Two-Dimensional One-Way Jumping
Finite Automata

S James Immanuel\textsuperscript{1} and D G Thomas\textsuperscript{2}
\textsuperscript{1,2}Department of Mathematics, Madras Christian College,
Tambaram, Chennai – 600 059, India
\textsuperscript{1}james_imch@yahoo.co.in
\textsuperscript{2}dgthomasmcc@yahoo.com

Abstract
In this paper, we extend the recently introduced concept of one-way jumping finite automata to two-dimensional languages. We define left and right variant of the two-dimensional one-way jumping finite automata. We give the comparison of the family of languages accepted by these automata with the family of languages accepted by two-dimensional jumping finite automata, then with other families of picture languages and discuss their closure properties.

1 Introduction
In this paper, we extend the concept of one-way jumping finite automata introduced in [5] to two-dimensional languages or picture languages. One-way jumping finite automata is one of the variants of Jumping finite automata [1] recently introduced in the literature. Jumping finite automata were proposed as automata that process information in a discontinuous way. Jumping finite automata work like classical finite automata except that they can jump in either direction over a part of the input word after reading a symbol and continue processing from there.

The two-dimensional jumping finite automata introduced in [6], is an extension of Jumping finite automata for string languages. A
two-dimensional jumping finite automaton work on an input array row by row using the normal jump operation. After the automaton has completely replaced every element of some row in the array with a dummy symbol, it jumps to some other row of the array using a row jump and continue the computation from there. In [7], some new results and new variants of two-dimensional jumping finite automata is obtained by restricting the movement of the jumps, and by using various row start configurations.

In [5], one-way jumping finite automata is proposed, both right and left variant. Right one-way jumping finite automata move similar to jumping finite automata except for that the read head moves deterministically left to right starting from the left most letter in the input and when it moves to the end of the input word, then it returns to the beginning of the input word and continues the computation. Furthermore, if there are some symbols to read by the rule, then jumping finite automata reads the symbol which is the nearest to read head in the right direction of it. Left one-way jumping finite automata are defined similarly.

In this paper, we first define the two-dimensional one-way jumping finite automata, both left and right variant. We then give the comparison of the families of languages accepted by these automata with the family of languages accepted by two-dimensional jumping finite automata, families of Siromoney matrix languages and recognizable languages. We report the results that are obtained but omit certain proofs which will be published elsewhere.

2 Preliminaries

In this section we recall some notions related to formal language theory and array grammars (refer to [2]). Let $\Sigma$ be a finite alphabet, $\text{card}(\Sigma)$ denotes the cardinality of $\Sigma$. $\Sigma^*$ is the set of words over $\Sigma$ including the empty word $\lambda$. $\Sigma^+ = \Sigma^* - \{\lambda\}$. A rectangular $m \times n$ array $A$ is written in the form, $A = \begin{bmatrix} a_{i1} & \cdots & a_{in} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$ 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 $\Sigma$ is denoted by $\Sigma^{**}$ which also includes the empty array $\Lambda$ (zero rows and zero columns). $\Sigma^{++} = \Sigma^{**} - \{\Lambda\}$. For $a \in \Sigma, A \in \Sigma^{**}, \text{alph}(A)$ denotes the set of symbols occurring in $A$. 

International Journal of Pure and Applied Mathematics Special Issue 488
\[ |A|_a \] denotes the number of occurrences of \( a \) in \( A \). \( |A|_c \) is the number of columns in \( A \) and \( |A|r \) is the number of rows in \( A \). Let \( A = (a_{11} \ldots a_{1p} \ldots a_{m1} \ldots a_{mp}) \) and \( B = (b_{11} \ldots b_{1q} \ldots b_{n1} \ldots b_{nq}) \). Then \( A \ominus B = (a_{11} \ldots a_{1p} b_{11} \ldots b_{1q} \ldots a_{m1} \ldots a_{mp} b_{n1} \ldots b_{nq}) \) is the column catenation defined only when \( m = n \). Similarly the row catenation, \( A \oplus B = (a_{11} \ldots a_{1p} \ldots a_{m1} \ldots a_{mp} \ b_{11} \ldots b_{1q} \ldots b_{n1} \ldots b_{nq}) \) is defined only when \( p = q \).

### 3 2-D One-Way Jumping Finite Automata

In this section, we define two variants of two-dimensional jumping finite automata called as two-dimensional left and right one-way jumping finite automata respectively. Every input array \( A \in \Sigma^* \) of size \((m, n)\) given to a two-dimensional right one-way jumping finite automaton, is considered as an array \( \hat{A} \) of size \((m, n+2)\) obtained by adjoining a special symbol \( \$ \in \Sigma \) at the left end and \# \in \Sigma \) at the right end of each row.

**Definition 1.** A two-dimensional right one-way jumping finite automaton, 2-ROWJFA for short, is defined as 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.
- \( \Sigma \) is an input alphabet.
- \( R_1 \subseteq (Q \times \Sigma^* \times Q) \)
- \( R_2 \subseteq (Q \times \Sigma^* \times Q') \cup \{r \times \# \times r | r \in Q'\}, \$ \in \Sigma \),
- \( R_3 \subseteq Q' \times \# \times Q, \# \notin \Sigma \),
- \( s \in Q \) is the initial state
- \( F \subseteq Q \) is the set of final states

Members 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 \( py \rightarrow q \in R \). The rules in \( R_1 \) perform right only jump operation, denoted by \( \bigtriangledown \), the rules in \( R_2 \) perform normal transitions from left to right, denoted by \( \rightarrow \) and the rules in \( R_3 \) perform row jumping operation, denoted by \( \bowtie \).

The right only jump operator \( \bigtriangledown \) is defined as follows: Let \( xyz \) with \( x, y, z \in \Sigma^* \) be any row of an input array \( A \). Let \( py \rightarrow q \in R_1 \)
with \( y \in \Sigma^* \), \(|y|_c = k\), \( y' = \Xi^k \), \( x \in (\Sigma \backslash \Sigma_p) \cup \{\Xi\})^*\) where \( \Sigma_p = \{b \in \Sigma | (p, b, q) \in R_1\ \text{for some } q \in Q\}, \ z \in (\Sigma \cup \{\Xi\})^* \) where \(|\Xi|_c = 1\). \( M \) makes a right jump from \( pxyz \) to \( qy'zx \) using the rule \( py \rightarrow q \in R_1 \), written as \( pxyz \mathcal{C} qy'zx \) where \( y \) is replaced by \( \Xi^{|w|} \). Here \( \Xi \) helps to maintain the length of the row, so that the initial rectangular form of the array \( A \) is maintained.

If there is some \( y \) for which no rule in \( R_1 \) can be applied, \( M \) halts without accepting the input array \( A \). The right only jump operator is continually applied until all the symbols in the current row are completely read and replaced with \( \Xi \)'s. When there is no \( y \) implying that \(|xyz|_c = |xyz|_\Xi\), then \( M \) is in state \( q \) reading \( \$ \) at the left end of the row. Now a rule \( qs \rightarrow r \in R_2 \) is applied which allows \( M \) to enter check and row jump state \( r \in Q' \). Now a rule \( r\Xi \rightarrow r \in R_2 \) is continually applied to check whether every symbol in the current row has been read and replaced with \( \Xi \). After the check, \( M \) reads the symbol \( \# \) at the right end of the row. Now \( M \) performs another jump called as the row jump, denoted by \( \mathcal{C} \).

The row jump operator \( \mathcal{C} \) is defined as follows: Let \( X, Z, X', Z' \in \Sigma^* \), \( u' \in \Sigma^* \), \( u \in \{\Xi\}^* \), \( r\# \rightarrow p \in R \), for some \( p \in Q \); \( M \) makes a row jump from \( \begin{array}{c|c}
 X \hline
 Z
\end{array} \) to \( \begin{array}{c|c}
 X' \hline
 Z'
\end{array} \) denoted by, \( \begin{array}{c|c}
 X \hline
 Z \end{array} \mathcal{C} \begin{array}{c|c}
 X' \hline
 Z'
\end{array} \) either up or down with \( X \ominus Z = X' \ominus u' \ominus Z' \), \(|X|_c = |Z|_c = |X'|_c = |Z'|_c = |u'|_c = |Z|_r + |Z'|_r + 1 \), \( u \) is any row of either \( X \) or \( Z \). The row \( u \in \Xi^{\Sigma|c|} \) is removed and \( Z \) shifts one row up after the row jump is performed.

A successful computation by a 2-ROWJFA \( M \) is as follows:

The read head of \( M \) starts at the leftmost symbol (after \#) of any row from the given input array, \( A \). Also each and every row computation begins on the leftmost symbol of the row. After reading a symbol \( M \) replaces the read symbol with \( \Xi \) and moves rightward. On the working row, \( M \) can jump over a part of the row after reading and replacing a symbol with \( \Xi \). If the read head of \( M \) reaches the right end of the current row, then it continues again from the left end. If every symbol in the current row has been read and replaced with \( \Xi \), \( M \) jumps to \( \$ \) and enters a check and row jump state. Now \( M \) checks whether every symbol has been read and replaced with \( \Xi \) using normal transition rules and on reaching \# \( M \) performs a row jump, after deleting the current row, to some
other row of the input and continues its computation. The right only jump relation $\rightsquigarrow$ 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 $\tilde{\varepsilon}$s and $M$ enters the final state. In this case $A$ is accepted.

Combining the right only jump operator $\rightsquigarrow$, the normal transition $\rightarrow$ and the row jumping operator $\triangleright$ we define complete right row jump operator $\circlearrowright$ as follows:

**Definition 2.** Let $M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)$ be a 2-ROWJFA. Let $U \in \Sigma^\ast$, with $U = ::$ and each $u_i = u_{i1}u_{i2} \ldots u_{im}$ where $u_{if} \in \Sigma$, $\forall i = 1, 2, \ldots, m$ and $f = 1, 2, \ldots, n$. Let us consider some $u_j = u_{j1}u_{j2} \ldots u_{jn}$ and $u_k = u_{k1}u_{k2} \ldots u_{kn}$ with $j \neq k$ and $j, k = 1, 2, \ldots, m$ along with some rules $s_ju_{j1} \rightarrow s_{j2} \in R_1$, $s_j(n+1)\varepsilon \rightarrow s_j(n+2)\varepsilon \rightarrow s_j(n+2)\tilde{\varepsilon} \rightarrow s_k1 \in R_3$ where $s_j1, s_j(n+1), s_k1 \in Q, s_j(n+2) \in Q'$. We define $\circlearrowright$ as,

$s_j1u_{j1}u_{j2} \ldots u_{jn}\tilde{\varepsilon} \circlearrowright s_k1u_{k1}u_{k2} \ldots u_{kn}$.

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

$L(M) = \left\{ u \in \Sigma^\ast \mid u_1^u_2 \ldots u_m^u_m \circlearrowright^* f, f \in F, k = 1, 2, \ldots, m \right\}$.

Family of languages accepted by 2-ROWJFA is denoted by 2-ROWJ.
by 2-LOWJFA by 2-LOWJ.

Example 1. Let us consider an example for a 2-ROWJFA.

\[ M = (\{s, p, q\}, \{q\}, \{X, \bullet\}, \{sX \rightarrow p, p\bullet \rightarrow p\}, \{p\$ \rightarrow q, q\$ \rightarrow q\}, \{q\# \rightarrow s\}, s, \{s\}) \]

\[ L(M) = \{(X(\bullet)^m)_n|m, n \geq 1\} \]

4 Some Basic Properties

In this section we give some basic properties of two-dimensional right and left one-way jumping finite automata.

Theorem 1. 2-ROWJ is incomparable with 2-LOWJ but not disjoint.

Proof. 2-ROWJ − 2-LOWJ ≠ ∅ follows from example 1, as \(L(M)\) cannot be accepted by any 2-LOWJFA. Consider a 2-LOWJFA, \(M' = (\{s, p, q\}, \{q\}, \{X, \bullet\}, \{p \leftarrow sX, p \leftarrow p\bullet\}, \{q \leftarrow p\$, q \leftarrow q\$\}, \{s \leftarrow q\#\}, s, \{s\})\) accepting \(L(M') = \{((\bullet)^mX)_n|m, n \geq 1\}\). Clearly this language cannot be accepted by any 2-ROWJFA. Hence, 2-LOWJ − 2-ROWJ ≠ ∅.

Now to prove 2-ROWJ ∩ 2-LOWJ ≠ ∅, consider the language \(L = \{((\bullet)^m)_n|m, n \geq 1\}\). Clearly this language can be accepted by some 2-ROWJFA as well as 2-LOWJFA. Hence the proof.

Theorem 2. Let \(M = (Q, Q', \Sigma, R_1, R_2, R_3, s, F)\) be a 2-ROWJFA or a 2-LOWJFA such that \(\text{card}(\Sigma) = 1\). Then \(L(M)\) is a RML.

Theorem 3. 2-ROWJ and 2-LOWJ are incomparable with \(\mathcal{L}(2-RJFA)\) but not disjoint.

Proof. The languages \(L_1 = \{(X(\bullet)^m)_n|m, n \geq 1\}\) and \(L_2 = \{((\bullet)^mX)_n|m, n \geq 1\}\) cannot be accepted by any 2-RJFA. Also, the language \(L = \{((\bullet)^m)_n|m, n \geq 1\} \in \mathcal{L}(2-RJFA)\).

5 Comparison With Other Families

In this section we give the comparison of the family of languages accepted by 2-ROWJFA and 2-LOWJFA with families of Siromoney
matrix languages [8] and recognizable languages [3, 4].

**Theorem 4.** \( \mathcal{L}(RML_T), \mathcal{L}(CFML_T) \) and \( \mathcal{L}(CSML_T) \) are incomparable with both 2-ROWJ and 2-LOWJ but not disjoint.

**Theorem 5.** REC is incomparable with both 2-ROWJ and 2-LOWJ but not disjoint.

### 6 Closure Properties

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

**Theorem 6.** Both 2-ROWJ and 2-LOWJ are closed under union.

**Proof.** Let us consider \( M_1 = (Q_1, Q'_1, \Sigma_1, R_1, R_2, R_3, s_1, F_1) \) and \( M_2 = (Q_2, Q'_2, \Sigma_2, R'_1, R'_2, R'_3, s_2, F_2) \) to be 2-ROWJs. Let \( L(M_1) \) and \( L(M_2) \) be the language generated by \( M_1 \) and \( M_2 \) respectively. Without any loss of generality, we assume that \( Q_1 \cap Q_2 = \emptyset \) and \( s \notin (Q_1 \cup Q_2) \).

Define the 2-ROWJ, \( M = (Q_1 \cup Q_2 \cup \{s\}, Q'_1 \cup Q'_2, \Sigma_1 \cup \Sigma_2, R_1 \cup R'_1 \cup \{s \rightarrow s_1, s \rightarrow s_2\}, R_2 \cup R'_2, R_3 \cup R'_3, s, F_1 \cup F_2) \)

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

**Theorem 7.** Both 2-ROWJ and 2-LOWJ are not closed under row catenation, column catenation, row kleene star, column kleene star, quarter turn, half turn, transpose and reflection on rightmost vertical.

**Theorem 8.** Both 2-ROWJ and 2-LOWJ are closed under reflection on base.

### 7 Conclusion

In this paper, we have extend the concept of one-way jumping finite automata to two-dimensional languages. We have studied two variants of these automata and have given some comparison results and closure properties. We also find as in the string language case, the languages accepted by two-dimensional right one-way jumping
finite automata is different from the languages accepted by two-dimensional left one-way jumping finite automata. We can also consider four more interesting variants of these automata by considering right only jump along with up only row jump and down only row jump respectively and similarly with left only jump.

References


