Vol. 33, No. 1, 2024
(this volume contains accepted papers online and is not closed yet)

Machine GRAPHICS & VISION

International Journal

Published by
The Institute of Information Technology
Warsaw University of Life Sciences – SGGW
Nowoursynowska 159, 02-776 Warsaw, Poland
in cooperation with
The Association for Image Processing, Poland – TPO

An Improved Generative Design Approach based on Graph Grammar for Pattern Drawing

Yufeng Liu*, Yangchen Zhou, Fan Yang, Song Li and Jun Wu
College of Information Engineering, Nanjing University of Finance and Economics, Nanjing, China
**Corresponding author: Yufeng Liu (yfengliu28@126.com)*

Abstract Generative design is used to efficiently generate design solutions with powerful computational methods. Generative design based on shape grammar is currently the most commonly used approach, but it is difficult for shape grammar to formally analyze the generated pattern. Graph grammar derived from one-dimensional character grammar is mainly used for generating and analyzing abstract models of visual languages. However, there is a significant gap between the generated node-edge graphs and the representation of shape appearance. To address these problems, we propose an improved generative design approach based on virtual-node based continuous Coordinate Graph Grammar (vcCGG). This approach defines a new type of grammatical rule named node transformation rules to convert nodes into shapes with node transformation applications. By combining node transformation applications and L-applications in vcCGG, we can generate a node-edge graph as the structure of the pattern through L-applications, and then draw the shape outline, next adjust the positions of these shapes, thus relating abstract structures and the physical layouts of visual languages. At the end of the paper, we provide an example application of this approach: generating an illustration from Emma Talbot using a combination of node transformation applications and L-applications.

Keywords: generative design; graph grammar; shape grammar; node transformation rules; pattern drawing.

1. Introduction

Design is a complex solution process that involves professional knowledge, innovative ability, comprehensive experience, aesthetic literacy, and use of scientific technology. With the rapid development and popularization of new intelligent design automation technologies such as machine learning, additive manufacturing, artificial intelligence, and cloud computing, design approaches are constantly expanding. As a developing design approach, generative design has been extensively studied in academia. Since the introduction of generative design based on shape grammar, as proposed by G. Stiny and J. Gips in 1971 [18], generative design has been introduced into different fields such as architectural design [5], product customization design [9], and visual communication design [14].

Shape grammar is a generation system oriented toward design. It is a design inference approach based on rules, using simple shapes as basic elements to establish the rules for the generation of complex shapes. The foundational rules involve spatial transformations such as translation, scaling, rotation and mirroring, which make one shape part of another shape. With limited predefined rules, there can be an infinite number of

designs generated through shape grammar. Following predefined rules, shape grammar can iteratively replace shapes to generate various patterns. However, shape grammar can generate only the shapes that consist of simple shapes such as lines, points and rectangles. Therefore, it is not yet widely used in computer-aided architectural design (CAAD) applications. Most designers design buildings manually or semi-automatically on CAD platforms, e.g. Revit and AutoCAD.

Shape grammar focuses on generative design, while graph grammar derived from one-dimensional character grammar focuses on modeling and analyzing the syntax and semantics of visual languages. Shape grammar supports only unidirectional workflows. It takes the initial shape and transformation rules as inputs to generate a preliminary design and then adjusts the preliminary design by the rules to generate the final design. In contrast, graph grammars have a bidirectional workflow across derivation and specification. Similarly, the graph grammar derivation process derives graphs by repeatedly applying given productions. The graph grammar reduction process, on the other hand, takes graphs and productions as inputs to parse the graphs by applying productions in a bottom-up fashion. However, there is a significant gap between the generated node-edge graphs and the representation of shape appearance for graph grammar.

In our previous work, we proposed an enhanced grammar system for shape generation [12]. This system defines shape rules to transform edges into shapes by shape applications, which builds an inherent relation between abstract structures and physical layouts of visual languages. The main weakness of this system is the position invariance that reduces the flexibility of design. To address the aforementioned issue, our research focuses on an analysis of semantic relations among shapes that make up a pattern. We propose a generative design approach based on vCGG (virtual-node based Coordinate Graph Grammar) [10]. Our approach defines a new type of grammatical rule named node transformation rules to convert nodes into shapes with node transformation applications. By combining node transformation applications and L-applications in vCGG, we can generate a node-edge graph as the structure of the pattern through L-applications, and then draw the outlines of shapes with node transformation applications, next adjusting the positions of these shapes.

In summary, this paper presents an improved generative design approach that automatically generates or validates patterns conforming to the specified rules. First, the structure of the target pattern is generated through vCGG, and then the nodes are converted into shapes according to the node transformation rules. Finally, the position of the shape is adjusted based on the edge attributes, and the target pattern is generated. This approach can set L-applications and node transformation rules in advance for drawing patterns, and can also formally validate a target pattern to determine whether it belongs to the pattern generated by the specified rules.

This paper addresses the aforementioned problems and makes the following contributions:

- An improved approach for grammar specification, grammar induction, generation and validation of pattern based on the vCGG formalism.
- A complete graph grammar for specifying and analyzing patterns that are composed of multiple geometric shapes.
- According to the concrete requirements, productions and transformation rules are designed to achieve customized designs.

The rest of this paper is organized as follows. Section 2 reviews the related works, including patterns generated by shape grammars, several typical graph grammars and our approach. Section 3 introduces the approach framework, including vCGG and node transformation rules. Next, Section 4 gives an example of the Cloud & Bunny rabbit pattern from Emma Talbot. Section 5 compares our approach and other generative design approaches. Finally, Section 6 concludes the paper and mentions future work.

2. Related works

In 1971, G. Stiny and J. Gips proposed that shape grammar is a generative system oriented toward design. G.Stiny detailed the concept and entire application process of shape grammar in 1980 [17]. Design based on shape grammar was first applied in the field of architectural design. M. Agarwal and J. Cagan [1] proposed the coffee machine shape grammar as the first application of shape grammar in product design, demonstrating its use for generating single products before gradually being applied to product design more broadly. The coffee machine grammar is a parametric grammar consisting of 100 manually created rules and labeled two-dimensional shape grammar implemented through a Java-based application program. Its objective is to provide designers with selectable design inspirations during the conceptual exploration phase. However, this method has limitations because its conceptual nature lacks practical production benefits, resulting in visual operational difficulties due to numerous labels.

H. H. Chau [3] concluded, through analysis of various electronic and fast-moving consumer products, that the appearance of these products is largely determined by straight lines, arcs, and their orthogonal projections. M. Pugliese and J. Cagan [13] summarized previous research methods and found that grammar has become a design tool for creating structures and functional requirements. However, there is no specific method for establishing and maintaining product brand characteristics in the field of product generation design. The field faces two challenges: engineers and designers need tools to help understand, express, and maintain product brands, and engineers, designers, and brand strategists need a common platform to discuss product brands. X. Chen et al. [4] focused on geometric shape in packaging design, proposing an application of shape grammar for packaging design research with personal care bottles as an example in experimentation. S. Wannarumon et al. [20] proposed a method for generating jewelry designs using shape grammar to support designers in exploring shapes as inspiration

sources with ring design as practice examples. S. Garcia and L.Romao [7] coded various types embedded in multifunctional chair classes to develop generative design tools usable during the chair concept design stage. Y. Yu et al. [21] proposed a method of generating origami pattern based on shape grammar recursive applications of shape rewriting rules. In addition, shape grammar provides a perspective and modeling technique for creating origami tessellation patterns.

Compared to shape grammar in the field of design, graph grammar has the characteristics of automated generation and specification. Designers can explore different design options by defining symbols, rules, and parameters, quickly generate a large number of design schemes, and make adjustments and modifications when necessary to improve design efficiency and innovation. H. Bunke [2] proposed attributed programmed graph grammars as a generative tool in image understanding. Based on that, an image understanding system was built to extract descriptions from input images, where a system consists of two major subsystems for preprocessing and segmentation, and understanding, respectively. H. Göttler et al. [8] described the data structures in terms of attributed graphs and their changes in terms of attributed graph productions in an object-oriented manner, applying Graph Grammar to CAD systems.

In the field of architectural design, X. Wang et al. [19] presented a generic approach for grammar specification, grammar induction, validation, and design generation of house floor plans using their path graphs based on the reserved graph grammar formalism (RGG). This approach validates floor plans in different styles with user-specified graph productions and the derivation process is capable of generating floor plan designs. G. Ślusarczyk [23] proposed a framework for supporting the design process by defining design requirements over graph-based representations of designs. First, hierarchical layout graph grammars are used to generate hierarchical layout hypergraphs (HL-graphs) that represent designs; then, local and global graph requirements are defined over HL-graphs, which correspond to design constraints. The proposed ontological interpretations transform first-order and monadic second-order logic formulas expressing design criteria into equivalent local and global graph requirements. The satisfiability of graph requirements by representations of designs allows for checking correctness of design solutions. In subsequent research, G. Ślusarczyk et al. [24] proposed CP-graph grammars to support building layout design, where the grammar rules are combined with semantic-driven embedding transformations and the derivations in this type of grammars are defined. The possibility of relating attributes of right-hand sides to that of the left-hand sides enables the system to capture parametric modelling knowledge. The proposed generative method allows the system to automatically model alternative floor layouts with similar structures but different geometry and parameters, which can be easily adapted to different use case scenarios and environmental conditions.

Apart from the architectural design, graph grammar has been applied to different

fields, including mechanical parts description [6], XML validation [16], cluster analysis [22], entity-relationship (E-R) diagram validation [11], and Web pattern recognition and validation [15]. Overall, graph grammar is a powerful tool for defining and validating graph models, hence the generative design method in this paper is proposed within the framework of graph grammar.

Because patterns are composed of various styles of shapes, there is a positional correlation between each shape. The structure of patterns is generated through graph grammar, which abstracts the positional relationships between various shapes. Then we convert the node-edge graph generated by graph grammar into shapes through node transformation rules, enabling graph grammar to generate shapes and draw patterns. Moreover, graph grammar parsing can check whether a target pattern belongs to the pattern set defined by the rules.

3. Improved generative design approach framework

VCGG is divided into virtual-node based discrete Coordinate Graph Grammar (vdCGG) and virtual-node based continuous Coordinate Graph Grammar (vcCGG) based on different granularity descriptions of spatial semantics. Due to the strict coordinate matching mechanism required in this approach, we choose vcCGG as the basic framework. Below is the theoretical framework of the improved approach.

Definition 3.1. *A **directed graph** G on a given label set L is a 2-tuple (N, E). L consists of a virtual label set L_v and a real label set L_r, where L_r consists of a nonterminal label set L_{NT} and a terminal label set L_T. N is a node set and consists of a virtual node set N_V and a real node set N_r, where N_r consists of a nonterminal node set N_{NT} and a terminal node set N_T. E is a directed edge set.*

Mapping for G includes the following:

- $f_{NL} : N \to L$ is a mapping that assigns a label $l \in L$ to node $n \in N$;
- $f_{NC} : N \to R \times R$ is a mapping that assigns a 2D coordinate $c \in R \times R$ to node $n \in N$;
- $f_{EN_s} : E \to N$ is a mapping that assigns the start node to directed edge $e \in E$;
- $f_{EN_e} : E \to N$ is a mapping that assigns the end node to directed edge $e \in E$.

Definition 3.2. *A **production** p: $G_L := G_R$ is made up of a left-hand-side (or left graph) G_L and a right-hand-side (or right graph) G_R. For a production, there exists a bijection $f_{NN} : G_L.N_v \leftrightarrow G_R.N_v$ between $N_v \in G_L$ and $N_v \in G_R$, where $G_L.N_v$ is a virtual node set N_v of G_L and $G_R.N_v$ is a virtual node set N_v of G_R.*

A production also satisfies the following conditions:

- $\forall n((n \in G_L.N_v) \Rightarrow (f_{NC}(n) = f'_{NC}(f_{NN}(n))))$, where f_{NC} is a mapping that assigns a coordinate to node $n \in G_L$ and f'_{NC} is a mapping that assigns a coordinate to $n \in G_R$;

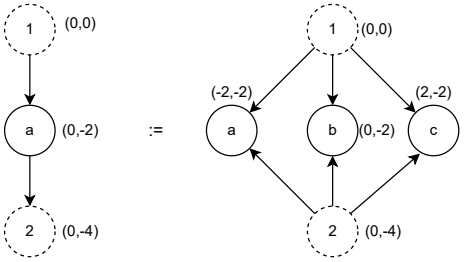

Fig. 1. vcCGG production.

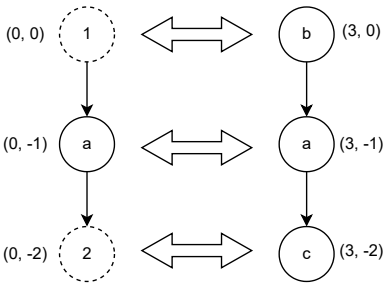

Fig. 2. The isomorphic graphs in vcCGG.

- $\forall n((n \in G_L.N_v) \Rightarrow (f_{NL}(n) = f'_{NL}(f_{NN}(n))))$, where f_{NL} is a mapping that assigns a label to node $n \in G_L$ and f'_{NL} is a mapping that assigns a label to $n \in G_R$;
- $\forall n_1, n_2((n_1, n_2 \in G_L.N_v) \wedge (n_1 \neq n_2) \Rightarrow (f_{NL}(n_1) \neq f_{NL}(n_2)))$;
- $\forall n_1, n_2((n_1, n_2 \in G_R.N_v) \wedge (n_1 \neq n_2) \Rightarrow (f'_{NL}(n_1) \neq f'_{NL}(n_2)))$.

VcCGG stipulates that there is a bijection between the virtual node sets at G_L and G_R, and the corresponding nodes have the same labels and coordinates. In addition, to avoid ambiguity during graph embedding, each virtual node in the same graph must have a unique label, which can be represented by a unique integer.

For example, Fig. 1 is a legal vcCGG production, where the dashed circle represents the virtual nodes and the solid circle represents the real nodes. There is a bijection between the left and right graphs of the production, and the corresponding nodes have the same labels '1', '2' and equal coordinates (0, 0) and (0, 4).

Definition 3.3. *Let G and Q be directed graphs. G and Q are* **isomorphic**, *denoted as $G \approx Q$, if and only if the following conditions hold:*

- *There exists a bijection between the nodes of G and Q, namely, $f_{NN} : G.N \leftrightarrow Q.N$;*

- *There exists a bijection between the edges of G and Q, namely,* $f_{EE} : G.E \leftrightarrow Q.E$*;*
- $\forall n((n \in G.N) \vee (n \in Q.N) \Rightarrow (f_{NL}(n) \in L_v) \vee (f'_{NL}(f_{NN}(n)) \in L_v) \vee (f_{NL}(n) = f'_{NL}(f_{NN}(n))))$*, where* f_{NL} *is a mapping that assigns a label to node* $n \in G$*;* f'_{NL} *is a mapping that assigns a label to* $n \in Q$*;*
- $\forall e((e \in G.E) \vee (e \in Q.E) \Rightarrow (f_{NN}(f_{EN_s}(e)) = f_{EN_s}(f_{EE}(e))))$*;*
- $\forall e((e \in G.E) \vee (e \in Q.E) \Rightarrow (f_{NN}(f_{EN_e}(e)) = f_{EN_e}(f_{EE}(e))))$.

When determining whether a pair of graphs satisfies the isomorphic condition, virtual nodes have a higher abstract degree than real nodes and can match any labeled node. Fig. 2 is an example of graph isomorphism in vcCGG, where all nodes and edges satisfy a bijective relationship. Real node 'a' and the corresponding nodes must have the same label, while virtual nodes '1' and '2' can match any labeled node. In Fig. 2, node '1' matches 'b' and node '2' matches node 'e'.

Definition 3.4. *Let G be a directed graph referred to as the host graph and Q be the subgraph of G. Let* $G_{L|R}$ *be the left or the right hand-side of a production. Q is called a* **redex** *of G with respect to* $G_{L|R}$*, denoted as* $Q \in \text{redex}(G, G_{L|R})$ *if and only if the following conditions hold:*

- $Q \approx G_{L|R}$*;*
- $\forall n((n \in Q.N \wedge ((f'_{NL}(f_{NN}(n)) \in L_r)) \Rightarrow (d_s(n) = d_s(f_{NN}(n))) \wedge (d_e(n) = d_e(f_{NN}(n))))$*;*
- $\forall n_1, n_2((n_1, n_2 \in Q.N) \Rightarrow (f_{NC}(n_1) - f_{NC}(n_2) = f'_{NC}(f_{NN}(n_1)) - f'_{NC}(f_{NN}(n_2))))$.

The nodes of a redex could be divided into two types: the nodes matched by the virtual nodes (context nodes) of the production, and the nodes matched by the non-virtual nodes (inner nodes) of the production. All the edges between the redex and the rest host graph are only allowed to be connected with the former type of nodes.

Definition 3.5. *A* **L/R application** *to graph G is a process that generates graph G′ using production p:* $G_L := G_R$*, denoted as* $G \rightarrow^p G'$*(L-application) or* $G \rightarrow^p G'$*(R-application).*

The L-application in vcCGG is as follows:

1. Generate an instance of the production as a copy of the production.
2. Translate the coordinates of the instance's G_R by the offset between any matched nodes in the redex Q and G_L.
3. Delete edges in the redex Q and nodes that match the real nodes in G_L from the host graph.
4. According to the mapping between the virtual node of G_L and the redex Q, glue the virtual node of G_R to the corresponding node in the redex Q and remove the virtual label from the host graph.

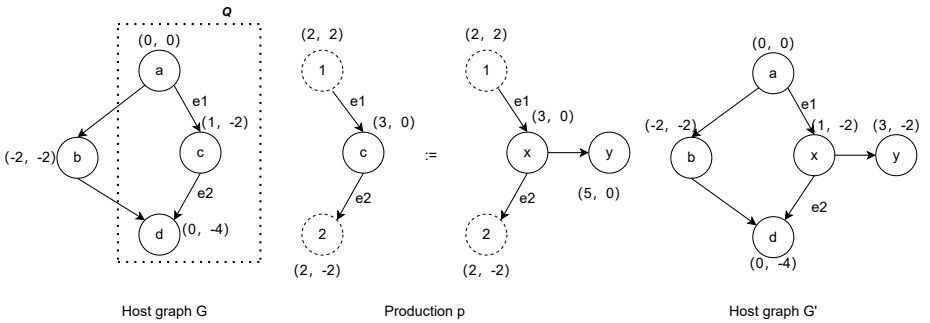

Fig. 3. New host graphs generated by a production.

Fig. 3 depicts an L-application process that generates new host graph G' using production p: $G_L := G_R$.

1. Generate an instance of production p.
2. Find a redex of G with respect to G_L: In the host graph G, we denote a graph in the dashed box as graph Q. $Q \approx G_L$ and the coordinate differences of the corresponding nodes are (2, 2), so $Q \in \text{redex}(G, G_{L|R})$.
3. Subtract all node coordinates of G_R (2, 2).
4. Delete edge 'e1', 'e2' and node 'c' from G.
5. Glue virtual node '1' of G_R to real node 'a' of G and virtual node '2' of G_R to real node 'd' of G; and remove the virtual label from the host graph.

Definition 3.6. *A* **node transformation rule** *is a 4-tuple(cset, cpoint, ops, parm), where*

- *cset is a set of coordinates as the points to represent a shape;*
- *cpoint is the mean point of cset;*
- *ops is the operations performed on the cset, such as translation, rotation, scaling, etc.;*
- *parm is the parameter of the ops, such as the offset of translation or the angle of rotation.*

Given a node transformation rule, the node transformation application is a process that draws the outline of a shape from the perspective of the user using node transformation rules. Below are the steps for a node transformation application:

1. Draw a shape based on the outline described by a node's cset, and make the cpoint coincide with the node. As shown in Fig. 4, a node transformation rule is to transform a node into a rectangle. Use this node transformation rule for node A and B: make the cpoint of this rectangle coincide with node A and B, and transform edge e_1 connecting A and B to line segment l_1;

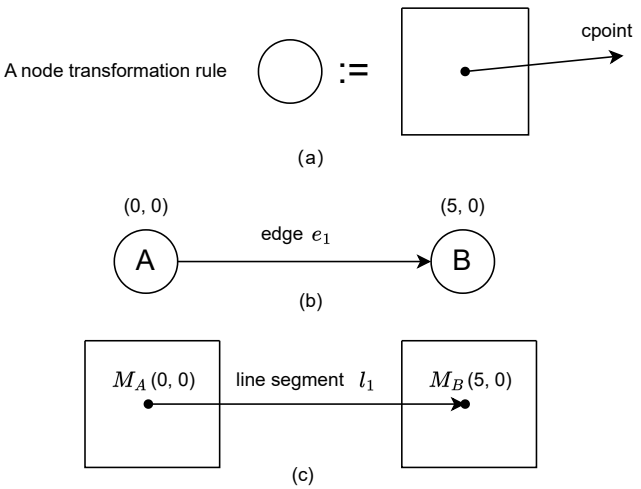

Fig. 4. Demonstration figure of step 1.

2. As shown in Fig. 5, deform the shape by the following methods according to ops and parm:
   (a) Translation: Let A be a shape, and the position of A can change along the X and Y axes, i.e.,
   $\forall(x, y) \in A, (x', y') = (x + a, y + b)$,
   where a is the distance that the position of A changes on the X axes and b is the distance that the position of A changes on the Y axes.
   (b) Scale: Let A be a shape that can expand or shrink in a certain proportion, i.e.,
   $\forall(x, y) \in A, \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$, where S is the factor by which shape A expands or shrinks.
   (c) Stretch: Let A be a shape that can be elongated or shortened along the X and Y axes. Specifically, if the factors of elongation or shortening along the X and Y axes are equal, A can be considered to be scaled, i.e.,
   $\forall (x, y) \in A, \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$,
   where Sx is the factor by which A is elongated or shortened along the X axes and Sy is the factor by which A is elongated or shortened along the Y axes.
   (d) Rotate:Let A be a shape that can rotate $\theta\,(0 < \theta < 2\pi)$ counterclockwise around the cpoint $M_A(X_A, Y_A)$, i.e.,

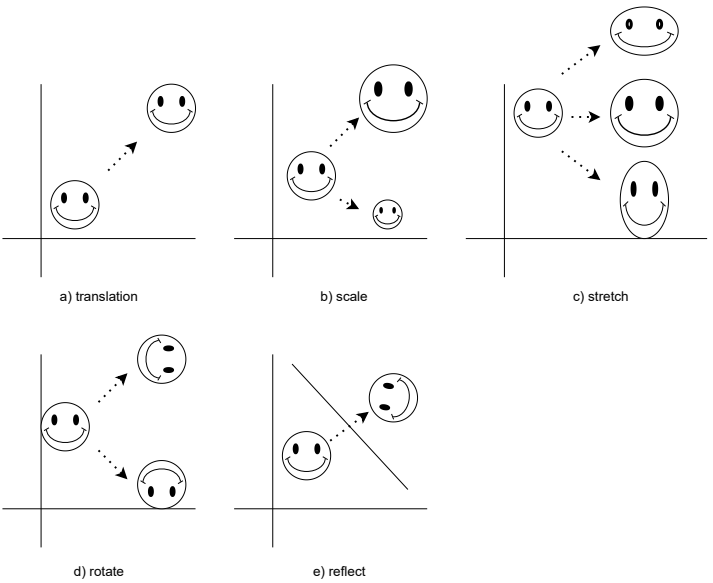

Fig. 5. A new shape formed by 5 operations.

$\forall\,(x,y) \in A,\ (x',y') = ((X_A - x)\,cos\theta - (Y_A - y)\sin\theta + X_A,\ (X_A - x)\sin\theta + (Y_A - y)\cos\theta + Y_A)$.

(e) Reflect: Let A be a shape. $\forall l : PX + QY + M = 0(P^2 + Q^2 > 0)$, new shape A' is a mirror image of A across line l, i.e.,
$\forall\,(x,y) \in A, (x',y') = \left(x - \frac{2P(Px+Qy+M)}{P^2+Q^2}, y - \frac{2Q(Px+Qy+M)}{P^2+Q^2}\right)$.

3. Render the shape from the user's perspective based on the outline described by the cset through its own operations.
4. Adjust the position of the shape based on the attributes of the line segment l_1.

Definition 3.7. *For shape A and shape B, A and B are* **separated** *if and only if* $\exists l : Px + Qy + M = 0(P^2 + Q^2 > 0)$*, A and B are on both sides of line l, as shown in Fig. 6.*

As shown in Fig. 7, for shape A and B, M_A is the cpoint of A and M_B is the cpoint of B. M_A and M_B are connected through a directed line segment l_{AB}, where M_A is the start point of l_{AB} and M_B is the end point of l_{AB}. The position of M_A will change according to the attribute of l_{AB}, and the position of A will be changed following the changes in M_A position. The attribute of l_{AB} is 'far from d' or 'near d', where d is the distance at which the M_A position changes. When using node transformation rules to transform

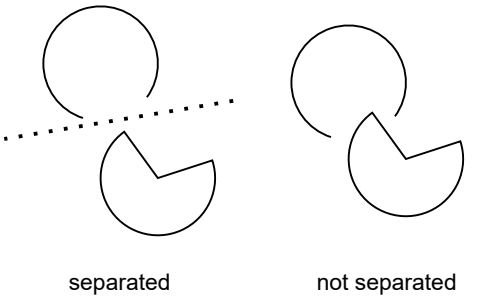

Fig. 6. The two shapes are separated or not.

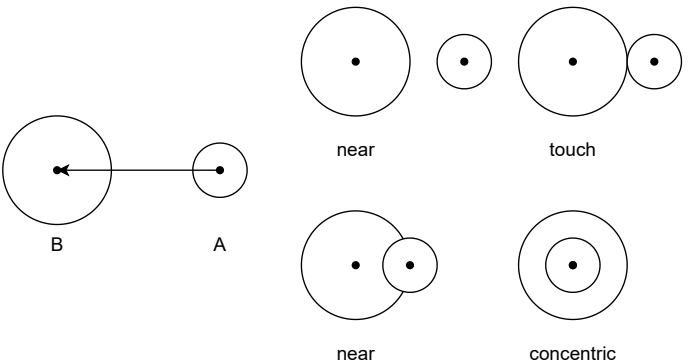

Fig. 7. A is near B; A touches B; A is concentric to B.

node A and B into shape A and B, it is necessary to ensure that they are separated. Therefore, if the attribute of l_{AB} is 'far from d', regardless of the value of d, A and B are still separated. So, we won't limit the value of d when the attribute of l_{AB} is 'far from d'.

Definition 3.8. *If the attribute of l_{AB} is 'near d', A may* **touch** *B or be* **concentric** *with B during the process of changing the position of A.*

- *Touch: $A.cset \wedge B.cset \neq \emptyset$ for the first time;*
- *Concentric: M_A coincides with M_B.*

For convenience, when users want A to touch B or be concentric with B, they can

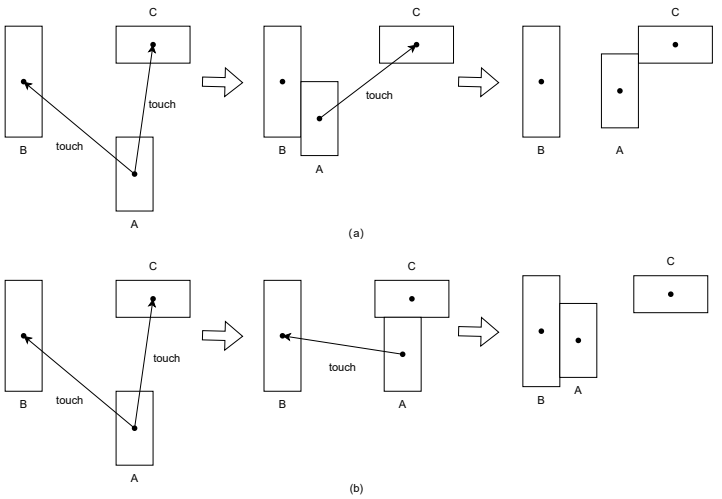

Fig. 8. The final position of A will change due to the order of touch B or C.

set the l_{AB} attribute to 'touch' or 'concentric'. Before the position of M_A changes, make $D_{\max} = |M_A - M_B|$. So, $0 < d \leq D_{\max}$ when the attribute of l_{AB} is 'near d'.

As shown in Fig. 8, for shape A, when M_A is the starting point of two or more directed line segments, the position of A must to be changed at least twice, and different changing sequences can lead to different positions. As shown in the Fig. 8, A needs to touch both B and C, and the final position of A will change based on the order of it touches B or C. Therefore:

- When the X coordinate of the end nodes is different, the position of start node first changes toward the end node with a smaller X coordinate;
- When the X coordinate of the end nodes is the same, the position of start node first changes toward the end node with a smaller Y coordinate.

4. An example on rabbit pattern

This section gives an example to illustrating an application of the improved approach, where a set of designed productions and node transformation rules are used to generate a section of the Cloud & Bunny rabbit pattern from Emma Talbot. Emma is passionate about mixed media research and enjoys using various media to create textures, patterns, and collages to integrate into her artistic creations. The Cloud & Bunny rabbit pattern is composed of simple geometric shapes such as arcs, rectangles, triangles, etc., forming

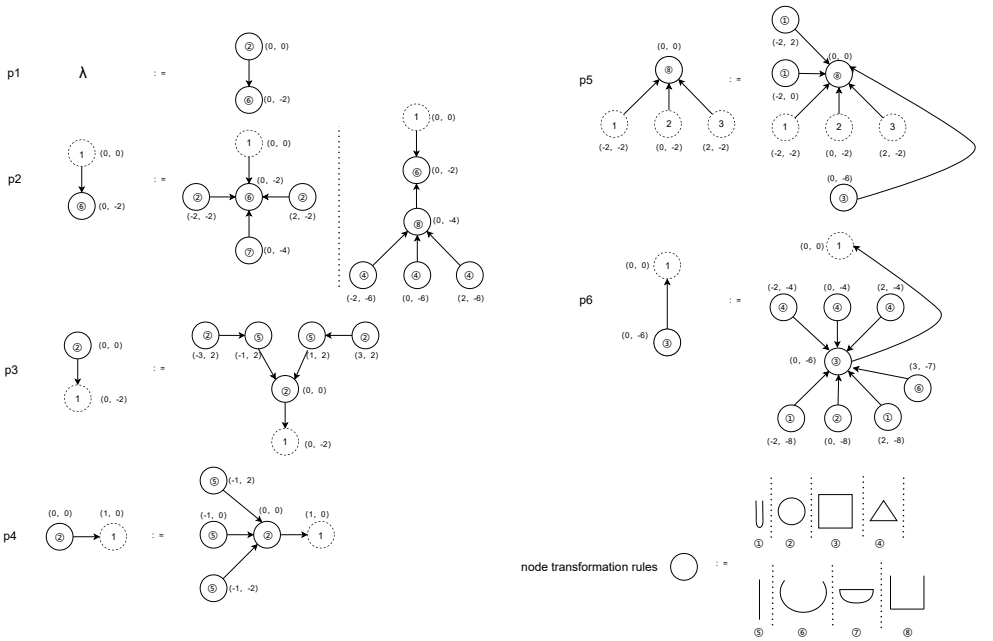

Fig. 9. Productions for a bunny rabbit.

patterns of rabbits, flowers, and clouds. In this paper, a rabbit pattern is selected as the generated pattern. Fig. 9 shows a set of vcCGG productions and eight node transformation rules as a grammar set for the rabbit pattern, where the vcCGG productions are used for the abstract models of pattern and node transformation rules describe physical layouts. For the vcCGG productions, the initial symbol 'λ' denotes the beginning of graph grammar. 'λ' is used to generate the right graph of p1 through production p1 and then generate the target structure of the pattern based on the remaining productions p2-p6. For the productions in Fig. 9, virtual nodes, which are represented by a dashed circle and labeled '1', '2', and '3', are used to match coordinates; real nodes, which are represented by a solid circle and labelled '①', '②', and '③', are converted into shapes. For the node transformation rules in Fig. 9, we set eight shapes to generate the final pattern, including circle, rectangle, triangle, etc.

Fig. 10 shows a process of generating a rabbit pattern using the productions and node transformation rules above. When using an L-application to generate the structure of the target pattern, an attribute is assigned to each generated edge. The attribute can be 'near', 'touch' or 'concentric'. If the attribute is 'near', the distance needs to be

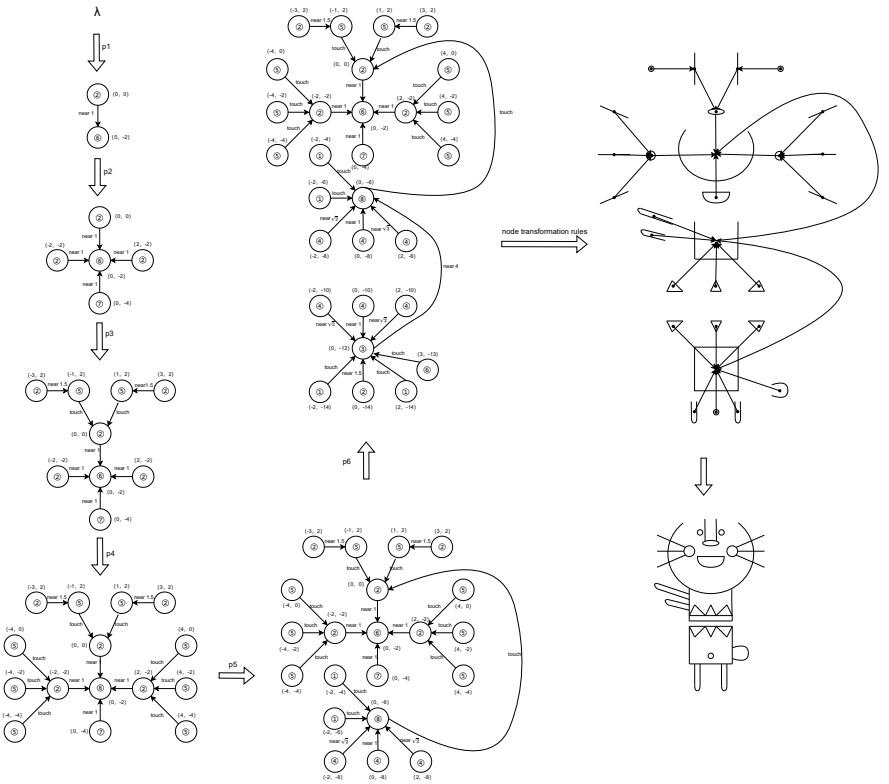

Fig. 10. Generation of a bunny rabbit.

given as parameter. When using node transformation rules for the final generated node-edge graph, each node is traversed and converted into a shape based on the associated label. Then, each edge is traversed, the position of each shape is adjusted based on the attribute of each edge, and the target pattern is ultimately obtained.

5. Comparisons with other generative design approaches

In this section, we compare our approach proposed in this paper with shape grammar, edge transformation grammar [12] and CP-graph grammar [24]. Shape grammar is a design inference approach based on rules, using simple shapes as basic elements to establish the rules for the generation of complex shapes. Edge transformation grammar

defines shape rules to transform edges into shapes by shape applications. The CP-graph grammar is used to automatically generate CP-graphs corresponding to new layout designs with non-geometrical properties (like sizes, areas) specified by graph attributes.

As Table 1 shows, these approaches can all design shapes through derivation. However, when drawing patterns using shape grammar, different shapes of a pattern are related only in terms of position and have no semantic relations. Therefore, it is difficult to formally analyze the generated pattern. Our approach based on vcCGG can formally validate a target pattern to determine whether it belongs to the pattern generated by the specified rules by combining node transformation applications and L-applications. Moreover, after designing the transformation rules for shape grammar, edge transformation grammar and CP-graph grammar, they are unable to adjust the size and position of the shape, resulting in a lack of position and size variability. However, for our approach, after generating the structure of the target pattern through vCGG, the nodes which are converted into shapes according to the node transformation rules can adjust the size and position of themselves. Therefore, in terms of position and size variability, our approach is superior to shape grammar and edge transformation grammar.

Tab. 1. Comparison between approach in this paper, shape grammar, edge transformation grammar and CP-graph grammar.

Approach	Derivation	Parsing	Positional and size variability
Our approach	✓	✓	✓
Shape grammar	✓	×	×
Edge transformation system	✓	✓	×
CP-graph grammar	✓	✓	×

6. Conclusions

When designers use shape grammar to generate patterns, there are no semantic relations among the various shapes that make up the pattern or the small patterns that make up the large patterns. Therefore, it is difficult to formally analyze the generated patterns. In addition, graph grammar is primarily used for generating and analyzing abstract models of visual languages. There is a significant gap between the generated node-edge graphs and the visual representation of shapes, so few researchers have applied these concepts in the design field.

This paper proposes an improved generative design approach for pattern drawing, which introduces node transformation rules in the framework of vcCGG. First, the structure of the target pattern is generated through vcCGG, and then the nodes are converted into shapes according to the node transformation rules. Finally, the position of each

shape is adjusted based on the edge attributes, and the target pattern is generated. In this approach, L-applications and node transformation rules are set in advance for drawing patterns, and a target pattern can be formally analyzed to determine whether it is a pattern generated based on the specified rules.

In the future, we plan to improve the theoretical framework of the improved approach and consider adding gray values to the node transformation rules. If it goes well, we plan to add RGB to it so that the improved approach can be used to design the colored patterns. Moreover, we plan to develop a support system for this approach with a friendly GUI for end users to design graph productions and node transformation rules. The system platform will provide support for grammatical operations and the implementation of related applications.

Acknowledgement

This work was supported by the National Natural Science Foundation of China within the grant No. 62002155 and the National Key Research and Development Program of China within the grant No. 2022YFB3305504.

References

[1] M. Agarwal and J. Cagan. A blend of different tastes: The language of coffeemakers. *Environment and Planning B: Planning and Design*, 25:205–226, 1998. doi:10.1068/b250205.

[2] H. Bunke. Graph grammars as a generative tool in image understanding. In: *Graph-Grammars and Their Application to Computer Science*, pp. 8–19, 1983. doi:10.1007/BFb0000096.

[3] H. H. Chau. *Preserving Brand Identity in Engineering Design Using a Grammatical Approach*. Ph.D. thesis, The University of Leeds, School of Mechanical Engineering, and Keyworth Institute of Manufacturing and Information Systems, 2002. https://www.researchgate.net/publication/286452884_Preserving_brand_identity_in_engineering_design_using_a_grammatical_approach.

[4] X. Chen, A. McKay, A. de Pennington, and H. H. Chau. Package shape design principles to support brand identity. *Proc. 14th IAPRI World Conference on Packaging*, pp. 1–14, 2004. https://www.researchgate.net/publication/267797410_PACKAGE_SHAPE_DESIGN_PRINCIPLES_TO_SUPPORT_BRAND_IDENTITY.

[5] G. Díaz, R. F. Herrera, F. Muñoz-La Rivera, and E. Atencio. Generative design for dimensioning of retaining walls. *Mathematics*, 9(16):1918, 2021. doi:10.3390/math9161918.

[6] M. Flasiński. Use of graph grammars for the description of mechanical parts. *Computer-Aided Design*, 27:403–433, 1995. doi:10.1016/0010-4485(94)00015-6.

[7] S. Garcia and L. Romao. A design tool for generic multipurpose chair design. In: *Proc. Computer-Aided Architectural Design Futures*, pp. 600–619, 2015. doi:10.1007/978-3-662-47386-3_33.

[8] H. Göttler, J. Günther, and G. Nieskens. Use graph grammars to design CAD-systems! In: *Graph-Grammars and Their Application to Computer Science*, pp. 396–410, 1990. doi:10.1007/BFb0017402.

[9] M. Lee, Y. Park, H. Jo, K. Kim, S. Lee, et al. Deep generative tread pattern design framework for efficient conceptual design. *Journal of Mechanical Design*, 144(7):011703, 2022. doi:10.1115/1.4053469.

[10] Y. Liu and Y. Fan. VCGG: Virtual-node based spatial graph grammar formalism. *Journal of Software*, 32:3669–3683, 2021. doi:10.13328/j.cnki.jos.006164.

[11] Y. Liu, X.-Q. Zeng, and Y. Zhu. Application of graph grammar EGG to design of ER diagrams. *Computer Engineering and Design*, 2014(3):1071–1075, 2014. https://api.semanticscholar.org/CorpusID:63040768.

[12] Y. Liu, Y. Zhou, F. Yang, and H. Sun. An enhanced grammatical approach for graph drawing. In: *Conf. International Conference on Artificial Intelligence, Virtual Reality, and Visualization AIVRV 2022*, p. 1258803, 2023. doi:10.1117/12.2667201.

[13] M. Pugliese and J. Cagan. Capturing a rebel: modeling the Harley-Davidson brand through a motorcycle shape grammar. *Research in Engineering Design*, 13:139–156, 2002. doi:10.1007/s00163-002-0013-1.

[14] C. Qian, R. Tan, and W. Ye. An adaptive artificial neural network-based generative design method for layout designs. *International Journal of Heat and Mass Transfer*, 184:122313, 2022. doi:10.1016/j.ijheatmasstransfer.2021.122313.

[15] A. Roudaki, J. Kong, and K. Zhang. Specification and discovery of web patterns: a graph grammar approach. *Information Sciences*, 328:528–545, 2016. doi:10.1016/j.ins.2015.08.052.

[16] G. Song and K. Zhang. Visual xml schemas based on reserved graph grammars. In: *Conf. International Conference on Information Technology: Coding and Computing. ITCC 2004*, pp. 687–691, 2004. doi:10.1109/ITCC.2004.1286546.

[17] G. Stiny. Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design*, 7(3):343–351, 1980. doi:10.1068/b070343.

[18] G. Stiny and J. Gips. Shape grammars and the generative specification of painting and sculpture. In: *Proc. Conf. International Federation for Information Processing IFIP 1971*, pp. 125–135, 1971. https://api.semanticscholar.org/CorpusID:36431081.

[19] X. Wang, Y. Liu, and K. Zhang. A graph grammar approach to the design and validation of floor plans. *The Computer Journal*, 63:137–150, 2019. doi:10.1093/comjnl/bxz002.

[20] S. Wannarumon, P. Pradujphongphet, and I. Bohez. An approach of generative design system: Jewelry design application. *IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 1329–1333, 2014. doi:10.1109/IEEM.2013.6962626.

[21] Y. Yu, T.-C. Hong, A. Economou, and G. Paulino. Rethinking origami: A generative specification of origami patterns with shape grammars. *Computer-Aided Design*, 137:103029, 2021. doi:10.1016/j.cad.2021.103029.

[22] K.-B. Zhang, M. A. Orgun, and K. Zhang. A prediction-based visual approach for cluster exploration and cluster validation by HOV3. In: *Proc. Knowledge Discovery in Databases. PKDD 2007*, pp. 336–349, 2007. doi:10.1007/978-3-540-74976-9_32.

[23] G. Ślusarczyk. A graph grammar approach to the design and validation of floor plans. *Computer-Aided Design*, 95:24–39, 2017. doi:10.1016/j.cad.2017.09.004.

[24] G. Ślusarczyk, B. Strug, A. Paszyńska, E. Grabska, and W. Palacz. Semantic-driven graph transformations in floor plan design. *Computer-Aided Design*, 158:103480, 2023. doi:10.1016/j.cad.2023.103480.

Yufeng Liu received the Ph.D. degree from Hehai University, Nanjing, China. He is now an associate professor in College of Information Engineering, Nanjing University of Finance and Economics, China. His main research interests include software engineering, machine learning and visual language.

Yangchen Zhou received the bachelor degree from Nanjing University of Finance and Economics, China. He is now studying for a master's degree at College of Information Engineering, Nanjing University of Finance and Economics. His main research interests include graph grammar and generative design.

Fan Yang received the Ph.D. degree from Changchun University of Science and Technology, Changchun, China. He is now an associate professor in College of Information Engineering, Nanjing University of Finance and Economics, China. His main research interests include multimodal data fusion, machine learning, and deep learning.

Song Li received the Ph.D. degree from Anhui University, Hefei, China. He is now a lecturer in College of Information Engineering, Nanjing University of Finance and Economics, China. His main research interests include cloud security and applied cryptography.

Jun Wu received the Ph.D. degree from Nanjing University, Nanjing, China. He is now a lecturer in College of Information Engineering, Nanjing University of Finance and Economics, China. His main research interests include computational economics, algorithmic game theory, and mechanism design.