CHAPTER IV

P SYSTEMS AND SEQUENTIAL/PARALLEL ARRAY GENERATION

4.1 INTRODUCTION

The array P systems considered in chapter 3, can generate arrays that are not necessarily rectangular arrays. In this chapter we concentrate on generation of rectangular arrays. One of the earliest rectangular array generating models was proposed by Siromoney et al. [22], motivated by “kolam” patterns. In this model, now called as two-dimensional (2D) matrix grammar, generation of rectangular arrays takes place in two phases with a sequential mode of rewriting in the first phase generating strings of intermediate symbols and a parallel mode of rewriting these strings in the second phase to yield rectangular picture patterns. Although these grammars are elegant and simple to handle, the generative power is weak in the sense these grammars for example cannot handle rectangular picture arrays that maintain a proportion between length and breadth.

In this chapter we introduce a new class of array P systems called sequential/parallel rectangular array P systems generating pictures of rectangular arrays. These P systems have rectangular arrays as objects in the membranes and rules in the membranes are either context-free or regular with sequential horizontal rewriting or right-linear rules with vertical rewriting in parallel as in the two-dimensional (2D) matrix grammars [22]. These P systems have more generative power and accordingly, they can generate picture patterns
that cannot be handled by 2D matrix grammars. As an application we also exhibit certain “kolam” patterns generated by these P systems. These patterns cannot be generated by any 2D matrix grammars.

4.2 TWO-DIMENSIONAL (2D) MATRIX GRAMMARS

We recall some of the basic notions before recalling 2D matrix grammars.

Let $\Sigma$ be a finite alphabet. $\Sigma^*$ is the set of all words over $\Sigma$ including the empty word. A picture $A$ over $\Sigma$ is a rectangular $m \times n$ array of elements of $\Sigma$. The set of all pictures over $\Sigma$ is denoted by $\Sigma^{**}$. A picture language or a two dimensional language over $\Sigma$ is a subset of $\Sigma^{**}$. The column concatenation $A\Phi B$ of an $m \times p$ array $A$ and an $n \times q$ array $B$ is defined only when $m = n$ and the row concatenation $A\theta B$ of $A$ and $B$ is defined only when $p = q$.

**Definition 4.2.1 (Two-Dimensional (2D) Matrix Grammars)**

A 2D matrix grammar is a 2-tuple $(G_1, G_2)$ where $G_1 = (H_1, I_1, P_1, S)$ is a Regular, CF or CS grammar; $H_1$ is a finite set of symbols called horizontal nonterminals; $I_1 = \{S_1, S_2, \ldots, S_k\}$, a finite set of symbols called intermediates; $H_1 \cap I_1 = \phi$; $P_1$ is a finite set of rules called horizontal rules; $S$ is the start symbol; $S \in H_1$; $G_2 = (G_{21}, G_{22}, \ldots, G_{2k})$ where $G_{2i} = (V_{2i}, T, P_{2i}, S_i)$, $1 \leq i \leq k$ are regular grammars; $G_{2i}$ is a finite set of symbols called vertical nonterminals; $V_{2i} \cap V_{2j} = \phi$ for $i \neq j$; $T$ is a finite set of terminals; $P_{2i}$ is a finite set of right-linear rules of the form $X \rightarrow aY$ (called vertical nonterminal rules) or $X \rightarrow a$ (called vertical terminal rules) where $X, Y \in V_{2i}$, $a \in T$; $S_i \in V_{2i}$ is the start symbol of $G_{2i}$.
G is a regular, (context-free, context-sensitive) 2D matrix grammar if \( G_1 \) is regular, (context-free, context sensitive) respectively. Derivations are defined as follows: First a string \( w \) over \( I_1 \) is generated horizontally using the horizontal rules. Vertical derivations then proceed in parallel using the rules of \( G_{2i} \). All symbols in the horizontal string \( w \) are rewritten in the vertical direction using vertical nonterminal rules for all symbols of \( w \) or vertical terminal rules for all symbols of \( w \), thus generating rectangular arrays over \( T \) when the vertical derivation terminates.

The set \( L(G) \) consists of all \( m \times n \) arrays generated by \( G \). We denote the picture language classes of regular, CF, CS 2DMatrix grammars by 2DRML, 2DCFML, 2DCSML respectively.

### 4.3 SEQUENTIAL/PARALLEL ARRAY P SYSTEMS

Now we introduce a new kind of array P systems by having the objects in the membranes as rectangular arrays and the rules as either context-free or regular rules or sets of vertical nonterminal/terminal rules as considered in a 2D matrix grammar.

**Definition 4.3.1**

A sequential / parallel Rectangular Array P system of degree \( m \geq 1 \) is a construct

\[
\Pi = (V_1 \cup V_2, I, T, \mu, F_1, \ldots, F_m, R_1, \ldots, R_m, i_0)
\]

where \( V = V_1 \cup V_2 \) is the total alphabet; \( V_1 - I \) is the set of horizontal nonterminals;
\[ I \subset V_1 \text{ is the set of intermediates; } V_2 - T \text{ is the set of vertical nonterminals; } T \subseteq V_2 \text{ is the set of terminals; } V_2 - T \text{ includes the elements of } I; \mu \text{ is a membrane structure with } m \text{ membranes labeled in a one-to-one way with } 1, 2, \ldots, m; F_1, \ldots, F_m \text{ are finite sets of rectangular arrays over } V \text{ associated with the } m \text{ regions of } \mu; R_1, \ldots, R_m \text{ are finite sets of rules associated with the } m \text{ regions of } \mu; \text{ the rules can be either horizontal context free rules of the form } A \rightarrow \alpha, A \in V_1 - I, \alpha \in V_1^* \text{ or a set of right linear vertical nonterminal rules of the form } X \rightarrow aY, X, Y \in V_2 - T, a \in T \text{ or a set of right-linear vertical terminal rules of the form } X \rightarrow a, X \in V_2 - T, a \in T. \]

The horizontal C F rules can be in particular regular rules of the form \( A \rightarrow wB, A \rightarrow w, A \in V_1 - I, w \in I^* \). Horizontal rules and sets of vertical rules have attached targets, here, out, in (in general, here is omitted). A membrane has either horizontal rules or sets of vertical rules; horizontal rules are applied in a sequential manner; the vertical rules in a parallel manner in the vertical direction as in a 2D matrix grammar. Finally, \( i_0 \) is the label of an elementary membrane of \( \mu \) (the output membrane).

A computation in a sequential / parallel array P 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 in each region, which can be rewritten by a horizontal rule or a set of vertical rules associated with that region (membrane), should be rewritten; the rectangular 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 and 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; a configuration is reached
where no rule can be applied to the existing 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 arrays computed (we also say generated) by a system Π
is denoted by RAL (Π). The family of all array languages RAL (Π). generated by systems Π as above, with at most m membranes, with horizontal
rules of type $\alpha \in \{\text{REG, CF}\}$ is denoted by S/PRAP_m(α).

**Example 4.3.1.**

Consider the Sequential/Parallel Rectangular Array P system belonging
to the class S/PRAP_4(REG)

$$
\Pi_1 = (V_1 \cup V_2, I, T, [1 [2 [3 [4]_3]_2]_1, A S_2 B, \phi, \phi, R_1, R_2, R_3, R_4, 4).$$

$$V_1 = \{ A, B, C, S_1, S_2 \}, V_2 = \{ S_1, S_2, D, a, b \}, I = \{ S_1, S_2 \},$$

$$T = \{ a, b \}$$

$$R_1 = \{ A \rightarrow S_1 A (in) \}$$

$$R_2 = \{ B \rightarrow S_1 B (out), B \rightarrow S_1 C (in) \}$$

$$R_3 = \{ A \rightarrow S_1 (here), C \rightarrow S_1 (in) \}$$
\[
R_4 = \{ \{ S_1 \to a \ , S_2 \to a \} (\text{here}) , \{ D \to b , S_2 \to a \} (\text{here}) , \{ D \to b , \}
\]
\[
S_2 \to a \} (\text{here}) \}
\]

The array object AS\_2B (which is indeed a string) is initially in the membrane 1 and the other membranes do not have objects. The rule A → S\_1 A is applied to AS\_2B to yield S\_1AS\_2B, which is sent to the inner membrane 2. If the rule B → S\_1 B is applied in membrane 2, S\_1AS\_2S\_1B is generated and sent back to membrane 1 as the target attached to the rule is out and the process repeats but if the rule used is B → S\_1 C with target in, then S\_1AS\_2S\_1C is sent to inner membrane 3 wherein S\_1S\_1S\_2S\_1S\_1 is generated and sent to inner membrane 4. The process can be repeated to generate S\_1^nS\_2S\_1^n. In membrane 4 the first of the three sets of vertical rules is applied followed by the second, a certain number of times, the computation halting with the application of the third of the sets of vertical rules. One of the rectangular arrays generated is shown in Figure 4.1. Any premature application of C → S\_1 in membrane 3 sends S\_1AS\_2S\_1S\_1 to membrane 4 where it gets stuck as the sets of vertical rules do not have a rule for A.

\[
\begin{array}{cccccccc}
a & a & a & a & a & a & a & a \\
b & b & b & a & b & b & b & b \\
b & b & b & a & b & b & b & b \\
b & b & b & a & b & b & b & b \\
b & b & b & a & b & b & b & b \\
\end{array}
\]

\[
\begin{array}{cccccccc}
a & a & a & a & a & a & a & a \\
a & b & b & b & b & b & b & b \\
a & b & b & b & b & b & b & b \\
a & b & b & b & b & b & b & b \\
a & b & b & b & b & b & b & b \\
\end{array}
\]

**Fig. 4.1: Array of Token T**

**Fig. 4.2: Token T of a's**
The picture language generated by $\Pi_1$ consists of rectangular arrays over $a$ (Figure 4.1) describing token T (Figure 4.2) with equal "horizontal arms" when $b$ is interpreted as blank.

Now we examine the generative power of these Sequential/parallel rectangular array P systems.

**Theorem 4.3.1.**

i) $\text{S/P RAP}_4(\text{REG}) \supset \text{RML}$

ii) $\text{S/P RAP}_4(\text{CFG}) \cap \text{CFML} \neq \emptyset$

**Proof**

The proper inclusion in statement (i) follows from Example 4.3.1, as no RMG can generate picture arrays describing token T as in Fig4.2 since the horizontal “arms” of Token T are equal in length.

Given a RMG, $G = (G_1, G_2)$ the inclusion in (i) is seen by constructing a sequential / parallel rectangular array P system with just 2 membranes. The membrane structure is $[1 [2]_2 ]_1$. Initially the start symbol S of $G_1$ is in membrane 1 and there are no objects in 2. The rules in membrane 1 are the horizontal regular nonterminal rules with target *here* and horizontal regular terminal rules, with target *in*. The rules in membrane 2 are of two kinds: a set consisting of all vertical right linear nonterminal rules of $G_2$ with the target *here* and another set consisting of all vertical right linear terminal rules of $G_2$ with target *here*. It can be seen that this array P system generates the picture language generated by $G$. 
The proof of ii) is due to the fact that the set of arrays describing token T of a’s with equal horizontal “arms” can indeed be generated by a CF 2Dmatrix grammar [22].

**Theorem 4.3.2**

\[ S/PRAP_{4}(REG) \subset S/PRAP_{4}(CFG) \]

**Proof**

The inclusion is clear from the definitions. The proper inclusion can be seen as follows: Consider the Sequential / Parallel Rectangular Array P system in the class S/PRAP$_4$ (CFG)

\[ \Pi_2 = (V_1 \cup V_2, I, T, [1 [2 [3 [4 ]3 ]2 ]1, AC, \varphi, \varphi, R_1, R_2, R_3, R_4, 4]). \]

\[ V_1 = \{A, C, S_1, S_2, S_3\}, V_2 = \{S_1, S_2, S_3, a, b, c\}, I = \{S_1, S_2, S_3\}, \]

\[ T = \{a, b, c\} \]

\[ R_1 = \{A \rightarrow S_1AS_2 (in), \} \]

\[ R_2 = \{C \rightarrow S_3C (out), C \rightarrow S_3D (in)\} \]

\[ R_3 = \{A \rightarrow S_1S_2 (here), D \rightarrow S_3 (in)\} \]

\[ R_4 = \{\{S_1 \rightarrow a , S_2 \rightarrow b , S_2 \rightarrow c \} (here), \{S_1 \rightarrow a, S_2 \rightarrow b, S_1 \rightarrow c\} (here) \}

\[ S_1 \quad S_2 \quad S_3 \]

The array object (in fact a string) AC is initially in membrane 1 with other membranes being empty. The rule A \rightarrow S_1AS_2 generates S_1AS_2C which is sent to membrane 2 wherein the application of C \rightarrow S_3D generates S_1AS_2S_3D which is then sent to membrane 3. Here the string is changed to S_1S_2S_2S_3S_3.
On the other hand if the rule $C \rightarrow S_3C$ is applied instead of $C \rightarrow S_3D$, then $S_1AS_2S_3C$ is sent back to membrane and the process can be repeated. Ultimately strings of the form $S_1^nS_2^nS_3^n$ are generated in membrane 3. These strings are sent to membrane 4 wherein pictures as in Fig.4.3 are generated.

The picture language generated by $\Pi_2$ consists of rectangular arrays of three equal size blocks of $a$’s, $b$’s, $c$’s (Figure 4.3).

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

Fig. 4.3: Array of three equal size blocks of $a$’s, $b$’s, $c$’s

4.4 APPLICATION TO “KOLAM” PATTERN GENERATION

"Kolam" refers to decorative artwork drawn on the floor with the kolam drawing generally starting with a certain number pattern of points and curly lines going around these points. Classification of kolam patterns based on their generation by different array grammars has been considered by Siromoney et al [24]. The Sequential/parallel rectangular array P systems introduced here, being more powerful than the 2D matrix grammars, are suitable in generating kolam patterns that cannot be generated by regular 2D matrix grammars. The
approach for generation of kolam patterns adopts a technique referred to as Narasimhan’s method of kolam generation [24]. The kolam patterns are coded as rectangular arrays of symbols. These arrays are generated using the P systems introduced here and then substitution of the basic units of the kolam pattern takes place yielding the desired patterns.

As an illustration we consider the kolam pattern in Figure 4.4.

The kolam pattern in Figure 4.4 can be expressed as an array (Figure 4.6) using primitive patterns (Figure 4.5) [32].
The set of such kolam patterns can be generated by a Sequential/parallel rectangular array P system similar to the P system $\Pi_2$ in the proof of Theorem 4.3.2 with slight modifications but cannot be generated by any CF 2D matrix grammar as the “middle part” and the “left / right parts” have equal number of columns in the kolam.

4.5 CONCLUSION

We have introduced here a new type of array P system called S/P rectangular array P system based on 2D matrix grammars. We have exhibited the suitability of these systems for Kolam pattern generation.