B. Dave, A. I. Li, N. Gu, H.-J. Park (eds.), *New Frontiers: Proceedings of the 15th International Conference on Computer-Aided Architectural Design Research in Asia CAADRIA 2010*, 127–136.

A FRAMEWORK TO INTEGRATE GENERATIVE DESIGN TECHNIQUES FOR ENHANCING DESIGN AUTOMATION

NING GU
School of Architecture and Built Environment, University of Newcastle, Australia, ning.gu@newcastle.edu.au

VISHAL SINGH
Design Lab, University of Sydney, Australia
vsin9057@mail.usyd.edu.au

and

KATHRYN MERRICK
University of New South Wales, Australia, k.merrick@adfa.edu.au

Abstract. This paper presents and demonstrates a computational framework that facilitates the integration of different generative design techniques to enhance design automation. The framework is based on the evaluation and comparison of four main generative design algorithms. Effectiveness of the framework is demonstrated through an example scenario. Compared to most existing generative design systems that are based on one of the techniques, which often bias the generative design process in a certain direction, new generative design systems by applying the proposed framework will provide the trigger at each stage as demonstrated in the example scenario for the designer to perceive the emergent designs from different viewpoints. This advantage will enhance design generation and automation by assisting the designer in making more informed decisions in understanding and selecting the suitable generative techniques for different design needs.

Keywords. Generative design systems; shape grammars; L-systems; cellular automata; genetic algorithms.

1. Introduction

Design automation is an important research area in the architecture, engineering and construction (AEC) domain, having the potential in significantly cutting time, cost, as well as other design and human resources involved in a project life cycle. Generative design systems are essential computational approaches for achieving design automation. This paper presents and demonstrates a computational framework for enhancing design automation by facilitating the use of different generative design techniques for different needs. Most existing generative design systems are developed based on one of the generative design techniques. This limitation often compromises the performance of the system in certain aspects for although there are overlaps and similarities across these generative techniques, each of them appears to be more suitable than others for specific design tasks.

This paper aims to bridge this gap through the development of a computational framework that facilitates the integration of different generative design techniques to enhance design automation. The framework is based on the evaluation and comparison of four main generative design algorithms: shape grammars (Stiny and Gips, 1972), L-systems (Lindenmayer, 1968), cellular automata (Wolfram, 2002), and genetic algorithms (Holland, 1975). Effectiveness of the framework is demonstrated through an example scenario. The framework enables designers to produce different design languages rather than hand crafting individual designs. The actual design tasks are carried out and (semi-)automated by the system according to the design knowledge or constraints set by the designers, during real-time interactions with the system. Compared to most existing generative design systems that are based on one of the techniques, which often bias the generative design process in a certain direction, new generative design systems by applying the proposed framework will provide the trigger at each stage as demonstrated in the example scenario for the designer to perceive the emergent designs from different viewpoints. This advantage will enhance design generation and automation by assisting the designer in making more informed decisions to select the “right” techniques for the “right” tasks.

2. Literature review of generative design techniques

Various computational approaches have been developed to support design automation. These generative design systems range from simple rule based systems that produce random outcomes to more advanced AI based systems that try to reproduce aspects of human design processes. Design is a co-evolutionary process where part of the knowledge and understanding of the

goals and solutions is generated as the design progresses (Schön, 1992). Gero (1998) describes design as a situated activity. A situated stance emphasises the designers' interaction with the design representation and how the designers' past experience and the environment influence the emergent design solutions. A situated view suggests that the same designers may generate different solutions in different situations. Based on this implicit assumption various creative design techniques such as brainstorming, TRIZ and morphological charts have been developed to modify the designer's situatedness and facilitate greater design exploration. Thus, generative design techniques that are applied to produce emergent forms also provide new ways for designers to see the design problem and the solution space, enriching the design process and the design outcome.

2.1. OVERVIEWS OF THE FOUR GENERATIVE DESIGN TECHNIQUES

The four commonly used generative design techniques are reviewed below.

Shape grammars (SG). SG is a set of shape rules that can be applied to generate a set or language of designs (Knight, 2000). SG is both descriptive and generative. The rules themselves are the descriptions of the forms of the generated designs. SG can be used as (1) design tools to generate vast varieties of design languages, (2) analysis tools for existing designs in order to better understand these designs and to generate shape rules that produce the designs and other similar designs. Stiny (1980) defines four basic components of SG: (1) S: a finite set of shapes, (2) L: a finite set of symbols, (3) R: a finite set of shape rules, and (4) I: initial shape. SG is typically used for generating patterns or 2D compositions, spatial layouts, and 3D compositions (Duarte, 2005; Stiny and Mitchell, 1978; Halatsch et al., 2008).

L-systems (LS). LS (Lindenmayer Systems) are mathematical algorithms first developed by biologist Lindenmayer (1968). Essentially, LS is a set of production rules that can be applied recursively through string rewriting. In design, LS is in principle a design grammar. LS has been distinguished from SG as being operated on strings (the symbolic representation of the shape) rather than directly on the shape itself (Parish and Muller, 2001). LS is typically used to generate repetitive patterns, fractals and natural organic forms like plants (Lindenmayer and Rozenberg, 1972), textures, etc. In design, LS has been used to generate road networks (Parish and Muller, 2001), city planning (Kelly and McCabe, 2007) and building forms (Mueller et al., 2006).

Cellular automata (CA). CA is a collection of cells on a grid of a specified shape that evolved over time according to a set of rules driven by the state of neighbouring cells (Wolfram, 2002). The complexity of the system is partly determined by the type of grid, which can range from one-dimensional

lines, two-dimensional grids to Cartesian grids in arbitrary dimensions. CA is always context-sensitive and defined in terms of cell states and states of the neighbouring cells. CA tends to follow the approach of "form follows function" because the emergent form is the result of the desired functionality (cell states). CA is useful for study of social effects that can be simulated using the neighbouring conditions, e.g., urban design (Herr and Kvan, 2005), zoning and building massing (Krawczyk, 2002).

Genetic algorithms (GA). GA is inspired from natural evolutionary processes. GA uses the analogs of evolutionary operators on a population of states in a search space to find those states that optimised by a fitness function. The search space consists of character strings of fixed or variable length (genotype) composed of the elements of a given alphabet (allele). The genotype space is mapped onto another (phenotype) search space. The fitness function is defined as a function of a state in the phenotype space. Various modifications of GA have been reported in literature. In design they have been used for design optimisation (Caldas, 2001), spatial layout (Gero and Kazakov, 1998), architectural forms (Caldas, 2001) and so on.

2.2. COMPARISON OF THE FOUR GENERATIVE DESIGN TECHNIQUES

Technical aspects. In general, each of these techniques defines a finite set of elements and/or symbols that can be operated upon using a finite set of production rules and/or operators. CA, LS and SG are generally represented in graphical forms that result from transformations and operations (i.e., addition, rotation, subtraction, etc.) on the initial elements, Further, there have been instances where some of these techniques are shown to be equivalent, for example, CA and LS (Alfonseca and Ortega, 2000). There are also examples where some of these techniques are used in conjunction with other techniques to manipulate the terminal elements (Jacob, 1996) or derive rules (Speller et al., 2007).

Table 1 highlights each of these techniques in terms of their (1) components and rules to define and implement the system, (2) advantages in design generation, (3) limitations, and (4) design accuracy (predictability), which can relate to the level of designer intervention. It is important to note there are always exceptions, for example, even for SG and LS it is possible to have systems that are constrained enough to generate valid designs that may not require validations by the designer. However, greater the number of constraints, fewer is the opportunity for exploration and emergence.

Design aspects. Each technique is useful for design exploration in their own way. GA is mainly used for optimisation. The design quality in GA tends to increase in each generation. Unlike CA, which is always constrained and

context-sensitive, SG and LS can be designed with(out) constraints depending on the level of intervention desired by the designer. SG and LS typically follow "function follows form", i.e., once the form is generated, it is evaluated for functionalities. Table 2 compares the techniques in terms of (1) suitability to design phases and approaches for application, (2) type and complexity of the design problem, (3) characteristics of the design outcome.

Table 1: Technical aspects of the four generative design techniques.

	SG	LS	CA	GA
Comp'ts	Set of terminal shapes (SG) or symbols (LS), operators, and rules Initial shape.		Grid and cells Set of state rules Initial cell states	Genotypes Phenotypes Population Operations Fitness function.
Rules	Usually, one of the rules fires if conditions match LHS.	Usually, as many rules are applied at the same time as defined.	Very few rules relating to current states, relevant rules fire to change states.	Usually, one operation is applied at a time on a population sub-set.
Advt.	Geometric.	Symbolic.	Context sensitive Constrained Bottom-up.	Multiple solution alternatives Optimisation.
Lims	Trade off between usefulness v.s. emergence exploration.		Constrained by cell geometry and definition.	Near optimal solutions but rarely optimal.
Acurc.	In general, often requires post-generation analysis by user.		Depends on problem representation and solution interpretation.	Depends on choice of genotypes and the fitness function.

System development. GA can be implemented as a pure bottom-up design approach as once the genotypes, phenotypes and fitness function are identified the mechanism is fixed. However, GA's efficiency lies in their selections and representations. CA also tends to be easier to model because CA is generally defined in terms of states, e.g., the rules corresponding to the state changes in CA can be easily identified because the relationships between the neighbouring cells and potential configurations are easy to visualise exhaustively (the emergent configurations are not necessarily easy to visualise). Developing CA is straightforward because often it is used to simulate and explore social behaviours. Hence, a pure bottom-up approach works and emergent behaviours are generally considered to be valuable.

LS and SG are not clearly defined as much in terms of the states as often the emergent and unexpected qualities of the design outcome are desirable. Thus, in many cases, a re-engineering or backward tracking approach is adopted to generate terminal and non-terminal elements, and a set of production rules

that may generate the desired outcomes. In many cases, not all results are promised to be functional, and, hence, developing such systems may take a greater number of iterations. Thus, the development of LS and SG is rarely a purely bottom-up approach (Tapia, 1999).

Table 2. Design aspects of the four generative design techniques.

	SG	LS	CA	GA
Phase	Design exploration: space layout (SG), patterns and compositions.		Grid-based design and planning.	Design exploration, optimisation.
Approach	Emergent Function follows form Iterative and re-engineering based approaches.		Study of neighbouring effects, growing patterns, social phenomenon etc. Form follows function.	Design optimisation Combined and morphological designs.
Charact's D' Problem	Patterns Architectural forms and styles Usually 2D, 3D applicable.	Natural and organic forms, road networks, terrains, textures Usually 2D.	Urban planning, zoning, block design and massing Usually 2D, 3D applicable.	Component based designs Optimisation.
	Geometric Emergent and exploratory Solutions usually require validation.		Emergent and normative Context-sensitive Usually predictable solutions.	Optimised.

Table 3. Development issues of the four generative design techniques.

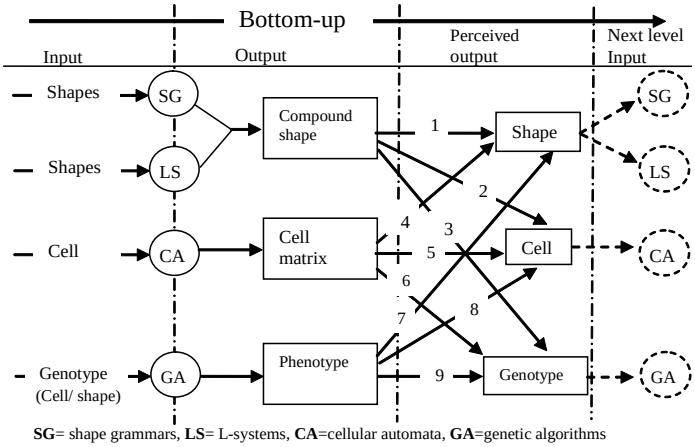
	SG	LS	CA	GA
User intervention	High level of user intervention to validate outcomes Iterative process.		Low level of user intervention once cell dimension, state rules and initial states are defined.	Low level of user intervention once fitness functions, genotypes and termination conditions are defined.
Challenges	Difficult to foresee expected outcomes. Hence, often requires re-engineering and backtracking.		Pure bottom-up Easy to identify possible cell states and neighborhood conditions.	Choice of representation can be challenging.

From the above summaries and comparisons, there are noticeable differences and similarities across the four generative design techniques. They are each unique in terms of design generation. Therefore they can be applied to solve different design problems and to suit different design preferences. Based on a situated view, the remainder of the paper presents and demonstrates a computational framework that facilitate the integration of the four different generative techniques for designers to explore the diverse solution spaces,

guiding them in selecting and applying the suitable generative design techniques for different design needs.

3. A framework to integrate generative design techniques

Fig. 1 shows a computational framework to assist designers in facilitating the integration of different generative design techniques. The framework is based on a bottom-up approach, where the designer starts with the smaller design components and tasks. However, following a similar approach, the framework can also facilitate the top-down approach, for example, given the site (e.g., a city), the designer may start with the initial zoning tasks. Once the first level zoning is completed, the sub-zones (e.g., a residential zone) can be further designed using the generative techniques. Thus, the designer is able to step back and select the alternative top-down and bottom-up approach as needed, in an iterative way.



SG= shape grammars, **LS**= L-systems, **CA**=cellular automata, **GA**=genetic algorithms

1: A compound shape generated at the lower level is used as a terminal shape for generation of further complex shapes at the next level.
2: A compound shape generated at the lower level suitable for repetition over a grid, where neighbouring relationships can be defined. In that case a cell envelope that encloses the shape can be used in CA at the next level.
3: Multiple compound shapes generated at the lower level can be used to generate further complex shapes using GA. In that case each compound shape is a genotype for GA for the next level.
4: Output of CA from the lower levels can be perceived as a shape. This shape can be used for complex shape generations at the next level.
5: Cell matrix at the lower level is part of a larger cell matrix.
6: Multiple cell matrices at the lower level can be combined together to generate new designs using GA at the higher level with each cell matrix as a genotype.
7: Outcome of GA at the lower level creates a compound shape, which is used as terminal shape for further shape generation at higher level.
8: Outcome of GA at lower level is suitable for repetition over a grid, where neighbouring relationships can be defined in CA at the next level.
9: Phenotypes at the lower levels are perceived as genotypes for GA at the higher

Figure 1. A framework to facilitate the integration of generative design techniques.

4. An example scenario for framework demonstration

The scenario is to demonstrate the use of the computational framework. The design brief is to generate a layout plan for an exhibition space of a given dimension and the design needs to meet the constraints of providing (1) four exhibitions zones with similar capacities and (2) adequate display panels.

The scenario adopts a bottom-up design approach. The designer starts by composing the panel and uses SG to generate a compound shape that satisfies the requirements. Next the designer may intend to use this as a modular unit. At this stage, the designer may adopt different techniques. As described in "1" in fig. 1, the designer adopts this compound shape as the terminal shape for generating further complex shapes using SG. Thus, in this case the designer uses SG for local module as well as for the overall panel layout. Alternatively, as shown in "2" in fig. 1, the designer may choose to repeat the same module by dividing the exhibition space into four similar zones. However, the designer may require defining neighbouring conditions such as how the panels in different cells/zones are related, e.g., the modules in each cell are arranged such that in each of the neighbouring cells the red panels face different directions. Thus, in the second case, the compound shape generated from SG in the lower level can be used to define cell characteristics at the higher level, where CA is used to generate the higher level space layout. These are illustrated in the top half of fig. 2. Similarly, in another case, the designer may start with CA by considering an envelope around each panel as the basic cell dimension. Using the neighbouring relationships a space layout in the local cell matrix is obtained. At the next higher level, the designer has the option to choose SG ("4" in fig. 1) or CA ("5" in fig. 1) as the technique to develop space layouts at the next higher level, as shown in the bottom half of fig. 2. Beyond what was illustrated in fig. 2, the designer may generate different modular alternatives from the basic panel using SG. Once alternatives are obtained the designer may create further space layouts using GA by combining modules and applying genetic operations. Further, the designer may also be able to configure these modules to optimise the desired function such as maximising the circulation space or maximising the number of available panels, and so on.

As fig. 2 demonstrates, the different generative techniques can be used in conjunction to generate different design alternatives to suit different design needs and preferences. The proposed framework facilitates the use of multiple generative techniques during the iterative design process by providing the trigger to change the designer's situated state. At each stage, once the designer has taken an action and looks at the outcomes, one has the opportunity to perceive the design task differently, which will assist one in making more

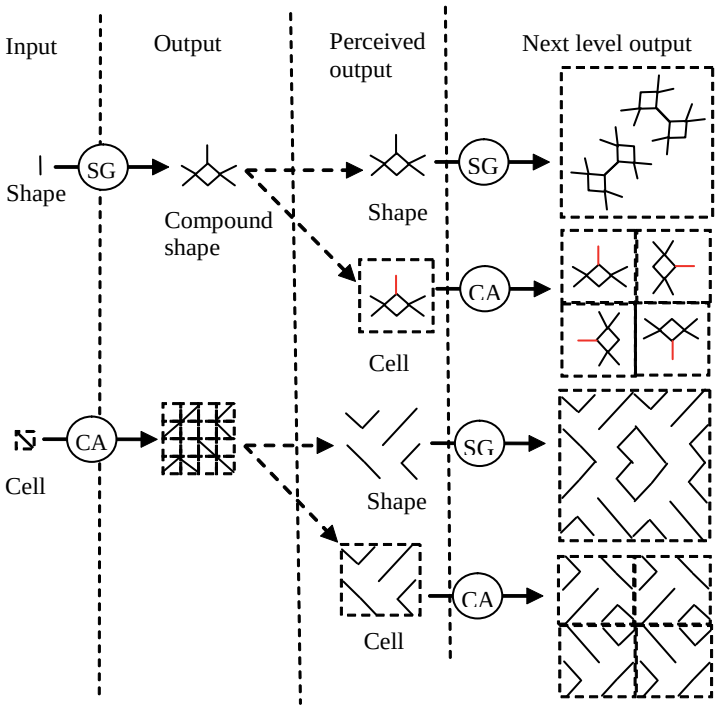


Figure 2: Design generation using multiple generative techniques.

informed decision in selecting the suitable generative design techniques to continue the design process.

5. Summary

The paper presents and demonstrates a computational framework to facilitate the integration of different generative design techniques. A review of the four main techniques namely shape grammars, L-systems, cellular automata and genetic algorithms is presented. The review shows that although there are overlaps and similarities across different techniques, each of them appears to be more suitable than others for specific design tasks. Hence, given the different strength of each technique, a computational framework that facilitates the integration of these techniques is developed. The framework enhances design automation by assisting the designer in marking more informed decision in understanding and selecting suitable generative techniques to suit their needs and preferences. Currently, existing generative design systems are based on one of the techniques, which often bias the generative design process in certain directions, i.e., designers working on a system based on SG tend to interact with and perceive the design artefacts as shapes. Generative design systems

by applying the proposed framework on the other hand provide the trigger at each stage for the designer to perceive the emergent designs from different viewpoints. The future project extension includes (1) the development of an example scenario that examines the framework in a top-down approach; (2) the development of an example scenario that examines the framework in a mix of bottom-up and top-down approaches; (3) prototype development of a trial system in Second Life.

References

Alfonseca, M. and Ortega, A.: 2000, Representation of some cellular automata by means of equivalent L-systems, *Complexity international*, **7**.

Caldas, L. G.: 2001, An evolution-based generative design system: using adaptation to shape architectural form, PhD thesis, MIT, Boston.

Duarte, J. P.: 2005, A discursive grammar for customizing mass housing: the case of Siza's houses at Malagueira, *Automation in construction*, **14**(2), 265–275.

Gero, J. S.: 1998, Conceptual designing as a sequence of situated acts, *in* I. F. Smith (ed.), *Artificial intelligence in structural engineering*, Springer, Berlin, 165–177.

Gero, J. S. and Kazakov, V.: 1998, Evolving design genes in space layout problems, *Artificial intelligence in engineering*, **12**(3), 163–176.

Halatsch, J., Kunze, A. and Schmitt, G.: 2008, Using shape grammars for master planning, *Proceedings of design computing and cognition 2008*, 655–673.

Herr, C. and Kvan, T.: 2005, Using cellular automata to generate high-density building form, *Proceedings of CAAD Futures 2005*, 249–258.

Holland, J.H.: 1975, *Adaptation in natural and artificial systems*, Michigan University Press.

Jacob, C.: 1996, Evolving evolution programs: genetic programming and L-systems, *Proceedings of genetic programming 1996*, Stanford, 107–115.

Kelly, G. and McCabe, H.: 2007, Citygen: an interactive system for procedural city generation, *Proceedings of game design and technology 2007*, 8–16.

Krawczyk, R.: 2002, Experiments in architectural form generation using cellular automata, *Proceedings of eCAADe 2002*.

Lindenmayer, A.: 1968, Mathematical models for cellular interaction in development, I. Filaments with one-sided inputs, *Journal of theoretical biology*, **18**, 280–289.

Lindenmayer, A. and Rozenberg, G.: 1972, Developmental systems and languages, *Proceedings of ACM symposium 1972*, Denver, Colorado, ACM, 214–221.

Mueller, P., Wonka, P., Haegler, S., Ulmer, A. and Gool, L.V.: 2006, Procedural modeling of buildings, *ACM transactions on graphics*, **25**(3), 614–623.

Parish, Y. I. H. and Muller, P.: 2001, Procedural modeling of cities, *Proceedings of SIGGRAPH 2001*, 301–308.

Schön, D.: 1992, Designing as a reflective conversation with the materials of a design situation, *Knowledge-Based Systems*, **5**(1), 3–14.

Speller, T. H., Whitney, D. and Crawley, E.: 2007, Using shape grammar to derive cellular automata rule patterns, *Complex systems*, **17**, 79–102.

Stiny, G.: 1980, Introduction to shape grammars, *Environment and planning B*, **7**, 343–351.

Stiny, G. and Gips, J.: 1972, Shape grammars and the generative specification of painting and sculpture, *Proceedings of information processing 1972*, Amsterdam, North Holland.

Stiny, G. and Mitchell, W.J.: 1978, Palladian grammar, *Environment and planning B*, **5**, 5–18.

Tapia, M.: 1999, A visual implementation of a shape grammar system, *Environment and planning B*, **26**, 59–73.

Wolfram, S.: 2002, *A new kind of science*, Wolfram Media.