CHAPTER 4

ONE-DIMENSIONAL ARRAY GRAMMARS

4.1 INTRODUCTION

Formal language theory initially dealt with string generating grammars. This was extended to array grammars describing pictures of arrays. One-dimensional array grammars are considered in the literature (Freund and Păun 1993) as string generating devices in which arrays of dimension one are treated as strings but with a difference that in rewriting such strings, only symbols at the left and right ends of a currently generated string can add symbols to the string whereas any symbol in between can be replaced only by a symbol. In case of one-dimensional array grammar \( L(REG) = L(CF) \), in contrast to the Chomsky case.

Parallel communicating grammar systems have been introduced by Csuha-$\ddot{\text{j}}$-Varju et al (1994) as a theoretical model of parallel distributed computation. A new grammar system called splicing grammar system was introduced by Dassow and Mitrana (1996), which utilizes the splicing operation introduced by Head (1987). Here splicing grammar systems with the component grammars as the one-dimensional context-free and regular array grammars are considered. The resulting splicing one-dimensional context-free array grammar system (S1d-CFAGS) is introduced to generate certain non-context-free language as well.
First, the notions of one-dimensional array grammars (Freund and Păun 1993) are recalled.

**Definition 4.1.1**

A one-dimensional (1d-array) over \( V \) is a two sided infinite row of squares marked with symbols in \( V \cup \# \), where \( \# \in V \) is the blank symbol.

That is arrays of the form \( "\#w\#" \) for some \( w \in V^+ \) with the squares marked by symbols in \( V \) being non-empty, finite and connected are only considered.

The set of all one-dimensional arrays over \( V \) is denoted by \( V^{+1} \). Any subset of \( V^{+1} \) is called a 1d array language.

**Definition 4.1.2**

The 1d-array language generated by \( G \) is \( L(G) = \{ x \in T^+ | \# S\# \Rightarrow^* x \} \). For a 1d-array \( \"\#w\#\" \) for some \( w \in V^+ \), \( str(w) \) and \( str(L) \) are defined as follows,

\[
str(w) = x \quad \text{and} \quad str(L) = \{ str(x) | x \in L \}, \ L \text{ is a 1d-array language.}
\]

**Definition 4.1.3**

A one-dimensional context-free array grammar (1d-CFAG) \( G = (N, T, S, P, \#) \), where \( N \) is the set of non-terminals, \( T \) is the set of terminals, \( S \in N \) is the start symbol, \( \# \) the blank symbol and \( P \) is a finite set of rules of the form \( \alpha \rightarrow \beta \in P \), such that \( \alpha \) contains exactly one non-terminal possibly surrounded by \( \# \)'s; \( \beta \in (N \cup T)^+ \) and \( | \alpha | = | \beta | \).
G is a one-dimensional regular array grammar (1d-REGAG) if \( P \) is a finite set of rules of the forms: \(#A \rightarrow Ba, A# \rightarrow aB, A \rightarrow a\), where \( A, B \in N, a \in T\).

**Definition 4.1.4**

For a 1d-CFAG and \( x, y \in (N \cup T)^+ \), the relation \( \Rightarrow \), called direct derivation is defined as follows: \( x \Rightarrow y \) if

1. \( x = \#^wV\#^w \), for some \( V \in (N \cup T)^+ \);
2. there is a rule \( \alpha \rightarrow \beta \in P \) such that \( \alpha \) is a substring of \( x \) and \( y \) is obtained from \( x \) by replacing \( \alpha \) in \( x \) by \( \beta \).

The reflexive transitive closure of \( \Rightarrow \) is denoted by \( \Rightarrow^* \).

A similar kind of derivation can be defined for a 1d-REGA.

### 4.2 SPICING 1D-ARRAY GRAMMAR SYSTEMS

Based on the notion of a splicing grammar system introduced by Dassow and Mitrana (1996), splicing 1d-array grammar system is defined as follows.

A splicing 1d-CF(REG) array grammar system (S1d-CF(REG)AGS) of degree \( n \) is a construct \( \Gamma = (N, T, (S_1, P_1), (S_2, P_2), \ldots, (S_n, P_n), M) \) where

1. \( N, T \) are disjoint alphabets and \( P_i, 1 \leq i \leq n \) are finite sets of 1d-CF array (or 1d-REG) grammar rules over \( N \cup T \cup \# \).
2. \( M \) is a finite subset of \( (N \cup T)^* \{\_\}(N \cup T)^* \{\_\}(N \cup T)^* \) with \( \_\), \$ two distinct symbols not in \( N \cup T \). Each element of \( M \) is a splicing rule.
The set $P, 1 \leq i \leq n$ are called the components of $\Gamma$. Here, 1d-array grammars of the form $G_i = (N, T, S_i, P_i, \#), 1 \leq i \leq n$ are considered. A configuration means an $n$-tuple consisting of 1d-arrays over $N \cup T$.

For two configurations,

$$x = (x_1, x_2, \ldots, x_n), \quad x_i = \text{"}w_i\text{"}, \quad w_i \in (N \cup T)^+ N(N \cup T)^*, \quad 1 \leq i \leq n$$

$$y = (y_1, y_2, \ldots, y_n), \quad y_i = \text{"}v_i\text{"}, \quad v_i \in (N \cup T)^*, \quad 1 \leq i \leq n$$

Define $x \Rightarrow \Gamma y$ or $x \Rightarrow y$ if and only if any of the following two conditions holds:

i) for each $1 \leq i \leq n$, $x_i \Rightarrow_{P_i} y_i,$

ii) there exist $1 \leq i, j \leq n$ such that

\[
\begin{align*}
    x_i &= x'_i \alpha \beta x_i'', \\
    x_j &= x'_j \gamma \delta x_j'', \\
    y_i &= x'_i \alpha \delta x_i'', \\
    y_j &= x'_j \gamma \beta x_j'',
\end{align*}
\]

For $\alpha \beta \gamma \delta \in M$, where $\alpha, \beta, \gamma, \delta$ do not contain # and $y_k = x_k$, for $k \neq i, j$.

In the derivation $x \Rightarrow \Gamma y$, in a $1d$-AGS, (i) defines a rewriting step as in a 1d-array grammar, but (ii) defines a splicing step, corresponding to a communication step in a parallel communicating grammar system. There is no priority of any of these operations over the other.

The language generated by the $i^{th}$ component is defined by

$$L_i(\Gamma) = \{ x_i \mid \text{"}S_i\text{"} \text{"}v_i\text{"} \Rightarrow^{*} (x_1, x_2, \ldots, x_n), \quad x_j \in (N \cup T)^{+1}, \quad j \neq i \}.$$
where $\Rightarrow^*$ is the reflexive and transitive closure of the relation $\Rightarrow$.

Two kinds of languages (Dassow and Mitrana 1996) can naturally be associated to a splicing 1d-array grammar system. One of them is the language generated by a single component which can be chosen as the first component. This language will be called the individual language of the system. The second associated language will be the total language, namely

$$L_t(\Gamma) = \bigcup_{i=1}^{n} L_i(\Gamma)$$

$Isags_{L_n}(X)$ and $Tsags_{L_n}(X)$ respectively denote the families of individual and total 1d-$X$ array languages generated by Splicing 1d-$X$ array grammar system for $X \in \{1d$-REGA, 1d-CFA$\}$.

**Example 4.2.1**

Consider the Splicing 1d-Regular Array Grammar System

$(1d$-REGAGS) $\Gamma_1 = (N, T, (S_1, P_1), (S_2, P_2), M),$

\[
\begin{align*}
N &= \{S_1, S_2, X, Y\} \\
T &= \{a, b\} \\
P_1 &= \{S_1\# \rightarrow aS_1, S_1\# \rightarrow aX\} \\
P_2 &= \{#S_2 \rightarrow S_2b, #S_2 \rightarrow Yb\} \\
M &= \{a \| X \$ Y \\| b\}
\end{align*}
\]

This system produces the languages

\[str(L_1(\Gamma_1)) = \{(a^nb^n | n \geq 1)\}\]

\[str(L_2(\Gamma_1)) = \emptyset\]

Here the total language is

\[str(L_t) = str(L_1(\Gamma_1))\]
Example 4.2.2
Consider the $S1d$-REGAGS
\[
\Gamma_2 = (N, T, (S_1, P_1), (S_2, P_2), (S_3, P_3), M),
\]
\[
N = \{S_1, S_2, S_3, A, X, Y, Z\}
\]
\[
T = \{a, b, c\}
\]
\[
P_1 = \{S_1\# \rightarrow aS_1, S_1\# \rightarrow aX\}
\]
\[
P_2 = \{#S_2 \rightarrow Ac, #A \rightarrow Ab, #A \rightarrow Yb\}
\]
\[
P_3 = \{#S_3 \rightarrow S_3c, #S_3 \rightarrow Zc\}
\]
\[
M = \{aXb, bYc, cZc\}
\]

This system produces the languages
\[
str(L_1(\Gamma_2)) = \{(a^n b^{n-1} c^n \mid n \geq 2) \cup (a^n b^{n-1} c \mid n \geq 1)\}
\]
\[
str(L_2(\Gamma_2)) = \emptyset
\]
\[
str(L_3(\Gamma_2)) = \emptyset
\]

Here the total language is
\[
str(L_i) = str(L_i(\Gamma_2))
\]

Example 4.2.3
Consider the splicing $1d$-CF Array Grammar System ($S1d$-CFAGS)
\[
\Gamma_3 = (N, T, (S_1, P_1), (S_2, P_2), (S_3, P_3), M),
\]
\[
N = \{S_1, S_2, S_3, A, X, Y, Z, A, D\}
\]
\[
T = \{a, b, c\}
\]
\[
P_1 = \{S_1\# \rightarrow aS_1, S_1\# \rightarrow aX\}
\]
\[
P_2 = \{#S_2 \rightarrow DbA, #D \rightarrow Db, #D \rightarrow Yb\}
\]
\[
P_3 = \{#S_3 \rightarrow S_3c, #S_3 \rightarrow Zc\}
\]
\[
M = \{aXb, bYc, cZc\}
\]

This system produces the languages
\[
str(L_1(\Gamma_3)) = \{(a^n b^n c^n \mid n \geq 2)\}
\]
\[
str(L_2(\Gamma_3)) = \emptyset
\]
\( \text{str}(L_3(\Gamma_3)) = \emptyset \)

Here the total language is
\( \text{str}(L_i) = \text{str}(L_i(\Gamma_3)) \)

**Remark 4.2.4**

Freund and Păun (1993) have proved that one-dimensional REG and CF array grammars can generate only regular string languages. This concludes that Splicing 1d-REG and 1d-CF array grammar systems can even generate some non-context free languages.

**Theorem 4.2.5**

For \( Y \in \{I, T\} \),

1. For \( X \in \{1d-REGA, 1d-CFA\} \), \( \text{REG} = \text{str}(YsagsL_i(X)) \)
2. \( \text{REG} \subset \text{str}(YsagsL_2(1d-REGA)) \subset \text{str}(YsagsL_2(1d-CFA)) \)
3. \( \text{str}(YsagsL_3(1d-REGA)) \) and \( \text{str}(YsagsL_3(1d-CFA)) \) contain non-context-free languages.

**Proof**

The first statement follows from the definitions and the results (Lemma 1 and Lemma 2) in Freund and Păun (1993) that \( \text{str}(1d-REGA) = \text{str}(1d-CFA) = \text{REG} \). The inclusions in the second statement are straight-forward and the proper inclusion \( \text{REG} \subset \text{str}(YsagsL_2(1d-REGA)) \) follows from Example 4.2.1 and the third statement is a consequence of Examples 4.2.2 and 4.2.3.

**Remark 4.2.6**

For \( Y \in \{I, T\} \), it is not known whether the inclusion \( \text{str}(YsagsL_2(1d-REGA)) \subset \text{str}(YsagsL_2(1d-CFA)) \) is proper or not.
Theorem 4.2.7

For $Y \in \{I, T\}$,

1. $\text{REG} = \text{str}(YsagsL_1(1d-CFA)) 
   \subset \text{str}(YsagsL_2(1d-CFA)) \subset \ldots \subset \text{str}(YsagsL_n(1d-CFA)) \subset \ldots$

2. $\text{REG} = \text{str}(YsagsL_1(1d-REGA)) 
   \subset \text{str}(YsagsL_2(1d-REGA)) \subset \ldots \subset \text{str}(YsagsL_n(1d-REGA)) \subset \ldots$

Proof

The inclusions are clear from the definitions.

The proper inclusion in the first statement $\text{str}(YsagsL_n-1(1d-CFA)) \subset \text{str}(YsagsL_n(1d-CFA))$ for $n > 1$ is a consequence of the fact that the $1d$-array language

$L_1 = \{ w^{k+1} a_{i+1}^{k+1} \ldots a_n^{k+1} \# w : k \geq 1 \}$

is generated by the following $S1d-CFAGS$ with $n$ components

$\Gamma_4 = (N, T, (S_1, P_1), (S_2, P_2), ..., (S_n, P_n), M)$

$N = \{ S_1, \ldots, S_n, X_2, \ldots, X_m, Y_1, \ldots, Y_n, A_1, \ldots, A_n \}$

$T = \{ a_1, \ldots, a_n \}$

$P_i = \{ S_i \# \rightarrow a_iA_1, A_i \# \rightarrow a_iA_j, A_i \# \rightarrow a_iY_1 \}$

For $2 \leq i \leq n-1$,  

$P_i = \{ S_i \# \rightarrow X_i a_iA_i, A_i \# \rightarrow a_iA_i, A_i \# \rightarrow a_iY_i \}$

$P_n = \{ S_n \# \rightarrow X_n a_nA_n, A_n \# \rightarrow a_nA_n, A_n \rightarrow a_n \}$

$M = \{ a \sqcup YS X_{i+1} \sqcup a_{i+1} : 1 \leq i \leq n-1 \}$

It can be seen that a $S1d-CFAGS$ with $n-1$ components cannot generate $L_1$.

The proper inclusion in the second statement $\text{str}(YsagsL_n-1(1d-REGA)) \subset \text{str}(YsagsL_n(1d-REGA))$ for $n > 1$ is a consequence of the fact that the $1d$-array language
\[ L_2 = \{ \# a_1^{k+1} a_2^k \ldots a_n^k \#^w : k \geq 1 \} \]

is generated by the following \( S1d-CFAGS \) with \( n \) components

\[
\Gamma_5 = (N, T, (S_1, P_1), (S_2, P_2), \ldots, (S_n, P_n), M),
\]

\[
N = \{ S_1, \ldots, S_n, Y_1, \ldots, Y_m, A_1, \ldots, A_n \}
\]

\[
T = \{ a_1, \ldots, a_m, b_2, \ldots, b_n \}
\]

\[
P_i = \{ S_i \# \rightarrow a_i A_1, A_1 \# \rightarrow a_i A_i, A_i \# \rightarrow a_i Y_1 \}
\]

For \( 2 \leq i \leq n-1 \),

\[
P_i = \{ S_i \# \rightarrow b_i A_i, A_i \# \rightarrow a_i A_i, A_i \# \rightarrow a_i Y_i \}
\]

\[
P_n = \{ S_n \# \rightarrow b_n A_n, A_n \# \rightarrow a_n A_n, A_n \rightarrow a_n \}
\]

\[
M = \{ a_i \# Y_i \# b_{i+1} \# a_{i+1} : 1 \leq i \leq n-1 \}
\]

It can be seen that a \( S1d-REGAGS \) with \( n-1 \) components cannot generate \( L_2 \).

**Remark 4.2.8**

For \( Y \in \{ I, T \} \), it is not known whether the inclusion \( \text{str}(YsagsL_n(1d-REGA)) \subseteq \text{str}(YsagsL_n(1d-CFA)) \) for \( n \geq 2 \) is proper or not.

Freund and Păun (1993) have also considered matrices of one-dimensional array grammar rules with the application of the rules of a matrix in the usual way, in the sense that the rules are applied one after another in the order in which they appear in a matrix.

The power of splicing \( 1d-CF \) array grammar system when matrix rules are allowed in the components is indicated by means of an example.

**Example 4.2.9**

Consider the Splicing \( 1d-CF \) array grammar system with matrix rules.

\[
\Gamma_6 = (N, T, (S_1, P_1), (S_2, P_2), M)
\]

\[
N = \{ S_1, S_2, A, B, D, E, X_1, X_2, Y_1, Y_2 \} \]
\(T = \{a, b, c, d\}\)

\(P_1 = \{\{S_1 \rightarrow AcB\}, \#A \rightarrow Aa, B \rightarrow aB\}, \{A \rightarrow a, B \#\rightarrow adX_1\}, \{A \rightarrow b, B \#\rightarrow bdX_2\}\) 

\(P_2 = \{\{S_2 \rightarrow DcE\}, \#D \rightarrow Da, E \#\rightarrow aE\}, \{\#D \rightarrow Y_1 a, E \rightarrow a\}, \{\# D \rightarrow Y_2 b, E \rightarrow b\}\) 

\(M = \{d \downarrow X_1 \$Y_1 \uparrow \lambda, d \downarrow X_1 \$Y_2 \uparrow \lambda, d \downarrow X_2 \$Y_1 \uparrow \lambda, d \downarrow X_2 \$Y_2 \uparrow \lambda\}\) 

\(\Gamma_6\) generates the following 1d-context-free array language 

\(L_6 = \{"^\#xcM_i(x)dy\$cM_j(y)\$^\# | x, y \in \{a, b\}^+\}\). It is of interest to note that \(str(L_6)\) cannot be generated by any 1d-CF array grammar even with matrix rules (Freund and Păun 1993).

Here splicing grammar systems (Dassow and Mitrana 2004) with component grammars as 1d-CF and REG array grammars are considered. It remains to investigate where exactly these classes lie in the Chomsky hierarchy.

4.3 SEQUENTIAL/PARALLEL ARRAY GRAMMARS

One of the rectangular array generating models is the 2d matrix grammars of Siromoney et al (1972). Here analogous to this model but incorporating the features of 1d-array grammars of Freund and Păun (1993). A two-dimensional rectangular array generating model is introduced.

Definition 4.3.1

A 2d matrix grammar is a 2-tuple \(G = (G_1, G_2)\) where \(G_1 = (H_1, I_1, P_1, S)\) is a regular, context-free grammar; \(H_1\) is a finite set of horizontal non-terminals, \(I_1 = S_1, S_2, \ldots, S_k\), a finite set of 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, $V_{2i}$ is a finite set of vertical non-terminals, $V_{2i} \cap V_{2j} = \emptyset$ for $i \neq j$, $T$ is a finite set of terminals, $P_{2i}$ is a finite set of right linear rules of the form $X \to aY$ or $X \to a$ 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 2d matrix grammar if $G_1$ is regular, context-free respectively. Derivations are defined as follows: First a string over $I_1$ is generated horizontally using the horizontal rules. Vertical derivations then proceed in parallel using the vertical phase non-terminal rules of $G_{2i}$ generating rectangular arrays over $T$ when the vertical derivation terminates using the terminal rules of the vertical phase. The set $L(G)$ consists of all $m \times n$ arrays generated by $G$. The picture language classes of regular, context-free 2d matrix grammars are denoted by $2d$-RML, $2d$-CFML respectively.

**Definition 4.3.2**

Sequential/parallel array grammar (S/PAG) is a 2-tuple $G = (G_1, G_2)$ where $G_1$ is a 1d-CFA or 1d-REGA generating in the horizontal phase, strings (over $k$ symbols, called intermediates) with a sequential rewriting; $G_2$ consists of $k$ grammars that are 1d-CFA or 1d-REGA. A row of intermediates is rewritten in the vertical phase in parallel generating picture patterns of rectangular arrays of symbols. A row is rewritten either by a set of non-terminal rules or by a set of terminal rules.

Four classes of these grammars are obtained namely $(R:R)$ S/PAG, $(R:CF)$S/PAG, $(CF : R)$S/PAG, $(CF : CF)$S/PAG according as the first or the second phase has 1d-REGA or 1d-CFA rules. The corresponding families of array languages are denoted by $(x : y)$S/PAL, where $x, y \in \{R, CF\}$. 
Example 4.3.3

The S/PAG with the following rules

\[
S \rightarrow S_2 Z, \quad Z \rightarrow S_1 Z, \quad Z \rightarrow S_1 U, \quad U \rightarrow S_2
\]

\[
C
\]

\[
S_1 \rightarrow X, \quad C \rightarrow C, \quad C \rightarrow \bullet, \quad D \rightarrow \bullet, \quad D \rightarrow \bullet,
\]

\[
\begin{array}{c}
D \\
\bullet \\
A \\
A \\
S_2 \rightarrow X, \quad A \rightarrow X, \quad A \rightarrow X, B \rightarrow X, \quad B \rightarrow X
\end{array}
\]

where \( S, Z, U \) are the non-terminals of the horizontal phase; \( S_1, S_2 \) are the intermediates; \( \bullet, X \) are the terminals of the vertical phase, generates \( H \) type patterns (not necessarily with equal number of rows above and below the middle line of \( X \)'s) as in Figure 4.1.

\[
\begin{array}{cccccc}
 & & & & & \\
X & \bullet & \bullet & \bullet & \bullet & X \\
X & \bullet & \bullet & \bullet & \bullet & X \\
X & \bullet & \bullet & \bullet & \bullet & X \\
X & X & X & X & X & X \\
X & \bullet & \bullet & \bullet & \bullet & X \\
X & \bullet & \bullet & \bullet & \bullet & X
\end{array}
\]

Figure 4.1 Picture pattern of Token \( H \)

Thorem 4.3.4

(i) \((R : R)S/PAL = (CF : R)S/PAL\)

(ii) \((R : CF)S/PAL = (CF : CF)S/PAL\)
Proof

The equalities in (i) and (ii) follow from Freund and Păun (1993), as 1d-REGA and 1d-CFA have equal string generative power. In statements (i) and (ii), the second phase has the same type of grammar in the equalities.

Theorem 4.3.5

\[ 2d-RML = (R : R)S/PAL \subset (R : CF)S/PAL \]

Proof

The equality of 2d-RML with \((R : R)S/PAL\) is due to the fact that the generative power of 1d-REGA equals the generative power of usual regular grammars (Freund and Păun 1993). The second phase of a RMG involves only regular grammars. The proper inclusion of \((R : R)S/PAL\) in \((R : CF)S/PAL\) follows as a consequence of Example 4.3.3, since picture patterns describing token \(H\) cannot be generated by any RMG (Siromoney and Siromoney 1977).

Remark 4.3.6

As a consequence of theorem 4.3.5, in contrast to the string generating one-dimensional grammars, the extension to two dimensions considered here, makes a difference when 1d-REG and 1d-CF type rules are used in the second phase of generation, whereas 1d-REGA and 1d-CFA have equal generative power.

Theorem 4.3.7

The classes \((R : CF)S/PAL\) and 2d-CFML are incomparable.

Proof

The language of picture pattern describing token I of X’s (without any proportion between the width and the height) as in Figure 4.2 is known (Siromoney et al 1972) to be a 2d-CFML, but the language cannot be generated by any \((R : CF)S/PAG\), as the first phase involves a CFL which is
non-regular of the form \( \{ S_1^n S_2 S_3^n \mid n \geq 1 \} \) where \( S_1, S_2 \) are intermediate symbols.

\[
X \quad X \quad X \quad X \quad X \quad X \quad X
\]

\[
\cdot \quad \cdot \quad \cdot \quad X \quad \cdot \quad \cdot \quad \cdot
\]

\[
\cdot \quad \cdot \quad \cdot \quad X \quad \cdot \quad \cdot \quad \cdot
\]

\[
\cdot \quad \cdot \quad \cdot \quad X \quad \cdot \quad \cdot \quad \cdot
\]

\[
\cdot \quad \cdot \quad \cdot \quad X \quad \cdot \quad \cdot \quad \cdot
\]

\[
X \quad X \quad X \quad X \quad X \quad X \quad X
\]

**Figure 4.2 Token I**

On the other hand, the picture pattern describing token \( H \) as in Example 4.3.3 is a \((R:CF)S/PAL\) but this cannot be generated by any \( 2d-CFMG \) as the second phase of a \( 2d-CFMG \) cannot maintain a row of \( X \)'s as seen in the Figure 4.1.

The application of the array grammars considered here for the generation of interesting picture patterns is indicated in the following example.

**Example 4.3.8**

The \((R:CF)S/PAG\) with the following rules generates the picture pattern as in Figure 4.4.

\[
S \to S_1 A, \quad A \to S_2 B, \quad B \to S_4 C, \quad C \to S_2
\]

\[
X \quad X
\]

\[
S_1 \to \begin{array}{llllll}
& l & , & X & \to & X \\
Y & b & , & X & \to & b, \quad Y & \to & b, \quad Y & \to & b, \quad S_1 & \to & r
\end{array}
\]

\[
X \quad X
\]

\[
Y
\]
where $s_1$, $s_2$, $s_3$, $s_4$, $p$, $r$, $l$, $d$, $f_1$, $f_2$ respectively stand for the primitive patterns given by Nagata and Thamburaj (2006) (read row-wise, the first four are saddles; the fifth is a pupil; sixth and seventh are drops; eighth is a diamond; the ninth and tenth are fans as in Figure 4.3). An array generated by this grammar is shown in Figure 4.4. The corresponding picture pattern is shown in Figure 4.5.
Figure 4.4 Array for picture pattern of Example 4.3.8

Figure 4.5 Picture pattern of Example 4.3.8

4.4 CONCLUSION

In this chapter the splicing grammar systems of Dassow and Mitrana (1996) are studied by taking the component grammars as one-dimensional array grammars of Freund and Păun (1993). This naturally gives a higher generative power for the resulting Splicing 1d-CF (REG) array grammar system. This is extended to two-dimensions on lines of the model of Siromoney et al (1972). Unlike the 1d case, the CF and Regular 2D models obtained do not have the same generative power. Also the resulting model differs from the Siromoney et al (1972) model.