

Array Grammars and *Kolam*

GIFT SIROMONEY

*Department of Statistics, Madras Christian College,
Madras, India 600059*

AND

RANI SIROMONEY AND KAMALA KRITHIVASAN

*Department of Mathematics, Madras Christian College,
Madras, India 600059*

Communicated by R. Narasimhan

Received September 15, 1973

Kolam designs of South Indian folk art are treated as examples of two-dimensional picture languages. Many of the complicated *kolam* patterns are seen to be generable by context-free array grammars. Two methods of *kolam* generation are discussed. One method is to generate the *kolam* patterns by array grammars with a finite number of primitives as terminals. The other is to generate labeled dots by array grammars and then to give a finite number of simple instructions to draw the *kolam* pattern on the framework of labeled dots.

1. INTRODUCTION

Theoretical models for generating two-dimensional arrays are proposed in [6,7]. These models describe a wide variety of interesting classes of pictures, and simple transformations of a picture such as translation, reflection, half-turn, magnification, and conjugation are obtained easily. These models are extended to n dimensions and the three-dimensional models describe crystal symmetry and growth of crystals [8]. Array automata are defined corresponding to array languages [4] and characterizations of context-free and regular matrices given in [5]. The definitions of the array models were motivated by the patterns found in *kolam* [3].

In South Indian villages, the courtyard in front of each house is decorated every morning by drawing traditional designs called the *kolam*. The decoration of the floor with *kolam* designs is carried out by young women who deftly design with pinches of rice flour held between their fingers. On festive occasions, the *kolam* designs are more elaborate and complicated. One method of covering a large area of floor space is to enlarge directly a small design but this method is seldom used. The different designs have names of their own and larger versions of the same kind of *kolam* are drawn using complicated structures. Each kind of *kolam* pattern can be treated as a two-dimensional language and a grammar can be constructed to generate it. The number of designs that can be constructed from each kind of *kolam* pattern is infinite.

In this paper we classify different types of *kolam* according to the patterns and note that most of the complicated patterns can be generated by context-free array grammars, where terminals are chosen to be compound primitives. We present two different methods. The first method is one of a syntactic approach. We regard a pattern to consist of smaller components concatenated together and the grammar rules aid in the generation of these picture languages. Just as in string languages, $a_1 \cdot \cdot \cdot a_n$ represents the concatenation (juxtaposition) of the terminals linearly; in array languages,

$$\begin{array}{ccc} a_{11} & \cdot \cdot \cdot & a_{1n} \\ \cdot \cdot \cdot & \cdot \cdot \cdot & \cdot \cdot \cdot \\ \cdot \cdot \cdot & \cdot \cdot \cdot & \cdot \cdot \cdot \\ a_{m1} & \cdot \cdot \cdot & a_{mn} \end{array}$$

denotes the concatenation in two directions (row and column) of the terminals—which themselves are compound primitives made up of straight lines and simple curves. The grammar generates the two-dimensional array of terminals, and if, in addition, the fixed distance between the neighboring terminals (both horizontal and vertical) is specified, the pattern is completely described.

The second method is to draw the dots (*pulli*) first as is done traditionally by the womenfolk. Array grammars generate classes of these labeled dots. Then a finite number of specific instructions are given as to how the *kolam* should be drawn. These instructions are simple in nature such as “join one type of labeled dot to another type of labeled dot by a straight line,” “draw a circle or arc around one type of labeled dot,” “thread a line from one type of labeled dot to another through a third type,” and so on.

Apart from the simplicity with which these patterns can be described, the advantage of these models is that a single grammar with a finite number of rules and a finite number of terminals or a finite number of instructions can generate an infinite set consisting of the same kind of *kolam* patterns of different sizes.

The preliminary section is devoted to a review of array grammars and related concepts from [6,7]. The reader is assumed to be familiar with simple notions from formal language theory [2]. In Section 3, we classify the different patterns of *kolam* into distinct types and give context-free array grammars to generate these picture classes where the terminals are compound primitives. In Section 4, we note that context-free array grammars are sufficient to generate labeled dots and additional instructions are given as to how the *kolam* should be drawn.

2. PRELIMINARIES

In this section we review some of the definitions and results in [6] and [7]. We give only the definitions of those models which are used in this paper.

Notation. Let I be an alphabet—a finite nonempty set of symbols. A matrix (or array) over I is an $m \times n$ rectangular array of symbols from I ($m, n \geq 1$).

The set of all matrices over I (including Λ) is denoted by I^{**} and $I^{++} = I^{**} - \{\Lambda\}$.

Definition 2.1. If

$$X = \begin{matrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{matrix}, Y = \begin{matrix} b_{11} & \cdots & b_{1n'} \\ \vdots & \ddots & \vdots \\ b_{m'1} & \cdots & b_{m'n'} \end{matrix},$$

the column catenation $X \oplus Y$ is defined only when $m = m'$ and is given by

$$\begin{matrix} a_{11} & \cdots & a_{1n}b_{11} & \cdots & b_{1n'} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn}b_{m1} & \cdots & b_{mn'} \end{matrix},$$

and the row catenation $X \ominus Y$ is defined only when $n = n'$ and is given by

$$\begin{matrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \\ b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m'1} & \cdots & b_{m'n} \end{matrix}.$$

We use \oplus to denote either \oplus or \ominus . Also, when there is no ambiguity and the meaning is clear, then the operator \oplus is left out.

Definition 2.2. If M and M' are two sets of matrices, the column product $M \oplus M' = \{X \oplus Y/X \text{ in } M, Y \text{ in } M'\}$ and the row product $M \ominus M' = \{X \ominus Y/X \text{ in } M, Y \text{ in } M'\}$.

Definition 2.3. If $A = [a_{ij}]$, $B = [b_{ij}]$, $a_{ij}, b_{ij} \in \{0,1\}$, then

$$A \otimes B = c_{ij}, \quad i = 1, \dots, m, \quad j = 1, \dots, n,$$

where

$$c_{ij} = \begin{cases} 0 & \text{if } a_{ij} = b_{ij} = 0, \\ 1 & \text{otherwise.} \end{cases}$$

$A \circ B$ is read as A is superimposed on B . It is immediately seen that "superimposition" is commutative and associative.

Definition 2.4. A right-linear matrix grammar (RLMG) is a two-tuple $G = (G_1, G_2)$, where $G_1 = (V_1, I_1, P_1, S)$ is a right-linear grammar (RLG) with

V_1 = a finite set of horizontal nonterminals,

I_1 = a finite set of intermediates $= \{S_1, \dots, S_k\}$,

P_1 = a finite set of right-linear production rules called horizontal production rules, and

S is the start symbol; $S \in V_1$, $V_1 \cap I_1 = \emptyset$; and

$G_2 = \cup_{i=1}^k G_{2i}$, where $G_{2i} = (V_{2i}, I_{2i}, P_{2i}, S_i)$, $i = 1, \dots, k$, are k right-linear

grammars with

- I_2 = a finite set of terminals,
- V_{2i} = a finite set of vertical nonterminals,
- S_i the start symbol, and
- P_{2i} a finite set of right-linear production rules,
- $V_{2i} \cap V_{2j} = \emptyset$ if $i \neq j$.

The derivations are obtained by first applying the horizontal productions and then the vertical productions.

The set of all matrices generated by G is defined to be $M(G) = \{m \times n \text{ arrays } [a_{ij}], i = 1, \dots, m, j = 1, \dots, n, m, n \geq 1 / S \xRightarrow[G_i]{*} S_1 \cdots S_n \downarrow_{G_i}^* [a_{ij}]\}$ and is called a regular matrix language (RML).

If $G_1, G_{21}, \dots, G_{2k}$ generate finite languages, then $M(G)$ is called a finite matrix language.

We now define a generative model whose grammars consist of array rewriting rules, which is more powerful than the matrix model. In [6,7], we have defined models corresponding to the different types of languages of the Chomskian hierarchy. Here, we review only those models which are needed for the generation of *kolam* patterns.

Definition 2.5. Let $G = (V, I, P, S)$ be an array (rewriting) grammar (AG), where $V = V_1 \cup V_2$, V_1 a finite set of nonterminals, V_2 a finite set of intermediates, I a finite set of terminals, $P = P_1 \cup P_2$, P_1 a finite set of nonterminal rules, P_2 a finite set of terminal rules, $S \in V_1$ is the start symbol, and each intermediate in V_2 generates an intermediate language.

The AG is called (CF:R)AG if

- (i) P_1 , the finite set of nonterminal rules, are ordered pairs (u, v) (written $u \rightarrow v$) with $u \in V_1$, and v in $(V_1 \cup V_2)^+$ or $(V_1 \cup V_2)_+$,
- (ii) P_2 , the finite set of terminal rules, are ordered pairs (u, v) , u in V_1 and v in I^{++} , and
- (iii) the intermediate languages are such that each intermediate in V_2 generates a regular set whose terminals are either a finite number of arrays with the same number of rows or a finite number of arrays with the same number of columns.

Notation. We write $\alpha \xRightarrow{P_i} \beta$, or $\alpha \xRightarrow{P_i^*} \beta$, when a rule in P_1 or P_2 is used in the derivation. When the meaning is clear, we omit P_1, P_2 . $\xRightarrow{*}$ is the reflexive, transitive closure of \Rightarrow .

If A is an intermediate, then the intermediate matrix language generated by A is $M_A = \{X/A \xRightarrow{*} X \in (x_1, \dots, x_p)^+, x_1, \dots, x_p \in I^{++} \text{ and have same number of rows}\}$ or $M_A = \{X/A \xRightarrow{*} X \in (x_1, \dots, x_p)_+, x_1, \dots, x_p \in I^{++} \text{ and } x_1, \dots, x_p \text{ have same number of columns}\}$.

Derivations proceed as follows: Starting with the start symbol S , the non-terminal rules are applied without any restriction just as in string grammars (except that \oplus acts both horizontally and vertically) until all the nonterminals are replaced, introducing parentheses wherever necessary, since the operator \oplus is not associative. Now replace for each intermediate A in V_2 elements

from M_A (the intermediate matrix language generated by A) subject to the conditions imposed by the row and column catenation operator. The replacements start from the innermost parenthesis and proceed outwards. The derivation comes to a dead end if the condition for row or column catenation is not satisfied.

Definition 2.6. $M = \{X/S \xrightarrow[G]{*} X, X \text{ in } I^{++}\}$ is a (context-free:regular) array language (CF:R)AL if there exists a (CF:R)AG G such that $M = M(G)$.

3. CLASSIFICATION OF KOLAM PATTERNS

In this section we classify the different kinds of *kolam* patterns according to the grammars generating them. In fact, these patterns provided the initial motivation for the definitions of array grammars. We note that they fall into three distinct classes and we shall call them type i , $i = 1, 2, 3$.

Very often when a *kolam* is drawn, it is done mainly with reference to the number and pattern of dots (*pulli*). So we illustrate the classification and hierarchy with the help of binary pictures with dots and blanks only. In all the examples that follow, B stands for blank and in the figures drawn the corresponding entry is left blank. Then we give grammars with other primitives and generate the different types of *kolam*.

Type 1. Finite Matrix Kolam

To this class belong all finite *kolam* patterns. It consists of distinct single patterns without any repetition or recursive element. Each pattern is an entity by itself and the dots for these belong to the class of finite matrix languages. Thus, *kolam* patterns of this type fall into the class of finite matrix languages. To draw the actual *kolam*, a finite set of instructions can be given as in Section 4. We illustrate with two examples drawn in Figs. 1 and 2 ("the hanging lamp" and "the sandal cup").

Since we are more interested in describing and generating infinite classes each consisting of a single kind of *kolam* pattern of different sizes, we shall consider the remaining two classes in greater detail.

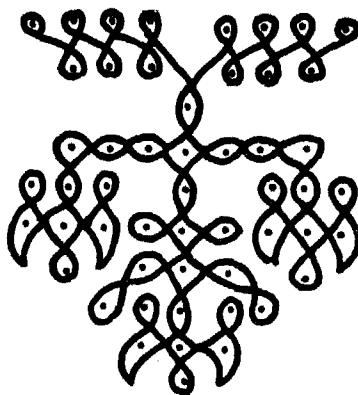


FIG. 1. The hanging lamp.

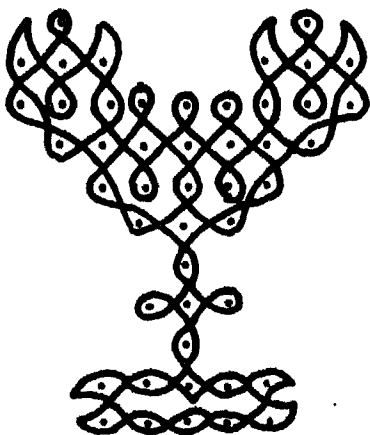


FIG. 2. The sandal cup.

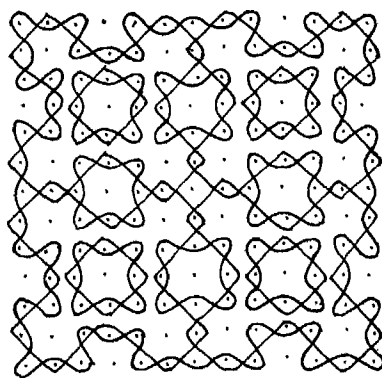


FIG. 3. The vine creeper.

Type 2. Regular Matrix Kolam

To this class belong all the *kolam* patterns where the dots (*pulli*) form $m \times n$ rectangular arrays, $m, n \geq 1$. Rectangular arrays of dots are generated by RLMG, and RLMG whose terminals are compound primitives can be constructed to generate *kolam* patterns. We have given one example ("aasanapalakai") in [6]. Further examples of this type include "the vine creeper," "the swing plank," and "the flower cradle" [3]. Figures 3, 4, and 5 show one model each of these *kolam* patterns.

Type 3. (CF: R) Array Kolam

(CF:R)AG are sufficient to generate dots for this class of *kolam*. But the *pullis* themselves fall into different types of patterns. So we further classify this class by different types of *pulli* patterns.

Type 3a. The dots for this type of *kolam* are in the form of a pyramid (or isosceles triangle) described by the infinite sequence of matrices $\{M_n/n \geq 1\}$

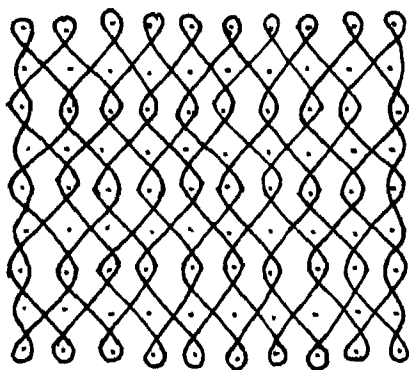


FIG. 4. The swing plank.

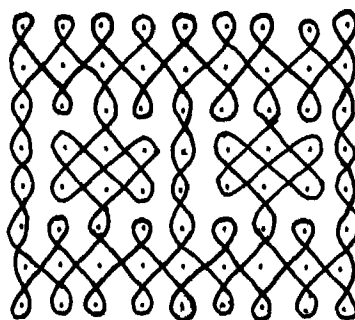


FIG. 5. The flower cradle.

where

$$M_{n+1} = Q_{n+1} \oplus X \oplus \tilde{Q}_{n+1},$$

$$Q_{n+1} = B \ominus (Q_n \oplus A).$$

$\{Q_n\}$ is generated by regular array grammars and X chosen from regular intermediate languages. Thus, this type of *kolam* is generated by (CF:R)AG.

We illustrate with the help of an example.

Example 3.1. The grammar for the dots is given by

$$G = (V, I, P, S), \text{ where } V = V_1 \cup V_2, \quad I = \{\cdot, B\}, \quad V_1 = \{S\}, \quad V_2 = \{A, C\},$$

$$P_1 = \{S \longrightarrow (A \oplus S \oplus A) \ominus C\}, \quad P_2 = \{S \longrightarrow B : B\},$$

$$L_A = \{(B)_n / n \geq 1\}, \quad L_C = \{(\cdot)^n / n \geq 1\}.$$

The *kolam parvatham* ("the mountain top") is described by $\{M_n / n \geq 1\}$ defined as follows:

$$M_{n+1} = Q_n X \tilde{Q}_n, \text{ where the grammar for } Q_n \text{ is given by } G = (V, I, P, S),$$

$$V = V_1 \cup V_2, \quad V_1 = \{S\}, \quad V_2 = \{A, C\}, \quad I = \{b, c, d, a_1, a_2, a_3, B\},$$

$$a_1 = \ominus, \quad a_2 = \emptyset, \quad a_3 = \varnothing, \quad b = \diamond, \quad c = \diamond, \quad d = \diamond,$$

$$P_1 = \left\{ S \longrightarrow C \ominus (S \oplus A) \right\}, \quad L_A = \left\{ (a_3 c) \ominus \begin{pmatrix} b & b \\ a_3 & c \end{pmatrix}_n \ominus \begin{pmatrix} b & b \\ a_2 & a_2 \end{pmatrix} / n \geq 1 \right\},$$

$$P_2 = \left\{ S \longrightarrow \begin{pmatrix} B & B & B \\ B & B & d \\ B & a_3 c \\ a_1 b & b \\ B & a_2 a_2 \end{pmatrix} \right\}, \quad L_C = \left\{ \begin{pmatrix} B \\ B \end{pmatrix}^{2n+1} \oplus \begin{pmatrix} B & B \\ B & d \end{pmatrix} / n \geq 1 \right\},$$

$$L_X = \left\{ a_3 \ominus \begin{pmatrix} b \\ c \end{pmatrix}_n \ominus \begin{pmatrix} b \\ a_2 \end{pmatrix} / n \geq 1 \right\}.$$

In this and the following examples, we have used \tilde{Q}_n to stand for the reflection of Q_n as a picture and not as a matrix. But the corresponding grammar for the matrix requires only a slight modification and can be written down similarly to G .

M_1 , M_2 , and M_3 of the *parvatham* are shown in Fig. 6.

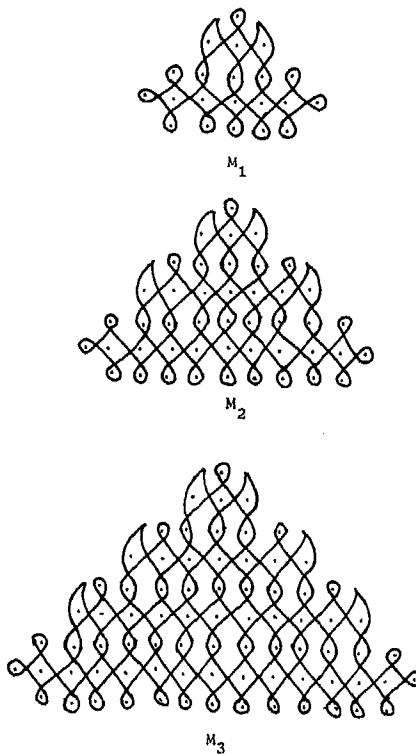
Type 3b. The dots for this class of *kolam* are in the form of a diamond obtained by combining two pyramids base to base. This class consists of infinite sequences of matrices $\{M_n / n \geq 1\}$ where $M_{n+1} = (Q_{n+1} \oplus X \oplus \tilde{Q}_{n+1}) \ominus Y \ominus (Q_{n+1} \oplus X \oplus \tilde{Q}_{n+1})$, Q_n is a (R:R)AL, and X, Y chosen from regular intermediate languages. Thus, this type of *kolam* is also generated by (CF:R)AG.

The grammar G in Example 3.1 can be modified to construct a grammar to generate Q_n in the following example.

Example 3.2. M_1 , M_2 , and M_3 of *katharikōl* (scissors) are shown in Fig. 7.

Type 3c. The dots for this type of *kolam* are in the form of a diamond (or hexagon) where the dots for any two consecutive rows are placed diagonally and not one above the other. The recursive definition of M_n is similar to that of type 3b. We illustrate with examples.

Example 3.3. This example illustrates type 3c.

FIG. 6. The mountain top. M_1, M_2, M_3 .

Let

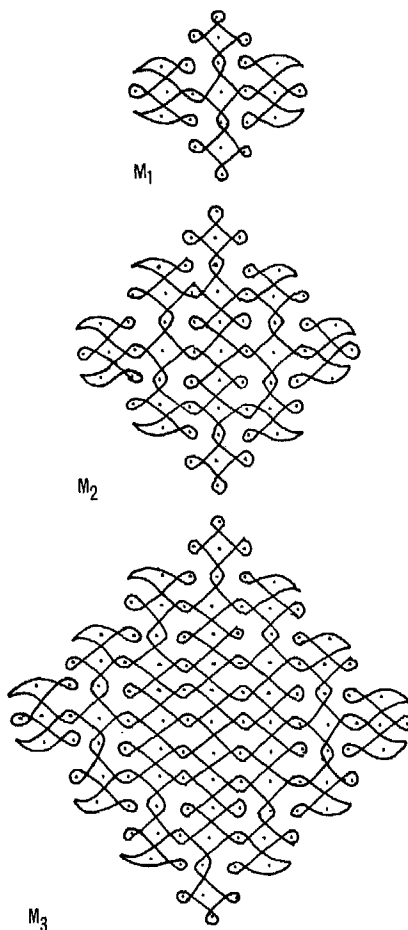
$$M_{n+1} = \begin{pmatrix} Q_n \\ Y \\ \bar{Q}_n \end{pmatrix} X \begin{pmatrix} \bar{Q}_n \\ Y \\ Q_n \end{pmatrix} \quad \text{where}$$

$$L_X = \left\{ \begin{pmatrix} a_1 B & a_2 \end{pmatrix} \ominus \begin{pmatrix} B & h & B \\ h & B & h \end{pmatrix}_n \ominus \begin{pmatrix} B & h & B \\ a_5 B & a_6 \end{pmatrix} \mid n \geq 0 \right\},$$

$$L_Y = \{a_3(B \ h)^n \cup a_3(B \ h)^n B/n \geq 0\}, \quad \text{and}$$

\bar{Q}_n , Q_n , and \tilde{Q}_n are the reflections about the rightmost vertical, base, and half-turn of \tilde{Q}_n as a picture. The grammar for Q_n is given by $G = (V, I, P, S)$, where

$$\begin{aligned} V &= V_1 \cup V_2, & V_1 &= \{S\}, & V_2 &= \{A, C\}, & I &= \{a_1, a_2, h, a_3, a_5, b, B\}, \\ P_1 &= \{S \longrightarrow (A \ominus S) \oplus C\}, & P_2 &= \{S \longrightarrow B\}, & L_A &= \{(B)^n/n \geq 1\}, \\ L_C &= \left\{ \begin{pmatrix} B \\ b \end{pmatrix} \ominus \begin{pmatrix} B \\ h \end{pmatrix}_n \begin{pmatrix} B \\ b \end{pmatrix} \ominus \begin{pmatrix} B \\ h \end{pmatrix}_n \ominus B/n \geq 0 \right\}. \end{aligned}$$


 FIG. 7. Scissors. M_1 , M_2 , M_3 .

By choosing

$$\begin{aligned}
 h &= \text{hexagon with a central dot}, & a_1 &= \text{zigzag line}, & a_2 &= \tilde{a}_1, \\
 a_3 &= \text{zigzag line}, & a_5 &= \text{zigzag line}, & a_6 &= a_5, & b &= \text{hexagon},
 \end{aligned}$$

we get the class of *kolam vilvathalam*, one member of which is given in Fig. 8. By changing h , a 's, and b to different primitives, we get different varieties of *kolam*—*māvilaikothu* (mango leaves), *pārijāthakodi* (the *parijātha* creeper), and *pāharkāi* (bitter gourds). One member of each *kolam* is drawn in Fig. 8. The grammar can be slightly modified to generate hexagonal patterns of the same *kolam*. Two such patterns are given in Fig. 9. Another hexagonal pattern is given in Fig. 10.

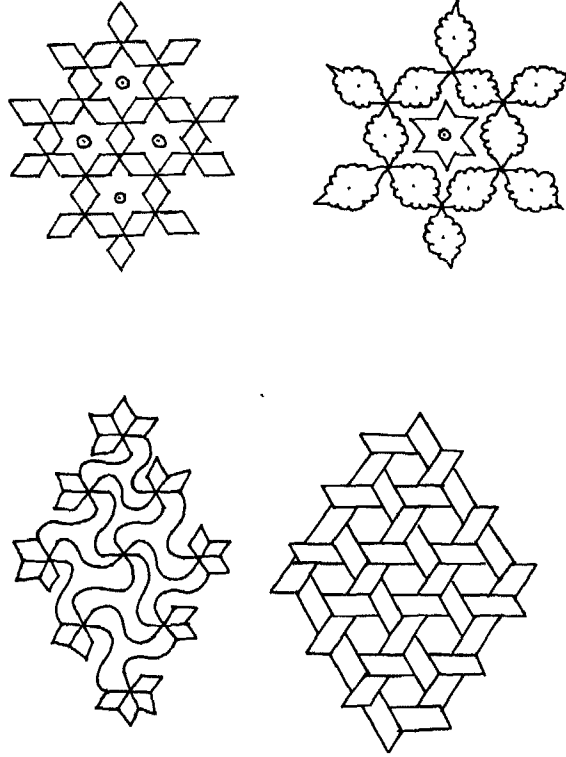


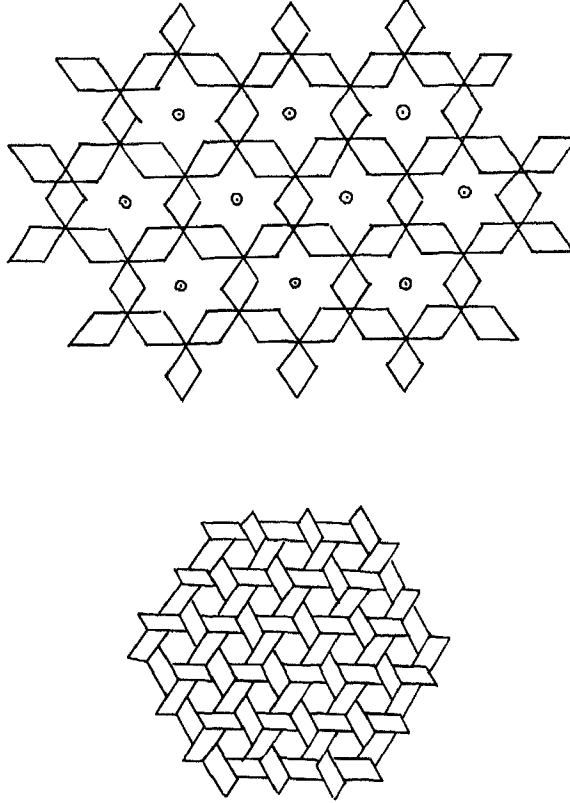
FIG. 8. (Left top) *Vilvathalam*, (right top) bitter gourds, (left bottom) the *partjatha* creeper, (right bottom) mango leaves.

Type 3d. The dots for this *kolam* are n^2 square arrays with slight variations along the corners, but since the primitives cannot be chosen to form $m \times n$ rectangular array patterns, this class cannot be generated by RLMG but belongs to (CF:R)AL. The sequence $\{M_n/n \geq 1\}$ is such that M_{n+1} contains M_n (with minor variations) in its interior, but the grammar is still of type 3(b) and is (CF:R).

Example 3.4 This example illustrates type 3d. M_1 , M_2 , and M_3 of the *kooja* are drawn in Fig. 11.

Type 3e. This type is naturally described by the recursive equation $M_{n+1} = M_n \ominus (M_n \oplus X \oplus M_n) \ominus M_n$, where M_1 is a single primitive. This class is also generated by (CF:R)AG. But it has a simpler grammar when we introduce the additional operation of superimposition. In Section 2, we defined superimposition with reference to the two terminals 0 and 1. Here we take 0 to be a dot (.) or blank and 1 to be any primitive.

Example 3.5. This example illustrates type 3e. This can be generated by (CF:R)AG but the grammar is long. However, the recursive definition $M_{n+1} = M_n \ominus (M_n \oplus C \oplus M_n) \ominus M_n$ can be brought out more effectively by introducing simple superimposition rules.

FIG. 9. (Top) *Vilvathalam*, (bottom) *mango leaves*.

Let $G = (V, I, P, S)$, where $V = V_1 \cup V_2$, $V_1 = \{S_1, S_2, S_3, S_4, X_1, X_2\}$,

$$V_2 = \{A, C\}, I = \{a, b, c, B\},$$

$$L_A = \{B^n \oplus c \oplus B^n / n \geq 1\}, \quad L_C = \{(B)_n \ominus c \ominus (B)_n / n \geq 1\},$$

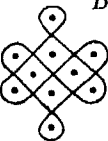

$$P_1 = \{S \longrightarrow S_1 \otimes S_2, S_1 \longrightarrow X_1 \oplus S_3 \oplus X_1,$$

$$X_1 \longrightarrow (X_1 \oplus X_1) \ominus (B)^n \ominus (X_1 \oplus X_1),$$

$$S_3 \longrightarrow S \ominus A \ominus S, \quad S_2 \longrightarrow X_2 \ominus S_4 \ominus X_2, \quad S_4 \longrightarrow S \oplus C \oplus S,$$

$$X_2 \longrightarrow (X_2 \ominus X_2) \oplus (B)_n \oplus (X_2 \ominus X_2) \subset$$

$$P_2 = \left\{ S \longrightarrow \begin{array}{ccc} B & a & B \\ b & c & b \\ B & a & B \end{array}, \quad X_1 \longrightarrow (B \ B)_7, \quad X_2 \longrightarrow \begin{pmatrix} B \\ B \end{pmatrix}^7 \right\}.$$

If we take $a =$ , $c =$ , $b = a$ turned through 90° ,

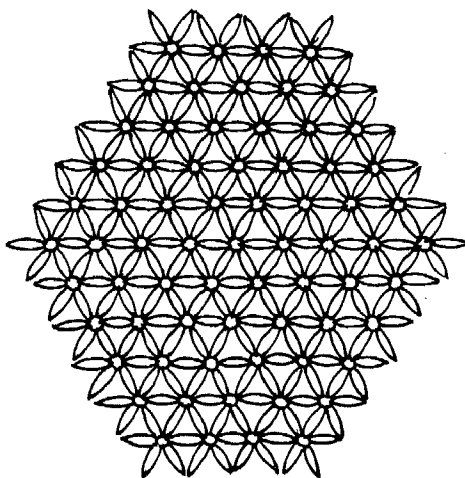
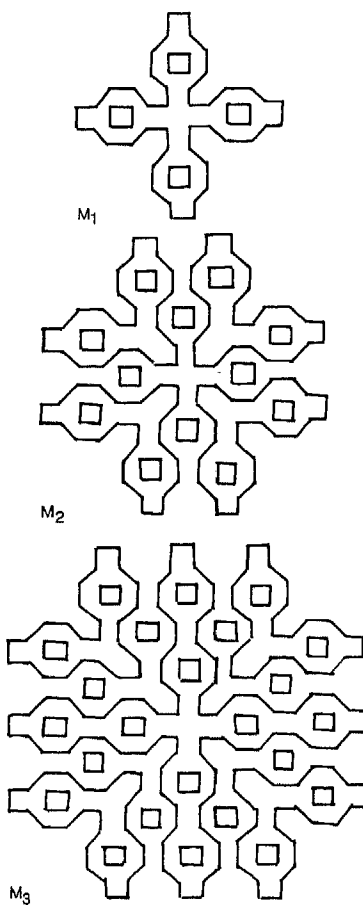


FIG. 10. Pineapple flowers.

FIG. 11. The *kooja*. M_1 , M_2 , M_3 .

we get the *kolam mittāipottalam*. If we take

$$a = b = \begin{array}{c} \diamond \\ \diagup \quad \diagdown \\ \diamond \quad \diamond \\ \diagdown \quad \diagup \\ \diamond \end{array}, \quad c = \diamond, \quad \text{we get}$$

Krishnan salangai. M_1 , M_2 , and M_3 of this *kolam* are shown in Fig. 12.

4. NARASIMHAN'S METHOD OF KOLAM GENERATION

In Section 3 we noted that (CF:R)AG are sufficient to generate the dots for most of the *kolam* patterns. Further, (CF:R)AG whose terminals are compound primitives generate the different *kolam* patterns. One may think of this as a “rubber stamp” method—where compound primitives (regarded as terminals) are printed at specified intervals (so that there is no overlapping).

However, in actual practice, the womenfolk who draw *kolam* in front of their houses every morning lay out the *pulli* first, and then straight lines or threads or circles are laid out connecting or entwining these points. In this section we show that it is possible to simulate this kind of generation also provided we use grammars to generate labeled dots first and then give simple instructions as to how to join the dots by a straight line or a thread to draw

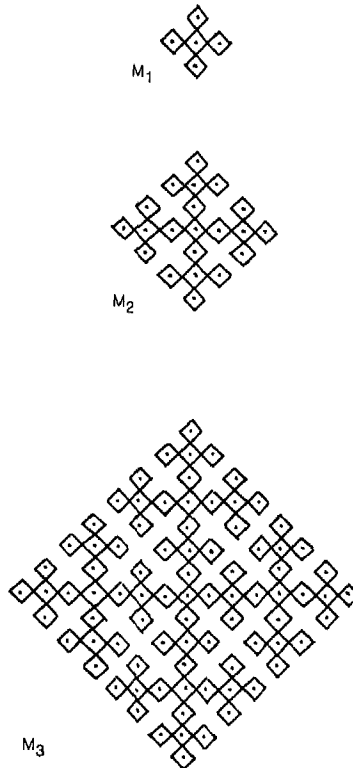


FIG. 12. Anklets of Krishna. M_1 , M_2 , M_3 .

circles or other curves around them. We note that in this case also, the same class of grammars generate the labeled dots of patterns as the unlabeled ones given in Section 3. We illustrate with examples.

Example 4.1. This example illustrates type 3a. The *kolam parvatham* of Example 3.1 is given in Fig. 6. We note that a (CF:R)AG can be given to generate the labeled dots. In addition, the following finite set of instructions is given to generate the *kolam* in Fig. 13.

- (1) Join two ∇ dots by an arc through a \blacktriangledown dot around a \triangle dot.
- (2) Join two nearest \blacktriangledown dots by straight lines (to form diamonds around \diamond dots).
- (3) Join \blacktriangledown dots around a \blacksquare dot by two arcs (say, a C-curve and its reflection).
- (4) Join two \blacktriangledown dots through a \square dot by a thread. Similarly for the *kolam katharikōl* (Fig. 7), the labeled dots can be generated and a set of instructions given.

Example 4.2. A (CF:R)AG is given for generating the labeled dots M for the *kolam vilvathalam* (Fig. 8).

$$M = \begin{pmatrix} Q \\ C \\ Q \end{pmatrix} D \begin{pmatrix} \tilde{Q} \\ \tilde{C} \\ \tilde{Q} \end{pmatrix} \quad \text{where}$$

$$L_C = \{B B (\nabla B \nabla B \blacksquare B)^n\} \cup \{B B (\nabla B \nabla B \blacksquare B)^n \nabla B \nabla B / n \geq 1\},$$

$$L_D = \left\{ \begin{pmatrix} B \blacktriangledown B \\ \nabla B \nabla \\ B \blacktriangledown B \\ \nabla B \nabla \\ B \blacksquare B \\ \nabla B \nabla \end{pmatrix}_n \ominus \begin{pmatrix} B \blacktriangledown B \\ \nabla B \nabla \\ B \blacktriangledown B \end{pmatrix} / n \geq 1 \right\}.$$

Q is generated by the (R:R)AG $G = (V, I, P, S)$, where

$$V = V_1 \cup V_2, \quad V_1 = \{S\}, \quad V_2 = \{E, F\}, \quad I = \{B, \nabla, \blacktriangledown, \blacksquare\},$$

$$P_1 = \{S \longrightarrow (F \oplus S) \ominus E\},$$

$$L_F = \{(B B B)_n / n \geq 1\},$$

$$P_2 = \left\{ S \longrightarrow \begin{pmatrix} B B B B B \\ B B B B B \\ \blacktriangledown B \nabla B \nabla \\ B \nabla B \blacktriangledown B \end{pmatrix} \right\},$$

$$L_E = \left\{ \begin{pmatrix} B B B B \\ \blacktriangledown B \nabla B \\ B \nabla B \blacktriangledown \end{pmatrix} \begin{pmatrix} B \nabla B \nabla B \blacksquare B \\ \nabla B \blacktriangledown B \nabla B \\ B \nabla B \nabla B \blacktriangledown \end{pmatrix}_n \nabla B / n \geq 0 \right\} \cup \left\{ \begin{pmatrix} B B B B \\ \blacktriangledown B \nabla B \\ B \nabla B \blacktriangledown \end{pmatrix} \begin{pmatrix} B \nabla B \nabla B \blacksquare B \\ \nabla B \blacktriangledown B \nabla B \\ B \nabla B \nabla B \blacktriangledown \end{pmatrix}_n \nabla B \blacktriangledown B / n \geq 0 \right\}.$$

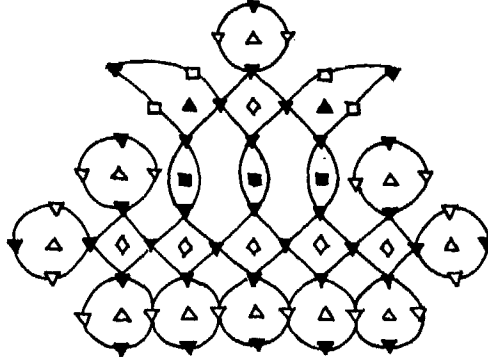


FIG. 13. The mountain top.

The set of instructions is as follows:

- (1) Join a \blacktriangledown dot to the nearest ∇ dot (The six nearest ∇ neighbors for interior \blacktriangledown dots and two nearest ∇ neighbors for exterior \blacktriangledown dots).
- (2) Draw a small circle around \blacksquare dots. One member of M (labeled dots and the *kolam*) is given in Fig. 14.

Example 4.3. We give the grammar for the labeled dots and a set of instructions for the *kolam māvilaikothu* (Fig. 8).

$$M = \begin{pmatrix} Q \\ C \\ R \end{pmatrix} D \begin{pmatrix} Q' \\ C' \\ R' \end{pmatrix} \quad \text{where}$$

$$L_C = \{B(\blacktriangle B \blacksquare B \blacklozenge B)^n \blacktriangle B \blacksquare B/n \geq 1\} \cup \{B(\blacktriangle B \blacksquare B \blacklozenge B)^n \blacktriangle/n \geq 1\},$$

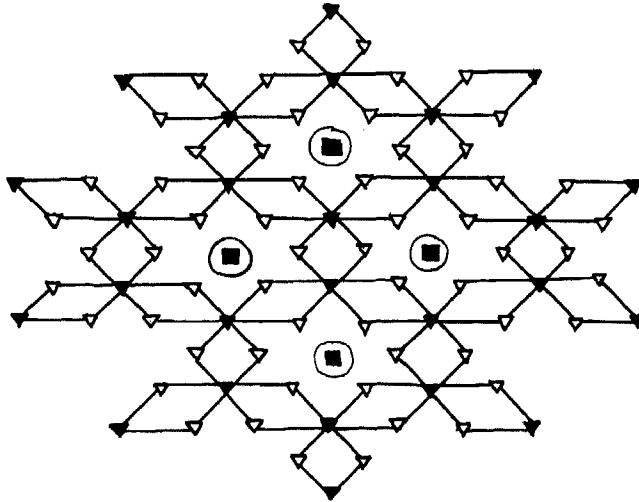


FIG. 14. Vilvathalam.

$$L_D = \left\{ \begin{array}{c} B \\ \blacktriangledown \\ B \\ \blacklozenge \\ B \\ \blacktriangledown \\ B \\ n \end{array} \right\} \quad n \geq 1 .$$

We shall give the (R:R)AG to generate Q . The grammars for Q' , R , R' can be similarly given.

$$G = (V, I, P, S), \quad V = V_1 \cup V_2, \quad I = \{\blacktriangledown, \blacktriangle, \nabla, \triangle, \blacklozenge, \blacksquare, \square, \diamond, B\},$$

$$V_1 = \{S\}, \quad V_2 = \{E, F\}, \quad P_1 = \{S \longrightarrow (F \oplus S) \ominus E\},$$

$$P_2 = S \longrightarrow \begin{array}{c} B \ B \ B \ B \ B \\ B \ B \ B \ B \ B \\ B \ \diamond \ B \ \triangle \ B \\ B \ B \ \blacktriangledown \ B \ \nabla \end{array} ,$$

$$L_E = \left\{ \begin{pmatrix} B & B & B \\ B & \diamond & B \\ B & B & \blacktriangledown \end{pmatrix} \begin{pmatrix} B & \blacktriangle & B & \blacksquare & B & \blacklozenge \\ \triangle & B & \blacktriangledown & B & \diamond & B \end{pmatrix}^n \begin{pmatrix} B & \blacktriangle & B & \blacksquare & B \\ \triangle & B & \blacktriangledown & B & \diamond \\ B & \nabla & B & \square & B \end{pmatrix} / n \geq a \right\} \cup \left\{ \begin{pmatrix} B & B & B \\ B & \diamond & B \\ B & B & \blacktriangledown \end{pmatrix} \begin{pmatrix} B & \blacktriangle & B & \blacksquare & B & \blacklozenge \\ \triangle & B & \blacktriangledown & B & \diamond & B \\ B & \nabla & B & \square & B & \blacktriangledown \end{pmatrix}^n \begin{pmatrix} B & \blacktriangle \\ \triangle & B \\ B & \nabla \end{pmatrix} / n \geq a \right\},$$

$$L_F = \{(B \ B \ B)_n / n \geq 1\}.$$

The set of instructions is:

- (1) Draw circles around \blacklozenge dots.
- (2) Join a \diamond dot to a \blacktriangledown dot through a \triangle dot.
- (3) Join a \triangle dot to a \blacktriangledown dot through a \blacktriangle dot.
- (4) Join a \blacktriangle dot to a \blacktriangledown dot through a \square dot.
- (5) Join a \square dot to a \blacktriangledown dot through a ∇ dot.
- (6) Join a ∇ dot to a \blacktriangledown dot through a \blacksquare dot.
- (7) Join a \blacksquare dot to a \blacktriangledown dot through a \diamond dot.
- (8) After completing instructions 1-7, join (at the corners) \diamond , \triangle , \blacktriangle , \square , ∇ , \blacksquare dots, respectively, to the nearest \blacktriangledown dot (Fig. 15).

Example 4.4. The grammars for the labeled dots for *parijathakodi* (Fig. 8) can be similarly given. The set of instructions is:

- (1) Join a \blacktriangle dot to a \blacktriangle dot through a ∇ dot by a straight line.
- (2) Join a \triangle dot to a \triangle dot through a ∇ dot by a straight line.
- (3) Join a \blacksquare dot to a \blacksquare dot through a ∇ dot by a straight line.
- (4) Join a \blacktriangle dot to a \blacktriangle dot through three collinear \blacktriangledown dots by a thread.
- (5) Join a \triangle dot to a \triangle dot through three collinear \blacktriangledown dots by a thread.
- (6) Join a \blacksquare dot to a \blacksquare dot through three collinear \blacktriangledown dots by a thread.

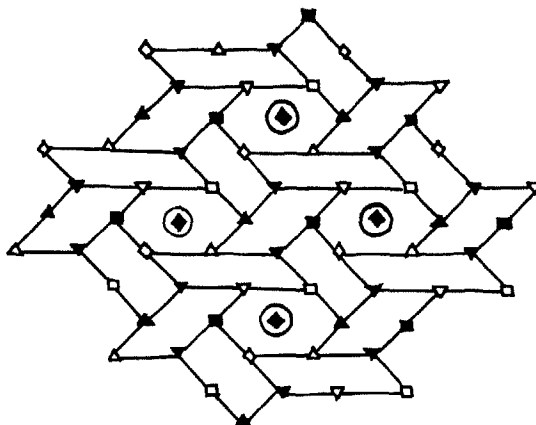


FIG. 15. Mango leaves.

(7) Join \diamond dots to the nearest Δ , \blacktriangle , \blacksquare dots by straight lines. One member is given in Fig. 16.

Example 4.5. Kooja of Fig. 10.

The labeled dots are in the form of $M = \begin{smallmatrix} Q & \bar{Q} \\ Q & \bar{Q} \end{smallmatrix}$ where the (R:R)AG for Q is given by

$$\begin{aligned} G &= (V, I, P, S), & V &= V_1 \cup V_2, & I &= \{\nabla, \diamond, \blacktriangledown, \blacksquare, \Delta, \triangle, B\}, \\ V_1 &= \{S\}, & V_2 &= \{E, F\}, & P_1 &= \{S \longrightarrow (S \oplus E) \oplus F\}, \end{aligned}$$

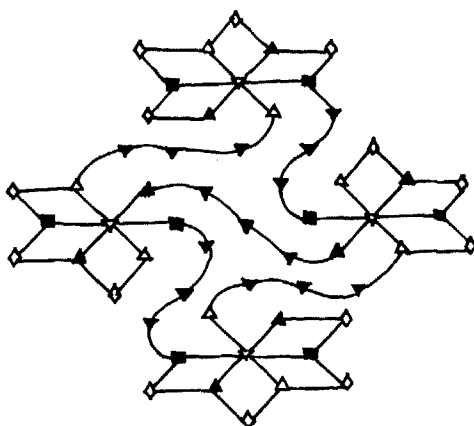


FIG. 16. The *pārijātha* creeper.

$$P_2 = S \longrightarrow \left\{ \begin{array}{cccccc} B & B & B & B & B & \nabla \\ B & B & B & B & B & \diamond \\ B & B & B & B & \square & \blacktriangledown \\ B & B & B & B & \triangle & \blacktriangledown \\ B & B & \square & \triangle & B & \blacksquare \\ \nabla & \diamond & \blacktriangledown & \blacktriangledown & \blacksquare & \diamond \end{array} \right\}$$

$$L_E = \left\{ \left(\begin{array}{c} \nabla B \\ \diamond B \\ \blacktriangledown \blacksquare \\ \blacktriangledown \triangle \\ \blacksquare \blacktriangledown \\ \diamond \end{array} \right)_n \mid n \geq 1 \right\} \cup \left\{ \left(\begin{array}{c} B \nabla \\ B \diamond \\ \blacksquare \blacktriangledown \\ \triangle \blacktriangledown \\ \blacktriangledown \blacksquare \\ \blacktriangledown \diamond \\ \square \blacktriangledown \\ \triangle \blacktriangledown \end{array} \right)_n \mid n \geq 1 \right\}$$

$$L_F = \left\{ \begin{array}{c} \nabla \diamond \\ B B \end{array} \left(\begin{array}{c} \blacktriangledown \blacktriangledown \blacksquare \diamond \\ \square \triangle \blacktriangledown \blacktriangledown \end{array} \right)^n B \square / n \geq 1 \right\} \cup \left\{ \begin{array}{c} B B \triangle \diamond \\ \nabla \diamond \blacktriangledown \blacktriangledown \end{array} \left(\begin{array}{c} \blacktriangledown \blacktriangledown \triangle \diamond \\ \triangle \diamond \blacktriangledown \blacktriangledown \end{array} \right)^n B \triangle \diamond / n \geq 1 \right\}$$

The set of instructions is:

- (1) Join a \blacktriangledown dot to the nearest \blacktriangledown dot.
- (2) Join a ∇ dot to the nearest ∇ dot.
- (3) Join a \diamond dot to the nearest \square dot.
- (4) Join a \square dot to the nearest \triangle dot.
- (5) Join a \triangle dot to the nearest \blacksquare dot.
- (6) Join a \blacksquare dot to the nearest \diamond dot.

One member is given in Fig. 17.

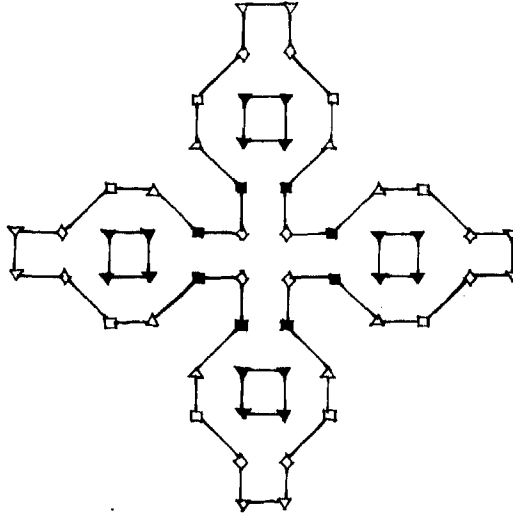


FIG. 17. The *kooja*.

Example 4.5. This example illustrates the *kolam* in Fig. 12. The grammar for the dots is the same as in Example 3.5. The instruction is to draw a (vertical) diamond around each point. The size of the diamond (square turned through 45°) is such that if d is the distance between two points, the side of each square is $d/2^{1/2}$.

5. Conclusion

Most of the interesting classes of *kolam* patterns are described by context-free array grammars. The operation of insertion on string languages [1] is extended to insertion on array languages and it is shown that this new operator yields new picture classes not generable by the earlier models [4]. We find that with the help of the "insertion" operator, a new class of *kolam* patterns not generable by CF array grammars is generated.

In addition to the study of the generation of *kolam* patterns, the enumeration of the number of different patterns that can be drawn with the same number of *pullis* and the number of closed curves in a single pattern seems to be interesting. This study can be done with reference to a particular type (Fig. 18) of *kolam* called the *hirudaya kamalam* (lotus of the heart). Several rays consisting of several *pullis* are drawn emanating from the center. The *kolam* is drawn by joining points from one ray to the next. It is interesting to note that we can enumerate the number of possible closed curves that can be drawn and the number of possible *hirudaya kamalam* making use of elementary results in number theory.

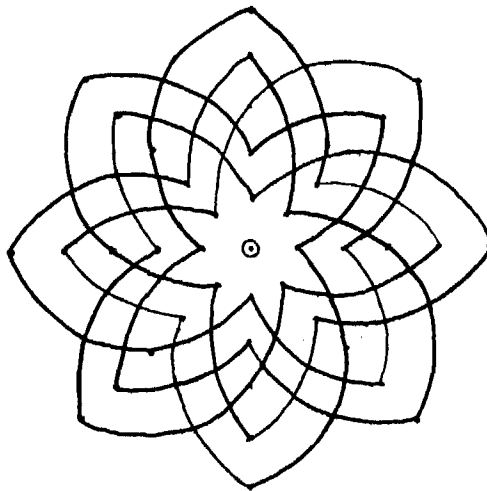


FIG. 18. Lotus of the heart.

ACKNOWLEDGMENT

The authors thank Professor R. Narasimhan for suggesting an alternative method of generating *kolam* patterns discussed in Section 4 of this paper and Mr. S. Govindarajo for the diagrams.

REFERENCES

1. S. A. GREIBACH, Syntactic operators on full semi-AFLs, *J. Comput. System Sci.* **6**, 1972, 30-76.
2. J. E. HOPCROFT AND J. ULLMAN, *Formal Languages and Their Relation to Automata*, Addison-Wesley, Reading, MA, 1969.
3. P. V. JAGADISA AYYAR, *Kolam Series*, Oxford University Press, Madras, 1937.
4. K. KRITHIVASAN AND R. SIROMONEY, Array automata and operations on array languages, *Internat. J. Comput. Math.* (to appear).
5. K. KRITHIVASAN AND R. SIROMONEY, Characterizations of regular and context-free matrices, *Internat. J. Comput. Math.* (to appear).
6. G. SIROMONEY, R. SIROMONEY, AND K. KRITHIVASAN, Abstract families of matrices and picture languages, *Computer Graphics and Image Processing* **1**, 1972, 284-307.
7. G. SIROMONEY, R. SIROMONEY, AND K. KRITHIVASAN, Picture languages with array rewriting rules, *Information and Control* **22**, 1973, 447-470.
8. R. SIROMONEY, K. KRITHIVASAN, AND G. SIROMONEY, n -Dimensional array languages and description of crystal symmetry I and II, *Proc. Indian Acad. Sci.*, **78**, 1973, 72-88, 130-139.