A Membrane Computing Model for Generation of Picture Arrays

Pradeep Isawasan[1], Ibrahim Venkat[1], Ravie Chandren Muniyandi[2], and K.G. Subramanian[3](✉)

[1] School of Computer Sciences, Universiti Sains Malaysia, 11800 Gelugor, Penang, Malaysia
[2] Faculty of Information Science and Technology, School of Computer Science, Universiti Kebangsaan Malaysia, 43600 Bangi, Malaysia
[3] Department of Mathematics and Computer Science, Faculty of Science, Liverpool Hope University, Hope Park, Liverpool L16 9JD, UK
kgsmani1948@gmail.com

Abstract. In the bio-inspired area of membrane computing, P system is a versatile model providing a rich framework for many computational problems. Array P system and its variant with parallel rewriting facilitate the study of picture languages within this area of membrane computing. Here another variant of array P system, called tabled parallel array P system (TPAP), is introduced, by endowing it with the features of parallel rewriting and tables of array-rewriting rules. The generative power of TPAP as well as the ability of this system in describing picture patterns are investigated.

Keywords: Bio-inspired computing · Membrane computing · P system · Picture language

1 Introduction

Membrane computing (MC) is an emerging area of natural computing, initiated by Paun [6] around the year 2000. The novel computing model proposed in MC, called membrane system (subsequently referred to as P system in honour of its originator) was inspired from the structure and functioning of living cells. The P system has proved to be a suitable framework for solving many computational problems in different fields of research and investigation [3,7]. Several research directions have emerged applying techniques of membrane computing [6,7]. One such study is on problems related to digital images falling under the broad area of computer vision [3].

On the other hand, motivated by problems arising in image processing and pattern recognition, a variety of two-dimensional (2D) array grammars, as generalizations of formal string grammars [9], have been introduced and investigated [2,4,8,15]. These 2D grammars generating picture languages consisting of digitized images or picture arrays, have also turned out to be potential tools for

H. Badioze Zaman et al. (Eds.): IVIC 2015, LNCS 9429, pp. 155–165, 2015.
DOI: 10.1007/978-3-319-25939-0_14

dealing with application problems [12,15]. The two areas of picture grammars and P systems have been linked in [1], thus providing enriched techniques for dealing with application problems in the broad area of computer vision (see, for Example [3] p. 617).

A variant of array P system [1], known as parallel array P system was recently introduced in [14] and a further improvement in this system was made in [5], in terms of reduction in the number of membranes used in generating certain picture languages. In formal language theory, one of the main studies is on the language generating capability of the grammars, referred to as the generative capacity, which depends on the types of rules. Also a standard technique to increase the generative capacity is to endow the rules with additional features. In this paper, the parallel array-rewriting P system is investigated by incorporating in the regions of the P system, the feature of having tables of rules, well-known in formal language theory, especially in Lindenmayer systems [10] and examine the generative power. We also provide an application to generation of picture patterns.

2 Preliminaries

We recall needed notions and results on array grammars [2,4] and array P systems [1]. We refer to [9] for concepts related to formal language theory.

Let V be a finite alphabet. In the two-dimensional plane $\mathbf{Z}^2$, a non-empty finite array $\mathcal{A}$ over V, also called a picture array, is made of a finite number of unit squares (also called cells or pixels) in the plane, with each square of $\mathcal{A}$ being labelled by a symbol of V. An empty square in the plane is indicated by labelling it with the *blank symbol* $\# \notin V$. The collection of all non-empty, connected finite arrays over V is denoted by V^{++}. An array language is a subset of V^{++}. More precisely, an array is a mapping $\mathcal{A} : \mathbf{Z}^2 \to V \cup \{\#\}$ with a finite support, given by $supp(\mathcal{A}) = \{v \in \mathbf{Z}^2 \mid \mathcal{A}(v) \neq \#\}$. We can specify an array by listing the *pixels* v of the support, along with the symbols in the respective pixels. For example, Fig. 1 shows a picture representing the English alphabetic letter Y that has its cells labeled by a. If we assume that the cell having label a in the bottommost pixel of the vertical arm of the letter Y has coordinates $(0, 0)$, then the array in Fig. 1 is given by listing the $(cordinate, label)$ pairs of all the cells belonging to the picture array as follows: $Y = \{((0,0),a),((0,1),a),((0,2),a),((0,3),a),((-1,4),a),((-2,5),a)\} \cup \{((-3,6),a),((1,4),a),((2,5),a),((3,6),a)\}$. Since only the relative positions of the symbols in a picture are needed for describing a picture, we can use a pictorial method to denote a picture array indicating only the non-blank labels of the cells, without mentioning their coordinates. For example, the array in Fig. 1 is shown in this manner, where the symbols a constitute the body of the picture representing the letter Y. An array production or array rule r over V, written as $\mathcal{A} \to \mathcal{B}$ is a triple $r = (W, \mathcal{A}, \mathcal{B})$, where W is a finite subset of $\mathbf{Z}^2$ and $\mathcal{A}, \mathcal{B}$ are arrays with their supports included in W. For two arrays $\mathcal{C}, \mathcal{D}$ over V and a production r as above, we write $\mathcal{C} \Rightarrow_r \mathcal{D}$ if $\mathcal{D}$ can be obtained by replacing by

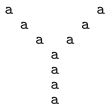

Fig. 1. A picture array representing the letter Y

$\mathcal{B}$, a subarray of $\mathcal{C}$ identical to $\mathcal{A}$, in the sense that the subarray of $\mathcal{C}$ is geometrically identical to $\mathcal{A}$ and the corresponding pixels in the subarray and $\mathcal{A}$ have the same label. The reflexive and transitive closure of the relation $\Rightarrow$ is denoted by $\Rightarrow^*$.

An array production $r = (W, \mathcal{A}, \mathcal{B})$ is called (i) context-free, if $supp(\mathcal{A}) \subseteq supp(\mathcal{B})$ and $card(supp(\mathcal{A})) = 1$, where $card(Z)$ is the number of labelled cells in the array Z and (ii) regular if it is in any one of the following forms: $A\ \# \rightarrow a\ B$, $\#\ A \rightarrow B\ a$, $\begin{matrix}\#\\A\end{matrix} \rightarrow \begin{matrix}B\\a\end{matrix}$, $\begin{matrix}A\\\#\end{matrix} \rightarrow \begin{matrix}a\\B\end{matrix}$, $A \rightarrow B$, $A \rightarrow a$, where A, B are nonterminals and a is a terminal.

An array grammar is a construct $G = (N, T, \#, \{((0,0), S)\}, P)$, where N, T are disjoint sets symbols, respectively called nonterminal symbols and terminal symbols, $\# \notin N \cup T$ is the blank symbol, $S \in N$ and P is a finite set of array rewriting rules $\mathcal{A} \rightarrow \mathcal{B}$ such that at least one pixel of $\mathcal{A}$ is marked with an element of N; usually, the *axiom array* $\{((0,0), S)\}$ will be simply written as S.

An array grammar is context-free or regular if all its rules are context-free (CF) or regular respectively. There is a unique non-blank pixel marked with a nonterminal in the left hand array of each context-free or regular rule. The array language generated by G is $L(G) = \{\mathcal{A} \in T^{++} \mid \{((0,0), S)\} \Rightarrow^* \mathcal{A}\}$. The families of array languages generated by context-free, and regular array grammars are denoted by ACF and $AREG$ respectively. The following strict inclusion is known [1]: $AREG \subset ACF$.

Example 1. We give an illustration of derivation in an array grammar with rules

$$p_1 : \begin{matrix}\#\\S\\\#\end{matrix} \rightarrow \begin{matrix}A\ B\\a\\C\end{matrix},\ p_2 : \#_A \rightarrow {}^A{}_a,\ p_3 : {}_B\# \rightarrow {}_a{}^B,\ p_4 : \begin{matrix}C\\\#\end{matrix} \rightarrow \begin{matrix}a\\C\end{matrix},$$

$p_5 : A \rightarrow a$, $p_6 : B \rightarrow a$, $p_7 : C \rightarrow a$ where S, A, B, C are nonterminals and a is a terminal symbol. A sample derivation starting from the start symbol S with the rules applied in the sequence $p_1, p_2, p_5, p_3, p_6, p_7$, is given below:

$$S \Rightarrow \begin{matrix}A\ B\\a\\C\end{matrix} \Rightarrow \begin{matrix}A\\a\ B\\a\\C\end{matrix} \Rightarrow \begin{matrix}a\\a\ B\\a\\C\end{matrix} \Rightarrow \begin{matrix}a\\a\ a\ B\\a\\C\end{matrix} \Rightarrow \begin{matrix}a\\a\ a\ a\\a\\C\end{matrix} \Rightarrow \begin{matrix}a\\a\ a\ a\\a\\a\end{matrix}.$$

We note that the rewriting is sequential with only one rule applied in a single step of derivation and hence the picture array Y generated need not have all three arms equal in length, where the length of an arm is the number of symbols a along an arm, counting from the "centre" symbol a.

In [14], a variant of the array P system of [1], called parallel array P system (PAP) was introduced by incorporating the feature of parallel rewriting of arrays in the regions. We now recall the parallel array P system.

Definition 1 *[14]. A parallel array P system* (PAP) *is a construct* $\Pi = (V, T, \#, \mu, F_1, \ldots, F_m, R_1, \ldots, R_m, i_o)$, *where the components are defined as follows:* V *is the alphabet of nonterminals and terminals,* $T \subseteq V$ *is the terminal alphabet,* $\# \notin V$ *is the blank symbol,* μ *is a membrane structure with* m *membranes labelled in a one-to-one way with* $1, 2, \ldots, m$; $F_1, \ldots, F_m$ *are finite sets of arrays over* V *associated with the* m *regions of* μ; $R_1, \ldots, R_m$ *are finite sets of array rewriting rules over* V *associated with the* m *regions of* μ; *the array-rewriting rules (context-free or regular) of the form* $\mathcal{A} \to \mathcal{B}(tar)$ *have attached targets "here", "out" or "in" (The meaning of here is that the array remains in the same region, out means that the array exits the current membrane, and in means that the array is immediately sent to one of the directly lower membranes, nondeterministically chosen if several exist; if no internal membrane exists, then a rule with the target indication in cannot be used)(in general, we omit mentioning "here"); finally,* i_o *is the label of an elementary membrane of* μ *which is the output membrane.*

The application of context-free rules in processing an array in a region is done in a parallel manner as described below: We require that context-free array rules are applied to all the nonterminals in the array being processed in a region. In other words for every nonterminal A in an array being processed in a region, if there is a context-free array rule that can rewrite a subarray containing this nonterminal A with other pixels, if any, in this subarray having the blank symbol #, then a set of such rules is used to rewrite all such nonterminals in the array. Since every rule in a region has one of the target indications "here", "in", "out", we require that all the context-free array rules applied to an array in a region should have the same target indication. If in a region, no set of rules having the same target indication is available for rewriting all the nonterminals in an array in that region, then the array is not processed and remains in the same region. Also, if two context-free array rules $\mathcal{A} \to \mathcal{B}, \mathcal{C} \to \mathcal{D}$ *when applied to an array overlap in their application in the sense that arrays* $\mathcal{C}$ *and* $\mathcal{D}$ *have to use some common pixels for successfully applying the rules, then the array is not rewritten. In other words, we consider only the overlap-free case. The families of all array languages generated by parallel array P systems as above, with at most m membranes, with CF and regular array rules are respectively denoted by* $PAP_m(CF)$ *and* $PAP_m(REG)$.

Remark 1. (i) The situation of *deadlock* that might arise when a set of context-free rules with different target indications are applied to an array, is avoided in a parallel array P system by requiring that all the rules applied in parallel to an array have the same target indication.

(ii) The feature of rewriting in a derivation step all nonterminals in an array in parallel in a PAP is on the lines of the standard technique of parallel rewriting in Lindenmayer systems in the string case [10]. Recently, in [5], besides this kind of parallelism, the feature of maximal parallelism in the style of

membrane computing is also considered and results in [14] are improved in terms of lesser number of membranes for the constructions involved.

We now illustrate derivation in a parallel array P system having the same rules as given in Example 1.

Example 2. Consider the following parallel array P system with array context-free rules: $\Pi_1 = (\{A, B, C, a\}, \{a\}, \#, [_1\, [_2\,]_2\,]_1, \{ \begin{smallmatrix} A & & B \\ & a & \\ & C & \end{smallmatrix} \}, \emptyset, R_1, R_2, 2)$, with R_1 containing the rules $p_1, \cdots, p_7$ as in Example 1 but the rules p_5, p_6, p_7 having the target indication *in* and R_2 is empty.

Starting with the axiom array $\begin{smallmatrix} A & & B \\ & a & \\ & C & \end{smallmatrix}$ in region 1, all three arms are grown together, one symbol at a time, by applying in parallel the rules p_1, p_2 and p_3 as many times as needed. Note that these rules have the same target indication (*here*, which is understood if it is not mentioned). When the rules p_5, p_6 and p_7 (having the same target indication *in*) are used, the derivation halts and the array in the shape of Y with equal arms enters region 2, where it is collected in the language generated. Note that there are no rules in region 2 and hence no array in region 2 can evolve further.

3 Tabled Parallel Array P System

We now introduce a variant of PAP, called Tabled parallel array P system ($TPAP$), by employing a well-known technique, called tables of rules, of grouping rules, especially used in Lindenmayer systems [10]. This enables a specific collection of rules being used at a time and enhances the generative power. This technique has been adopted in array generating systems as well (see, for example, [13]).

Definition 2. *A tabled parallel array P system* ($TPAP$) *is a construct*

$$\Pi = (V, T, \#, \mu, F_1, \ldots, F_m, R_1, \ldots, R_m, i_0),$$

where the components are as in PAP except that the finite sets of array rewriting rules $R_1, \ldots, R_m$ contain tables of array-rewriting rules (context-free or regular) of the form $t = \{\mathcal{A}_i \to \mathcal{B}_i \mid 1 \leq i \leq m,\ m \geq 1\}\,(tar)$ ($\mathcal{A}_i$ and $\mathcal{B}_i$ are arrays) with an attached target indicated by tar, which can be "here", "out", "in" (in general, we omit mentioning "here"), with the usual interpretation. The application of a table of rules in processing an array in a region is done in a parallel manner in the sense that all the nonterminals in the array being processed in a region, are rewritten by the rules in a table. Also we consider as in the case of PAP only the overlap-free case. The families of all array languages generated by tabled parallel array P systems as above, with at most m membranes, with CF and regular array rules are respectively denoted by $TPAP_m(CF)$ and $TPAP_m(REG)$.

Example 3. Consider the following tabled parallel array P system with array context-free rules: $\Pi_2 = (\{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{X}, \mathtt{W}, \mathtt{Y}, \mathtt{Z}, a\}, \{a, b\}, \#, [_1\, [_2\,]_2\,]_1,\ \{ \begin{smallmatrix} A & D & C \\ a & a & B \\ A & D & C \end{smallmatrix} \}, \emptyset,$ $R_1 = \{t_1, t_2(in)\}, R_2 = \emptyset, 2)$, with

$$t_1 = \{\, \mathtt{A} \to \mathtt{A}\,,\ \mathtt{D} \to \mathtt{D}\,,\ \mathtt{C}\,\# \to \mathtt{D}\,\mathtt{C}\,,\ \mathtt{B}\,\# \to \mathtt{a}\,\mathtt{B}\,\}\,, t_2 = \{\, \mathtt{A} \to \mathtt{X}\,,\ \mathtt{B} \to a\,,\ \mathtt{C} \to \mathtt{Y}\,,\ \mathtt{D} \to \mathtt{Z}\,,\}$$

$t_3 = \left\{ \begin{matrix}\# \\ X\end{matrix} \to \begin{matrix}X \\ a\end{matrix}, \begin{matrix}X \\ \#\end{matrix} \to \begin{matrix}a \\ X\end{matrix}, \begin{matrix}\# \\ Z\end{matrix} \to \begin{matrix}Z \\ b\end{matrix}, \begin{matrix}Z \\ \#\end{matrix} \to \begin{matrix}b \\ Z\end{matrix}, \begin{matrix}Y \\ \#\end{matrix} \to \begin{matrix}a \\ Y\end{matrix}, \begin{matrix}\# \\ Y\end{matrix} \to \begin{matrix}Y \\ a\end{matrix}, \right\}, t_4 = \{ X \to a, Y \to a, Z \to b \}.$

Starting with the axiom array $\begin{matrix}A\ D\ C \\ a\ a\ B \\ A\ D\ C\end{matrix}$ in region 1, if rules in table t_1 are applied in parallel, then a column is added to the array yielding $\begin{matrix}A\ D\ D\ C \\ a\ a\ a\ B \\ A\ D\ D\ C\end{matrix}$ and the process can be repeated till t_2 is applied which changes nonterminals A, C, D in X, Y, Z and the array is sent to region 2. Here if the rules of table t_3 are applied, then a row above and a row below with reference to the middle row are added and the process can be repeated any number of times, thus yielding an array of the form shown in Fig. 2*a*. An application of the rules of the table t_4 changes all the nonterminals into corresponding terminals yielding arrays over $\{a, b\}$ of the form shown in Fig. 2*b*. These arrays are collected in the picture language generated by the $TPAP\,\Pi_2$. We note that if the symbol b is interpreted as blank, then these arrays represent the letter H with equal length vertical arms made of the symbol a, above and below the middle horizontal row also made of the symbol a, one member of which is shown in Fig. 2*c*.

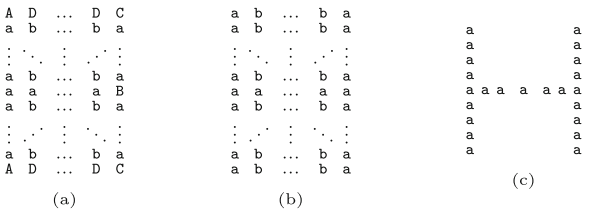

Fig. 2. (a): An array generated in an intermediate step of derivation in Example 3 (b): An array generated in a completed derivation in Example 3 (c) An array representation of the letter H

The array productions we have so far considered are known as the isometric variety in the sense that the arrays in the left and right sides of the rule are geometrically identical in shape. In contrast to this, in the non-isometric variety, the rules that rewrite or generate arrays are analogous to the string grammar rules in the sense that application of a rule $u \to v$, where u, v are either strings or arrays, would mean that enough 'space' is created by 'pushing' symbols, if needed, for v to replace u. There are many array grammar models of the non-isometric variety generating $m \times n\,(m, n \geq 1)$, rectangular arrays of symbols with m number of rows and n number of columns and among these we consider here the two-dimensional right-linear grammar with tables of rules [13], which we call here as a tabled two-dimensional right-linear grammar, consistent with the terminology used in [4].

Definition 3. *A tabled two-dimensional right-linear grammar (2TRLG) [13] is* $G = (V_h, V_v, V_i, T, S, R_h, R_v)$ *where* V_h, V_v, V_i *are horizontal, vertical and intermediate finite sets of nonterminals;* $V_i \subset V_v$; T *is a finite set of terminals;* $S \in V_h$ *is the start symbol;* R_h *is a finite set of horizontal rules of the form* $X \to AY, X \to A\ X, Y \in V_v, A \in V_i$; R_v *is a finite set of tables of vertical rules with a table consisting of either only rules of the form* $X \to aY$ *or only rules of the form* $X \to a$, $X, Y \in V_i, a \in T$.

There are two phases of derivation in a 2TRLG. In the first phase, starting with S the horizontal rules are applied (as in a regular grammar) generating strings over intermediates. In the second phase each intermediate in such a string serves as the start symbol for the second phase. The vertical rules of a table are applied in parallel in this phase for generating the columns of the rectangular arrays over terminals. When the table with the terminating vertical rules of the form $B \to b$ is applied the vertical generation halts, with the array obtained collected in the picture language generated by the 2TRLG. Note that the picture language generated by a 2TRLG consists of rectangular arrays of symbols. When there are only two tables of vertical rules with one of these containing all the rules $X \to aY$ and the other table containing all the rules of the form $X \to a$, $X, Y \in V_i, a \in T$, then the 2TRLG is simply called two-dimensional right-linear grammar (2RLG) [4]. We denote by $L(2RLG)$ and $L(2TRLG)$ the corresponding families of array languages generated by tabled two-dimensional right-linear grammars and two-dimensional right-linear grammars. The following strict inclusion is known [13].

Lemma 1 *[13].* $L(2RLG) \subset L(2TRLG)$.

Lemma 2. $L(2TRLG) \subseteq TPAP_2(CF)$.

Proof. Given a 2TRLG $G = (V_h, V_v, V_i, T, S, R_h, R_v)$, we construct a TPAP Π_3 with two membranes and CF array rules as follows:

$\Pi_3 = (V_h \cup V_v \cup \{A' \mid A \in V_i\} \cup T, T, \#, [_1[_2]_2]_1, \emptyset, \{S\}, R_1, R_2, 2)$. R_2 consists of two tables of rules t_1, t_2 with target (out) for t_2, given by

$$t_1 = \{ \, {}_X{}^{\#} \to {}_{A'}{}^{Y} \, , \; {}_{A'} \to {}_{A'} \mid {}_X \to {}_{AY} \in R_h, \, X, Y \in V_h, A \in V_i\},$$

$$t_2 = \{ \, {}_{X \to A} \mid {}_X \to {}_A \in R_h, \, X \in V_h, A \in V_i\} \cup \{ \, {}_{A'} \to {}_A \mid A \in V_i\}(out)$$

For each table t in R_v with rules of the form $B \to aD$, $B, D \in V_v, a \in T$, R_1 consists of a corresponding table with rules of the form $\begin{smallmatrix} \mathtt{B} \\ \# \end{smallmatrix} \to \begin{smallmatrix} \mathtt{a} \\ \mathtt{D} \end{smallmatrix}$ while for each table t' in R_v with rules of the form $B \to a$, $B \in V_v, a \in T$, R_1 consists of a corresponding table with target (in) and rules of the form $\mathtt{B} \to \mathtt{a}$. We note that the tables of rules of R_2 simulate the derivations in the horizontal phase of G generating strings of intermediates. In fact, the rules with target indication *out* terminate a derivation whenever termination happens in the first phase of G and the string is sent to region 1. In this region 1, the tables of rules of R_1 simulate the parallel derivation of the second vertical phase of G generating rectangular arrays of the picture array language of G which are sent to region 2. □

Lemma 3. $TPAP_2(CF) \backslash L(2TRLG) \neq \emptyset$.

Proof. Consider the picture language L_c consisting of rectangular arrays with a middle row of symbol c and an equal number of rows above and below this row, with each of these rows made of symbol a. This language is generated by the tabled parallel array P system $\Pi_4 = (V, T, \#, [_1[_2]_2]_1, \emptyset, \{S\}, R_1, R_2, 2)$, with $V = \{S, A, B, a, c\}$, $T = \{a, c\}$; R_2 consists of tables of rules $t_1, t_2(out)$ and R_1 consists of tables $t_3, t_4(in)$; $t_1 = \{ \, {}_{\mathtt{S}} \to {}_{\mathtt{AS}} \, , \, {}_{\mathtt{S}} \to {}_{\mathtt{A}} \, , \, {}_{\mathtt{A}} \to {}_{\mathtt{A}} \}, t_2 = \left\{ \begin{smallmatrix} \# \\ \mathtt{A} \\ \# \end{smallmatrix} \to \begin{smallmatrix} \mathtt{B} \\ \mathtt{c} \\ \mathtt{B} \end{smallmatrix} \right\}$

$t_3 = \left\{ \begin{matrix} \mathtt{B} \\ \# \end{matrix} \rightarrow \begin{matrix} \mathtt{a} \\ \mathtt{B} \end{matrix}, \begin{matrix} \# \\ \mathtt{B} \end{matrix} \rightarrow \begin{matrix} \mathtt{B} \\ \mathtt{a} \end{matrix} \right\}, t_4 = \{ \mathtt{B} \rightarrow \mathtt{a} \}$. In region 1, there is no initial array but in region 2, starting with the initial symbol S, the rules of table t_1 generate arrays with one row of the symbol A (of any desired length). This is followed by the application of the rules of table t_2 generating an array of the form $\begin{matrix} \mathtt{B} & \mathtt{B} & \ldots & \mathtt{B} & \mathtt{B} \\ \mathtt{c} & \mathtt{c} & \ldots & \mathtt{c} & \mathtt{c} \\ \mathtt{B} & \mathtt{B} & \ldots & \mathtt{B} & \mathtt{B} \end{matrix}$ which is then sent to region 1. Here the application of the rules of table t_3 as many times as we need, add rows made of the symbol a, equal in number, above and below the rewritten array, with the rewriting finally terminated by an application of the rule of table t_4. The arrays generated are sent back to region 2, where they are collected in the picture array language generated by Π_3, constituting the language L_c. But this language cannot be generated by any $2TRLG$, since the tables of rules in such a grammar are regular array rules and hence there is no ability for the grammar to check the equality of the number of rows made of a, above and below the middle row of c, although a row of c can be generated. □

The following Theorem is a consequence of Lemmas 1, 2 and 3.

Theorem 1. $L(2RLG) \subset L(2TRLG) \subset TPAP_2(CF)$.

4 Application to Generation of Picture Patterns

Generation of picture patterns, referred to as "kolam" patterns (also called "floor designs") (Fig. 3), using array grammars is well-known [11].

The approach is to encode the picture pattern as an array over certain terminal symbols, usually rectangular array with certain number of rows and columns, and generate the array with the rules of an array grammar. Then substitute for each symbol some suitable basic unit of the picture pattern to be generated, yielding the picture pattern. This kind of picture pattern generation has been done using certain P systems also [13,14].

Here we construct a $TPAP$ π_p with only one membrane, generating a language L_p of picture arrays which can be interpreted to represent picture patterns, one member of which is shown in Fig. 3. The compound patterns corresponding to the terminal symbols are shown in Fig. 4*b*, while the primitive patterns involved in the compound pattern are shown in Fig. 4*a*. The $TPAP$ system π_p is given by $\pi_p = (V, T, [_1]_1, F_1, R_1, 1)$ where $V = \{A, B, C, D\}, T = \{a, b_{ud}, b_{lr}, c_{ud}, c_{lr}, p_u, p_d, p_r, p_l\}$, $F_1 = \left\{ \begin{matrix} & \mathtt{A} & \\ \mathtt{D} & \mathtt{a} & \mathtt{B} \\ & \mathtt{C} & \end{matrix} \right\}$;

R_1 consists of tables of rules t_1, t_2, t_3;

$$t_1 = \left\{ \begin{matrix} \# \\ A \end{matrix} \rightarrow \begin{matrix} A \\ b_{ud} \end{matrix}, \; B\ \# \rightarrow b_{lr}\ B, \; \begin{matrix} C \\ \# \end{matrix} \rightarrow \begin{matrix} b_{ud} \\ C \end{matrix}, \; \#\ D \rightarrow D\ b_{lr} \right\}$$

$$t_2 = \left\{ \begin{matrix} \# \\ A \end{matrix} \rightarrow \begin{matrix} A \\ c_{ud} \end{matrix}, \; B\ \# \rightarrow c_{lr}\ B, \; \begin{matrix} C \\ \# \end{matrix} \rightarrow \begin{matrix} c_{ud} \\ C \end{matrix}, \; \#\ D \rightarrow D\ c_{lr} \right\}$$

$$t_3 = \{ A \rightarrow p_u, \; B \rightarrow p_r, \; C \rightarrow p_d, \; D \rightarrow p_l \}$$

Starting with the axiom array $D \begin{matrix} A \\ a \\ C \end{matrix} B$ the rules of the table t_1 could be applied any number of times and likewise the rules of table t_2 could also be applied any number of times and there is no preference in the order of application of these

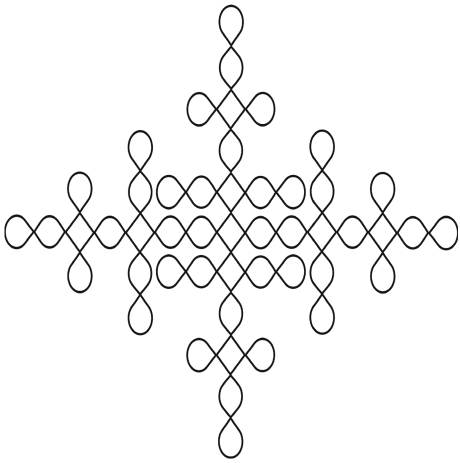

Fig. 3. Picture pattern corresponding to a member of L_p

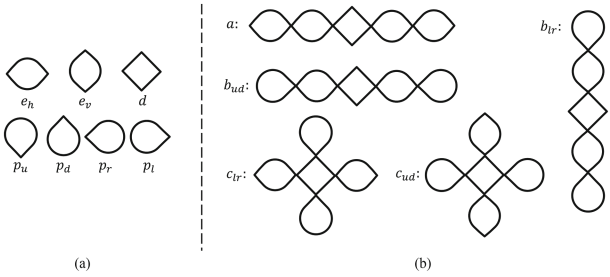

Fig. 4. (a) Primitive Patterns (b) Compound patterns

two tables. Once the rules of table t_3 are applied, the derivation ends generating a picture array over T which is collected in the language L_p. For example, the picture array in Fig. 5 is an element of L_p and the picture pattern corresponding to this picture array is given in Fig. 3. We note that in a picture array of L_p, the compound pattern b_{lr} will occur as many times as b_{ud} while c_{lr} will occur as many times as c_{ud} but b_{ud} and c_{ud} need not be in equal number.

Remark 2. The picture language of the $TPAP$ Π_p cannot be generated by any parallel array P system with just one membrane since the feature of grouping of rules is absent in PAP and hence the rules of tables t_1, t_2 cannot be applied independent of each other if only one membrane is used but with two membranes it is possible to generate the language L_p. It is straightforward to construct such a PAP and the details are omitted here.

$$\begin{array}{ccccccc} & & & p_u & & & \\ & & & c_{ud} & & & \\ & & & b_{ud} & & & \\ p_r & c_{lr} & b_{lr} & a & b_{lr} & c_{lr} & p_d \\ & & & b_{ud} & & & \\ & & & c_{ud} & & & \\ & & & p_d & & & \end{array}$$

Fig. 5. Picture array of the pattern in Fig. 3

5 Conclusion

In this paper, tabled parallel array P system is introduced and the generative power examined. An application in generating picture patterns is given. It will be interesting to find picture patterns generated by a $TPAP$ which will require $m \geq 2$ membranes but at the same time not able to generate by any PAP with m membranes.

Acknowledgements. The first author would like to thank Ministry of Higher Education for the award of MyPhD under which this research was jointly carried out by him. The second author gratefully acknowledges support for this research from an RUI grant 1001/PKOMP/811290 awarded by Universiti Sains Malaysia. The third author gratefully acknowledges support for this research from Science Fund of Ministry of Science, Technology and Innovation (MOSTI), Malaysia with grant code: 01-01-02-SF1104.

References

1. Ceterchi, R., Mutyam, M., Păun, G., Subramanian, K.G.: Array-rewriting P systems. Nat. Comput. **2**, 229–249 (2003)
2. Freund, R.: Array Grammars. Technical report 15/00, Research Group on Mathematical Linguistics, Rovira i Virgili University, Tarragona, p. 164 (2000)
3. Gheorghe, M., Păun, G., Pérez-Jiménez, M.J., Rozenberg, G.: Research frontiers of membrane computing: open problem and research topics. Int. J. Found. Comput. Sci. **24**(05), 547–623 (2013)
4. Giammarresi, D., Restivo, A.: Two-dimensional languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3, pp. 215–267. Springer, New York (1997)
5. Pan, L., Păun, G.: On parallel array P systems. In: Adamatzky, A. (ed.) Automata, Universality, Computation. Emergence, Complexity and Computation, vol. 12, pp. 171–181. Springer, Switzerland (2015)
6. Păun, G.: Computing with membranes. J. Comput. Syst. Sci. **61**, 108–143 (2000)
7. Păun, G., Rozenberg, G., Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press, Inc., New York (2010)
8. Rosenfeld, A., Siromoney, R.: Picture languages - a survey. Lang. Des. **1**, 229–245 (1993)
9. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, vol. 1–3. Springer, Berlin (1997)

10. Rozenberg, G., Salomaa, A.: The Mathematical Theory of L Systems. Academic Press, New York (1980)
11. Siromoney, G., Siromoney, R., Krithivasan, K.: Array grammars and kolam. Comput. Graph. Image Process. **3**, 6382 (1974)
12. Subramanian, K.G., Rangarajan, K., Mukund, M. (eds.): Formal Models, Languages and Applications. Series in Machine Perception and Artificial Intelligence, vol. 66. World Scientific Publishing, Singapore (2006)
13. Subramanian, K.G., Saravanan, R., Robinson, T.: P system for array generation and application to kolam patterns. Forma **22**, 47–54 (2007)
14. Subramanian, K.G., Isawasan, P., Venkat, I., Pan, L.: Parallel array-rewriting P systems. Rom. J. Inf. Sci. Technol. **17**(1), 103–116 (2014)
15. Wang, P.S.P. (ed.): Array Grammars, Patterns and Recognizers. Series in Computer Science, vol. 18. World Scientific, Singapore (1989)