Bag Context Shape Grammars

Blessing Ogbuokiri *Member, IAENG,* and Mpho Raborife

***Abstract*—Shape grammars have become established as a method of generating designs (images), especially in architecture, engineering and product design. Most shape grammar systems generate images in a way that is not always regulated. This is because they are context free. As such, the application of their rules is not always controlled. In this paper, we introduce Bag Context Shape Grammars (BCSGs), for the generation of images in a regulated manner. The BCSGs are also context free, but the application of a rule is controlled by a special vector of integers called the bag, which changes during a derivation. This paper goes on to prove that every puzzle grammar with permitting features can be converted to a Bag Context Shape Grammar (BCSG). Further, it demonstrates the conversion process with examples. Additionally, this paper considers a set of images and demonstrates that BCSGs can generate a set of images with fewer variables and rules. These BCSGs could offer a wide range of application areas such as the generation of distractor (similar) images for visual password systems or scheme.**

***Index Terms*—Formal Language, Shape Grammar, Bag Context Grammar, Image Generation.**

I. Introduction

FORMAL grammars were first introduced by Post and Mil and later improved by Thue and other researchers [1]. However, full use of formal grammars and languages did not begin until the mid–1950s [2]. The general system for representing languages is based on the formal notions of grammars. A grammar is a finite non-empty set of production rules with nonterminal symbols, terminal symbols and the start symbol [3]. Formal grammars can be classified as; unrestricted, context sensitive, context free and regular grammars [4], [2], [5]. Image generating grammars are developed based on these classes of formal grammars.

There are many image generating grammars that exist and more are still being developed. Such grammars include but are not limited to bag context picture grammars [6], random context picture grammars [7], [8], tree based picture grammars [9], [10], Iterated Function Systems [11], L-systems [12], puzzle grammars [7], [13], array grammars [7], [14], chain code picture grammars [15], [7], collage grammars [7], [16], and shape grammars [17], etc. In this paper, we focus on shape grammars. A shape grammar uses shapes and spatial rules to generate images [17], [18].

Shape grammars belong to the context free class of formal grammars, and were first introduced in 1971 by Stiny and Gips [17]. They presented a formalism (prescribed logical form) for the generation and arrangement of a class of geometric paintings with shape grammars. These paintings are the material representation of two dimensional shapes generated by shape grammars, having some algorithmic specifications in terms of recursive schemata as the basic formal component. Shapes are defined in the shape grammar formalism as labelled shapes and parameterized labelled shapes. Each such formalism defines the language of the grammar [19].

According to [17], a shape grammar consists of set of nonterminal shapes, set of terminal shapes, the starting shape and the set of productions or rules. A rule consists of two shapes one on the left and another on the right side of the rule. During derivation, images are generated from the shape grammar by beginning with the starting shape and recursively applying the shape rules without restriction or control. The shape grammar rules are applied by identifying the part of the shape that is similar to the left side of the rule in terms of both nonterminal and terminal shapes and replace it. The output of this application is the given shape with the right side of the rule substituted in the shape for an occurrence of the left side of the rule. When a terminal shape is added during the derivation it cannot be replaced, thus, the generation process is terminated when no rule in the grammar can be applied or when the shape is filled with the terminals only.

Shape grammars have been applied mostly in architectural building plans [20], decorative art [21], engineering [22], [23] and product designs [24], [25]. The purpose of shape grammars is to transform or to produce shapes (designs) from the premise of an existing one [26]. Moreover, there are three processes of transforming shape grammars [27]. They are; addition, subtraction, and substitution. Addition in shape grammars is where rules are added to the grammars. Subtraction is where a rule is deleted from the grammars, while substitution involves changing of the constructive mechanism of the grammar rule [28]. Different designs can be defined in different styles by applying addition, subtraction or substitution rules to the original rule [28], [29].

The use of shape grammars for image generation does not always produce images in a regulated manner. Some generate an infinite number of images without considering their similarities. As a result, these grammars may generate images that are entirely different from each other, or they may generate images, that are too small in some parts, using the same grammar. This is because they are context free and as such, results in the application of a rule that is not controlled. Furthermore, shape grammar rules are designed with a specific shape in mind and it may not be possible to use any other shape to implement images during rendering and after derivation.

However, research has shown that control or regulation can be added to the context free grammar rules and to the derivation process using different techniques such as Bag Context (BC) [30], as well as random context [31], etc. This paper therefore focuses on bag context because it is said to be a technique used to control when a context free grammar rule

Manuscript received December 6th, 2018; revised November 1st, 2019.

Blessing Ogbuokiri is a Ph.D. Candidate at the School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa. E-mail: ogbuokiriblessing@gmail.com.

Mpho Raborife is a Senior Lecturer at the Department of Applied Information Systems, University of Johannesburg, South Africa. E-mail: mraborife@uj.ac.za.

can be applied during derivation. BC controls the application of a rule by a special vector of integers called the bag. The bag changes during the derivation of an image. A rule in the grammar can only be applied if the bag at that point is within the range defined by the lower and upper limits of the rule. When a rule is applied, it causes the bag's values to change by adding the bag adjustment, which is part of the rule. This type of technique, when added to shape grammar rules, can also be used to generate an infinite number of images in a regulated manner.

In this paper, we introduce a new shape grammar class, called Bag Context Shape Grammar (BCSG) for the generation of images in a regulated manner. This idea was first proposed by S. Ewert (Personal communication, University of the Witwatersrand). BCSGs belong to the class of shape grammars that use context to control the application of context free rules.

The main contribution of this paper is to:

- formally define BCSGs and show that bag context can be added to shape grammar rules to generate images in a controlled manner.
- illustrate these grammars with some examples.
- compare BCSGs to a recent development in puzzle grammars with permitting features.

The remaining part of this paper is organised as follows: Section II presents the definition of bag context shape grammars, and some examples to illustrate how bag context shape grammars work. We compare BCSGs to the recent development in puzzle grammars with permitting features in Section III. Section IV presents the power of bag context shape grammars. Finally, Section V is the conclusion.

Next is the definition of bag context shape grammars in Section II.

II. Bag Context Shape Grammars

In this section, we first define a new additive shape grammar class that can allow the use of any shape to be used to implement images during rendering and after a derivation. By additive, we mean the application of a rule is done by building upon the axiom during the generative process. We give an example to demonstrate this notion. Then we formally define the bag context shape grammars and give some examples to illustrate the notion. Firstly, we present some notation and terms used in this section.

A. Terms and Notation

The symbol $\mathbb{N}$ represents the set of natural numbers $\{0, 1, 2, \dots\}$. $\mathbb{N}_+$ represents the set $\{1, 2, \dots\}$. For $k \in \mathbb{N}_+$, the set $\{1, 2, \dots, k\}$ is denoted by $[k]$. $\mathbb{Z}$ represents the set of integers $\{\dots, -2, -1, 0, 1, 2, \dots\}$, then $\mathbb{Z}^2 = \{(x, y) \mid x, y \in \mathbb{Z}\}$. $\mathbb{Z}_\infty$ represents $\mathbb{Z} \cup \{-\infty, \infty\}$. If $I = [k]$, then elements of $\mathbb{Z}_\infty^I$ are written as $k-$tuples. If we have a vector of the form $(k, k, \dots, k)$, then we denote it by **k**. For example, the vector $(3, 3, \dots, 3)$ will be denoted by **3**.

A point, (x, y) in $\mathbb{Z}^2$ is a position determined by a pair (x, y), where $x, y \in \mathbb{Z}$. We denote a point by p. A line, l, is the shortest distance between two points [32]. When we draw lines in geometry, we use an arrow at each end to show that it extends infinitely. In this part, a line segment, $\bar{l}$, is determined by any pair of two distinct points (p_1, p_2) on a line together with all the points of the line between p_1 and p_2, where p_1 and p_2 are called the endpoints. A shape, σ, is defined by the area enclosed by a finite set of line segments. A label can be denoted by any of the lowercase or uppercase letters, a–z and A–Z, respectively.

A labelled shape is a pair (A, σ), where A is the label of the shape taken from the uppercase or lowercase letters and the symbol σ is as defined above. A grid is a coordinate plane consisting of a space of small squares, with horizontal (x) axis and vertical (y) axis, see Fig. 1. Every labelled shape is presented in a unit square on the grid. The initial shape is usually labelled S. The labelled shape (A, σ), is denoted by $(A, (x, y))$ in the rest of this paper. Where $(x, y) \in \mathbb{Z}^2$ denotes the lower left hand corner of the unit square the shape, σ, will be drawn.

An additive shape grammar rule is commonly expressed in the form in Rule 1,

$$(A, (x, y)) \longrightarrow \{(A_1, (x_1, y_1)), (A_2, (x_2, y_2)), \dots, (A_i, (x_i, y_i))\} \quad \text{(Rule 1)}$$

where A is a variable (uppercase label) and $(x, y) \in \mathbb{Z}^2$ is as defined above. The arrow "$\longrightarrow$" is interpreted as "transformed to", $A_1, A_2 \dots A_i$ represent variable(s) and terminal(s) (lowercase labels), for $i \in \mathbb{N}_+$ and (x_1, y_2), (x_1, y_2), $\dots$, $(x_i, y_i) \in \mathbb{Z}^2$.

The interpretation is as follows: a rule as in Rule 1 can be applied to a developing image if the developing image contains the variable A. Then, we take the set difference of the developing image and the variable A and take the set union of the developing image and $\{(A_1, (x_1, y_1)), (A_2, (x_2, y_2)), \dots, (A_i, (x_i, y_i))\}$.

The number (n) of times a rule (r) is applied during the generative process is denoted by $r^{(n)}$. For instance, $2^{(5)}$ means that rule 2 is applied five times. The operations of shape union and difference treat shapes in the same basic way as the set theoretic operations of union and difference treat sets.

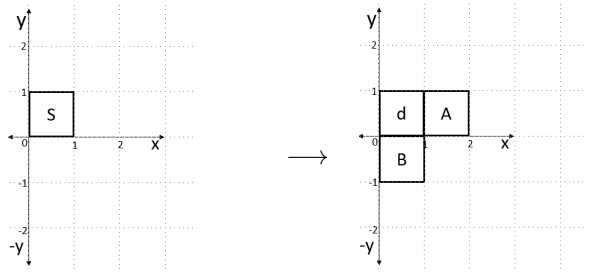
Next, we demonstrate how an additive shape grammar rule is rendered graphically. Let

$(S, (x, y)) \longrightarrow \{(d, (x, y)), (A, (x+1, y)), (B, (x, y-1))\}$

be a rule, that is also the same as

$$(S, (0, 0)) \longrightarrow \{(d, (0, 0)), (A, (1, 0)), (B, (0, -1))\},$$

when $x, y = 0$. The rule above will produce Fig. 1 when rendered using square shapes.



(a) (b)

Fig. 1: A simple shape grammar rule on the grid.

For visualisation purposes, every terminal is associated with a shape and the shape is filled with a chosen colour. The image is rendered to any size according to what is needed.

B. Formal Definition of Additive Shape Grammars

The definition of additive shape grammars is motivated by definition of shape grammars in [19], and has been modified where appropriate.

Definition 1 *An Additive Shape Grammar (ASG), $G = (V_{\mathrm{M}}, V_{\mathrm{T}}, R, (S, (x, y)))$, has a finite alphabet V of labels, consisting of disjoint subsets V_{M} of variables and V_{T} of terminals. R is the set of rules of the form $(A, (x, y)) \longrightarrow \{(A_1, (x_1, y_1)), (A_2, (x_2, y_2)), \ldots, (A_i, (x_i, y_i))\}$ where $A \in V_{\mathrm{M}}$, $\{A_1, A_2, \ldots, A_i\} \subseteq V$ and $(x, y), (x_1, y_1), \ldots, (x_i, y_i) \in \mathbb{Z}^2 \; \forall \; i \in \mathbb{N}_+$, $(S, (x, y))$ is the initial labelled shape, with $S \in V_{\mathrm{M}}$.*

Definition 2 *A pictorial form or evolving image is any set (composition) of labelled shapes in the plane denoted by Π. If Π is a pictorial form, we denote by $l(\Pi)$ the set of labels used in Π.*

Definition 3 *For an ASG G and pictorial forms Π and Γ, there is a derivation step from Π to Γ, if there is a rule $(s, (x, y)) \longrightarrow \{(s_1, (x_1, y_1)), (s_2, (x_2, y_2)), \ldots, (s_i, (xi, y_i))\}$ in R, where Π contains a labelled shape $(s, (x, y))$: $s \in V_M$ and Γ is $(\Pi \setminus \{(s, (x, y))\}) \cup \{(s_1, (x_1, y_1)), (s_2, (x_2, y_2)), \ldots, (s_i, (x_i, y_i))\}$: $\{s_1, s_2, \ldots, s_i\} \subseteq V$ for $i \in \mathbb{N}_+$ as usual.*

We denote the derivation step by $\Pi \Longrightarrow \Gamma$. This simply means that Γ is directly derived from Π. If there is a sequence of zero or more derivation steps from Π to Γ, then we denote that by $\Pi \Longrightarrow^* \Gamma$. We now say that Γ is derived from Π.

Definition 4 *An image is a pictorial form Π with $l(\Pi) \subseteq V_{\mathrm{T}}$.*

Definition 5 *The gallery $\mathcal{G}(G)$ generated by an additive shape grammar G is the set of images, Π, derivable from the initial shape $(S, (x, y))$, represented as: $\mathcal{G}(G) = \{\Pi : (S, (x, y)) \Longrightarrow^* \Pi\}$.*

Next, we demonstrate how additive shape grammars work in Example II.1.

Example II.1 *We want to generate the gallery of images that consist of the letter E (see Fig. 2). Let $G_{\mathrm{E}} = (V_M, V_T, R, (S, (x, y)))$, where $V_M = \{S, A, B, C, D, E\}$, $V_T = \{ d \}$, $(S, (x, y)) = (S, (0, 0))$, and R is shown in Fig. 3.*

Some of the images in $\mathcal{G}(G_{\mathrm{E}})$ when rendered using square shapes with the label d associated with dark colour are shown in Fig. 2. The images were scaled to the same size for presentation purposes as in Fig. 2. The illustration of a derivation of G_{E} is given in Fig. 4, having the lower lefthand corner of the initial shape S start at $x, y = 0$.

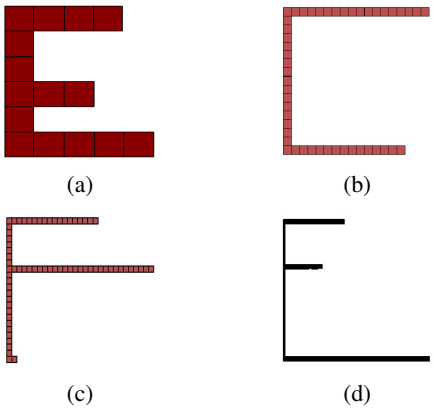


Fig. 2: Some images in $\mathcal{G}(G_{\mathrm{E}})$.

The main purpose of our type of additive shape grammars is to allow any shape to be used during image rendering after derivation. The gallery in Fig. 5 shows that different shapes can be used to implement image rendering when the derivation is complete. Some images obtained when the derivation in Fig. 4 is rendered with different shapes are shown in Fig. 5.

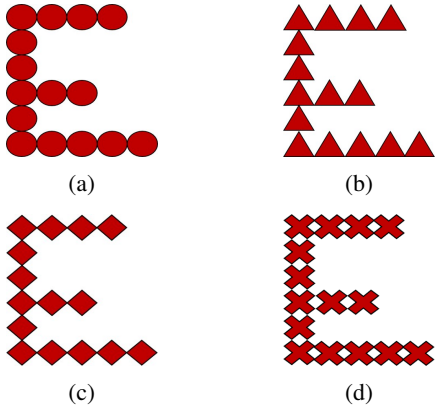


Fig. 5: Image rendering of the derivation in Fig. 4 with different shapes.

Observe that after Rule 2, any of the Rules 3–6 in Fig. 3 are applicable and can be applied at any time. The same goes for Rules 7–10 after Rule 6, and Rules 11 and 12 after Rule 8. This can be seen as a weakness, as the images in the gallery may not only contain the letter E (see Figs. 2(b) and (c)). This means that the additive shape grammars, in as much as they can allow for any shape to be used to render images after derivation (see Fig. 5), may not always generate images in a regulated manner using the same grammar. This simply means that additive shape grammars alone may not be able to generate images in a regulated manner at all times, thus, the need for the concept of bag context to be added to the additive shape grammar rules to generate similar images of choice in a controlled manner. This control to the additive shape grammar rules and to the derivation process with the help of bag context will enable us to generate a gallery of images that will contain only images of $E's$ whose legs are an equal length with the upper and lower spines the same length also. This approach is what we called BCSGs in this paper.

Informally, a bag context shape grammar may be defined as a shape grammar G in which the application of a rule is regulated by a special vector of integers called the bag β. For a rule r to be applied by G at a particular point in a derivation, the bag β must be within the range, determined by the lower limit and upper limit L_{b} and $U_{\mathrm{b}} \in \mathbb{Z}^I_\infty$, which are defined in r. The bag adjustment δ is made together with the application of the rule, that is, δ is also defined in r. Next, we present the formal definition of BCSGs in Section II-C.

C. Formal Definition of Bag Context Shape Grammars

This section is motivated by the definition of bag context tree grammars in [30], [6] and the definition of additive shape grammars in Section II-B of this paper.

Definition 6 *A BCSG is a grammar with the form $G = (V_{\mathrm{M}}, V_{\mathrm{T}}, R, (S, (x, y)), I, \beta_0)$, where V_{M}, V_{T}, and $(S, (x, y))$ are as in Definition 1, R is the set of shape grammar rules, where every rule is of the form $(A, (x, y)) \longrightarrow \{(A_1, (x_1, y_1)), (A_2, (x_2, y_2)), \ldots, (A_i, (x_i, y_i))\}$ $(L_{\mathrm{b}}, U_{\mathrm{b}} \; ; \delta)$, where $(A, (x, y)) \longrightarrow \{(A_1, (x_1, y_1)), (A_2, (x_2, y_2)), \ldots, (A_i, (x_i, y_i))\}$ is as in Definition 1, $L_{\mathrm{b}}, U_{\mathrm{b}} \in \mathbb{Z}^I_\infty$, and δ is the bag adjustment, $\delta \in \mathbb{Z}^I$. I is the finite bag index set of the form $[k]$ and β_0 is the initial bag, $\beta_0 \in \mathbb{Z}^I$.*

Definition 7 *Let a configuration be a pair (Π, β) where Π is pictorial form and β is the bag. For a BCSG G and configurations (Π, β) and (Γ, β'), there is a derivation step from (Π, β) to (Γ, β'), if there is a rule $(s, (x, y)) \longrightarrow \{(s_1, (x_1, y_1)), (s_2, (x_2, y_2)), \ldots, (s_i, (x_i, y_i))\}$ $(L_{\mathrm{b}}, U_{\mathrm{b}} \; ; \delta)$ in R, where Π contains a labelled shape $(s, (x, y))$: $s \in V_M$ with $L_{\mathrm{b}} \leq \beta \leq U_{\mathrm{b}}$. $\Gamma = (\Pi \setminus \{(s, (x, y))\}) \cup \{(s_1, (x_1, y_1)), (s_2, (x_2, y_2)), \ldots, (s_i, (x_i, y_i))\}$ and $\beta' = \beta + \delta$.*

$$
R = \left\{
\begin{array}{lcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y)),(B,(x,y-1))\} & \text{(Rule 2)}\\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\} \mid & \text{(Rule 3)}\\
 & & \{(d,(x,y))\} & \text{(Rule 4)}\\
(B,(x,y)) & \longrightarrow & \{(d,(x,y)),(B,(x,y-1))\} \mid & \text{(Rule 5)}\\
 & & \{(d,(x,y)),(C,(x+1,y)),(D,(x,y-1))\} & \text{(Rule 6)}\\
(D,(x,y)) & \longrightarrow & \{(d,(x,y)),(D,(x,y-1))\} \mid & \text{(Rule 7)}\\
 & & \{(d,(x,y)),(E,(x+1,y))\} & \text{(Rule 8)}\\
(C,(x,y)) & \longrightarrow & \{(d,(x,y)),(C,(x+1,y))\} \mid & \text{(Rule 9)}\\
 & & \{(d,(x,y))\} & \text{(Rule 10)}\\
(E,(x,y)) & \longrightarrow & \{(d,(x,y)),(E,(x+1,y))\} \mid & \text{(Rule 11)}\\
 & & \{(d,(x,y))\} & \text{(Rule 12)}
\end{array}
\right\}
$$

Fig. 3: The set of rules for G_{E} in Example II.1.

$$
\begin{aligned}
\{(S,(0,0))\} &\overset{2}{\Rightarrow} \{(d,(0,0)),(A,(1,0)),(B,(0,-1))\}\\
&\overset{3^{(2)},4}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(B,(0,-1))\}\\
&\overset{5^{(2)},6}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(C,(1,-3)),(D,(0,-4))\}\\
&\overset{9,10}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(D,(0,-4))\}\\
&\overset{7,8}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(d,(0,-4)),(d,(0,-5)),\\
&\quad (E,(1,-5))\}\\
&\overset{11^{(3)},12}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(d,(0,-4)),(d,(0,-5)),\\
&\quad (d,(1,-5)),(d,(2,-5)),(d,(3,-5)),(d,(4,-5))\}
\end{aligned}
$$

Fig. 4: A derivation of the picture in Fig. 2(a) for G_{E} in Example II.1.

We denote the derivation step by $(\Pi,\ \beta) \Longrightarrow (\Gamma,\ \beta')$. *This simply means that* $(\Gamma,\ \beta')$ *is directly derived from* $(\Pi,\ \beta)$. *If there is a sequence of zero or more derivation steps from* $(\Pi,\ \beta)$ *to* $(\Gamma,\ \beta')$, *then we denote that by* $(\Pi,\ \beta) \Longrightarrow^{*} (\Gamma,\ \beta')$. *We now say that* $(\Gamma,\ \beta')$ *is derived from* $(\Pi,\ \beta)$.

Definition 8 *An image is a pictorial form* Π *with* $l(\Pi) \subseteq V_T$ *and the bag,* β.

Definition 9 *The gallery* $\mathcal{G}(G)$ *generated by a BCSG* G *is the set of images,* Π, *derivable from the initial labelled shape* $(S,\ (x,\ y))$ *and the initial bag* β_0, *represented as:* $\mathcal{G}(G) = \{\Pi : ((S,\ (x,\ y)),\beta_0) \Longrightarrow^{*} (\Pi,\beta)$, *for some* $\beta \in \mathbb{Z}^{I}_{\infty}$ *and* $l(\Pi) \subseteq V_T\}$. *The class of all galleries generated by BCSGs is denoted by* $BCSG_{\mathrm{G}}$.

Next, we demonstrate BCSGs in Example II.2. In Example II.2, we generate images with the following: three legs of equal length and a spine. The second leg of the letter *E* to be generated divides the spine into an upper and a lower half which are the same length.

Example II.2 *We extend Example II.1 by adding bag context to generate the images in Fig. 6. Let* $G_{\mathrm{E-bag}} = (V_M, V_T, R, (S,\ (x,\ y)),\ \{1,\ 2,\ 3,\ 4\}, 0)$, *where* $V_M = \{S,\ A,\ B,\ C,\ D,\ E\ \}$, $V_T = \{\ d\ \}$, $(S,\ (x,\ y)) = (S,(0,0))$, *and* R *is shown in Fig. 7.*

Some of the images in the gallery produced by $G_{\mathrm{E-bag}}$ when rendered using square shapes with the label d associated with dark colour are shown in Fig. 6. The images were scaled to the size as in Fig. 6 for presentation purposes.

The strategy here is that the first, third, and fourth bag positions are used to control how the first, second and third legs grow. The second bag position is used to control the upper and lower spines of the letter *E*. The first (top) leg of the letter E is formed before any other part. The reason for this is that the number of times the rule that generates the top leg is applied is used to control the other legs. The upper spine is formed immediately after the top leg. The second leg is formed after the upper spine. Then the lower spine is formed followed by the third leg formation. The second leg must be exactly at half the spine which makes the upper and lower spines to be the same length. The three legs must also be the same length.

$$
R = \Bigg\{
$$

$$
\begin{aligned}
(S,(x,y)) &\longrightarrow \{(d,(x,y)),(A,(x+1,y)),(B,(x,y-1))\}(0,0;(1,1,0,0)) && \text{(Rule 13)}\\
(A,(x,y)) &\longrightarrow \{(d,(x,y)),(A,(x+1,y))\}((1,1,0,0),\infty;(1,0,0,0)) \mid && \text{(Rule 14)}\\
&\quad \{(d,(x,y))\}((2,1,0,0),(\infty,\infty,\infty,1);(-1,-1,0,1)) && \text{(Rule 15)}\\
(B,(x,y)) &\longrightarrow \{(d,(x,y)),(B,(x,y-1))\}((1,0,0,1),(\infty,\infty,\infty,1);(0,1,0,0)) \mid && \text{(Rule 16)}\\
&\quad \{(d,(x,y)),(C,(x+1,y)),(D,(x,y-1))\}\\
&\qquad ((1,2,0,1),(\infty,\infty,0,1);(0,0,0,1)) && \text{(Rule 17)}\\
(D,(x,y)) &\longrightarrow \{(d,(x,y)),(D,(x,y-1))\}((0,1,1,3),(0,\infty,\infty,3);(0,-1,0,0)) \mid && \text{(Rule 18)}\\
&\quad \{(d,(x,y)),(E,(x+1,y))\}((0,0,1,3),(\infty,0,\infty,3);(0,0,0,1)) && \text{(Rule 19)}\\
(C,(x,y)) &\longrightarrow \{(d,(x,y)),(C,(x+1,y))\}((1,1,0,2),(\infty,\infty,\infty,2);(-1,0,1,0)) \mid && \text{(Rule 20)}\\
&\quad \{(d,(x,y))\}((0,1,2,2),(0,\infty,\infty,2);(0,0,0,1)) && \text{(Rule 21)}\\
(E,(x,y)) &\longrightarrow \{(d,(x,y)),(E,(x+1,y))\}((0,0,1,4),(\infty,\infty,\infty,4);(1,0,-1,0)) \mid && \text{(Rule 22)}\\
&\quad \{(d,(x,y))\}((1,0,0,4),(\infty,\infty,0,4);0) && \text{(Rule 23)}
\end{aligned}
$$

$$
\Bigg\}
$$

Fig. 7: The set of rules for $G_{\mathrm{E-bag}}$ in Example II.2.

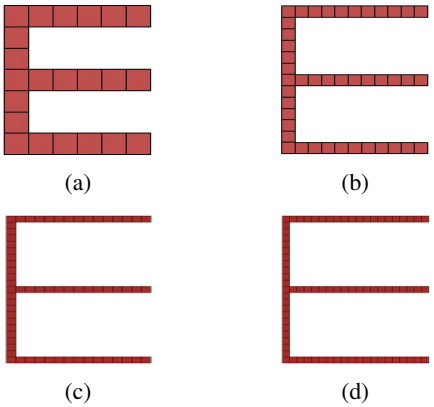


Fig. 6: Some images in $\mathcal{G}(G_{\mathrm{E-bag}})$

Next, we explain how the rules in Fig. 7 are applied to form the image in Fig. 6(a).

To achieve the strategy, the derivation starts with the initial labelled shape in the pictorial form, $\Pi = \{(S,(x,y))\}$. The application of Rule 13 of Fig. 7 replaces the initial labelled shape $(S,(x,y))$ with the shapes labelled $(d,(x,y))$, $(A,(x+1,y))$ and $(B,(x,y-1))$ in the set. The bag adjustment, (1, 1, 0, 0) is added to the bag, (0, 0, 0, 0) which is (0 + 1, 0 + 1, 0 + 0, 0 + 0) then the bag becomes (1, 1, 0, 0).

We can apply Rule 14 twice because the bag at each application is within the defined range. That is the bag is greater than or equal to the lower limit, (1, 1, 0, 0) and less than or equal to the upper limit, ∞. The bag adjustment, (1, 0, 0, 0) is added to the bag at each application of Rule 14. This is followed by the application of Rule 15 because it is also within the defined range.

At this point, the bag is (2, 0, 0, 1). We apply Rule 16 twice to form the upper spine; the bag is (2, 2, 0, 1). Rule 17 is then applied to begin the formation of the second leg and the lower spine; the bag adjustment, (0, 0, 0, 1) is added to the bag, (2, 2, 0, 1) then the bag becomes (2, 2, 0, 2).

We further apply Rule 20 twice and Rule 21 once to form the second leg. At this point, the bag adjustment, (0, 0, 0, 1) is added to the bag, (0, 2, 2, 2) the bag becomes (0, 2, 2, 3). We apply Rule 18 twice followed by Rule 19 because the bag at each application is within the defined range. At this point, the bag adjustment, (0, 0, 0, 1), is added to the bag, (0, 0, 2, 3) then the bag becomes (0, 0, 2, 4) and the lower spine is formed.

Finally, we apply Rule 22 twice and Rule 23 once to form the third leg. The bag adjustment, (0, 0, 0, 0) is added to the bag, (2, 0, 0, 4) the bag becomes (2, 0, 0, 4). At this point the picture formation is completed and the letter E is generated (see Fig. 6(a)).

We can also generate more images by repeating the process and applying Rules 14 and 16 as long as they are within the defined range at each application. This is because the number of times Rule 14 is applied determines the number of times Rules 20 and 22 will be applied and the number of times Rule 16 is applied determines the number of times Rule 18 will be applied with the help of the bag. Then, the process can be terminated with Rules 15, 21 and 23 respectively in order to form a complete picture (see Figs. 6(b)–(d)).

An illustration of one of the derivations of $G_{\mathrm{E-bag}}$ using the picture in Fig. 6(a) is given in Fig. 8. The derivation has the lower lefthand corner of the initial shape S start at $x, y = 0$. The derivation is summarised in Table I showing the operations in the bag when an adjustment is made.

Some images obtained when the derivation in Fig. 8 is rendered with different shapes are shown in Fig. 9.

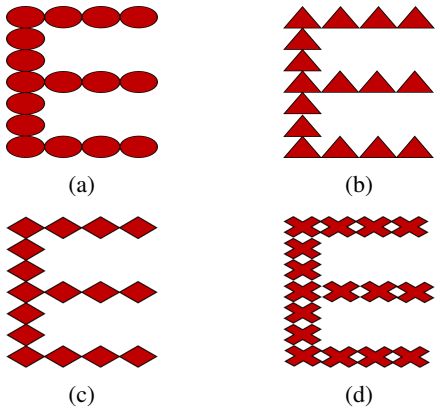


Fig. 9: Image rendering of the derivation in Fig. 8 with different shapes.

In Example II.3 we consider a gallery of images that consists of carpets where each colour fills or dominates an entire row, see Fig. 10.

Example II.3 *Let* $G_{\mathrm{Carpet-A}} = (V_M, V_T, R, S, \{1, 2, 3\}, 0)$, *where* $V_M = \{S, A, B, C\}$, $V_T = \{d, w\}$, $(S,(x,y)) = (S,(0,0))$, *and* R *is the set in Fig. 11.*

The first and second bag positions are used to control the variables $A - C$, as such, they ensure a colour dominates a row at

$$
\begin{aligned}
\{(S,(0,0))\} &\overset{13}{\Longrightarrow} \{(d,(0,0)),(A,(1,0)),(B,(0,-1))\}\\
&\overset{14^{(2)},15}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(B,(0,-1))\}\\
&\overset{16^{(2)},17}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(C,(1,-3)),(D,(0,-4))\}\\
&\overset{20^{(2)},21}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(d,(3,-3)),(D,(0,-4))\}\\
&\overset{18^{(2)},19}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(d,(3,-3)),(d,(0,-4)),\\
&\quad (d,(0,-5)),(d,(0,-6)),(E,(1,-6))\}\\
&\overset{22^{(2)},23}{\Longrightarrow} \{(d,(0,0)),(d,(1,0)),(d,(2,0)),(d,(3,0)),(d,(0,-1)),(d,(0,-2)),\\
&\quad (d,(0,-3)),(d,(1,-3)),(d,(2,-3)),(d,(3,-3)),(d,(0,-4)),\\
&\quad (d,(0,-5)),(d,(0,-6)),(d,(1,-6)),(d,(2,-6)),\\
&\quad (d,(3,-6))\}
\end{aligned}
$$

Fig. 8: A derivation of the picture in Fig. 6(a) for $G_{\mathrm{E-bag}}$.

TABLE I: The bag during the derivation in Fig. 8.

Rules	β	δ
13	0	(1, 1, 0, 0)
$14^{(2)}$	(1, 1, 0, 0)	(1, 0, 0, 0)
15	(3, 1, 0, 0)	(-1, -1, 0, 1)
$16^{(2)}$	(2, 0, 0, 1)	(0, 1, 0, 0)
17	(2, 2, 0, 1)	(0, 0, 0, 1)
$20^{(2)}$	(2, 2, 0, 2)	(-1, 0, 1, 0)
21	(0, 2, 2, 2)	(0, 0, 0, 1)
$18^{(2)}$	(0, 2, 2, 3)	(0, -1, 0, 0)
19	(0, 0, 2, 3)	(0, 0, 0, 1)
$22^{(2)}$	(0, 0, 2, 4)	(1, 0, -1, 0)
23	(2, 0, 0, 4)	0

a time. The third bag position is used to ensure the number of rows are exactly the same as the number of columns.

Some images in $\mathcal{G}$ produced by $G_{\mathrm{Carpet-A}}$ are shown in Fig. 10.

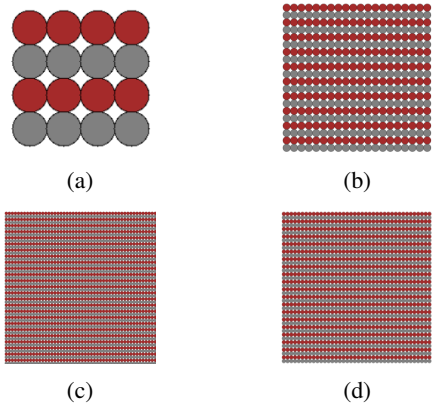


Fig. 10: Some of the images in $\mathcal{G}$ of $G_{\mathrm{Carpet-A}}$.

The strategy is that the number of times Rule 25 of Fig. 11 is applied determines the size of the image. Then only one colour is allowed to dominate a row at a time.

To achieve the strategy, the application of Rule 24 of Fig. 11 starts the derivation. Then Rule 25 is applicable as long as the bag at each application is within the defined range. That is the bag is greater than or equal to the lower limit, (1, 0, 1) and less than or equal to the upper limit, (∞, 0, ∞). The bag adjustment, (1, 0, 1) is added to the bag which automatically increments the first and third bag positions by 1 at each application. This is to control the variables, while the number of rows and columns are kept in check.

The application of Rule 26 adds the bag adjustment, (0, 0, -1) to the bag which automatically decrements the third bag position by 1. This also begins the formation of a new row. Then Rules 27 and 28 followed by Rules 30 and 31 are applicable as long as the bag at each application is within the range. This process continues until the third bag position is 0 then, any of Rule 29 or 32 of Fig. 11 is applicable to complete the formation of the image. This strategy is sufficient to generate the images in Fig. 10.

Next, we extend Example II.3 to generate a gallery of images that consists of a light colour on the upper right corner of the first dark row. See Example II.4. The number of the light colour on the lower left corner of the last dark row is twice the number of the light colour on the upper right corner of the first dark row. See Fig. 12.

Example II.4 *Let* $G_{\mathrm{Carpet-B}} = (V_M, V_T, R, S, \{1, 2, 3, 4, 5\}, 0)$, *where* $V_M = \{S, A, B, C\}$, $V_T = \{d, w\}$, $(S,(x,y)) = (S,(0,0))$, *and* R *is the set in Fig. 13.*

The strategy is that the first and second bag positions are used to control the variables A–C which help to keep colour on a row. The third bag position is used to ensure the number of rows is the same as the number of columns. The fourth and fifth bag positions are used to ensure the number of the light colour on the lower left corner of the last dark row of the image is twice the number of the light colour on the upper right corner of the first dark row of the image.

Some images in $\mathcal{G}$ produced by $G_{\mathrm{Carpet-B}}$ are shown in Fig. 12.

To achieve the strategy, the application of Rule 33 of Fig. 13 starts the derivation. Then Rule 25 is applicable as long as the bag at each application is within the defined range, this automatically adds the bag adjustment (1, 0, 1, 0, 0) to the bag at each application. Hence the application of Rule 35 of Fig. 13 enables Rules 36 and 38 to be applicable. This is to ensure that a light colour appears at the upper right corner of the first dark row.

Next, we extend Example II.4 to demonstrate how BCSGs can generate images with sub-images in Example II.5.

Example II.5 *We generate a gallery of images that consists of light or dark colour on the upper right and lower left corners. The top left corner of each image in the gallery consists of dark and light*

$$
R = \left\{
\begin{array}{lcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}(0,0;(1,0,1)) & \text{(Rule 24)}\\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}((1,0,1),(\infty,0,\infty);(1,0,1)) \mid & \text{(Rule 25)}\\
 & & \{(d,(x,y)),(B,(x,y-1))\}((2,0,2),(\infty,0,\infty);(0,0,-1)) & \text{(Rule 26)}\\
(B,(x,y)) & \longrightarrow & \{(B,(x-1,y)),(w,(x,y))\}((1,0,0),(\infty,\infty,\infty);(-1,1,0)) \mid & \text{(Rule 27)}\\
 & & \{(w,(x,y)),(C,(x,y-1))\}((0,1,1),(0,\infty,\infty);(0,0,-1)) \mid & \text{(Rule 28)}\\
 & & \{(w,(x,y))\}((0,(0,\infty,0);0) & \text{(Rule 29)}\\
(C,(x,y)) & \longrightarrow & \{(d,(x,y)),(C,(x+1,y))\}((0,1,1),(\infty,\infty,\infty);(1,-1,0)) \mid & \text{(Rule 30)}\\
 & & \{(d,(x,y)),(B,(x,y-1))\}((1,0,1),(\infty,\infty,\infty);(0,0,-1)) \mid & \text{(Rule 31)}\\
 & & \{(d,(x,y))\}((0,0,0),(\infty,0,0);0) & \text{(Rule 32)}
\end{array}
\right\}
$$

Fig. 11: The set of rules for $G_{\mathrm{Carpet-A}}$.

$$
R = \left\{
\begin{array}{lcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\}(0,0;(1,0,1,0,0)) & \text{(Rule 33)}\\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x+1,y))\} & \text{(Rule 34)}\\
 & & ((1,0,1,0,0),(\infty,0,\infty,0,0);(1,0,1,0,0)) \mid & \text{(Rule 35)}\\
 & & \{(w,(x,y)),(B,(x,y-1))\} & \\
 & & ((2,0,2,0,0),(\infty,0,\infty,\infty,\infty);(0,0,-1,1,0)) \mid & \text{(Rule 36)}\\
 & & \{(w,(x,y)),(A,(x+1,y))\} & \text{(Rule 37)}\\
 & & ((2,0,2,0,0),(\infty,0,\infty,\infty,\infty);(1,0,1,1,0)) & \text{(Rule 38)}\\
(B,(x,y)) & \longrightarrow & \{(B,(x-1,y)),(w,(x,y))\} & \text{(Rule 39)}\\
 & & ((1,0,0,0,0),(\infty,\infty,\infty,\infty,0);(-1,1,0,0,0)) \mid & \text{(Rule 40)}\\
 & & \{(w,(x,y)),(C,(x,y-1))\} & \text{(Rule 41)}\\
 & & ((0,1,1,1,0),(0,\infty,\infty,\infty,0);(0,0,-1,0,0)) & \text{(Rule 42)}\\
(C,(x,y)) & \longrightarrow & \{(d,(x,y)),(C,(x+1,y))\} & \text{(Rule 43)}\\
 & & ((0,1,1,1,0),((\infty,\infty,\infty,\infty,0));(1,-1,0,0,0)) \mid & \text{(Rule 44)}\\
 & & \{(d,(x,y)),(B,(x,y-1))\} & \text{(Rule 45)}\\
 & & ((1,0,1,1,0),((\infty,0,\infty,\infty,0));(0,0,-1,0,0)) \mid & \text{(Rule 46)}\\
 & & \{(w,(x,y)),(C,(x+1,y))\} & \text{(Rule 47)}\\
 & & ((0,1,0,1,0),(\infty,\infty,0,\infty,\infty);(0,-1,0,-1,1)) \mid & \text{(Rule 48)}\\
 & & \{(w,(x,y)),(C,(x+1,y))\} & \text{(Rule 49)}\\
 & & ((0,1,0,0,1),(\infty,\infty,0,0,\infty);(0,-1,0,0,-1)) \mid & \text{(Rule 50)}\\
 & & \{(d,(x,y)),(C,(x+1,y))\} & \text{(Rule 51)}\\
 & & ((0,1,0,0,0),(\infty,\infty,0,0,0);(0,-1,0,0,0)) \mid & \text{(Rule 52)}\\
 & & \{(d,(x,y))\}(0,0;0) & \text{(Rule 53)}
\end{array}
\right\}
$$

Fig. 13: The set of rules for $G_{\mathrm{Carpet-B}}$.

colours, such that each colour dominates a row. This feature is replicated on the lower right corner of the image. See Fig. 14.

Let $G_{\mathrm{Carpet-C}} = (V_M, V_T, R, S, \{1, 2, 3, 4, 5\}, 0)$*, where* $V_M = \{S, A, B, C, A', A''\}$*,* $V_T = \{d, w\}$*,* $(S,(x,y)) = (S,(0,0))$*, and* R *is the set in Fig. 15.*

Some pictures in $\mathcal{G}$ produced by $G_{\mathrm{Carpet-C}}$ are shown in Fig. 14.

The strategy here is that the variables A, A', and A'' in Example II.5 are used for the upper left corner of the image. The variables B and C can be interchangeably used for the lower left and upper right corners respectively or the upper right corner can also be mirrored on the lower left corner depending on which rule is applied during derivation. Meanwhile, the lower right corner of the image is a reflection of the upper left corner. The derivation starts from the upper left corner to the lower left corner then to the upper right and lower right corners of the image respectively. The size of the upper left corner of the image determines the size of the other three corners (parts) of the image. See Fig. 14.

To achieve this strategy, the first and second bag positions are used to control the light and dark colours respectively. The third bag position is used to ensure the number of rows is equal to the number of columns. The fourth and fifth bag positions are used to control the lower left, upper right and lower right corners of the image respectively. The lower right corner mimics the upper left corner of the image. Such that the number of light and dark colours in the upper left corner is exactly the same as the lower

$$
R = \Bigg\{
$$

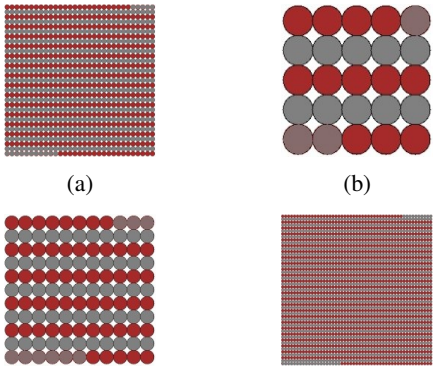
$$
\begin{aligned}
(S,(x,y)) &\longrightarrow \{(d,(x,y)),(A,(x+1,y))\}(0,0;(1,0,1,0,0)) && \text{(Rule 54)}\\
(A,(x,y)) &\longrightarrow \{(d,(x,y)),(A,(x+1,y))\}((1,0,1,0,0),(\infty,0,\infty,0,0);(1,0,1,0,0)) \mid && \text{(Rule 55)}\\
&\quad \{(b,(x,y)),(B,(x+1,y),(A',(x-1,y))\} && \\
&\qquad ((1,0,1,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,0)) && \text{(Rule 56)}\\
&\quad \{(b,(x,y)),(C,(x+1,y),(A',(x-1,y))\} && \\
&\qquad ((1,0,1,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,0)) && \text{(Rule 57)}\\
(A',(x,y)) &\longrightarrow \{(A',(x-1,y)),(w,(x,y))\} && \\
&\qquad ((1,0,0,0,0),(\infty,\infty,\infty,\infty,0);(-1,1,0,0,0)) \mid && \text{(Rule 58)}\\
&\quad \{(w,(x,y)),(A'',(x,y-1))\} && \\
&\qquad ((0,1,1,0,0),(0,\infty,\infty,\infty,0);(0,0,-1,1,0)) \mid && \text{(Rule 59)}\\
&\quad \{(d,(x,y)),(B,(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);0) \mid && \text{(Rule 60)}\\
&\quad \{(w,(x,y)),(C,(x,y-1))\}((0,1,0,0,0),(0,\infty,0,\infty,0);0) \mid && \text{(Rule 61)}\\
&\quad \{(d,(x,y))\}(0,(0,\infty,0,\infty,\infty);0) && \text{(Rule 62)}\\
(A'',(x,y)) &\longrightarrow \{(d,(x,y)),(A'',(x+1,y))\}((0,1,0,0,0),\infty;(1,-1,0,0,0)) \mid && \text{(Rule 63)}\\
&\quad \{(d,(x,y)),(A',(x,y-1))\}((1,0,1,0,0),\infty;(0,0,-1,1,0)) \mid && \text{(Rule 64)}\\
&\quad \{(d,(x,y)),(C,(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);0) \mid && \text{(Rule 65)}\\
&\quad \{(d,(x,y)),(B,(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);0) \mid && \text{(Rule 66)}\\
&\quad \{(d,(x,y))\}(0,(\infty,0,0,\infty,\infty);0) && \text{(Rule 67)}\\
(C,(x,y)) &\longrightarrow \{(d,(x,y)),(C,(x+1,y))\} && \\
&\qquad ((0,1,0,0,0),(\infty,\infty,\infty,\infty,0);(1,-1,0,0,0)) \mid && \text{(Rule 68)}\\
&\quad \{(C,(x-1,y)),(d,(x,y))\} && \\
&\qquad ((1,0,1,0,1),(\infty,\infty,\infty,\infty,1);(-1,1,0,0,0)) \mid && \text{(Rule 69)}\\
&\quad \{(d,(x,y)),(C,(x,y-1))\} && \\
&\qquad ((0,1,0,1,1),(0,\infty,\infty,\infty,1);(0,0,1,-1,-1)) \mid && \text{(Rule 70)}\\
&\quad \{(d,(x,y)),(C,(x,y-1))\} && \\
&\qquad ((1,0,0,1,0),(\infty,0,\infty,\infty,0);(0,0,1,-1,1)) \mid && \text{(Rule 71)}\\
&\quad \{(w,(x,y)),(A'',(x,y-1))\} && \\
&\qquad ((0,1,0,0,1),(0,\infty,0,\infty,1);(0,0,0,0,-1)) && \text{(Rule 72)}\\
&\quad \{(w,(x,y)),(A',(x,y-1))\} && \\
&\qquad ((1,0,0,0,0),(\infty,0,0,\infty,0);(0,0,0,0,0)) && \text{(Rule 73)}\\
&\quad \{(d,(x,y))\}(0,(0,\infty,\infty,0,\infty);(0,0,0,0,-1)) && \text{(Rule 74)}\\
(B,(x,y)) &\longrightarrow \{(w,(x,y)),(B,(x+1,y))\} && \\
&\qquad ((0,1,0,0,0),(\infty,\infty,\infty,\infty,0);(1,-1,0,0,0)) \mid && \text{(Rule 75)}\\
&\quad \{(B,(x+1,y),(w,(x,y)))\} && \\
&\qquad ((1,0,1,0,1),(\infty,\infty,\infty,\infty,1);(-1,1,0,0,0)) \mid && \text{(Rule 76)}\\
&\quad \{(w,(x,y)),(B,(x,y-1))\} && \\
&\qquad ((1,0,0,0,0),(\infty,0,\infty,\infty,0);(0,0,-1,1,1)) \mid && \text{(Rule 77)}\\
&\quad \{(w,(x,y)),(B,(x,y-1))\} && \\
&\qquad ((0,1,1,0,1),(0,\infty,\infty,\infty,1);(0,0,-1,1,-1)) \mid && \text{(Rule 78)}\\
&\quad \{(d,(x,y))\}(0,(0,\infty,\infty,0,\infty);(0,0,0,0,-1)) && \text{(Rule 79)}\\
&\quad \{(w,(x,y)),(A'',(x,y-1))\}((0,1,0,0,1),(0,\infty,0,\infty,1);(0,0,0,0,-1)) && \text{(Rule 80)}\\
&\quad \{(w,(x,y)),(A',(x,y-1))\}((1,0,0,0,0),(\infty,0,0,\infty,0);(0,0,0,0,0)) && \text{(Rule 81)}
\end{aligned}
$$

$$
\Bigg\}
$$

Fig. 15: The set of rules for $G_{\mathrm{Carpet-C}}$.

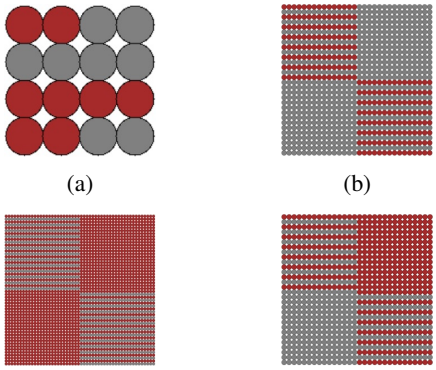
right corner. The size of the upper left corner is duplicated on the lower left corner with a dark or light colour. Then the lower left corner of the image is mirrored on the upper right corner of the image with the same or an opposite colour depending on which rule is applied.

The application of Rule 55 of Fig. 15 after Rule 54 starts the formation of the upper left corner of the image. The bag adjustment, (1, 0, 1, 0, 0) is added to the bag as long as the bag at each application is within the range, this continues until Rule 56 or 57 is applied. The application of Rule 56 or 57 begins the formation of a new row and creates room for the formation of the upper right corner. Any of Rules 58–67 is applicable as long as the bag at each application is within the range. This process will form the upper left corner of the image, then begins the formation of the lower left corner with the application of Rule 65 or 66. If Rule 65 is applied then Rules 68–74 are applicable when the bag is within the range at each application. On the other hand, If Rule 66 is applied then Rules 75–81 are applicable. The lower left corner formation



(a) (b) (c) (d)

Fig. 12: Some of the images in $\mathcal{G}$ of $G_{\text{Carpet}-\text{B}}$.



(a) (b) (c) (d)

Fig. 14: Some of the pictures in $\mathcal{G}$ of $G_{\text{Carpet}-\text{C}}$.

is completed by the application of Rule 74 or 79.

The formation of the upper right corner begins with the application of Rule 68 or 75 of Fig. 15. The bag adjustment, (1, -1, 0, 0, 0) is added to the bag until the second bag position is 0. Then any of Rules 69–71 or 76–78 is applicable so long as the bag at each application is within the range. This process continues until the third bag position is 0 then the upper right corner is complete. The formation of the lower right corner begins with the application of any of Rule 72, 74, 80 or 81, as long as the bag is within the range. At this point, any of Rules 58–67 is now applicable as long as the bag at each application is within the range as usual. This process continues until the lower right corner formation is complete. Then the entire picture formation is now complete. This strategy is sufficient to generate the images in Fig. 14.

Next, we compare the bag context shape grammars and basic puzzle grammars with permitting features in Section III.

III. Comparison of BCSGs and Basic Puzzle Grammars With Permitting Features

In this section, we compare BCSGs to their basic puzzle with permitting features (also called permitting basic puzzle grammar) counterparts. The choice of permitting basic puzzle grammar is that the model is one of the recent developments in puzzle grammar systems to the best of our knowledge. By permitting feature we mean that the grammars can control when a rule should be applied during derivation.

A. Permitting Basic Puzzle Grammars

We begin by understanding permitting basic puzzle grammar.

Definition 10 *A permitting basic puzzle grammar (PBPzG) $G = (V_M, V_T, R, S)$ has disjoint, nonempty, finite sets V_M and V_T of variables and terminals, respectively, a finite set of rules R, and a start symbol $S \in V_M$.*

A rule is of the form $(A \to \alpha, per)$, where $A \to \alpha$ is the basic puzzle grammar rule as in [33], [34], $A \in V_M$, $\alpha \subseteq (V_M \cup V_T)$ and $per \subseteq V_M$ [35]. If $per = \emptyset$, then it is not mentioned in the rule. A derivation starts with S written in a unit cell in the two-dimensional plane, with all other cells containing the blank symbol $\# \notin V_M \cup V_T$.

For a $PBPzG$ G and a pictorial form Π, a rule, $A \to \alpha$, per in R can be applied if $A \in V_M$ and $per \subseteq l(\Pi) \setminus \{A\}$. Then, in a derivation step, denoted by $\Longrightarrow^*_{\Pi}$, a nonterminal A in a cell is replaced by the right hand side of a rule with the left hand side A. In this replacement, the circled symbol of the right hand side of the rule used occupies the cell of the replaced symbol A and the non-circled symbol of the right hand side occupies the cell to the right or left or above or below the cell of the replaced symbol, depending on the type of rule used. The replacement is possible only if the cell to be filled in by the non-circled symbol contains a $\#$.

A picture generated by G is a connected, finite array of elements of V_T derived in one or more steps from the start symbol; the set of such pictures, the gallery, is denoted by $\mathcal{G}(G)$.

Example III.1 *Consider a PBPzG of the structure $G = (V_M, V_T, R, S)$ where $V_M = \{S, A, B, A', B', C, D\}$, $V_T = \{d\}$, R is the set of rules in Fig. 16 and S is an array of the form* $\begin{matrix} A \\ d \quad B \end{matrix}$.

$R = \Big\{$

$$S \longrightarrow \begin{matrix} \text{A} \\ \boxed{\text{d}}\ \text{B} \end{matrix} \quad \text{(Rule 82)}$$

$$A \longrightarrow \begin{matrix} A' \\ \boxed{\text{d}} \end{matrix}, \{\text{B}\} \quad \text{(Rule 83)}$$

$$B \longrightarrow \boxed{\text{d}}\ B', \quad \{A'\} \quad \text{(Rule 84)}$$

$$A' \longrightarrow A, \{B'\} \quad \text{(Rule 85)}$$

$$B' \longrightarrow B, \{A\} \quad \text{(Rule 86)}$$

$$A \longrightarrow C, \{B\} \quad \text{(Rule 87)}$$

$$B \longrightarrow D, \{C\} \quad \text{(Rule 88)}$$

$$C \longrightarrow d \quad \text{(Rule 89)}$$

$$D \longrightarrow d \quad \text{(Rule 90)}$$

$\Big\}$

Fig. 16: The set of PBPzG rules in Example III.1.

The derivation starts with S then Rule 83 is applicable because the permitting symbol B is present in the array. This grows the vertical arm by one square. At this point, Rule 84 can be applied because the permitting symbol, A' is present, hence this grows the horizontal arm by one square. Rule 85 can now be applied because the permitting symbol, B' is present followed by Rule 86 whose permitting symbol, A is present. The process can be repeated to grow both arms equal in length until Rules 87 and 88 are applied to change the symbol A to C and B to D. Then the process can be terminated by the application of Rules 89 and 90. This derivation generates a gallery in the shape of an L with base and height equal in length. The gallery generated by the grammar is presented in Fig. 17. Next, we compare BCSGs and PBPzG.

B. Comparison of BCSGs and PBPzG

We consider a permitting basic puzzle grammar, PBPzG, $G = (V_M, V_T, R, S)$. Suppose $V_M = \{A_1, A_2, \ldots, A_m\}$, and the order of the elements in V_M is arbitrary but fixed. For rule $A \to \alpha$, per,

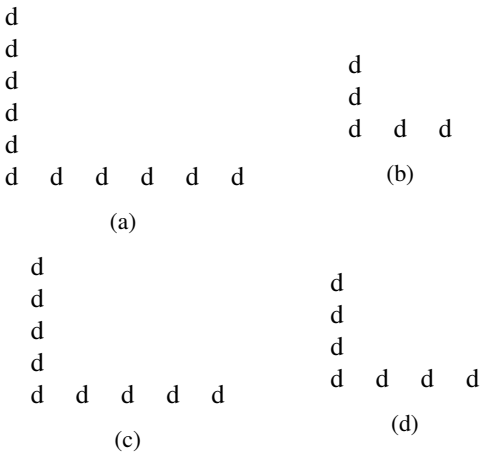


Fig. 17: Some images in $\mathcal{G}(G)$.

we denote by cnt, the occurrences of the variable of V_M in the left hand and right hand sides of the rule. In particular,

- $cnt(A)$ denotes the m–tuple in which the component corresponding to A is set to 1, whereas all other components are set to 0,
- $cnt(\alpha)$ denotes the m–tuple $(n_1, n_2, \ldots, n_m)$ such that n_i is equal to the number of occurrences of A_i in $[x_1, \ldots, x_m]$, and
- $cnt(per)$, denotes the $\sum_{A \in per} cnt(A)$.

The following Lemma is adapted from [6]. We show that bag context shape grammar is strictly more powerful than permitting puzzle grammar.

Lemma III.1 *For every basic permitting puzzle grammar, G = (V_M, V_T, R, S) there is an equivalent BCSG G' = (V_M, V_T, R', $(S, (x, y))$, I, β_0). That is, $\mathcal{G}(G) \subseteq \mathcal{G}(G')$.*

Proof: Without loss of generality, assume that $V_M = \{A_1, A_2, \ldots, A_m\}$, $\forall\ m \in \mathbb{N}_+$. The bag index set $I = [m]$ records, at all times during a derivation, the number of occurrences of A_i in the ith position of the bag. Let the initial bag, $\beta_0 = cnt(S)$. For every permitting puzzle grammar rule, $A \rightarrow \alpha$, per in R, we write it to the equivalent bag context shape grammar rule $(A, (x, y)) \longrightarrow \{(A_1, (x_1, y_1)), (A_2, (x_2, y_2)), \ldots, (A_i, (x_i, y_i))\}$ $(L_b, U_b : \delta)$ in R', where

- $L_b = cnt(A) + cnt(per)$,
- $U_b = cnt(A) + \infty.cnt(V_M) = cnt(A) + \infty.cnt(V_M) = cnt(A) + \infty = \infty$, and
- $\delta = cnt(\alpha)$ - $cnt(A)$.

Then, $\mathcal{G}(G) \subseteq \mathcal{G}(G')$.

Next, we use Lemma III.1 to illustrate that the rules of Example III.1 of PBPzG can be converted to BCSG rules. We derive the values of the lower limit, L_b, the upper limit, U_b, and the bag adjustment, δ for each rule in Fig. 16, see Table II. Then, the values obtained in Table II are used to rewrite each rule in Fig. 16 to a BCSG rule, see Fig. 18 of Example III.2.

Example III.2 *We want to rewrite the rules of Example III.1 to BCSG rules using the values of L_b, U_b, and δ in Table II for each rule. Let a BCSG G_{bag} = (V_M, V_T, R', $(S, (x, y))$, $\{1, 2, \ldots, 7\}$, $\{1, 0, 0, 0, 0, 0, 0\}$), where V_M = {S, A, B, A', B', C, D}, V_T = {d}, R' is as shown in Fig. 18, and $(S, (x, y)) = ((d, (0, 0)), (A, (0, 1)), (B, (1, 0)))$.*

The derivation starts with S, then Rule 92 is applicable because the bag is within the defined range. That is the bag, (0, 1, 1, 0, 0, 0, 0) is greater than or equal to the lower limit, (0, 1, 1, 0, 0, 0, 0) and the bag is at the same time less than or equal to the upper limit, ∞, then the bag adjustment, (0, -1, 0, 1, 0, 0, 0) is added to the bag. Rule 93 is applicable followed by Rules 94 and 95 because they are within the range at each application. This process can be repeated to grow both arms equal in length until Rule 96 is applied followed by Rule 97. Then the process can be terminated by applying Rules 98 and 99 because they are within the defined range.

TABLE II: The conversion of the PBPzG rules in Fig. 16 to bag context.

Rules	L_b	U_b	δ
82	(1, 0, 0, 0, 0, 0, 0)	∞	(-1, 1, 1, 0, 0, 0, 0)
83	(0, 1, 1, 0, 0, 0, 0)	∞	(0, -1, 0, 1, 0, 0, 0)
84	(0, 0, 1, 1, 0, 0, 0)	∞	(0, 0, -1, 0, 1, 0, 0)
85	(0, 0, 0, 1, 1, 0, 0)	∞	(0, 1, 0, -1, 0, 0, 0)
86	(0, 1, 0, 0, 1, 0, 0)	∞	(0, 0, 1, 0, -1, 0, 0)
87	(0, 1, 1, 0, 0, 0, 0)	∞	(0, -1, 0, 0, 0, 1, 0)
88	(0, 0, 1, 0, 0, 1, 0)	∞	(0, 0, -1, 0, 0, 0, 1)
89	(0, 0, 0, 0, 0, 1, 0)	∞	(0, 0, 0, 0, 0, -1, 0)
90	(0, 0, 0, 0, 0, 0, 1)	∞	(0, 0, 0, 0, 0, 0, -1)

This derivation generates images in the shape of an L with base and height equal in length. The images are presented in Fig. 19 when the terminal d is replaced with a square shape which is filled with a dark colour. The images are also scaled to the same size for presentation purposes.

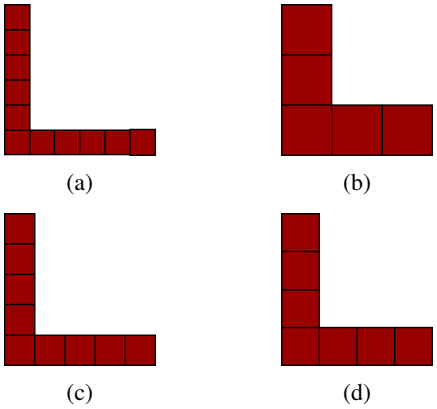


Fig. 19: Some images in $\mathcal{G}(G_{bag})$.

Clearly the gallery $\mathcal{G}(G) \subseteq \mathcal{G}(G_{bag})$, see Figs. 17 and 19. Next we show that BCSGs can generate the same gallery with fewer rules and variables.

IV. The Strength of BCSGs

Here we demonstrate the strength of BCSGs in terms of their ability to produce the same gallery in the shape of an L with base and height equal in length that permitting puzzle grammars produce (see Figure 17). At this time, we show how we can produce the same gallery using BCSGs with fewer variables and rules. We illustrate this notion in Example IV.1.

Example IV.1 *Consider the gallery of an L shape whose base and height are equal in length. Let BCSG G''_{bag} = (V_M, V_T, R''', $(S, (x, y))$, $\{1, 2, 3\}$, 0), where V_M = {S, A, B}, V_T = {d}, R''' is as shown in Fig. 20, and $(S, (x, y)) = (S, (0, 0))$.*

The strategy is that the first and the second bag positions are used to control how the base and the height of the L shape grow. The third bag position ensures that the base and height of the L are the same lengths.

To achieve the strategy, Rule 101 can be applied immediately after Rule 100, then the bag adjustment, (0, 0, 1) is added to the bag. Rule 101 is no longer applicable because the third bag position is now 1 until Rule 103 is applied, which automatically adds the bag adjustment (0, 0, -1) to the bag. The third bag position is 0 making Rule 103 inapplicable. This process can continue while the base

$$
R' = \left\{
\begin{array}{rcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x,y+1)),(B,(x+1,y))\} & \\
 & & ((1,0,0,0,0,0,0),\infty;(-1,1,1,0,0,0,0)) & \text{(Rule 91)} \\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A',(x,y+1))\} & \\
 & & ((0,1,1,0,0,0,0),\infty;(0,-1,0,1,0,0,0)) & \text{(Rule 92)} \\
(B,(x,y)) & \longrightarrow & \{(d,(x,y)),(B',(x+1,y))\} & \\
 & & ((0,0,1,1,0,0,0),\infty;(0,0,-1,0,1,0,0)) & \text{(Rule 93)} \\
(A',(x,y)) & \longrightarrow & \{(A,(x,y))\}((0,0,0,1,1,0,0),\infty;(0,1,0,-1,0,0,0)) & \text{(Rule 94)} \\
(B',(x,y)) & \longrightarrow & \{(B,(x,y))\}((0,1,0,0,1,0,0),\infty;(0,0,1,0,-1,0,0)) & \text{(Rule 95)} \\
(A,(x,y)) & \longrightarrow & \{(C,(x,y))\}((0,1,1,0,0,0,0),\infty;(0,-1,0,0,0,1,0)) & \text{(Rule 96)} \\
(B,(x,y)) & \longrightarrow & \{(D,(x,y))\}((0,0,1,0,0,1,0),\infty;(0,0,-1,0,0,0,1)) & \text{(Rule 97)} \\
(C,(x,y)) & \longrightarrow & \{(d,(x,y))\}((0,0,0,0,0,1,0),\infty;(0,0,0,0,0,-1,0)) & \text{(Rule 98)} \\
(D,(x,y)) & \longrightarrow & \{(d,(x,y))\}((0,0,0,0,0,0,1),\infty;(0,0,0,0,0,0,-1)) & \text{(Rule 99)}
\end{array}
\right\}
$$

Fig. 18: The set of rules for G_{bag} in Example III.2.

$$
R'' = \left\{
\begin{array}{rcll}
(S,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x,y+1)),(B,(x+1,y))\}(0,0;(1,1,0)) & \text{(Rule 100)} \\
(A,(x,y)) & \longrightarrow & \{(d,(x,y)),(A,(x,y+1))\}((1,1,0),(1,1,0);(0,0,1)) \mid & \text{(Rule 101)} \\
 & & \{(d,(x,y))\}((1,1,0),(1,1,0);(-1,0,1)) & \text{(Rule 102)} \\
(B,(x,y)) & \longrightarrow & \{(d,(x,y)),(B,(x+1,y))\}(1,1;(0,0,-1)) \mid & \text{(Rule 103)} \\
 & & \{(d,(x,y))\}((0,1,1),(0,1,1);(0,-1,-1)) & \text{(Rule 104)}
\end{array}
\right\}
$$

Fig. 20: The set of rules for G''_{bag} in Example IV.1.

and height grow at equal length until Rule 102 is applied followed by Rule 104.

The set of rules in Fig. 20 can generate the same gallery generated by permitting basic puzzle grammar rules of Examples III.1 with fewer variables and rules. Observe that there are seven variables in Examples III.1 while Example IV.1 has only three variables. Also, the set in Fig. 16 of Examples III.1 has eight number of rules while the number of rules in Fig. 20 is only five rules.

These can be regarded as strength of BCSGs because they can produce the same type of gallery as permitting basic puzzle grammars with fewer rules and variables, even with fewer bag positions. Next, we conclude this work in Section V.

V. Conclusion and Future Work

In this paper, we introduced bag context shape grammars (BCSGs). They are context free grammars, but the application of a rule is controlled by a special vector of integers, called the bag, which changes during a derivation. We demonstrated that these grammars can be used to generate images in a regulated manner. We compared BCSGs and permitting basic puzzle grammars (PBPzGs), a recent development in the basic puzzle grammar system that regulates the application of a rule during derivation.

We stated a Lemma that shows how any PBPzG can be converted to a BCSG and demonstrated this notion with an example. We then considered one set of images and showed that BCSGs can generate the same gallery with fewer variables, rules, and even fewer bag positions. These BCSGs can offer a wide range of application areas such as the generation of distractor (similar) images for visual password systems or scheme.

In the future, we will compare BCSGs and other recent developments in array grammars with permitting features. Also, when a PBPzG is converted to a BCSG using the above Lemma III.1, the resultant BCSG has a bag position for each variable of the PBPzG. By inspecting the BCSG, one can often reduce the number of bag positions. The question is whether there exist general rules for reducing bag positions.

Acknowledgements

The authors would like to thank the Department of Science and Technology (DST) and the Council for Scientific and Industrial Research (CSIR), Inter-bursary support programme, South Africa, for funding this research.

References

[1] E. L. Post, "Formal reductions of the general combinatorial decision problem," *American Journal of Mathematics*, vol. 65, no. 2, pp. 197–215, 1943.

[2] N. Chomsky, "Three models for the description of language," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 113–124, 1956.

[3] T. Jiang, M. Li, B. Ravikumar, and K. W. Regan, "Formal grammars and languages," in *Algorithms and Theory of Computation Handbook*. Chapman & Hall/CRC, 2010, pp. 20–26.

[4] N. Chomsky, "Formal properties of grammars, Handbook of Mathematical Pychology," *Wiley & Sons, New York*, vol. 2, pp. 232–418, 1963.

[5] ——, "Constraints on rules of grammars," *Linguistic Analysis*, vol. 2, pp. 303–351, 1976.

[6] S. Ewert, N. Jingili, L. Mpota, and I. Sanders, "Bag context picture grammars," *Journal of Computer Languages*, 2019, doi: https://doi.org/10.1016/j.cola.2019.04.001.

[7] S. Ewert, "Random context picture grammars," Ph.D. dissertation, Stellenbosch University, Stellenbosch, 1999.

[8] H. Kruger and S. Ewert, "Translating mutually recursive function systems into generalized random context picture grammars: reviewed article," *South African Computer Journal*, no. 36, pp. 99–109, 2006.

[9] F. Drewes, "Tree-based picture generation," *Theoretical Computer Science*, vol. 246, no. 1, pp. 1–51, 2000.

[10] B. Okundaye, "A Tree Grammar-Based Visual Password Scheme," Ph.D. dissertation, School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, 2015.

[11] S. Ewert and A. van der Walt, "Shrink indecomposable fractals," *Journal of Universal Computer Science*, vol. 5, no. 9, pp. 521–531, 1999.

[12] A. Lindenmayer, "Mathematical models for cellular interactions in development I. filaments with one-sided inputs," *Journal of Theoretical Biology*, vol. 18, no. 3, pp. 280–299, 1968.

[13] S. Rani, H. Abdul, M. Chandrasekaran, and K. G. Subramanian, "Stochastic puzzle grammars," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 6, pp. 257–272, 1992.

[14] Y. Yasunori, M. Kenichi, and S. Kazuhiro, "Context sensitivity of two-dimensional regular array grammars," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 3, no. 3 & 4, pp. 295–319, 1989.

[15] G. Roger, "A transformation system for generating description languages of chain code pictures," *Theoretical Computer Science*, vol. 68, pp. 239–252, 1989.

[16] F. Drewes, "Language theoretic and algorithmic properties of d-dimensional collages and patterns in a grid," *Journal of Computer and System Sciences*, vol. 53, pp. 33–60, 1996.

[17] G. Stiny and J. Gips, "Shape grammars and the generative specification of painting and sculpture." in *IFIP Congress (2)*, vol. 2, no. 3, 1971, pp. 125–135.

[18] T. Trescak, M. Esteva, and I. Rodriguez, "A shape grammar interpreter for rectilinear forms," *Computer-Aided Design*, vol. 44, no. 7, pp. 657–670, 2012.

[19] G. Stiny, "Introduction to shape and shape grammars," *Environment and Planning B*, vol. 7, no. 3, pp. 343–351, 1980.

[20] R. Stouffs, "Description grammars : A general notation," *Environment and Planning B: Urban Analytics and City*, vol. 45, no. 1, pp. 106–123, 2016.

[21] T. W. Knight, "Shape grammar and color grammar in design," *Environment and Planning B: Urban Analytics and City Science*, vol. 21, no. 6, pp. 705–735, 1994.

[22] G. Stiny and W. J. Mitchell, "The Palladian grammar," *Environment and Planning B: Planning and Design*, vol. 5, no. 1, pp. 5–18, 1978.

[23] D. A. Al-Kazzaz and A. H. Bridges, "A framework for adaptation in shape grammars," *Design Studies*, vol. 33, no. 4, pp. 342–356, 2012.

[24] K. Shea and J. Cagan, "The design of novel roof Trusses with shape annealing: Assessing the ability of a computational method in aiding structural designers with varying design intent," *Design Studies*, vol. 20, no. 1, pp. 3–23, 1999.

[25] T. H. Speller, D. Whitney, and E. Crawley, "Using shape grammar to derive cellular automata rule patterns," *Complex Systems-Champaign*, vol. 17, no. 1/2, p. 79, 2007.

[26] L. March and G. Stiny, "Spatial systems in architecture and design: Some history and logic," *Environment and Planning B: Planning and Design*, vol. 12, no. 1, pp. 31–53, 1985.

[27] T. W. Knight, *Transformations in design: A formal approach to stylistic change and innovation in the visual arts*. Cambridge University Press, 1995.

[28] ——, "Shape grammars and color grammars in design," *Environment and Planning B: Planning and Design*, vol. 21, no. 6, pp. 705–735, 1994.

[29] S. Ahmad and S. Chase, "Transforming grammars for goal driven style innovation," in *Predicting the future, Proceedings of the 25th Conference on Education in Computer Aided Architectural Design in Europe (eCAADe), Frankfurt am Main, Germany*, 2007, pp. 879–886.

[30] F. Drewes, C. du Toit, S. Ewert, B. van der Merwe, and A. P. van der Walt, "Bag context tree grammars," in *Proceedings of the 10th International Conference, DLT 2006, Santa Barbrara, CA*. Springer, June 26-29, 2006, pp. 226–237.

[31] S. Ewert, "Random context picture grammars: The state of the art," *Manipulation of Graphs, Algebras and Pictures. Essays Dedicated to Hans-Jörg Kreowski on the Occasion of His 60th Birthday*, pp. 135–147, 2009.

[32] C. Anthony and D. Robert, *Foundation Maths*, 6th ed. Edinburgh Gate, United Kingdom: Pearson Education Limited, 2016.

[33] K. Subramanian, R. Siromoney, V. Reda, and A. Saoudi, "Basic puzzle languages," *International Journal of Pattern Recognition and Artificial Intelligence,*, vol. 9, no. 5, pp. 763–775, 1995.

[34] ——, "Basic puzzle grammars and isosceles right triangles," *International Journal of Pattern Recognition and Artificial Intelligence,*, vol. 6, no. 5, pp. 799–816, 1992.

[35] P. Isawasan, R. Muniyandi, I. Venkat, and K. Subramanian, "Array-rewriting P systems with basic puzzle grammar rules and permitting features," in *International Conference on Membrane Computing, Cham, Switzerland*. Springer, 2017, pp. 272–285.