-RJFA) is closed under column shuffle only if every row computation of any 2-RJFA begins with the same start symbol of that respective 2-RJFA.*

Proof. Let us consider two 2-RJFAs, $M_1 = (Q_1, Q_1', \Sigma_1, R_1, R_2, R_3, s_1, \{s_1\})$ and $M_2 = (Q_2, Q_2', \Sigma_2, R_1', R_2', R_3', s_2, \{s_2\})$ accepting the language L_1 and L_2 respectively. Let every row computation of M_1 begin with s_1 i.e., $q\# \to s_1 \in R_3, \forall\ q \in Q_1'$ and every row computation of M_2 begin with s_2 i.e., $p\# \to s_2 \in R_3', \forall\ p \in Q_2'$. Without loss of generality, we assume that $Q_1 \cap Q_2 = \emptyset$. Define the 2-RJFA, $M = (Q_1 \cup Q_2, Q_2', \Sigma_1 \cup \Sigma_2, R_1'', R_2'', R_3'', s_1, \{s_2\})$ where, $R_1'' = R_1 \cup R_1' \cup \{r \to s_2 | r\$ \to q \in R_2\}$, $R_2'' = R_2'$, $R_3'' = \{p\# \to s_1 | p\# \to s_2 \in R_3'\}$.

To prove that $L(M) = L_1 \sqcup\!\sqcup_c L_2$, we observe how M works. On an input array $A \in (\Sigma_1 \cup \Sigma_2)^{**}$, M first runs M_1 on any row from A. If it ends in r, then using the rule $r \to s_2$ in R_1'', M runs M_2 on the remaining elements of the same row from A. After replacing every element in the row with $\boxtimes$, M enters a check and jump state in Q_2' after reading \$. On reading #, M jumps to another row using the rule $p\# \to s_1 \in R_3''$. Then again, M first runs M_1 on that row followed by M_2. This procedure is repeated for all the remaining rows and, if M_2 ends in the final state after computing the last remaining row, then M accepts A. Otherwise, M rejects A. Since, after reading any symbol in a row of an array A, the machine M can jump anywhere in the same row and the same holds for all such rows of A, we clearly see that, $L(M) = L_1 \sqcup\!\sqcup_c L_2$. □

Theorem 7.4. *$\mathfrak{L}$(2-RJFA) is closed under row shuffle only if every row computation of any 2-RJFA begins with the same start symbol of that respective 2-RJFA.*

Proof. Let us consider two 2-RJFAs, $M_1 = (Q_1, Q_1', \Sigma_1, R_1, R_2, R_3, s_1, \{s_1\})$ and $M_2 = (Q_2, Q_2', \Sigma_2, R_1', R_2', R_3', s_2, \{s_2\})$ accepting the language L_1 and L_2, respectively. Let every row computation of M_1 begin with s_1 and every row computation of M_2 begin with s_2. Without loss of generality, we assume that $Q_1 \cap Q_2 = \emptyset$. Define the 2-RJFA, $M = (Q_1 \cup Q_2, Q_1' \cup Q_2', \Sigma_1 \cup \Sigma_2, R_1'', R_2'', R_3'', s_1\ \{s_2\})$ where, $R_1'' = R_1 \cup R_1'$, $R_2'' = R_2 \cup R_2'$, $R_3'' = \{p\# \to s_2 \mid p\# \to s_1 \in R_3\} \cup \{q\# \to s_1 \mid q\# \to s_2 \in R_3'\}$.

To prove that $L(M) = L_1 \sqcup\!\sqcup_r L_2$, we observe how M works. On an input array $A \in (\Sigma_1 \cup \Sigma_2)^{**}$, M first runs M_1 on any row from A and replaces all elements in that row with $\boxtimes$ followed by a check. Then, using the rule $p\# \to s_2 \in R_3''$, M jumps to another row and runs M_2 replacing all the elements in that row with $\boxtimes$ followed by a check. Then, using the rule $q\# \to s_1 \in R_3''$, M makes a row jump to some other row. Thus, M runs M_1 on some other row followed by M_2 on another row. This procedure is repeated for all the remaining rows and, if M_2 ends in the final state after computing the last remaining row, then M accepts A. Otherwise, M rejects A. Since, after reading any symbol in a row of an array A, the machine M can jump anywhere in the same row and the same holds for all such rows of A, we clearly see that, $L(M) = L_1 \sqcup\!\sqcup_r L_2$. □

Theorem 7.5. *$\mathfrak{L}$(2-RJFA) is not closed under both row and column Kleene star and Kleene plus.*

Proof. Consider the language, $L = \{a \quad b\ ,\ b \quad a\}$, which is accepted by the 2-RJFA $M = (\{s, r, f, p\}, \{p\}, \{a, b\}, \{sa \to r, rb \to f\}, \{f\$ \to p, p\boxtimes \to p\}, \{p\# \to f\}, s, \{f\})$.

We prove by contradiction that there is no 2-RJFA that accepts $L^{*\textcircled{1}}$ or $L^{+\textcircled{1}}$. Suppose that there is a 2-RJFA, $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, \{s\})$ such that

$L(M) = L^{*\textcircled{1}}$. Let $A = \begin{matrix} a & b \\ b & a \end{matrix}$. Since $A \in L^{*\textcircled{1}}$, by theorem 4.3, M also accepts arrays like $\begin{matrix} a & a \\ b & b \end{matrix}$, $\begin{matrix} b & b \\ a & a \end{matrix}$ which are not in $L^{*\textcircled{1}}$, as $L^{2\textcircled{1}} = \left\{\begin{matrix} a & b \\ b & a \end{matrix}, \begin{matrix} a & b \\ a & b \end{matrix}, \begin{matrix} b & a \\ b & a \end{matrix}, \begin{matrix} b & a \\ a & b \end{matrix}\right\}$. Therefore, we arrive at a contradiction.

Hence, $\mathfrak{L}$(2-RJFA) is not closed under column Kleene star and Kleene plus.

Consider the language, $L = \left\{\begin{matrix} a \\ b \end{matrix}, \begin{matrix} b \\ a \end{matrix}\right\}$, which is accepted by the 2-RJFA $M = (\{s, r, t, f, p, q\}, \{p, q\}, \{a, b\}, \{sa \to r, tb \to f\}, \{r\$ \to p, p\boxtimes \to p, f\$ \to q, q\boxtimes \to q\}, \{p\# \to t, q\# \to f\}, s, \{f\})$

We prove by contradiction that there is no 2-RJFA that accepts $L^{*\ominus}$ or $L^{+\ominus}$. Suppose that there is a 2-RJFA, $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, \{s\})$ such that $L(M) = L^{*\ominus}$. Let $A = \begin{matrix} a \\ b \\ b \\ a \end{matrix}$. Since $A \in L^{*\ominus}$, by theorem 4.3, M also accepts arrays $\begin{matrix} a & b \\ a & b \\ b & a \\ b & a \end{matrix}$, $\begin{matrix} a & b \\ b & a \end{matrix}$ which are not in $L^{*\ominus}$, as $L^{2\ominus} = \left\{\begin{matrix} a & a & b & b \\ b & b & a & a \\ b & a & b & a \\ a & b & a & b \end{matrix}\right\}$. Therefore, we arrive at a contradiction.

Hence, $\mathfrak{L}$(2-RJFA) is not closed under row Kleene star and Kleene plus. □

Theorem 7.6. *Neither $\mathfrak{L}$(2-GRJFA) nor $\mathfrak{L}$(2-RJFA) is closed under quarter turn and transpose.*

Proof. Consider the language,

$$L = \left\{ X \in \{0,1\}^{**} \middle| X = \begin{matrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{matrix}, \begin{matrix} \text{where each } X_i \text{ is of size } 1 \times n \text{ and} \\ |X_i|_0 = |X_i|_1, \forall \quad i = 1, 2, \ldots, k \end{matrix} \right\}, \text{ here}$$

$X_i = a_{i1}a_{i2}\ldots a_{in}$ with $a_{ij} \in \{0,1\} \forall\ j = 1, 2, \ldots, n$. This language can be generated by a 2-RJFA, $M = (\{s, r, p\}, \{p\}, \{0,1\}, \{s0 \to r, r1 \to s\}, \{s\$ \to p, p\boxtimes \to p\}, \{p\# \to s\}, s, \{s\})$. We represent the quarter turn of this language by L^{QT} and we have, $L^{QT} = \{X \in \{0,1\}^{**} | X = X_k X_{k-1} \ldots X_2 X_1$, where each X_i is of size $n \times 1$ and $|X_i|_0 = |X_i|_1, \forall\ i = 1, 2, \ldots, k\}$, here $X_i = \begin{matrix} a_{i1} \\ \vdots \\ a_{in} \end{matrix}$ with $a_{ij} \in \{0,1\}$, for all $j = 1, 2, \ldots, n$. This language cannot be generated by any 2-RJFAs or 2-GRJFAs because the language does not satisfy the conditions given in theorem 4.3. This proves that neither $\mathfrak{L}$(2-GRJFA) nor $\mathfrak{L}$(2-RJFA) is closed under quarter turn.

We represent the transpose of the language L by L^T and we have, $L^T = \{X \in \{0,1\}^{**} | X = X_1 X_2 \ldots X_k$, where each X_i is of size $n \times 1$ and $|X_i|_0 = |X_i|_1, \forall\, i = 1, 2, \ldots, k\}$, here $X_i = \begin{matrix} a_{i1} \\ \vdots \\ a_{in} \end{matrix}$ with $a_{ij} \in \{0,1\}$, for all $j = 1, 2, \ldots, n$. By the same

argument as before, we show that there is no 2-RJFA or 2-GRJFA to accept L^T. This proves that neither $\mathfrak{L}$(2-GRJFA) nor $\mathfrak{L}$(2-RJFA) is closed under transpose. □

Theorem 7.7. *Neither $\mathfrak{L}$(2-GRJFA) nor $\mathfrak{L}$(2-RJFA) is closed under half turn, reflection on rightmost vertical and reflection on base.*

Proof. Let $L \in \mathfrak{L}$(2-RJFA). Since $rwcolperm(A) \subseteq L$ and $rowperm(A) \subseteq L$ by theorem 4.3 for all $A \in L$, we clearly see that $\mathfrak{L}$(2-RJFA) is closed under half turn, reflection on rightmost vertical and reflection on base.

Let $L \in \mathfrak{L}$(2-GRJFA). Let $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ be a 2-GRJFA such that $L(M) = L$. We construct a new 2-GRJFA $M' = (Q, Q', \Sigma, R_1', R_2, R_3, s, F)$ where R_1' is defined as follows:

(1) $pa \to q \in R_1'$, if $pa \to q \in R_1$ where $p, q \in Q$ and $a \in \Sigma$.
(2) $pa_n a_{n-1} \dots a_2 a_1 \to q \in R_1'$ if $pa_1 a_2, \dots a_{n-1} a_n \to q \in R_1$ where $p, q \in Q$ and $a_1, a_2, \dots, a_n \in \Sigma$.

We clearly see that $L(M')$ is the same language as the reflection of L on the rightmost vertical. This shows that $\mathfrak{L}$(2-GRJFA) is closed under reflection on the rightmost vertical.

Using a construction similar to that above and by conditions given in theorem 4.3, we can easily prove that $\mathfrak{L}$(2-GRJFA) is closed under half turn and reflection on base. □

8. Decidability properties

In this section, we discuss some of the decidability properties of the family of languages accepted by 2-GRJFA.

Theorem 8.1. *Both finiteness and infiniteness are decidable for $\mathfrak{L}$(2-GRJFA).*

Proof. Let $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ be a 2-GRJFA. $L(M)$ is infinite if and only if $pY \rightsquigarrow p \in \Delta(M)$, where $p \in Q, Y \in \Sigma^{**}$ such that p is both reachable and terminating in $\Delta(M)$. This check can be done by any graph searching algorithm. Therefore, the theorem holds. □

Corollary 8.2. *Both finiteness and infiniteness are decidable for $\mathfrak{L}$(2-RJFA).*

Theorem 8.3. *The membership problem is decidable for $\mathfrak{L}$(2-GRJFA).*

Proof. Let $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ be a 2-GRJFA. If $A \in \Sigma^{**}$, we may assume M is $\wedge$-free. If $A = \wedge$, then $A \in L(M)$ if and only if $s \in F$. So we assume that $A \neq \wedge$ and A is any $m \times n$ array. Set,

$$\Gamma = \left\{ (A_1, A_2, \dots, A_m) \middle| A_i = a_{i1}a_{i2}\dots a_{in} \in \Sigma^{**}, 1 \le i \le m, \begin{matrix} A_1 \\ \vdots \\ A_m \end{matrix} = A, m \ge 1 \right\},$$

$$\Gamma_c = \left\{ (B_1, B_2, \dots, B_m) \middle| (A_1, A_2, \dots, A_m) \in \Gamma, m \ge 1, \begin{matrix} B_1 \\ \vdots \\ B_m \end{matrix} \in rwcolperm(A) \right\}$$

and

$$\Gamma_r = \left\{ (C_1, C_2, \ldots, C_m) \middle| (A_1, A_2, \ldots, A_m) \in \Gamma, m \geq 1, \begin{matrix} C_1 \\ \vdots \\ C_m \end{matrix} \in rowperm(A) \right\}.$$

Now, if there exists $(B_1, B_2, \ldots, B_m) \in \Gamma_c$ combined with $(C_1, C_2, \ldots, C_m) \in \Gamma_r$ and $q_{i1}, q_{i2}, \ldots, q_{in}, q_{i(n+1)}, q_{i(n+2)}, q_{(m+1)1} \in Q$, for $i = 1, 2, \ldots, m, 1 \leq n \leq |A|_c$ such that $s = q_{11}, q_{(m+1)1} \in F$ and $q_{ij}a_{ij} \to q_{i(j+1)} \in R_1$, for all $i = 1, 2, \ldots, m, j = 1, 2, \ldots, n$ and $q_{i(n+1)}\$ \to q_{i(n+2)}$, $q_{i(n+2)}\boxtimes \to q_{i(n+2)} \in R_2$, $q_{i(n+2)}\# \to q_{(i+1)1} \in R_3$, then $A \in L(M)$, otherwise $A \notin L(M)$. Since Q, Γ_c and Γ_r are finite, this check can be done in a finite time. □

Corollary 8.4. *The membership problem is decidable for $\mathfrak{L}$(2-RJFA).*

Theorem 8.5. *The emptiness problem is decidable for $\mathfrak{L}$(2-GRJFA).*

Proof. Let $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ be a 2-GRJFA. $L(M)$ is empty if and only if no $f \in F$ is reachable in $\Delta(M)$. This check can be done by any graph searching algorithm. Hence the theorem holds. □

Corollary 8.6. *The emptiness problem is decidable for $\mathfrak{L}$(2-RJFA).*

9. Conclusion

In this paper we have introduced two-dimensional jumping finite automata to accept rectangular arrays of languages and have also investigated some of their properties. We have discussed the closure properties of the family of languages accepted by these automata. We have compared the family of languages accepted by these automata with the family of Siromoney matrix languages and REC languages. We have also discussed some of their decidability properties (emptiness, membership, finiteness and infiniteness). We can exhibit more comparison results for the family of languages accepted by these new automata and study their applications and various other properties. By restricting the movement of the jumps, we can bring out many variants of two-dimensional jumping finite automata using only left jump, right jump, up row jump, down row jump, left and up row jump, left and down row jump, right and up row jump, right and down row jump, and can examine their acceptance power. We can also consider the impact on these automata by using various row start fixed configurations for beginning the computation in each row of the input array, that is, an automaton can start its computation from the beginning, end or anywhere in each row of the input array. By combining the restricted movement of jumps with the various row start configurations, we can also obtain much more variants of these automata and their power of acceptance can also be examined.

References

[1] A. Meduna and Peter Zemek, *Jumping finite automata*, Int. J. Found. Comput. Sci. **24** (2012), 1555–1578.

[2] A. Meduna, *Automata and Languages: Theory and Applications*, Springer, 2000.

[3] M. Blum and C. Hewitt, *Automata on a 2-dimensional tape*, in: SWAT 1967, 8th Annual Symposium on Switching and Automata Theory, 1967, 155–160.
[4] D. Giammarresi and A. Restivo, *Two-dimensional languages, Handbook of formal languages*, in: G. Rozenberg and A. Salomaa (eds.), Handbook of Formal Languages **3**, 1997, 215–267.
[5] D. Giammarresi and A. Restivo, *Recognizable picture languages*, Int. J. Pattern Recogn. **6** (1992), 241–256.
[6] K. Inoue and I. Takanami, *A survey of two-dimensional automata theory*, Inf. Sci. **55** (1991), 99–121.
[7] S. James Immanuel and D. Gnanaraj Thomas, *Extension of jumping finite automata to two dimensional languages*, in: Proceedings of IWCIA 2015, Methods and applications in Image Analysis, RPS Singapore, 2015, 1–16.
[8] K. Kamala and R. Siromoney, *Array automata and operations on array languages*, Int. J. Computer Math., Sect A **4** (1974), 3–30.
[9] V. Lonati and M. Pradella, *Picture recognizability with automata based on Wang tiles*, in: J. van Leeuwen *at al.* (eds), SOFSEM 2010: Theory and Practice of Computer Science, Lecture Notes in Computer Science **5901**, Springer, 2010, 576–587.
[10] M. Pradella, A. Cherubini and S. Crespi Reghizzi, *A unifying approach to picture grammars*, Inform. Comput. **209** (2011), 1246–1267.
[11] A. Rosenfeld, *Picture Languages: Formal Models for Picture Recognition*, Academic Press, 1979.
[12] A. Rosenfeld and R. Siromoney, *Picture languages–a survey*, Languages of Design **1** (1993), 229–245.
[13] G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages* **1–3**, Springer, 1997.
[14] A. Salomaa, *Computation and Automata*, Cambridge University Press, 1985.
[15] G. Siromoney, R. Siromoney and K. Krithivasan, *Abstract families of matrices and picture languages*, Comput. Graphics Image Process. **1** (1972), 284–307.
[16] R. Siromoney, *Advances in array languages*, in: Ehrig *et al.* (eds): Graph Grammars and Their Applications to Computer Science, Lecture Notes in Computer Science **291**, Springer, Berlin, 1987, 549–563.
[17] G. Siromoney, R. Siromoney and K. Krithivasan, *Picture languages with array rewriting rules*, Inform. and Control **22** (1973), 447–470.
[18] R. Siromoney, K. G. Subramanian and K. Rangarajan, *Parallel/sequential rectangular arrays with tables*, Int. J. Computer Math. **6** (1977), 143–158.
[19] P. S. Wang, *Array Grammars, Patterns and Recognizers*, Series in Computer Science **18**, World Scientific, 1989.
[20] P. S. Wang, *Sequential/parallel matrix array languages*, J. Cybern. **5** (1975), 19–36.
[21] P. S. Wang, *Hierarchical structures and complexities of isometric patterns*, IEEE Trans. Pattern Anal. Mach. Intell. **5** (1983), 92–99.

S. James Immanuel, Department of Mathematics, Madras Christian College, Tambaram, Chennai – 600 059, India
e-mail: james_imch@yahoo.co.in

D. G. Thomas, Department of Mathematics, Madras Christian College, Tambaram, Chennai – 600 059, India
e-mail: dgthomasmcc@yahoo.com