P systems with array objects and array rewriting rules

K.G. Subramanian[1]*, R. Saravanan[2], M. Geethalakshmi[3], P. Helen Chandra[4] and M. Margenstern[5]

(1. Department of Computer Science, Sri Muthukumaran Institute of Technology, Chennai 600 069, India; 2. Department of Mathematics, Bharath Institute of Higher Education and Research, Chennai 600 073, India; 3. Department of Mathematics, Dr. MGR Janaki College, Adayar, Chennai, India; 4. Department of Mathematics, Jayaraj Annapackiam College for Women Periyakulam 625 601, India; 5. LITA, Universite Paul Verlaine-Metz, Ile du Saulcy, 57045 Metz Cedex, France)

Abstract Array P systems were introduced by Păun Gh. which is linking the two areas of membrane computing and picture grammars. Puzzle grammars were introduced by us for generating connected picture arrays in the two-dimensional plane, motivated by the problem of tiling the plane. On the other hand, incorporating into arrays the developmental type of generation used in the well-known biologically motivated L systems, Siromoney and Siromoney proposed a very general rectangular array generating model, called extended controlled tabled L array system (ECTLAS). In this paper we introduce two variations of the array P system, called BPG array P system and parallel array P system. The former has in the regions array objects and basic puzzle grammar rules (BPG), which are a specific kind of puzzle grammar rules. In the latter, the regions have rectangular array objects and tables of context-free rules. We examine these two types of P systems for their array generative power.

Keywords: membrane computing, P systems, array grammars.

The area of membrane computing was initiated by Păun[1] introducing a new computability model, now called as P system, which is a distributed, highly parallel theoretical computing model, based on the membrane structure and the behaviour of the living cells. Among a variety of applications of this model, the problem of handling array languages using P systems has been considered by Ceterchi et al.[2] by introducing array-rewriting P systems and thus linking the two areas of membrane computing and picture grammars.

On the other hand, in the study of generation and description of picture patterns considered as connected digitized, finite arrays of symbols, syntactic techniques for pattern recognition and image analysis have played a significant role on account of their structure-handling ability. Adapting the techniques of formal string language theory, various types of picture or array grammars have been introduced and investigated[3,4]. Puzzle grammars introduced in [5] are array generating two-dimensional grammars motivated by the problem of tiling the plane. A subclass called basic puzzle grammars was introduced by Subramanian et al.[6]. Another very general rectangular array generating model, called extended controlled tabled L array system (ECTLAS) was proposed by Siromoney and Siromoney[7], incorporating into arrays the developmental type of generation used in the well-known biologically motivated L-systems. In this nore we introduce two kinds of array P systems for generation of pictures which are arrays of symbols. In the first kind of array P systems we take the objects in the regions as arrays and rules to be basic puzzle grammar rules. We call these P systems as BPG array P systems. In the second kind, we take the objects in the regions as rectangular arrays. Tables of context-free rules are in the regions, with rewriting of a rectangular array being done at a time by rules in a single table, as in a ECTLA system of [7]. We call the resulting P system as parallel array P system. We examine these two types of picture array generating P systems for their generative power. Preliminary versions of these two systems were reported in [8,9].

1 Preliminaries

Let Σ be a finite alphabet. A word or string w over Σ is a sequence of symbols from Σ. The set of all words over Σ, including the empty word λ with no symbols, is denoted by Σ^*. An array over Σ consists of finitely many symbols from Σ placed at the points of $\mathbb{Z}^2$ (the two-dimensional plane), and the points of the plane not marked with symbols of Σ are assumed to have the blank symbol $\# \notin \Sigma$. We will pictorially represent the arrays, indicating their non blank pixels, whenever possible. The set of all arrays

* To whom correspondence should be addressed. E-mail: kgsmani1948@yahoo.com

over Σ will be denoted by Σ^{*2}. An array over $\{a\}$ describing the picture token T is shown in Fig. 1.

$$\begin{array}{c} a\ a\ a\ a\ a\ a\ a\ a\ a \\ a \\ a \\ a \\ a \end{array}$$

Fig. 1. An array describing picture token T.

An array, in particular, can be rectangular. A rectangular $m \times n$ array M over Σ is of the form

$$M = \begin{matrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{matrix}$$

where each $a_{ij} \in \Sigma$, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$. The set of all rectangular arrays over Σ is denoted by Σ^{**}, which includes the empty array Λ.

We now recall the definition of basic puzzle grammars introduced in [6]. These grammars constitute a special class of puzzle grammars defined in [5] for generation of arrays non-rectangular or rectangular. We refer to [5] for notions of puzzle grammars and to [3,4] for array grammars. For notions of formal language theory we refer to [10].

Definition 1. A basic puzzle grammar (BPG) is a structure $G = (N, T, R, S)$ where N and T are finite sets of symbols; $N \cap T = \emptyset$; elements of N are called non-terminals and elements of T, terminals; $S \in N$ is the start symbol or the axiom; R consists of rules of the following forms:

$A \to \textcircled{a}B$, $A \to a\ \textcircled{B}$, $A \to B\ \textcircled{a}$, $A \to \textcircled{B}a$

$A \to \begin{matrix} \textcircled{a} \\ B \end{matrix}$, $A \to \begin{matrix} a \\ \textcircled{B} \end{matrix}$, $A \to \begin{matrix} B \\ \textcircled{a} \end{matrix}$, $A \to \begin{matrix} \textcircled{B} \\ a \end{matrix}$, $A \to \textcircled{a}$

where $A, B \in N$ and $a \in T$. We may omit the circle in the rule with a single a on the right side.

Derivations begin with S written in a unit cell in the two-dimensional plane, with all the other cells containing the blank symbol #, not in $N \cup T$. In a derivation step, denoted by $\Rightarrow$, a non-terminal A in a cell is replaced by the right-hand member of a rule whose left-hand side is A. In this replacement, the circled symbol of the right-hand side of the rule used occupies the cell of the replaced symbol and the non-circled symbol of the right side occupies the cell to the right or the left or above or below the cell of the replaced symbol depending on the type of the rule used. The replacement is possible only if the cell to be filled in by the non-circled symbol contains a blank symbol.

The basic puzzle language (BPL) generated by the *BPG* G, denoted by $L(G)$, is the set of connected finite arrays over T, derivable in one or more steps from the axiom.

We denote the family of *BPL* by $L(BPL)$.

Remark 1. A regular array grammar (RAG)[11,12] consists of array productions of the following forms:

$A\# \to aB$, $\#A \to Ba$, $\begin{matrix} A \\ \# \end{matrix} \to \begin{matrix} a \\ B \end{matrix}$, $\begin{matrix} \# \\ A \end{matrix} \to \begin{matrix} B \\ a \end{matrix}$, $A \to a$

These are in fact included in the productions of a BPG. For example the equivalent of $A\# \to aB$ is

$$A \to \textcircled{a}B$$

We denote by $L(RAL)$ the family of the regular array languages (RAL) generated by *RAGs*. It is known that $L(RAL) \subset L(BPL)$[6].

We next recall extended controlled tabled L array systems (ECTLAS) introduced in [7] for generating rectangular arrays. These systems were introduced in [7] incorporating the Lindenmayer-like parallel rewriting in arrays. Here we restrict ourselves to ECT0LAS where tables of context-free rules only are used.

Definition 2. An extended, controlled tabled $0L$ array system (*ECT0LAS*) is a 5-tuple $G = (V, T, \wp, C, S, \#)$ where V is a finite nonempty set (the alphabet of G); $T \subseteq V$ is the terminal or target alphabet of G; $\wp$ is a finite set of tables, $\{t_1, t_2, \cdots, t_k\}$, and each t_i, $i = 1, \cdots, k$, is a left, right, up, or down table consisting respectively, of a finite set of left, right, up, or down rules only. The rules within a table are context-free in nature but all right hand sides of rules within the same table are of the same length; C is a control language over $\wp$; $S \notin V$ is the start rectangular array; # is an element not in V (the marker of G).

Let $G = (V, T, \wp, C, S, \#)$ be an *ECT0LAS*. Let

$$M_1 = \begin{matrix} a_{11} & \cdots & a_{1(n-1)} & a_{1n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & \cdots & a_{m(n-1)} & a_{mn} \end{matrix},$$

$$M_2 = \begin{matrix} a_{11} & \cdots & a_{1(n-1)} & \omega_{1n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & \cdots & a_{m(n-1)} & \omega_{mn} \end{matrix}$$

with a_{ij} in V and ω_{in} in V^*, $i=1,\cdots,m, j=1,\cdots,n$. We say that M_1 directly derives M_2 (by a right table R in $\wp$), denoted by $M_1 \Rightarrow^R M_2$, if M_2 is obtained by applying in parallel the rules in a right table to all the symbols in the rightmost column of M_1. Similarly we define $\Rightarrow^L, \Rightarrow^U, \Rightarrow^D$ corresponding to a left, up, or down table. We write $M_1 \Rightarrow M_2$ if either $M_1 \Rightarrow^R M_2$, or $M_1 \Rightarrow^L M_2$, or $M_1 \Rightarrow^U M_2$, or $M_1 \Rightarrow^D M_2$. We write $M_0 \Rightarrow^* M$ iff there exists a sequence of derivations $M_0 \Rightarrow^{p_{i1}} M_1 \Rightarrow^{p_{i2}} \cdots \Rightarrow^{p_{in}} M_n = M$, such that $p_{i1}p_{i2}\cdots p_{in} \in C$.

A set $\mathcal{M}(G)$ of arrays is called an extended controlled table $0L$ array language ($ECT0LAL$) iff there exists an extended controlled table $0L$ array system G such that $\mathcal{M}(G) = \{M \mid S \Rightarrow^* M, M \in T^{**}\}$. The family of extended controlled tabled $0L$ array languages is denoted by $L(ECT0LAL)$.

In particular,

1. if $V = T$ and S is a rectangular array M_0 (the axiom), G is a controlled tabled $0L$ array system;

2. if $C = \wp^*$, then there is no control and the order of application of the tables is arbitrary; G is then an extended tabled $0L$ array system; G is a tabled $0L$ array system if in addition $V = T$.

In these cases the corresponding family of languages is respectively denoted by $L(CT0LAL)$, $L(ET0LAL)$, $L(T0LAL)$.

2 Array rewriting P systems with BPG rules

Array P systems were introduced in [2] as an extension from the string rewriting P systems. In these systems arrays are placed in the regions of the system and they evolve by means of array-rewriting rules. In particular, the rules can be regular array rewriting rules. We now consider a variation of the array P systems of degree m by taking the rules in the membranes as basic puzzle grammar rules. We call the resulting array P system as BPG array P system of degree m ($EAP_m(BPG)$) (E refers to the fact that the system is extended with a terminal alphabet). When the rules in the membranes are taken as regular array grammar rules, we call the array P system as REG array P system of degree m ($EAP_m(REGA)$). The corresponding language classes of array P systems with regular and BPG rules are respectively denoted by $L(EAP_m(BPG))$, $L(EAP_m(REGA))$.

Definition 3. A BPG array P System of degree $m(\geqslant 1)$ is a contruct

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

where V is the total alphabet, $T \subseteq V$ is the terminal alphabet, $\#$ is the blank symbol, μ is a membrane structure with m membranes labeled in a one-to-one way with $1, 2, \cdots, m$; $F_1, \cdots, F_m$ are finite sets of arrays over V initially associated with the m regions of μ; $R_1, \cdots, R_m$ are finite sets of BPG rules over $V \cup T$ associated with the m regions of μ; the rules have attached targets, here, out, in (in general, here is omitted); finally, i_0 is the label of an elementary membrane of μ (the output membrane).

A computation in a BPG array P system is defined in the same way as in a string rewriting P system[1] with the successful computations being the halting ones; each array from each region of the system, which can be rewritten by a rule associated with that region (membrane), should be rewritten; this means that one rule is applied (the rewriting is sequential at the level of arrays); The array obtained by rewriting is placed in the region indicated by the target associated with the rule used; "here" means that the array remains in the same region, "out" means that the array exits the current membrane—thus, if the rewriting was done in the skin membrane, then it can exit the system; arrays leaving the system are "lost" in the environment, and "in" means that the array is immediately sent to one of the directly lower membranes, non-deterministically chosen if several exist (if no internal membrane exists, then a rule with the target indication in cannot be used).

A computation is successful only if it stops, i.e., if a configuration is reached where no rule can be applied to the existing arrays. The result of a halting computation consists of the arrays composed only of symbols from T placed in the membrane with label i_o in the halting configuration.

The set of all such arrays computed (we also say generated) by a system Π is denoted by $AL(\Pi)$. The family of all array languages $AL(\Pi)$ generated by systems Π as above, with at most m membranes, is denoted by $EAP_m(BPG)$. The regular array rewriting ($REGA$) rules are also BPG rules. So when $REGA$ rules alone are used in the regions, we

call the family as $EAP_m(REGA)$.

By definition it follows that

1. $EAP_m(X) \subseteq EAP_{m+1}(X)$ for $X \in \{REGA, BPG\}$

2. $EAP_m(REGA) \subseteq EAP_m(BPG)$

Theorem 1. (i) $L(RAL) \subset EAP_1(REGA)$

(ii) $L(BPL) \subset EAP_1(BPG)$

(iii) $EAP_1(REGA) \subset EAP_1(BPG)$

(iv) $EAP_1(REGA) \subset EAP_2(REGA)$

(v) $EAP_1(BPG) \subset EAP_2(BPG)$

(vi) $EAP_2(REGA) \subset EAP_2(BPG)$

Proof. The inclusions in (i)—(vi) are clear from the definitions.

The proper inclusion (i) can be seen as follows: Let

$$\Pi_1 = \left(\{A, B, a\}, \{a\}, \#, [\,]_1, \begin{Bmatrix} A\ a\ A \\ B \end{Bmatrix}, R_1, 1 \right)$$

$$R_1 = \{\# A \to Aa, A \# \to aA, \begin{matrix} B \\ \# \end{matrix} \to \begin{matrix} a \\ B \end{matrix}, A \to a, B \to a\}$$

The regular array P System Π_1 generates a language L_1 consisting of arrays in the shape of token T but not necessarily with equal "arms". But a regular array grammar cannot generate L_1 as the rewriting in a regular array grammar, when it reaches the "junction" in a T shaped array can either proceed horizontally (left or right) or vertically (down) and thus will fail to produce the third arm. The proper inclusion in (ii) can be seen similarly, by considering a language L_2 consisting of arrays in the shape of token T, not necessarily of equal "arms" but with "protrusions" i.e., with an extra symbol a above every alternate symbol "a" in the horizontal left and right "arms" in the T shaped array. The system Π_1 can be modified to take care of this feature by having suitable BPG rules and having the initial array in the following form: $\begin{matrix} & a & \\ A & a & A \\ & B & \end{matrix}$. Again BPG rules will fail to produce the "third arm", for a reason similar to the case of REGA rules. The proper inclusion in (iii) follows by noting that the language L_2 described above is in $EAP_1(BPG)$ but regular array rewriting rules alone are not enough to produce the "protrusions". The proper inclusion in (iv) can be seen as follows:

$$\Pi_3 = \left(\{A, B, a\}, \{a\}, \#, [\,[\,]_2]_1, \begin{Bmatrix} A & \\ a & B \end{Bmatrix}, \emptyset, R_1, R_2, 2 \right)$$

$$R_1 = \left\{ \begin{matrix} \# \\ A \end{matrix} \to \begin{matrix} A \\ a \end{matrix} (in) \right\}$$

$$R_2 = \{B \# \to aB(out), B \# \to aC, C \to a, A \to a\}$$

The non-terminals A and B take care of growing the vertical arm (in the skin membrane) and the horizontal arm (in membrane 2), step by step; at any time the computation stops in membrane 2 after using the rule $A \to a$, $B \# \to aC$ and $C \to a$, in any order. In case after terminating A, the array is sent out to skin membrane by the application of $B \# \to aB$, then it gets stuck in the skin membrane. The L shaped arrays over $\{a\}$ with equal "arms", collected in membrane 2 constitute the language generated. But this picture language L_3 cannot be generated by a regular array grammar in just a single membrane, as the rules of a regular array grammar cannot maintain equal growth between horizontal and vertical "arms".

The proper inclusion in (v) can be seen as follows: The array language L_4 consisting of picture arrays (one of these is shown in Fig. 2) with equal "arms" and "single protrusions" to the left of the vertical line of x's and below the horizontal line if x's, is generated by the following array P system with basic puzzle grammar rules.

$$\Pi_4 = \left(\{A, B, C, D, E, F, x\}, [\,[\,]_2]_1, \begin{matrix} & A & \\ x & x & B \\ & x & \end{matrix}, \emptyset, R_1, R_2, 2 \right)$$

$$R_1 = \left\{ A \to \begin{matrix} & C & \\ \textcircled{x} & & (in), C \to x\ \textcircled{D}(here), \\ & A & \end{matrix} \right.$$

$$\left. D \to \textcircled{x}(in) \right\}$$

$$R_2 = \left\{ B \to \textcircled{x}E(out), E \to \begin{matrix} \textcircled{F} \\ x \end{matrix} \right.$$

$$\left. F \to \textcircled{x}B(out), F \to \textcircled{x}U, A \to x, U \to x \right\}$$

$$\begin{array}{c} x \\ x\ x \\ x \\ x\ x\ x\ x\ x \\ x\quad x \end{array}$$

Fig. 2. Array with equal "arms" and "protrusions".

But this language cannot be generated by *BPG* rules in just a single membrane as these rules cannot maintain equal growth between the horizontal and vertical directions.

The proper inclusion in (vi) is due to the fact that REGA rules cannot produce the "protrusions" as in the arrays of L_4.

3 Array rewriting parallel P systems

We now consider another variation of the array P systems introduced in [2]. We take the objects in this array P System of [2] as rectangular arrays. Also we take tables of context free rules of the form $a \to \alpha$, $\alpha \in V^*$, for an alphabet V, in the regions of the P system with the application of a table being done as in a ECT0LAS[7]. We call the resulting array P system as parallel array P system (PAPS).

Definition 4. A parallel array P system (PAPS) is a construct

$$\Pi = (V, T, \#, \mu, F_1, \cdots, F_m, \Im_1, \cdots, \Im_m, i_0),$$

where V is the total alphabet, $T \subseteq V$ is the terminal alphabet, μ is a membrane structure with m membranes labelled in a one to one way with $1, 2, \cdots, m$; $F_1, \cdots, F_m$ are finite sets of rectangular arrays over V associated with the m regions of μ; $\Im_1, \cdots, \Im_m$ associated with m regions of μ are finite sets of left, right, up, down tables of context free rules over V of the form $a \to \alpha$, $a \in V$, $\alpha \in V^*$; all α's have the same length in a table. The tables have attached targets here, out, in; i_0 is the label of an elementary membrane of μ (output membrane).

When a set T is distinguished, we speak about an extended PAP system; when $V = T$ we have a non-extended system.

A computation in a PAP system is defined in the same way as in a string rewriting P system with the successful computations being the halting ones: each rectangular array, from each region of the system, which can be rewritten by a table of rules, rewriting being done as in a *ECT0LAS*[7] associated with that region (membrane), should be rewritten; this means that one table is applied and the rectangular array obtained by rewriting is placed in the region indicated by the target associated with the table used (here means that the rectangular array remains in the same region, out means that the rectangular array exits the current membrane-thus, if the rewriting was done in the skin membrane, then it can exit the system; rectangular arrays leaving the system are "lost" in the environment, and in means that the rectangular array is immediately sent to one of the directly lower membranes, nondeterministically chosen if several exist. If no internal membrane exists, then a table with the target indication in cannot be used.

A computation is successful only if it stops, i.e., if a configuration is reached where no table can be applied to the existing rectangular arrays. The result of a halting computation consists of the rectangular arrays composed only of symbols from T placed in the membrane with label i_0 in the halting configuration. The set of all such rectangular arrays computed by a system Π is denoted by $RAL(\Pi)$. The family of all array languages $RAL(\Pi)$ generated by systems Π as above, with at most m membranes, is denoted by $EPAP_m$; if non-extended systems are considered, then we write PAP_m.

By definition, the following inclusions are clear.

(i) $PAP_m \subseteq PAP_{m+1}$ for $m \geqslant 1$

(ii) $EPAP_m \subseteq EPAP_{m+1}$ for $m \geqslant 1$

(iii) $PAP_m \subseteq EPAP_m$ for $m \geqslant 1$

(iv) $L(T0LAL) \subseteq PAP_1$

(v) $L(ET0LAL) \subseteq EPAP_1$

Theorem 2. (i) $PAP_3 \setminus L(T0LAL) \neq \emptyset$

(ii) $EPAP_5 \setminus L(ET0LAL) \neq \emptyset$

Proof. (i) Let $\Pi_5 = (V, V, [_1[_2[_3]_3]_2]_1, M_0, \phi, \phi, \Im_1, \Im_2, \phi, 3)$, where

$$V = \{X, \cdot\}, \quad M_0 = \begin{array}{cc} X & \cdot \\ X & \cdot \\ X & X \end{array}$$

$$\Im_1 = \{(R_1, in)\}, \quad \Im_2 = \{(U, out), (R_2, in)\},$$

$$R_1 = \{X \to XX, \cdot \to \cdot\cdot\},$$

$R_2 = \{X \to X, \cdot \to \cdot\}$ are right tables.

$U = \{X \to \begin{array}{c} X \\ X \end{array}, \cdot \to \begin{array}{c} \cdot \\ \cdot \end{array}\}$ is an up table.

The axiom rectangular array M_0 is initially in the region 1. When the rules of the table R_1 are applied to this array it grows one column in the right and is then sent to region 2. If R_2 is applied, then this array is sent to region 3 where it remains for ever and the language generated collects this array. If U is applied to M_1 in region 2 the array grows upwards and is sent back to region 1. The derivation then continues.

The array language generated consists of arrays of the form in Fig. 3 where the array represents token L (. is interpreted as blank) with equal "arms". This array language belongs to PAP_3 but it is not in $L(T0LAL)$ as controlling the application of right and up tables is not possible in a $T0LAS$.

$$\begin{array}{c} X \cdot \cdot \cdot \cdot \\ X \cdot \cdot \cdot \cdot \\ X \cdot \cdot \cdot \cdot \\ X \cdot \cdot \cdot \cdot \\ X\,X\,X\,X\,X \end{array}$$

Fig. 3. Array describing token L.

(ii) Let

$$\Pi_6 = (V, T, [_1[_2[_3[_4[_5]_5]_4]_3]_2]_1, M_0, \phi, \phi, \phi, \phi, \Im_1, \Im_2, \Im_3, \Im_4, \Im_5, 5)$$

where

$$V = \{A, B, a, b\},\ T = \{a, b\},\ M_0 = \begin{matrix} b\ b \\ b\ b \end{matrix}$$

$$\Im_1 = \{(R_1, in), (R_2, here)\},$$

$$\Im_2 = \{(L_1, in), (L_2, out)\},$$

$$\Im_3 = \{(U_1, in), (U_2, out)\},$$

$$\Im_4 = \{(D_1, out), (D_2, in)\},$$

$$\Im_5 = \{(U_3, here)\},\ R_1 = \{b \to bA, a \to aa\},$$

$R_2 = \{B \to b, A \to b, a \to a\}$ are right tables.

$$L_1 = \{b \to Ab, a \to aa\},$$

$L_2 = \{A \to b, B \to b, a \to a\}$ are left tables.

$$U_1 = \left\{b \to \begin{matrix} a \\ b \end{matrix}, a \to \begin{matrix} a \\ a \end{matrix}, A \to \begin{matrix} B \\ a \end{matrix}\right\},$$

$U_2 = \{a \to a, B \to A\}$

$U_3 = \{B \to b, a \to a\}$ are up tables.

$$D_1 = \left\{b \to \begin{matrix} b \\ a \end{matrix},\ a \to \begin{matrix} a \\ a \end{matrix}, A \to \begin{matrix} a \\ B \end{matrix}\right\}$$

$$D_2 = \left\{b \to \begin{matrix} b \\ a \end{matrix},\ a \to \begin{matrix} a \\ a \end{matrix}, A \to \begin{matrix} a \\ b \end{matrix}\right\}$$

are down tables.

The axiom rectangular array is initially in the region 1. An application of the right table R_1 allows the array grow one column to the right and the array is immediately sent to region 2. Here the left table L_1 alone is applicable which grows the array in the left by one column. The array is then sent to region 3 where the up table U_1 alone is applicable and this allows the array to grow by one row upwards and is then sent to region 4; if the down table D_2 is applied the array grows one row downwards and the array is sent to region 5. Here the computation comes to an end with an application of the up table U_3, yielding a rectangular array over terminals and this is collected in the language L_6 generated. If D_1 were applied in region 4, the array grows one row downwards but is sent out to region 3. Only the up table U_2 can be applied without the top row growing but changing B's to A. The array is then sent out to region 2 where again a similar change of symbols happens in the leftmost column. Again the array is sent out to region 1 where an application of the right table R_2 alone is possible and the process repeats with an application of R_1 and so on. This kind of controlling growth one after another is not possible in an $ECT0LAS$. One of the arrays generated is shown in Figure 4. This proves that $EPAP_5 \setminus L(ET0LAL) \neq \emptyset$.

$$\begin{array}{c} b\ a\ a\ a\ a\ b \\ a\ b\ a\ a\ b\ a \\ a\ a\ b\ b\ a\ a \\ a\ a\ b\ b\ a\ a \\ a\ b\ a\ a\ b\ a \\ b\ a\ a\ a\ a\ b \end{array}$$

Fig. 4 An array in L_6.

Acknowledgement The authors thank the referees for their very useful comments. The author K. G Subramanian is grateful to Prof. M. Margenstern, University of Paul Verlaine-Metz, France, for his invitation and the University for the support to visit the university during May-June, 2006. Part of this work was done during this visit.

References

1 Păun Gh. Membrane Computing: An Introduction. Berlin, Heidelberg: Springer-Verlag, 2000

2 Ceterchi R, Mutyam M, Păun Gh, et al. Array-rewriting P systems. Natural Computing, 2003, 2: 229—249

3 Rosenfeld A. Picture Languages. New York: Academic Press, 1979

4 Rosenfeld A and Siromoney R. Picture languages—a survey. Languages of Design, 1993, 1: 229—245

5 Nivat M, Saoudi A, Subramanian KG, et al. Puzzle grammars and context-free array grammars. Int Journal of Pattern Recognition and Artificial Intelligence, 1991, 5: 663—676

6 Subramanian KG, Siromoney R, Dare VR, et al. Basic puzzle languages. Int Journal of Pattern Recognition and Artificial Intelligence, 1995, 9: 763—775
7 Siromoney R and Siromoney G. Extended controlled tabled L-arrays. Information and Control, 1977, 35(2): 119—138
8 Subramanian KG, Geethalakshmi M and Helen Chandra P. Array rewriting P systems generating rectangular arrays. Paper presented at the National conference on Intelligent Optimization Modeling, Gandhigram Rural Institute-Deemed University, Gandhigram, India, March 2006
9 Subramanian KG, Saravanan R and Rangarajan K. Array P systems and basic puzzle grammars. Paper presented at the National conference on Intelligent Optimization Modeling, Gandhigram Rural Institute-Deemed University, Gandhigram, India, March 2006
10 Salomaa A. Formal Languages. London: Academic Press, 1973
11 Yamamoto Y, Morita K and Sugata K. Context-sensitivity of two-dimensional array grammars. In: Array Grammars, Patterns and Recognizers. Singapore: World Scientific, 1989, 17—41
12 Wang PSP. Array Grammars, Patterns and Recognizers. Singapore: World Scientific, 1989