Solving Form-Making Problems Using Shape Algebras and Constraint Satisfaction

Samir Emdanat, Emmanuel George Vakalo, William Birmingham

Shape grammars are well known approaches in design space exploration. This paper reviews the current work on shape grammars in design and suggests that considerable gains can be attained by integrating parametric shape grammar based design approaches with distributed constraint-based problem solving. Parametric grammars are represent design topologies while distributed constrain satisfaction can be used to maintain consistency and produce the space of feasible design solutions. Designers' decision making can be coordinated such that constraints cannot be violated and designs that exhibit the highest utility (value) are selected.

Keywords: *Shape Grammar, Shape Algebra, and Constraint Satisfaction*

Introduction

Design is an intentional activity of analysis, evaluation, and synthesis that proceeds by continuously updating partial design representations to arrive at a fully specified design solution. It is also a distributed activity. Design teams from different areas of specialization coordinate their complex decision-making activities to resolve conflicts, achieve goals, and solve constraints. Grammar-based approaches to design encode design changes in the form of rules that apply repeatedly to a partially specified design state and systematically modify and evaluate it until a fully specified design solution that satisfies the initial problem requirements is reached. This paper reviews the current work on shape grammars in design. It suggests that considerable gains can be achieved by integrating shape grammar based design approaches with distributed constraint-based problem solving.

Shape Grammar Background

Th e shape grammar approach to design was initiated by the pioneering work of Stiny and Gips (Gips 1974; Stiny 1975; Stiny and Gips 1972; Stiny and Gips 1978). In these early works it was suggested that similar to natural languages, families of visually similar shapes could be characterized using re-write rules. Subsequently, many shape grammars were formulated to describe styles of existing buildings including Palladian villas (Stiny and Mitchell 1987), Queen Anne houses (Flemming 1987), and Wren church designs (Buelinckx 1993) among others.

Although, conceptually provocative, these early works remained pencil and paper exercises. The main strength of the grammar approach is that it provides a concise and computable representation of a design space. Using a finite set of rules, a grammar can represent an infinite range of design solutions that can be explored systematically. Recently, the focus of the shape grammar research shifted from conceptual demonstration to theoretical refinement and implementation. Krishnamurti (Krishnamurti 1980; Krishnamurti 1981) addressed the issues pertaining to the computability of shapes composed of points and lines. Later research extended this work to algebras of planes and solids (Krishnamurti 1992; Krishnamurti and Stouffs, 1997). A number of shape

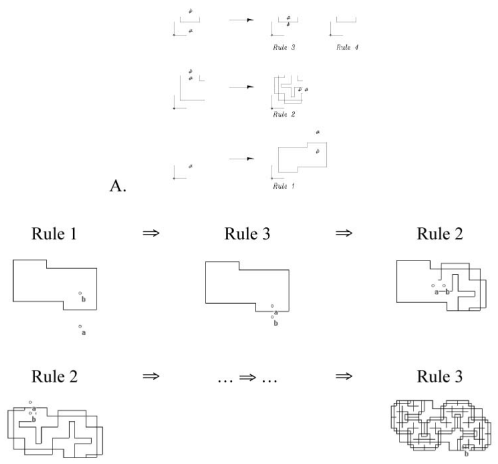

Figure 1 (right). An example of an early shape grammar from Stiny and Gips (Stiny and Gips 1978) the rules are used in deriving designs using Emdanat's shape grammar interpreter implementation (Emdanat 1998).

grammar interpreters for architectural design were implemented with varying levels of support of shape algebraic concepts and functionality (Heisserman 1991; Tapia 1996; Chase 1996; Emdanat 1997).

Early implementations of shape grammars were based on production systems. They were defined using a set of rules that operate on a current shape represented separately from the rules. The interpreter continuously compares the rules to the current shape and whenever a match is found between a rule (or a set of rules) and the current shape, they are applied to the current shape. Usually, conflicts arise and techniques similar to those used in production systems can be used to resolve the conflict (e.g., recency, specificity, and rule priority).

Current research in shape grammar focuses on how to control the application of the rules to generate interesting designs. Control guarantees that grammars generate what they are expected to generate and that the appropriate rules fire at the right moment. Flexibility in defining the shape rules can be achieved by allowing the shapes to be defined using variables and functions that determine the possible variable assignments. Current research in controlling the rule applications targets the use of stochastic search techniques such as shape annealing (see Brown and Cagan 1997) applied to a parametric representation of the shape rules. Performance measures are used to guide the applications of the rules rather than simple rule precedence operations. Brown et.al.,(1994) introduced constraint unification grammars as an approach to directing the grammar to produce a set of topologically feasible design solutions. The method maintains a set of parametric rules and constraints. The constraints are updated dynamically with each rule application and only the rules that satisfy the constraints apply. Once a topologically correct design solution is found, traditional constraint satisfaction techniques can be used to derive a particular design solution.

Recent work on the computability of shape algebras explored the potential of the algebraic approach for representing architectural form and its attributes. Theoretical issues pertaining to shape continuity and closure resulted in defining shapes as sets of continuous elements based on the sub part relation and the maximal element representation (Stiny 1994). The introduction of algebras of weights (Stiny 1992) extended the definition of shapes to include visual physical, and functional properties among others. Moreover, restricted forms of shape algebras were utilized to solve product optimization problems in other design-related domains including mechanical and structural engineering (Cagan and Mitchell 1993a ; Brown and Cagan 1996 ; Agarwal and Cagan 1998; Schmidt and Cagan 1998; Shea and Cagan 1998).

A shape grammar is defined using a set of re-write rules that are represented visually in the form of shape primitives (e.g., points, lines, planes, solids) and their attributes (e.g., color, material, physical characteristics, function). The rules consist of a left-hand side and a right hand side. The left-hand side of a rule defines the spatial as well as non-spatial conditions under which a rule applies to a given design state. A rule applies if, there is an affine transformation that makes every element in the left-hand-side a part (sub-shape) of the current shape. A shape grammar interpreter reads a set of rule representations and creates a computable specification of the design space.

The rules of a grammar operate on unique and unambiguous design representations according to the shape algebra definitions. A shape algebra defines the primitives of a shape language and the operations for manipulating these primitives by adding and removing primitives under a given set of restrictions. A shape algebra has the following characteristics: First, shapes are defined in spatial dimensions that are discrete, linear, planar, or volumetric or their combinations. Second, no two shapes of the same kind can occupy the same place at the same time. For example, if two lines are collinear and overlapping, then they are both subsumed into one line called a maximal line. Accordingly, shapes used in shape grammars are always maximal elements. Third, shapes are continuous entities. That is a shape exists as the continuous space represented by the elements of its boundary representation and consequently can contain infinitely many shapes of the same kind (i.e., subshapes).

Fourth, shapes are defined according to their boundaries. The boundaries of a shape are shapes that belong to a lesser spatial dimension. For example, Solids are bounded by Planes, which are bounded by lines, which are bounded by points. Fifth, shapes can only be added to or removed from other shapes that are of the same kind. Sixth, shapes have both spatial and non-spatial features referred to as labels and weights. Shapes associated with weights behave in the same way as other shapes thus extending the formalism uniformly.

The aforementioned properties of shape algebra facilitate an unambiguous shape representation. According to these properties, a shape can have one and only one representation. Moreover, this representation has no pre-defined structure. A shape representation is given structure prior to a rule application and then after the rule applies the resulting shapes return to their unstructured maximal element representation. In other words, the structure is maintained in the rule specification and not the current partial design. This makes it very easy to add, remove, or modify rules in a uniform and flexible way. The grammar rules are compact generative representations of an infinite number of designs that facilitate the systematic exploration of the design space. However, the flexibility of the representation is attained at a price. The continuous structuring and reduction of shapes is computationally expensive. For most practical purposes, shape grammar implementations restrict the expressiveness of the shape algebra to achieve better performance.

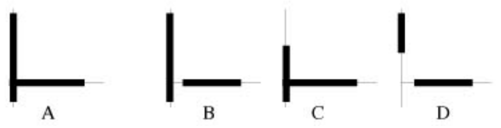

Figure 2 (left). A Shape A and some of Its Possible Subshapes.

Distributed Problem Solving Using Parametric Grammars

Currently, the use of shape grammars in design is limited to stand-alone systems modeling a design process that is controlled by a single designer. Little work has been done on scaling shape grammars to facilitate modeling the design process itself as a decision-making activity. As design is both an intentional and purposeful activity, designers can be viewed as rational decision-makers that make decisions about design changes that involve feasibility and value of a design solution to maximize its utility. Design agents make these decisions locally based on their expertise and communicate the results to coordinate their decision making with other design members.

This paper suggests that utilizing parametric shape grammars in distributed constraint-based problem solving can attain considerable gains. Constraint-based problem solving techniques are ideal for identifying the feasible design space in design problems where the design variables are known and certain relations connect their domain elements (attributes). However, they are limited in exploring design alternatives outside the initial problem specification. Generative systems provide a compact representation of the design space that can be subjected to systematic exploration to discover feasible design topologies. In addition, when coupled with efficient constraint satisfaction techniques, the exploration can be efficiently directed to feasible segments of the design space from which the designs that exhibit highest utility will be selected.

Adopting this approach to design promotes concurrent problem solving. We suggest that a concurrent engineering approach that uses algebras of parametric shapes can be used to model the design process and study its aspects formally. Algebras of parametric shapes provide a formal framework for representing large design spaces, the feasibility of solutions, and their value. Feasibility refers to parts of the design space that satisfy specific conditions without which a meaningful design solution cannot exist (e.g., laws of physics). The value of a design refers to its economic benefit. Constraints in this framework define the feasible design space over the range of variable assignments in a parametric design representation. A utility function provides a valuation on this space to identity designs that are of greater value. It also describes how tradeoffs can be made among feasible design alternatives. A concurrent problem-solving approach to design integrates the overall knowledge, resources and experience needed to solve a design problem as early as possible in the design cycle. Separating the types of decisions made during decision making during may provide better insight into designing efficient algorithms and heuristics to support the designers' activities.

Formulating a design problem via parametric grammars, constraints, and utility functions provides several substantial benefits over stand-alone shape grammar systems. It is possible to produce designs that have the greatest utility, which would be a more meaningful measure than the typical design goal (e.g. minimal cost or maximal performance). Using parametric shape rules, automated design agents, as well as, human designers can use a consistent formalism to communicate their design decisions. Parametric shape rules can be associated with constraints that get added to the current design every time a change is proposed. Rules that are consistent with the overall network of constraints can be added to the design resulting in topological design representations that are consistent with the network.

In more specific terms, a design problem can be cast as a multi-attribute distributed con constraint satisfaction problem (D'Ambrosio, Darr, and Birmingham 1996), in which a network of constraints and shared utility functions can be used to link designers. Designers operate in their own design space, where they make decisions (for finding an optimal design) that are consistent with respect to other designers. Additionally, since utility functions can be shared where necessary, it can be guaranteed that all designers are synergistically making decisions. In other words, all designers are optimizing the same objective, while still having control over their design spaces.

The constraint network can be represented as a multi-attribute domain constraint-satisfaction problem (MAD-CSP). This class of constraint satisfaction problems propagates intervals that naturally represent uncertainty in distributed design problems. They also lead to efficient algorithms for finding solutions since intervals are compact representations of highly complex data. Utility functions operate over complete designs. That is by knowing rough bounds on design attributes it would be possible to create utility functions that accurately represent a designer's preference (see D'Amrosio and Birmingham 1995). Consequently, these utility functions can be used to prune the feasible design space to produce designs that are close to the optimal.

Summary and Conclusions

The purpose of this paper was to review the current work on the shape grammar approaches to design and suggest that considerable gains can be attained by integrating parametric shape grammars with distributed constraint-based problem solving.

Parametric shape grammars are ideal tools for exploring large spaces of topological design representations. The rules of a grammar can be used to build ranges of topologically feasible design spaces that are consistent with the overall network of design constraints by eliminating the rules that violate the shared constraints. Constraint satisfaction techniques are ideal for determining the set of feasible variable assignments to a given parametric design specification to main a consistent solution that meets the design specification.

References

Agarwal, Manish and Jonathan Cagan. 1998. A Blend of Different Tastes: The Language of Coffee Makers. *Environment and Planning B: Planning and Design* 25: 205-26.

Brown, Ken and Jonathan Cagan. 1996. Modified Shape Annealing for Optimally-Directed Generation: Initial Results. *Advances in Formal Design Methods for CAD: Proceedings of the IFIP WG5.2 Workshop on Formal Design Methods for Computer-Aided Design*. London: Chapman & Hall, 67-80.

Brown, Ken and Jonathan Cagan. June 1997. Optimized Process Planning by Generative Simulated Annealing. *Ai Edam-Artificial Intelligence for Engineering Design Analysis and Manufacturing* 11, no. 3: 219-35.

Brown, Ken, C. McMahon, and J. Sims Williams. 1994. Constraint Unification Grammars: Specifying Languages of Parametric Designs. *Artificial Intelligence in Design '94*, Eds John Gero and Fay Sudweeks. 239-56. The Netherlands: Kluwer Academic Publishers.

Buelinckx, Hendrika. 1993. Wren's Language of City Church Designs: A Formal Generative Classification. *Environment and Planning B: Planning and Design* 20: 645-76.

Chase, Scott. 1996. Modeling Designs with Shape Algebras and Formal Logic. Ph.D. Dissertation, University of California.

D'Ambrosio, Joseph, Timothy Darr, and William Birmingham. March 1996. Hierarchical Concurrent Engineering in a Multiagent Framework. *Concurrent Engineering: Research and Applications (CERA)* 4, no. 1: 47-57.

D'Amrosio, J. and W. Birmingham. 1995. Preference Directed Design. *AI EDAM* 9: 219-30.

Emdanat, Samir. 1997. *Shape Grammars, Some Basic Implementations and New Results: Introduction to Building Grammars.* Ann Arbor, MI: College of Architecture and Urban Planning, University of Michigan, Doctoral Research Practicum.

Flemming, Ulrich. 1987. More than the Sum of Parts: The Grammar of Queen Anne Houses. *Environment and Planning B: Planning and Design* 14: 323-50.

Gips, James. 1974. Shape Grammars and Their Uses. Ph.D. Dissertation, Stanford University.

Heisserman, Jeff. 1991. Generative Geometric Design and Boundary Solid Grammars Department of Architecture. Ph.D. Dissertation, Carnegie Mellon University.

Krishnamurti, Ramesh. 1980. The Arithmetic of Shapes. *Environment and Planning B: Planning and Design* 7: 463-84.

Krishnamurti, Ramesh.1981. The Construction of Shapes. *Environment and Planning B: Planning and Design* 8: 5-40.

Krishnamurti, Ramesh.1992. The Maximal Representation of a Shape. *Environment and Planning B: Planning and Design* 19: 267-88.

Krishnamurti, Ramesh and C Giraud. 1986. Towards a Shape Editor: The Implementation of a Shape Generation System. *Environment and Planning B: Planning and Design* 13: 391-404.

Krishnamurti, Ramesh and Rudi Stouffs. 1997. Spatial Change: Continuity, Reversibility, and Emergent Shapes . *Environment and Planning B: Planning and Design* 24: 359-84.

Schmidt, L. C. and Jonathan Cagan. 1998. Optimal Configuration Design: an Integrated Approach

Using Grammars. *Journal of Mechanical Design* 120, no. 1: 2-9.

Shea, Kristina and Jonathan Cagan. 1998. Generating Structural Essays from Lanuages of Discrete Structures. *Artificial Interlligence in Design '98*, eds. John Gero and Fay Sudweeks. 365-84. Dordrecht: Kluwer Academic Publishers.

Stiny, George. 1975. Pictorial and Formal Aspects of Shape and Shape Grammars. Birkhauser.

Stiny, George.1992. Weights. *Environment and Planning B: Planning and Design* 19: 413-30.

Stiny, George.1994. Shape Rules: Closure, Continuity, and Emergence. *Environment and Planning B: Planning and Design* 21: s49-s78.

Stiny, George and James Gips. 1972. Shape Grammars and the Generative Specification of Painting and Sculpture. *Information Processing*, Eds C. Frieman. 1460-5.Vol. 71. North-Holland, Amsterdam: North-Holland Publishing Company.

Stiny, George and James Gips. 1978. *Algorithmic Aesthetics: Computer Models for Criticism and Design in the Arts.* Berkeley: University of California Press.

Stiny, George and William Mitchell. 1987. The Palladian Grammar. *Environment and Planning B: Planning and Design* 5: 5-18.

Tapia, Mark. 1996. From Shape to Style, Shape Grammars: Issues in Representation and Computation, Presentation and Selection. Ph.D. Dissertation, University of Toronto.

Samir Emdanat (1), Emmanuel George Vakalo (1),
William Birmingham (2)
(1) Doctoral Program in Architecture, CAUP
emdanat@umich.edu
(2) Artificial Intelligence Laboratory, EECS
The University of Michigan
Ann Arbor, Michigan 48109, USA