Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2000), *14*, 431–439. Printed in the USA.
Copyright © 2000 Cambridge University Press 0890-0604/00 $12.50

On the use of shape grammars as expert systems for geometry-based engineering design

MANISH AGARWAL AND JONATHAN CAGAN

Computational Design Laboratory, Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

(RECEIVED October 4, 1999; ACCEPTED May 4, 2000)

Abstract

Production systems have traditionally been used in engineering to aid the designer in generating designs in domains not significantly influenced by geometry. However, in geometric-based domains such as mechanical engineering, their popularity has been limited. This work proposes the use of shape grammars as the framework for engineering expert systems. Shape grammars are shown to have all the generation and analysis capabilities of traditional production systems, while being able to represent knowledge about both the functionality and the form of a product. Additionally, the parametric nature of shape grammars, their ability to deal with physical form rather than abstract elements, and their ability to recognize emergent shapes give them significant advantages over traditional production systems for geometry-based engineering design.

Keywords: Shape Grammars; Expert Systems; Production Systems; Design

1. INTRODUCTION

Engineering design has traditionally been a manual task. Designers, relying chiefly on their experience, generate designs based on the required specifications and only then represent them in CAD systems. Such an approach often leads to substantial turnaround times because of the labor-intensive nature of the undertaking. Further, creating radically new designs, and thereby obtaining significant cost and performance advantages, often requires designers to have an intuitive leap rather than the ability to apply a formal methodology. Finally, design experience is difficult to pass from one designer to another, resulting in long and wasteful learning curves.

There are two complementary aspects of the design process—the creation of designs to meet certain specifications and the analysis of the created designs to determine if indeed they do meet the requirements. A number of researchers, with the aim of speeding up the design cycle, have developed systems to aid the designers in one or the other (although, often, not both) of these aspects. Production systems have been developed to aid designers in assembling components to create designs. Simulation-based analysis systems, for example, finite element methods, have been developed to help the designer analyze parts of the design.

Although simulation tools have enjoyed significant success, production systems for geometry-based engineering domains (e.g., mechanical, electromechanical, and civil), which form the focus of this work, have not been as popular. Traditional production systems within these domains typically use rules to represent the available knowledge about the problem space, and then use those rules to create designs. Most of these systems, however, use abstract elements to create devices. Design, on the other hand, deals chiefly with physical shape and creating the form to meet the functional requirements. It is the interaction of form that often leads to creative designs by allowing new shapes to emerge, an important property of the design process (Coyne & Newton, 1989). We believe that it is the inability of traditional production systems to work directly with form that has limited them primarily to nongeometric domains and to the creation of only routine designs. Within the context of this paper we must first define our use of the terms "production system" and "expert system." The traditional "production systems" that are referred to in this work are rule-based systems that rely on the manipulation of symbols and

Reprint request to: Jonathan Cagan, Computational Design Laboratory, Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213. E-mail: cagan@cmu.edu

https://doi.org/10.1017/S089006040014507X Published online by Cambridge University Press

abstract entities, but not geometry; expert systems are used interchangeably with production systems, although expert systems imply the modeling of expertise.

We propose shape grammars as a new framework for engineering expert systems. Shape grammars are essentially a rule set defining how shapes in a set can be modified. They have typically been used as generative systems creating designs with specific characteristics in the field of architecture and, to a lesser extent, in mechanical engineering. Shape grammars can be parametric in nature, thereby allowing an infinite number of designs to be generated from the same rule set. This paper demonstrates that shape grammars, in addition to their generative properties, can also be used to represent engineering knowledge and analyze designs. They have the ability to deal directly with form while satisfying functional requirements. However, they can also recognize emergent shapes in the design process, making it possible for more creative designs to be generated. Based on these observations, we argue that shape grammars provide an effective framework for engineering expert systems. The rest of this paper is organized as follows: first some related work is discussed and a shape grammar for the generation of MicroElectro Mechanical Systems (MEMS) resonators is outlined; next the applicability of shape grammars to the three main aspects of the design process, namely knowledge representation, analysis, and generation, is presented; finally the paper concludes with some general observations.

2. RELATED WORK

2.1. Production systems

Production systems aid designers in the creation of designs through a systematic rule-based approach. The rule set is created so as to capture all available knowledge about the domain, and heuristics are used to guide the generation process. A number of production systems have been developed over the years for various application domains. Examples include R1 (XCON) (McDermott, 1981), used for configuring computer systems, AIR-CYL for air cylinder design (Brown & Chandrasekaran, 1986), and VT (Marcus et al., 1988), for the design of elevators. Further examples of production systems can be found in Liebowitz (1998).

Most of the application domains for production systems have been primarily nongeometric, for example, computer systems and chip design. Systems for geometry-intensive domains like mechanical configurations tend to represent components abstractly rather than as physical forms. As a result, these systems are typically used for creating only routine designs and not for generating innovative configurations. Further, the lack of geometric detail in most production systems has limited their ability to provide concurrent design and analysis, which means designs must be completed before they can be analyzed, leading to significant wasted computation on inferior designs. These issues, which we believe have limited the popularity of traditional production systems, can be addressed using a shape-grammar-based framework. Note that, although this work advocates a shape-grammar-based framework, it does not suggest that alternate representations with similar properties are not possible.

2.2. Shape grammars

A shape grammar (Stiny, 1980) derives designs in the language it specifies by successive application of shape transformation rules to some evolving shape, starting with an initial shape. It can be used to describe how complex shapes are built from simple entities and how a complex shape can be decomposed into simpler subshapes. Shape grammars have been successfully used for spatial design in the field of architecture, including villas in the style of Palladio (Stiny & Mitchell, 1978) and prairie houses in the style of Frank Lloyd Wright (Koning & Eizenberg, 1981).

There has been a limited application of shape grammars to engineering design. Fitzhorn (1990) presented a shape grammar specifying the languages of constructive solid geometry and boundary representations (i.e., realizable solids). Reddy and Cagan (1995) and Shea and Cagan (1997) presented a parametric shape grammar for the design of truss structures that uses the shape annealing technique of Cagan and Mitchell (1993) to generate optimal truss structures. Recent work by Agarwal and Cagan (1998) has focused on developing shape grammars for a class of individual products, in this case coffee makers, and associating cost expressions with the grammar rules (Agarwal et al., 1999). Agarwal et al. (2000) extended the idea of product representation by the development of a MEMS resonator shape grammar; further details of this work will be discussed in Section 3.

Although shape grammars form the focus of this paper, other grammars can also be defined. The term grammar was used first in a formal sense by Chomsky (1957). Although grammars defined by Chomsky generate one-dimensional sequences, grammars that generate arrays, trees, graphs, and so forth have since been developed.

Formally, a shape grammar has four components:

1. S is a finite set of shapes where a shape is a collection of lines and curves that form an entity;
2. L is a finite set of symbols;
3. R is a finite set of shape rules of the form $\alpha \rightarrow \beta$, where α is a labeled shape in $(S, L)^+$, and β is a labeled shape in $(S, L)^0$ where $(S, L)^+$ is the set of all labeled shapes made up of shapes in the set S and symbols in the set L and $(S, L)^0$ is the set that contains, in addition to all of the labeled shapes in the set $(S, L)^+$, the empty labeled shape $\langle s_\phi, \phi \rangle$; and
4. I is a labeled shape in $(S, L)^+$ called the initial shape.

All shapes that are generated using a shape grammar are made up of the primitive elements in the sets S and L that have been operated upon by the rules in the set R. A shape rule $\alpha \rightarrow \beta$ applies to a labeled shape γ when there is a transformation τ such that $\tau(\alpha)$ is a subshape of γ. The

https://doi.org/10.1017/S089006040014507X Published online by Cambridge University Press

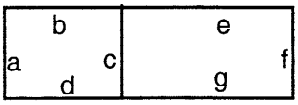


Fig. 1. Shape made up of seven lines labeled a through g.

transformation τ can either be a general Euclidean transformation or can be restricted to some special kinds of transformations like those that preserve isometry, for example. The shape generated by applying this rule to the shape γ is the shape $[\gamma - \tau(\alpha)] + \tau(\beta)$. In other words, applying the shape rule $\alpha \rightarrow \beta$ to the shape γ results in the replacement of the subshape α by the subshape β (or more correctly, $\tau(\alpha)$ by $\tau(\beta)$). Another shape rule can then be applied to this new shape. This must be done until the rule applied to the shape is the terminal rule after which no further rule from the set *R* can be applied to the generated shape or any of its subshapes.

Parametric shape grammars are an extension of shape grammars in which shape rules are defined by filling in the open terms in a general schema. A shape rule schema $\alpha \rightarrow \beta$ consists of parametric labeled shapes α and β, where no member of the family of labeled shapes specified by α is the empty labeled shape. An assignment *g* which gives specific values to all the variables in α and β determines a shape rule $g(\alpha) \rightarrow g(\beta)$, which can then be applied to a labeled shape in the usual way, resulting in a new labeled shape. Algebras of shapes can also be augmented by weights to obtain new algebras in which the shape union operation has been redefined to reflect different possible weights on different entities (Stiny, 1992).

What are the properties of shape grammars that make them useful for engineering? The single most important property from the perspective of design is the ability of shape grammars to deal directly with the parametric nature of shape, that is, points, lines, and planes, rather than their higher-level abstract counterparts. The shape rules in the set *R* are written in terms of actual shapes and the elements they operate on, that is, members of set *S* are also physical shapes. The other critical property that gives shape grammars their power is that the subshape rather than the subset operation is used to determine if one shape is a part of the other and that the basic elements that make up a shape are maximal rather than individual. Consider the shape in Figure 1, made up of seven lines labeled a through g. This shape is represented instead by five maximal lines a, c, f, (be) and (dg), where the notation (ij) is used to refer to a line comprised of the union of lines i and j. This, coupled with the subshape operation, allows the system to recognize not only the two input rectangles[1] abcd and cefg, but also the emergent rectangle a(be)f(gd). On the other hand, a system that represents the left-hand- and right-hand-side rectangles as symbols may not be able to identify the emergent rectangle as easily, if at all.

Finally, we comment on the relation between shape grammars and production systems. Shape grammars consist of a rule set that describes how to put shapes together, much like productions systems. In fact, Stiny and Gips (1980) describe a characterization of grammars that fits them neatly in the paradigm of production systems. In this article, however, we show that shape grammars have properties that make them more suitable for geometry-based engineering design than traditional production systems.

3. EXAMPLE: MEMS RESONATOR GRAMMAR

This section discusses a shape grammar for the design of MEMS resonators that will be used throughout this paper to illustrate properties of shape grammars. MEMS devices are electromechanical structures fabricated using IC technology. They integrate mechanical elements like beams and springs with electrical circuitry. MEMS devices have been used in various applications, for example, air-bag sensors, pressure sensors, and gyroscopes. A variety of different configurations are possible for these devices. The resonators produced by the grammar discussed here use comb drives for actuation, beams as springs and air viscosity for damping. A schematic of a typical resonator is shown in Figure 2e. Note that, due to space limitations, only certain

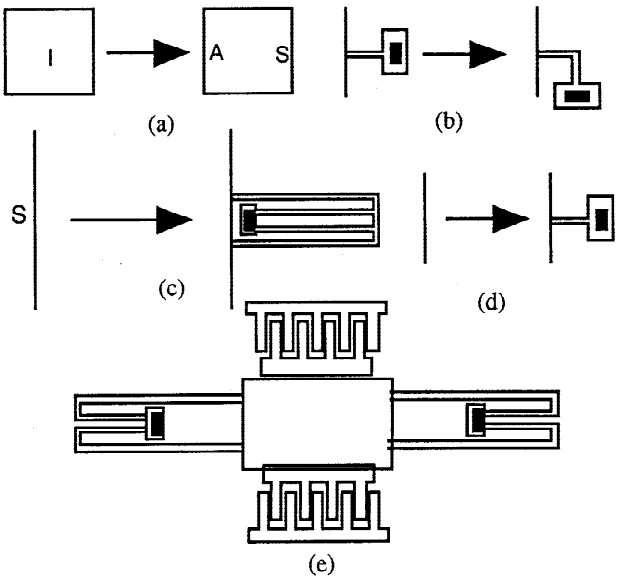


Fig. 2. MEMS resonator grammar. (a) Rule adding labels A and S, signifying the position of actuators and springs; (b) rule modifying an existing spring by adding another spring element and moving the anchor; (c) rule adding a folded-flexure spring to a side of the mass previously labeled S; (d) rule adding a simple beam element, on the side of the mass, to act as a spring; and (e) a typical resonator generated by the grammar.

[1] A rectangle is defined as a four-sided polygon with all internal angles equal to 90 degrees.

https://doi.org/10.1017/S089006040014507X Published online by Cambridge University Press

aspects of the grammar will be presented. The complete grammar can be found in Agarwal et al. (2000).

The resonator shape grammar is a parametric grammar augmented by weights and labels. The grammar starts with the central mass and builds the resonator around it. Actuators are designed before springs, though the terminal rule cannot be applied before the device has at least one actuator and one spring. The springs can be created in one of two ways—either incrementally using simple beam elements or by directly adding functional macroelements like serpentine springs. The mass, actuators, anchors and springs are assigned distinct decreasing weights and the shape addition operator is defined such that the mass dominates the actuators, actuators dominate anchors, and anchors dominate springs. Simply stated, this domination is achieved by allowing shapes of higher weights to overlap and erase shapes of lower weights. This also ensures that actuators are not broken up by anchors or springs, anchors are not broken up by springs, and so forth. Some representative rules in the grammar are shown in Figure 2a–2d. Note that all lines and points are parametric.

4. ENGINEERING KNOWLEDGE REPRESENTATION

There are two classes of engineering knowledge about a domain that need to be represented in any intelligent design generation and analysis system, namely, information about function and information about form. The function information is typically in the form of various attributes that a design must meet before it can be considered complete. For example, a MEMS device must have at least one mass, one spring, and one actuator before it can function as a valid resonator. Form information typically consists of a variety of shapes that can be used to satisfy the functional requirements. Traditional production (or expert) systems represent this knowledge through a set of rules that defines the valid configurations for components to ensure that all required functionality is met. Again, the difficulty there is the ability to represent, reason about, and interpret geometry in a natural way.

Shape grammars can be used to represent both function and form knowledge for engineering domains. Indeed the requirement that a MEMS resonator have at least one actuator and one spring before the terminal rule can be applied, for example, is an instance of the representation of the functional knowledge within the shape grammar paradigm. By far the simplest means of ensuring that all the functional requirements are met before the generation process terminates is through the use of labels. Assume that there are n attributes that must be met before a design can be considered complete. Also assume that a label i is associated with the shape when the ith requirement is met (e.g., a spring is added to the system). Further assume that there is a global label I that needs to be present on the shape for any shape rule to apply. Then, the rule

$$\mathrm{I}\ 1\ 2 \ldots n \rightarrow \mathrm{I}'\ 1\ 2 \ldots n$$

is the appropriate termination rule that ensures that all functional attributes will be met before the design generation process terminates.

Using a similar strategy, it is also possible to represent the form knowledge about a domain through the shape rules. Specifically, assume that the design process is broken up into two parts, function design and form design. The appropriate function design can be carried out using the rules discussed above, except that the rule

$$\mathrm{I}\ 1\ 2 \ldots n \rightarrow \mathrm{I}'\ 1\ 2 \ldots n$$

is the terminal rule for the function design phase of the process rather than for the whole process. Now, form design rules that capture the knowledge about the various possible means of satisfying a desired function attribute can be written based on the labels $1 \ldots n$. More concretely, assume that the function attribute j can be met through m possible form designs $\mathrm{A}_1, \mathrm{A}_2 \ldots \mathrm{A}_m$ (a spring can be a simple beam, a serpentine spring, a folded flexure spring, etc.). The rules for the form design corresponding to the function attribute j could then be of the form

$$\mathrm{I}'\ j \rightarrow \mathrm{A}_1\ \mathrm{I}'\ j'$$

$$\mathrm{I}'\ j \rightarrow \mathrm{A}_2\ \mathrm{I}'\ j'$$

$$\ldots$$

$$\mathrm{I}'\ j \rightarrow \mathrm{A}_m\ \mathrm{I}'\ j'.$$

Rule c in Figure 2 is an example of such a rule. The termination rule for the whole process would then be of the form

$$\mathrm{I}'\ 1'\ 2' \ldots n' \rightarrow \mathrm{I}''\ 1'\ 2' \ldots n'$$

assuming that the label I″ ensures that no further rules can be applied.

So far an assumption of a decoupling of function and form has been assumed. The same approaches can be used concurrently if function and form are integrated. In summary, it is possible to capture knowledge about both function design requirements and form design alternatives through the use of labels in shape rules.

5. SHAPE GRAMMARS AS AN ANALYSIS TOOL

5.1. Recognition

One of the most important applications of engineering expert systems is as an analysis tool, that is, to represent and interpret failure in systems. To do this, the first requirement of any system is to be able to recognize certain key elements in the design that can then be analyzed by calling the appropriate analysis routines. Production systems can be augmented rather easily to carry out such a recognition. Because a specific sequence of rules is used to create any given

https://doi.org/10.1017/S089006040014507X Published online by Cambridge University Press

design, the rule set can be examined to determine the key elements that were included in the design. However, such an approach would miss any emergent shapes in the design as discussed in Section 2 and thus may not be able to take advantage of analysis techniques for that shape.

We claim that shape grammars can be used as easily to determine the salient features of a design including any emergent shapes. Intuitively, recognizing a feature in a design involves some sort of pattern matching, that is, the current design state must be examined to determine where, if at all, a target shape occurs. Formally, a production rule of the form

$$A \rightarrow A,$$

where A is the element that needs to be recognized, can be used for such pattern matching. It is worthwhile to examine what effect this rule would have on a shape. Obviously, the shape itself would not change under such an identity transformation. However, before the system can apply such a rule, it must determine where in the current shape this rule could apply. This act forces the system to recognize the desired element in the shape. The system can be queried to determine where the rule was applied, in effect determining the location of the desired element. Although seemingly obvious, this rule is quite powerful. It enables the recognition of emergent shapes and, being parametric, recognizes an infinite number of such left-hand-side basis shapes. For a general discussion of this, and other seemingly trivial rules, see Stiny (1996).

An example of the identity rule, shown in Figure 3, can be used to recognize and isolate an infinite number of folded-flexure springs in a resonator. The power of the rule can be fully demonstrated through the recognition of *emergent* folded-flexure elements, that is, a folded-flexure spring that results by putting together elemental beams rather than a macroshape. Consider the generation sequence shown in Figure 4. The spring is created by adding a beam to the side of the mass and modifying it twice to create a U shape as shown in Figure 4b. Another similar spring is then created starting below the top spring. While the second spring folds over itself like the first, it does not stop at the anchor of the first spring, but continues further and stops at its own anchor.

Now consider the final spring produced. If the beam segment between the two anchors and the right anchor itself are disregarded, the spring is exactly the folded-flexure spring of Figure 3 and thus the rule of Figure 3 can correctly identify the desired spring structure. Formally, this is because the folded-flexure shown in the left-hand side of the rule in Figure 3 is a subshape of the shape in Figure 4c. The beam element between the two anchors and the right anchor itself are not relevant to the rule. This is similar to the recognition of the emergent rectangle in the example in Section 2. Such a recognition would be difficult in traditional production systems because it would require the system to check numerous combinations of beams to split the top beam, almost arbitrarily, in the bottom part of the flexure at the left anchor.

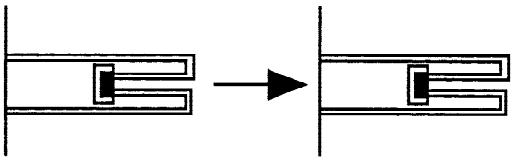


Fig. 3. Identity shape rule of the form $A \rightarrow A$ that can be used to recognize folded-flexure spring elements in MEMS resonators.

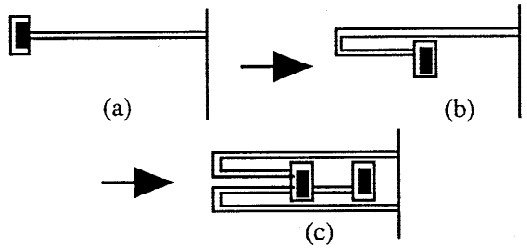


Fig. 4. Example generation sequence for a folded-flexure spring. Not all steps are shown. (a) The process is started by adding a beam to the mass; (b) the top part of the folded flexure is completed; and (c) the bottom part of the folded flexure is created in an analogous manner.

5.2. Analysis

Engineering designs need to be analyzed either to determine how they will work under certain conditions or to determine why a certain failure occurred. There are various kinds of analyses that can be carried out, ranging from those using full simulations to those using only analytical models. Engineering expert systems can be used to carry out either kind of analysis and possibly combinations of both. First, the elements are recognized at the appropriate level of detail depending on the required analysis. For example, springs in MEMS resonators can be recognized as made up of beams or as made up of macro-shapes like folded flexures. Of course, the inability to recognize emergent shapes and thus be able to call the appropriate analytical analysis routine may make such an analysis harder for traditional expert systems. However, once elements have been recognized, the geometry information can be passed to either a simulation package or to an analytical analysis routine.

We claim that shape-grammar-based systems are equally amenable to such hierarchical analyses. All that is needed is that rules of the form $A \rightarrow A$ be written for all possible levels of abstraction along with appropriate control strategies for ensuring that only rules at one level can be applied. A rather straightforward method of doing this is to augment the left- and right-hand sides of the rule $A \rightarrow A$ with global labels $1 \ldots k$ (signifying k levels of abstraction) to obtain the rules $A_1\ 1 \rightarrow A_1\ 1$, $A_2\ 2 \rightarrow A_2\ 2$, $\ldots$, $A_k\ k \rightarrow A_k\ k$. Depending upon the desired level of detail, the global label $1, 2 \ldots k$ can be associated with the initial shape. This then ensures

https://doi.org/10.1017/S089006040014507X Published online by Cambridge University Press

that only elements at the right level of abstraction (e.g., A_1 for level 1) will be recognized and analyzed. Another approach for recognizing elements at various levels of hierarchy can be derived by using rules of the form $A \rightarrow \phi$, where ϕ is the empty shape (Stiny, 1996). In this case, the following set of rules can be sequentially applied:

$$A_1 \rightarrow \phi,$$

$$A_2 \rightarrow \phi,$$

$$\cdots$$

$$A_k \rightarrow \phi.$$

The first rule is applied to part A_1 of the current shape, the second to part A_2 of the shape A_1, the third to part A_3 of the shape A_2, and so forth. Rules can be applied concurrently and each rule results in a mapping or copy of the subshape on which analyses can be applied. Such a rule set will result in the decomposition of the current shape into various sublevels of hierarchy. The existence of more than one rule at a certain level ($A_2 \rightarrow \phi$ and $A_2' \rightarrow \phi$, for example) can be used to determine alternative decompositions of the same current shape, a potentially powerful tool to investigate different aspects of the design at hand. For example, the different decompositions can correspond to analyses in different domains (say, electrical and mechanical).

As an example, consider the coupled resonator (filter) device shown in Figure 5a. A hierarchy of the features on the device is shown in Figure 5b. A set of rules of the form $A \rightarrow \phi$ are shown in Figure 5c. Using these rules, an individual resonator can be isolated for analysis. From that individual resonator, the comb drives and folded flexures of the resonator can be isolated for analysis, and finally an individual beam element from the folded flexure (or other subcomponent) can be isolated for localized analysis.

However, often in engineering systems, certain elements of a design become important only under certain conditions and thus may need to be analyzed only under those conditions. Such an analysis is typically carried out by traditional expert systems using conditional statements. Shape grammars are also capable of such an analysis through the use of labels. Specifically, assume that an element *A* needs to be analyzed (and, thus, recognized) only if a certain condition is met. A label, say *a*, can be associated with the shape if the condition is met. Then the rule $A\ a \rightarrow A\ a$ will recognize the element *A* only if the appropriate condition is met. Note that the condition itself may be the presence of another element, say *B*, in which case the rule $B \rightarrow B\ a$, will associate the appropriate label with the shape.

For example, the compliance (or lack, thereof) of a MEMS device in the *x* direction would be relevant only if there is an applied force in that direction. Thus, if a MEMS resonator had an excessive motion in the *x* direction, an expert system may execute the following algorithm:

```
If (x-direction-force != 0) then
    if (x-direction-stiffness < critical-stiffness) then
        {
        failure-mode = x-direction-stiffness;
        return (failure-mode);
        }
```

A similar analysis can be carried out using shape grammars through the use of labels. At the first stage, comb drive elements in the *x* direction are identified and, if found, an appropriate label is attached to the shape. This is done through a shape rule of the form $A \rightarrow A\ 1$, where A is the *x*-direction comb drive and 1 is the label signifying that a *x*-direction force is present. In the second stage the label 1 is used to indicate that the *x*-direction stiffness should be computed and compared with the critical stiffness in that direction. In short, the shape grammar paradigm allows, without the imposition of extra structure, conditional analysis to be carried out in a manner similar to traditional expert systems.

6. SHAPE GRAMMARS AS A GENERATIVE TOOL

This section describes the generative aspects of shape grammars, that is, the ability to create a device given the desired specifications, and compares them with the capabilities of a traditional production system. At a fundamental level, both systems work in a similar manner—rules are chosen from a rule set and applied to an evolving entity to change it some way. The difference, however, is that shape grammars deal with physical form, whereas traditional production systems usually represent shapes as symbols and thus deal with shape only at an abstract level. It is this difference that provides shape grammars with the additional properties discussed in this section and allows them to generate designs that go beyond standard topologies.

The shape generation process in a shape-grammar-based system proceeds in two distinct phases. First, the rule to be applied to the current shape is chosen from the set of applicable rules. Next, the chosen rule is instantiated by filling in the open terms in the general schema. Both the rules and the parameters can be chosen either by the designer or computationally through random selection or through a search. For example, optimization algorithms can be used as a means of efficiently carrying out directed search. Obviously, these strategies can be used in various combinations, the simplest being random or user-aided selection. A slightly more sophisticated approach is to maintain statistics on the progress of the system and use the information to guide future rule selection in a genetic algorithm or simulated annealing sense. This approach has been used with considerable success in simple application domains like dome design (Shea & Cagan, 1997). Dixon et al. (1987) discuss the use of traditional optimization algorithms on top of traditional production systems.

The final and perhaps most powerful approach to using shape grammars as a generative tool would be an auto-

https://doi.org/10.1017/S089006040014507X Published online by Cambridge University Press

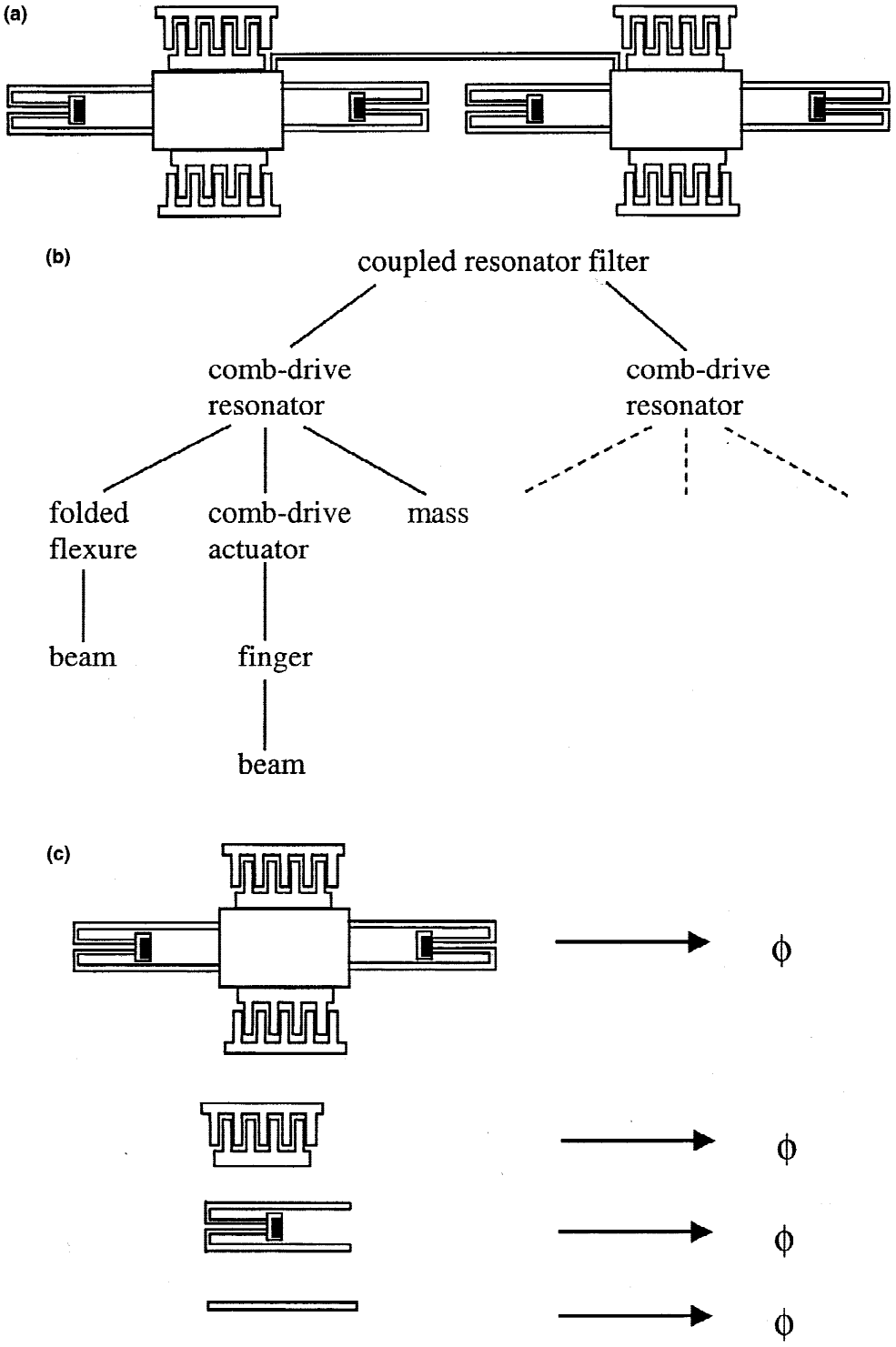


Fig. 5. Coupled resonator filter (a), hierarchical component breakdown (b), and removal rules to isolate subcomponents for analysis (c).

mated strategy where the system chooses rules to be applied, based on the specifications that need to be met, aided through immediate feedback about the performance of the partial design. Note that providing feedback about the partial (and complete) design stage may require one or more of the analysis techniques discussed in Section 5. Based on the results of the analysis, appropriate rules can be applied at a later stage in the same iteration or in subsequent iterations. To ensure that such a strategy will create designs that will satisfy all required functionality, shape rules similar to those

https://doi.org/10.1017/S089006040014507X Published online by Cambridge University Press

discussed in Section 4 need to be used to guide function and form design.

A shape grammar system that demonstrates a part of this paradigm to engineering design problems is the coffee maker grammar of Agarwal and Cagan (1998), a label-driven shape grammar augmented by expressions for manufacturing costs (Agarwal et al., 1999). The system can be used not only to generate existing and novel coffee makers, but also to cost the designs as they are being generated, thereby providing immediate feedback. A final appraisal of the cost of the design is also obtained at the end of the process. For existing designs, Agarwal et al. (1999) show that these costs match those in literature closely, thereby validating the approach. The coffee maker generative tool is not automated, yet recognition that the shape annealing method has been successful in automating design generation in several applications [truss design (Shea & Cagan, 1997) and process planning (Brown & Cagan, 1997)] lends credence to the idea that automated generative tools for sophisticated engineering shape grammars are feasible.

7. CONCLUSIONS

This article proposes a shape-grammar-based paradigm for engineering systems. We have argued that not only do shape grammars have all the generation abilities of traditional production systems, but their parametric nature and ability to recognize emergent shapes makes them more useful for geometry-based engineering domains. Shape grammars can also be used for analysis and for representing engineering knowledge. Additionally, and perhaps most importantly, shape grammars provide a unifying framework for all expert systems for geometry-based engineering domains.

Shape grammars have the ability to deal directly with form. This, coupled with the manner in which the subshape operation is defined, allows shape grammars to recognize emergent shapes and gives them the ability to create nonroutine designs. Additionally, shape grammars allow labels to be associated with shapes to carry nongeometric information and guide the generation process. Finally, their parametric nature allows the same small and finite rule set to generate an infinite number of designs, allowing a generative system to explore a wide variety of designs. Over the years, a number of difficult theoretical and implementation issues with shape grammars have been resolved by a number of researchers (e.g., Krishnamurti, 1980, 1981), thereby allowing them to be applied to geometry-based domains. By combining ideas from traditional production systems with the work done on shape grammars, it is possible to create useful expert systems that address issues of knowledge representation, analysis, and generation for various geometry-driven engineering domains.

ACKNOWLEDGMENTS

The authors are grateful to Prof. George Stiny for his comments on and suggestions for this manuscript. This research effort was sponsored by the Air Force Office of Scientific Research (AFOSR), Air Force Material Command, U.S. Air Force, under grant number F49620-98-1-0172. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR or the U.S. government.

REFERENCES

Agarwal, M., & Cagan, J. (1998). A blend of different tastes: The language of coffee makers. *Environment and Planning B: Planning and Design 25(2)*, 205–226.

Agarwal, M., Cagan, J., & Constantine, G.C. (1999). Influencing generative design through continuous evaluation: Associating costs with the coffeemaker shape grammar. *AI EDAM 13*, 253–275.

Agarwal, M., Cagan, J., & Stiny, G. (2000). A micro language: Generating MEMS resonators using a coupled form-function shape grammar. *Environment and Planning B: Planning and Design 27*, 615–626.

Brown, D.C., & Chandrasekaran, B. (1986). Knowledge and control for a mechanical design expert system. *IEEE Computer 19*, 92–100.

Brown, K.N., & Cagan, J. (1997). Optimized process planning by generative simulated annealing. *AI EDAM 11*, 219–235.

Cagan, J., & Mitchell, W.J. (1993). Optimally directed shape generation by shape annealing. *Environment and Planning B 20*, 5–12.

Chomsky, N. (1957). *Syntactic Structures*, Mouton, The Hague.

Coyne, R.D., & Newton S. (1989). A tutorial on neural networks and expert systems for design. In *Proc. Australasian Conference on Expert Systems in Engineering, Architecture and Construction*, 321–337.

Dixon, J.R., Howe, A.H., Cohen, P.R., & Simmons, M.K. (1987). DOMINIC I: Progress towards domain independence in design by iterative design. *Engineering with Computers 2*, 137–145.

Fitzhorn, P.A. (1990). Formal graph languages of shape. *AI EDAM 4(3)*, 151–164.

Koning, H., & Eizenberg, J. (1981). The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and Planning B 8*, 295–323.

Krishnamurti, R. (1980). The arithmetic of shapes. *Environment and Planning B 7*, 463–484.

Krishnamurti, R. (1981). The construction of shapes. *Environment and Planning B 8*, 5–40.

Liebowitz, J. (1998). *The Handbook of Applied Expert Systems*. CRC Press, Boca Raton, FL.

Marcus, S., Stout, J., & McDermott, J. (1988). VT: An expert elevator designer that uses knowledge based backtracking. *AI Magazine 9(1)*, 95–112.

McDermott, J. (1981). R1: The formative years. *AI Magazine 2(2)*, 21–29.

Reddy, G., & Cagan, J. (1995). An improved shape annealing algorithm for truss topology generation. *ASME Journal of Mechanical Design 117(2(A))*, 315–321.

Shea, K., & Cagan, J. (1997). Innovative dome design: Applying geodesic patterns with shape annealing. *AI EDAM 11*, 379–394.

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B 7*, 343–351.

Stiny, G. (1992). Weights. *Environment and Planning B: Planning and Design 19*, 413–430.

Stiny, G. (1996). Useless rules. *Environment and Planning B: Planning and Design 23*, 235–237.

Stiny, G., & Gips, J. (1980). Production systems and grammars: A uniform characterization. *Environment and Planning B 7*, 399–408

Stiny, G., & Mitchell, W.J. (1978). The Palladian Grammar. *Environment and Planning B 5*, 5–18.

Manish Agarwal received his Ph.D. from the Department of Mechanical Engineering at Carnegie Mellon University in 1999. His research is in the area of design theory and

https://doi.org/10.1017/S089006040014507X Published online by Cambridge University Press

methodology, including product design, qualitative optimization, and formal design techniques. He has been involved in developing shape-grammar-based design representations for a number of application domains. Dr. Agarwal received his B.Tech. from the Indian Institute of Technology, Kanpur, India, in 1995 and his M.S. from Carnegie Mellon University in 1997, both in Mechanical Engineering. He is a member of Sigma Xi National Honor Society and the ASME and SDPT professional societies. Dr. Agarwal is currently working at the Chicago office of McKinsey & Company.

Jonathan Cagan is a Professor of Mechanical Engineering at Carnegie Mellon University, with appointments in the School of Design, Biomedical and Health Engineering, and Computer Science. His research, teaching, and consulting are in the area of design theory, methodology, automation, and practice. He received his B.S. in 1983 and M.S. in 1985 from the University of Rochester, and his Ph.D. in 1990 from the University of California at Berkeley, all in Mechanical Engineering. Dr. Cagan is the recipient of the National Science Foundation's NYI Award and the Society of Automotive Engineer's Ralph R. Teetor Award for Education. He is a member of the Phi Beta Kappa, Tau Beta Pi, and Sigma Xi National Honor Societies, and the ASME, IDSA, AAAI, SAE, and ASEE professional societies. Dr. Cagan is a registered Professional Engineer.

https://doi.org/10.1017/S089006040014507X Published online by Cambridge University Press