CHAPTER THREE

Engineering Shape Grammars

Where We Have Been and Where We Are Going

Jonathan Cagan

INTRODUCTION

Shape grammars (Stiny, 1980; see also Chapter 2 in this book), originally presented in the architecture literature, have successfully been used to generate a variety of architectural designs including irregular Chinese lattice designs (Stiny, 1977), villas in the style of Palladio (Stiny and Mitchell, 1978), Mughul gardens (Stiny and Mitchell, 1980), prairie houses in the style of Frank Lloyd Wright (Koning and Eizenberg, 1981), Greek meander patterns (Knight, 1986), suburban Queen Anne houses (Flemming, 1987), and windows in the style of Frank Lloyd Wright (Rollo, 1995). The derivation of the grammars and the designs generated by their use have revealed systematic logic in the generation of classes of architectural designs. For example, Koning and Eizenberg's Frank Lloyd Wright prairie house grammar demonstrated that, through the application of a relatively simple set of predefined rules, an infinite number of designs illustrative of Frank Lloyd Wright's own prairie houses can be created. One could question whether Wright consciously or unconsciously used such a set of rules to generate his houses; or one could not care but rather just marvel at the power of a simple rule set, based in the geometry of shape, to generate such rich and representative designs.

We can next ask, if the logic of architecture can be revealed and capitalized upon to represent and generate creative designs within known styles, can the same be done in engineering? In other words, is there a logical structure for the synthesis of engineering design and, if so, can it be modeled and used to create new engineering artifacts? This chapter attempts to address these questions. The simple answers are yes: there is a logical structure for synthesis within engineering, rooted in the interaction between function and form, and styles that emerge as a result of this interaction, to drive realizable design details. Through shape grammars, classes of known artifacts can be recreated, but also through shape grammars, and often the same grammar, unique and novel solutions can be created. Further, within engineering design, spaces of artifacts can be explored through directed search algorithms to drive designs toward characteristics that best meet any given set of design objectives and constraints.

The remainder of this chapter will review shape grammars and the current state of the art in application to engineering design. Next we will explore the issues surrounding the design of an engineering shape grammar and the issues that remain to be solved as we look toward future theoretical and application contributions using shape grammars.

SHAPE GRAMMARS

A shape grammar is a form of production system (see Stiny and Gips, 1980). What differentiates a shape grammar production system from a traditional (first predicate) rule-based system are (1) geometry, (2) parameterization, and (3) emergence (Agarwal and Cagan, 2000). A shape grammar (Stiny, 1980, 1991) derives designs in the language that it specifies by successive application of shape transformation rules to some evolving shape, starting with an initial shape (I). In particular, given a finite set of shapes (S) and a finite set of labels (L), a finite set of shape rules (R) of the form $\alpha \rightarrow \beta$ transform a labeled shape α in $(S, L)^+$ into a labeled shape β 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 $\langle S_\phi, \phi \rangle$. 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. An assignment g that 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 on a labeled shape in the usual way to generate a new labeled shape.

The shape grammar formalism is defined in a variety of algebras built up in terms of a basic hierarchy (Stiny, 1992). The initial algebras U_{ij}, $0 \leq i \leq j$, in the hierarchy are given in this table:

$U_{0\,0}$	$U_{0\,1}$	$U_{0\,2}$	$U_{0\,3}$
	$U_{1\,1}$	$U_{1\,2}$	$U_{1\,3}$
		$U_{2\,2}$	$U_{2\,3}$
			$U_{3\,3}$

Each algebra U_{ij} in the table contains shapes made up of finitely many basic elements of dimension i that are combined in an underlying space of dimension $j \geq i$. For example, in the algebras $U_{0\,2}$, $U_{1\,2}$, and $U_{2\,2}$, shapes are defined in a plane with points, lines, and planes, respectively. Most shape grammars have been written as a Cartesian product of $U_{0\,2}$ and $U_{1\,2}$, using points and lines that act in planes, though more recent work discussed in this chapter presents $U_{2\,2}$ grammars. Algebras of shapes can also be augmented by labels (V_{ij}, $0 \leq i \leq j$) or weights (W_{ij}, $0 \leq i \leq j$) to obtain new algebras in which the shape operations have been redefined to reflect different possible labels and weights on different entities (Stiny, 1992).

Semantics of shape grammars can be considered descriptions of the function, purpose, or use of the shapes generated by a shape grammar. Stiny (1981) defines such descriptions by a description function h: $L_G \rightarrow D$, which maps the language L_G described by grammar G onto descriptions D. These descriptions form the theoretical basis to combine function and form so important in engineering design, because the semantics of the grammar enable the form to be viewed from a given functional perspective.

WHERE WE HAVE BEEN

EARLY WORK

To date there has been quite limited exploration into the application of shape grammars in engineering design. Within engineering, and in particular the mechanical, civil, and electromechanical disciplines, function and form and their relation are critical for understanding the completeness of any design. Initial explorations into the use of grammars in engineering design focused more on function than on shape. Early work centered on discrete function grammars through the use of labels and symbols alone. Mitchell (1991) presented function grammars as shape grammars that are limited to the generation of both realizable and functional designs, and he illustrated this point with an example of a function grammar for the design of primitive huts. Fenves and Baker (1987) presented a function grammar for the conceptual design of structures, using architectural and structural critics to guide the design configuration. In a similar fashion, Rinderle (1991) presented an attribute grammar, a string grammar with a set of attributes that describe parametric and behavioral properties that are attached to every symbol, for the configuration of discrete planar structures. Finger and Rinderle (1989) described a bond graph grammar for the form and function configuration of mechanical systems. In all of these grammars, function was the focus, and any use of form came in a symbolic fashion based on the functional properties.

Early work on shape-oriented grammars can be found in Spillers (1985), where heuristics based on a grammar approach were used to introduce new structural members into a truss structure. Here, as with Rinderle (1991) above, trusses could be generated, but only in a user-directed fashion to generate limited form types. Fitzhorn (1990) and Longenecker and Fitzhorn (1991) presented shape grammars specifying the languages of both manifold and nonmanifold constructive solid geometry and boundary representations (i.e., realizable solids).

All of these efforts provided an early view of how to model the relationship between function and form within a grammar formalism for engineering application. Two aspects of these works missing and critical for the advancement of shape grammars in engineering fields were (1) a detailed application illustrating the potential power of shape grammar representations, and (2) a means to seek out useful and preferred designs within a design space generated by a shape grammar. These two aspects were critical because until a shape grammar could model a detailed engineering process and be used to generate significant design solutions, it was unproven that shape grammars were practical, even on a research scale, in engineering domains. Also, pure generate and test does not in itself effectively solve engineering problems; rather engineers desire solutions that best meet often competing objectives within tight constraints. If such a solution were to fall out of the language of a shape grammar directly and consistently, then the grammar would have to be narrowly defined and obvious, thus prompting the question of its value in solving real engineering problems. Instead, desirable solutions had to be obtained from a more general grammar in a directed manner. Two somewhat concurrent efforts focused on these two aspects of engineering shape grammars as now discussed.

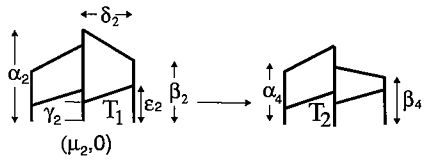
THE LATHE GRAMMAR

Brown, McMahon, and Sims-Williams (1994) developed a grammar that specifies the language of shapes manufacturable on a simplified axisymmetric lathe. The grammar represents drilling, external turning, boring, and external and internal gripping, and it provides a formal computable specification of the capabilities of the lathe, represented by ordered sets of simple two-dimensional parametric shapes. Labels attached to the shapes, together with their associated attribute values, indicate the type, position, and status of the machine tools and the grips. Each grammar rule represents a transition between a combined state and machine and a new state. Individual rules are represented as rewrite rules on sets of parametric attributed labeled shapes by using an extension of Stiny's attributed set grammar formalism (Stiny, 1982).

There are over 120 rules in the grammar, divided into six sets – one each for external gripping, internal gripping, drilling, turning, boring, and retracting the tool. The majority of the rules do not modify the shape, but examine the workpiece and collect data (using attribute values) on the state of the shape and the machine. These data then act as constraints on subsequent rule applications. At any given time, the state of the workpiece is represented by the current shape, and the state of the machine is represented by the attribute values and positions of the shape labels. The legal transitions between states of the workpiece and lathe are defined by the grammar rules. The changing values of the attributes, and the propagation of the labels over the shape as the shape changes, model the physical process that creates the finished object. The shapes in the language of the grammar then represent the parts that can be machined from the initial workpiece by using the lathe. An example rule from Brown's grammar is found in Figure 3.1, and an example of its application is found in Figure 3.2. Figure 3.3 illustrates a part generated by the grammar. This grammar illustrated the ability of shape grammars to model detailed realistic engineering systems.

SHAPE ANNEALING

In focusing on the second problem, that of directed search, Cagan and Mitchell (1993) introduced the shape annealing method, which combines a grammatical



C_L: $\delta_2 <= T_1.l,\ \alpha_2,\beta_2 < \eta$
$\gamma_2 < T_1.h + \delta_2 * T_1.m <= \alpha_2$
$\varepsilon_2 < T_1.h <= \beta_2$

C_R: $\alpha_4 = T_1.h + \delta_2 * T_1.m,\ \beta_4 = T_1.h$
$T_2.h = \alpha_4,\ T_2.l = T_1.l - \delta_2$

Figure 3.1. Example shape rule from Brown et al. (1994) (taken from Brown and Cagan, 1997).

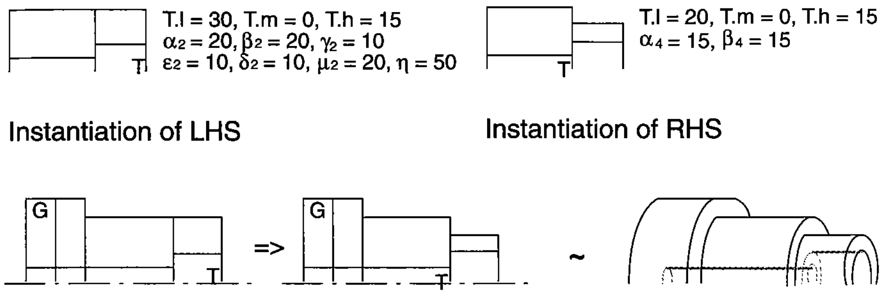

T.l = 30, T.m = 0, T.h = 15
α2 = 20, β2 = 20, γ2 = 10
ε2 = 10, δ2 = 10, μ2 = 20, η = 50
T
T.l = 20, T.m = 0, T.h = 15
α4 = 15, β4 = 15
T
Instantiation of LHS
Instantiation of RHS
G
T
=>
G
T
~

Figure 3.2. Application of rule from Figure 3.1 (Brown et al., 1994) (taken from Brown and Cagan, 1997).

formalism to define the language of design alternatives and stochastic optimization to search this language for optimally directed designs. The need for a stochastic search technique was emphasized based on the discrete and multimodal characteristics of the objective function evaluations of the spaces generated by shape grammars. Though any stochastic method was usable, we chose the simulated annealing method (Kirkpatrick, Gelatt, and Vecchi, 1983). The basic idea is that a rule satisfying the left-hand side is (randomly) selected and applied; the resulting design is then evaluated and compared with the previous state. If the new design is an improvement, it is chosen as the current design and the process continues from that design state. If, however, the new design is evaluated to be worse than the previous state, then there is still a probability that the new design is chosen as the current state; this probability is based on the metropolis algorithm (Metropolis et al., 1953) and starts out high and decreases to zero as the algorithm proceeds. The 1993 paper by Cagan and Mitchell described a simple "vanilla" annealing schedule; more sophisticated, dynamic schedules were introduced in more recent work as described below (see, e.g., Shea, Cagan, and Fenves, 1997).

Until the shape annealing technique was introduced, the basic assumption in shape grammars was that the right rule was chosen at the right time; the question of how the rule was chosen was never addressed. The motivation of the shape annealing technique was to do this automatically, driven by the objective function. The significance of this becomes clear when focusing on automated engineering synthesis; useful and desirable engineering systems are generally guided toward preferred instantiations through directed or optimizing search.

The primary application of the shape annealing technique has been in the area of structural design. The design representation used in the shape annealing method is based on an analogy to a network (Cagan and Mitchell, 1994). A network is a means of representing design objects and the coupled relations between these objects, where design objects and their relations can have both functional and spatial attributes. It

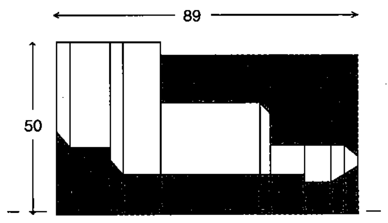


Figure 3.3. Standard part generated by Brown's lathe grammar (Brown et al., 1994) (taken from Brown and Cagan, 1997).

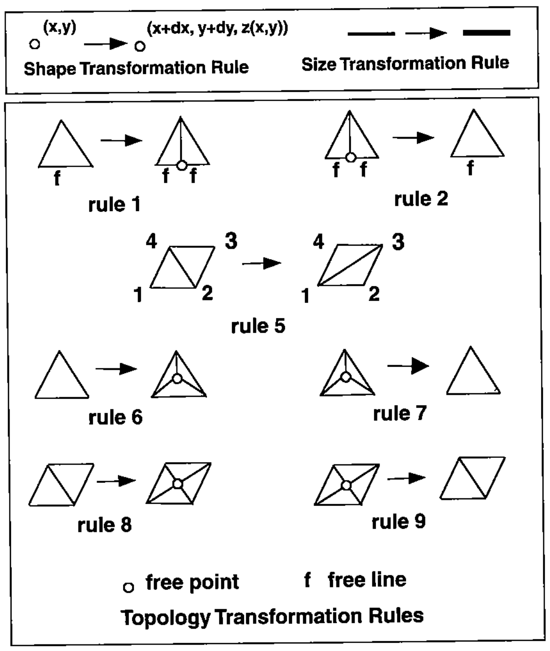


Figure 3.4. Space truss grammar (Shea and Cagan, 1997).

is these capabilities that make it amenable to structural synthesis: a network applied to discrete structures represents structural shape, where the design objects are joints and the relations between joints are the flow of force through structural members. A network, and thus a structure, can be evaluated and optimized based on global and local design goals of design object and relation attributes and acceptable levels of system flow, that is, behavioral constraints. The structural generation and optimization using shape grammars and shape annealing was originally introduced by Reddy and Cagan (1995a, 1995b). However, it was the work of Shea and Cagan (Shea, 1997; Shea et al., 1997; Shea and Cagan, 1997, 1999a, 1999b) that brought the technique to maturity as a consistent, robust method for structural synthesis. In this method, a simple shape grammar that represents a language of structures (mostly truss structures) is used; an example grammar, shown in Figure 3.4, represents a space truss grammar used to generate geodesic-like domes (Shea and Cagan, 1997). During shape annealing, a finite-element analysis is completed at every iteration to provide evaluation of the design objectives and constraints such as minimum weight and surface area while not failing by yield stress and buckling conditions. Dynamic annealing schedules and dynamic probability-based rule selection techniques are one contribution toward making the technique robust and efficient (see Shea, 1997, for details). Further, the method supports any articulable objective functions and constraints. A wide variety of both traditional and very interesting novel structures have been generated with the shape annealing technique. Example geodesic-like domes under various loading conditions and design objectives can be found in Figure 3.5.

One interesting aspect of this work is that syntax defines legal designs whereas semantics expresses design intent and supports evaluation. Function is implied in the grammar, but it is the external finite-element analysis that provides performance

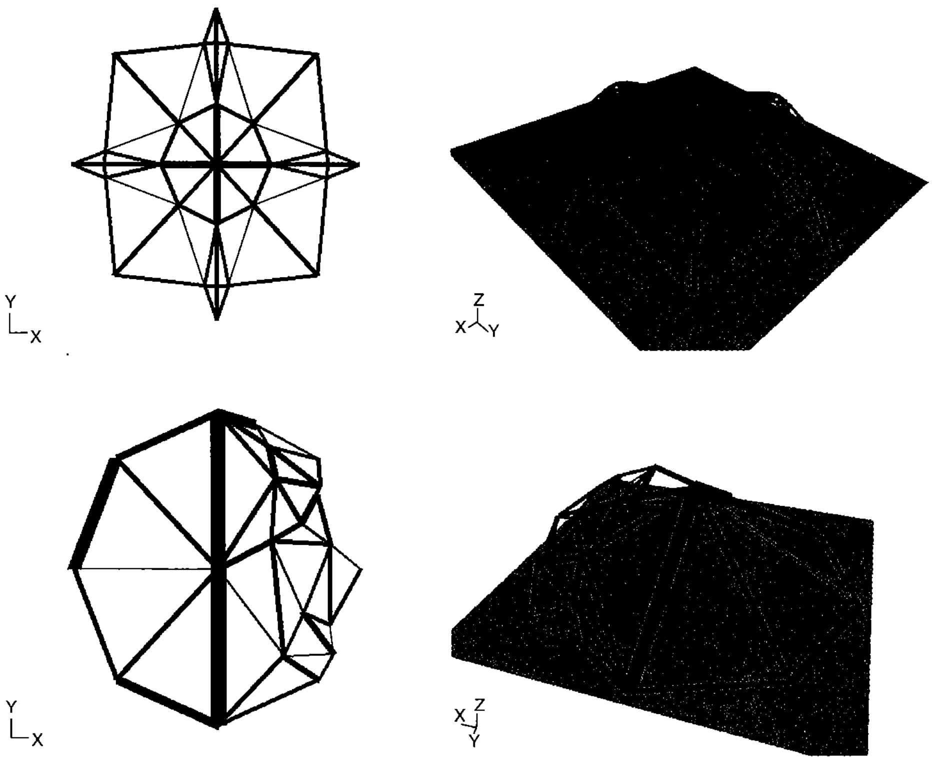


Figure 3.5. Symmetric and nonsymmetric geodesic-like domes generated by shape annealing (Shea and Cagan, 1997).

evaluation. Directed search through the simulated annealing algorithm seeks out desirable designs based on given objectives. The dome study (Shea and Cagan, 1997) and a roof study (Shea and Cagan, 1999a) best illustrate the varied design characteristics (called "design essays") that result from differing design objectives. The grammar itself, then, restricts the design space to geometrically feasible (though not necessarily functional) designs; the optimization drives solutions to be functional and high performing. This separation is important in engineering shape grammars; almost any specific problem will have a different set of objectives and constraints, so for a grammar to be useful beyond a single application it must be general, relying on the search strategy to filter and direct the generation of the solution.

Another significant application of shape annealing comes from the merging of Brown and coworkers' manufacturing grammar with shape annealing, resulting in a method to create optimal manufacturing process plans (Brown and Cagan, 1997). The underlying process model is developed by using a grammar of shape as described above. The formal semantics of the grammar interpret shape operations as manufacturing information, and the grammar and semantics are intended to provide a complete and correct specification of the domain. Planning is then a search process within the space defined by the grammar, using the semantics to guide the search. At any given stage, the approach can generate a possible operation by inspecting the current workpiece and the target part. Proposed operations are then translated into a sequence of grammar rules, which are applied to the current shape, and interpreted

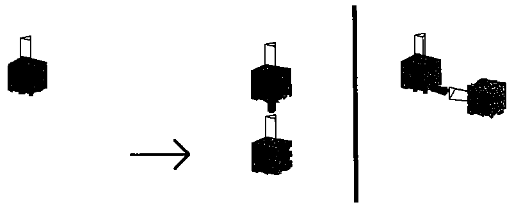


Figure 3.6. Example rule from the robot arm grammar (Wells, 1994).

as a detailed operation with associated costs. The cost of completing the plan is also estimated from the new state. The shape annealing algorithm directs the search by trying forward moves or backjumps at random, and accepting or rejecting the resulting states by using a probabilistic decision procedure. The algorithm probabilistically converges on near-optimal plans.

The system essentially works on two levels: a higher level for postulating operations (using macros), and a lower level for ensuring that the postulated operations are legal and for generating the consequences (through the grammar rules and semantics). What is also interesting is that at any intermediate stage, only a predictive measure can be used to evaluate the design state (you don't know what it will cost until its plan is completed); thus decisions as to whether to accept or reject a state are based on a heuristic cost formulation.

Shape annealing enabled computer-based directed design generation. The lathe grammar illustrated that real systems could be modeled. From this point several applications have been pursued for both modeling and search with engineering shape grammars, as next discussed.

THE ROBOT ARM GRAMMAR

Wells and Antonsson (Wells, 1994) presented an engineering shape grammar (that they term an *engineering grammar*) to generate configurations of modular reconfigurable robot arms. The grammar generates all nonisomorphic assembly configurations while simultaneously calculating kinematic properties of the arms. The grammar models the properties of dyads and connects them together by means of arms. Within the grammar is modeled knowledge about how dyads behave; thus function is modeled through constraints embodied within the grammar that restrict the possible configurations that can be generated. An example rule can be found in Figure 3.6, and example configurations generated by the grammar can be found in Figure 3.7.

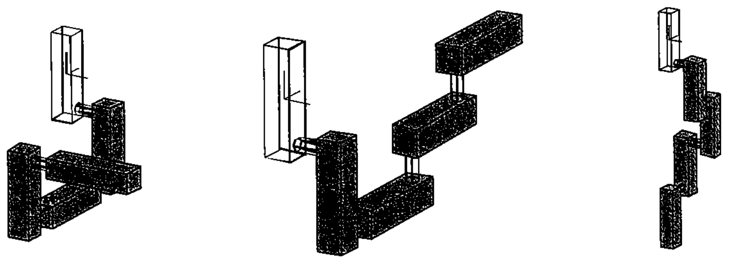


Figure 3.7. Example robot arms generated from Well's grammar (Wells, 1994).

MOVING ON: A COMMENT ON FUNCTION AND FORM BY MEANS OF ABSTRACTION GRAMMARS

While focusing on engineering design problems, the relationship between the form of the device being designed and the function of the device is, as mentioned, critical. Within the structures work discussed above, the grammar itself was simple, modeling fundamental stability relationships while relying on external evaluators to interpret the function and behavior of the design. Within the robot arm grammar, function is directly mapped to the rules that constrain configurations. As machines and other more intricate products are considered, function must be addressed in a more fundamental manner. Representing function and its relationship to form, however, is a challenging problem. Early work on function described above, and placed within the scope of design descriptions as laid out by Stiny (1981), begins to address issues of discretely separated relationships between function and form.

Schmidt and Cagan pursued function grammars as applied to machine design within a directed search framework (Schmidt, 1995; Schmidt and Cagan, 1992, 1995, 1997, 1998). Originally conceived of as a shape grammar formalism application within shape annealing (Schmidt and Cagan, 1992) in which machines would evolve out of a stochastic annealing process with machine components represented by shapes, it was quickly recognized that the intricacies of functional descriptions within a directed search framework were in and of themselves a difficult problem that had to be addressed. A hierarchical *abstraction grammar* was laid out to discretize a continuum of function to form into useful levels for product function and form representation. One implementation uses three levels of abstraction: energy, kinematic mode plus energy, and component level. The component level represents form, but not, in practice, shape; for atomic components this may be sufficient, but the assumption of a decoupling between function and shape is strong here. Because metrics do not exist on the abstract level, but only on the form levels, and yet design conceptualization occurs on the function levels, directed search occurs along the hierarchy through a recursive simulated annealing algorithm. Both string and graph grammars were developed, generating concepts for optimally performing drills and carts. More recently, Schmidt extended the graph grammar work to the design of planar mechanisms (Schmidt, Shetty, and Chase, 1998), in which interesting kinematic structures of epicyclic gear trains are generated from the grammar. Still, the coupling that can occur between function and form has to be addressed. See Chapter 6 in this book for more discussion on functional representation and synthesis.

CONSUMER PRODUCT DESIGN: THE COFFEE MAKER GRAMMAR

The next direction for shape grammar development has been in the area of product development, with consumer products as the initial focus. With the consideration of products again comes the consideration of function as well as shape. The design of consumer products (such as coffee makers, telephones, toasters, and flashlights) often tends to be driven by a basic functional decomposition, but the products themselves are differentiated by form. Thus function and related form can be treated through discrete subsystems. This is true of the coffee maker shape grammar introduced in

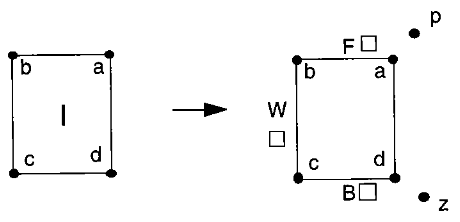


Figure 3.8. An initial shape rule from the coffee maker grammar that chooses a one-heater unit (Agarwal and Cagan, 1998).

Agarwal and Cagan (1998), the first to focus on a class of products. The grammar consists of 100 parametric rules, grouped within functional subsystems of the coffee maker, namely the filter, water storage container, and base, and each rule targets either the function or the form aspect of the coffee maker; not all 100 rules are used, but each functional unit must be designed. The grammar is a $V_{0\,2} \times V_{1\,2}$ grammar operating on three views (top, side, front), resulting in a complete three-dimensional representation of the product.

The initial shape for the coffee maker, that of the coffee pot region, is shown on the left-hand side of Figure 3.8. Points labeled a, b, c, and d are parametric points of the vertical cross section that allow the design of coffee makers of different sizes. The first set of rules, called initial shape rules (rules 1–5), distinguish between the two main classes of coffee makers, those with one heating element and those with two, and break apart the basic form into three regions (for the filter, base, and water storage units). Labels F, B, and W are used to distinguish among the three regions. A square label is associated with each of the three regions for a one-heater design, and a triangle label for a two-heater design. The rule shown in Figure 3.8, for example, chooses a one-heater unit. The points labeled by p and z indicate the rightmost bounds on the design. The points are constrained to have the same rightmost (or $x-$) coordinate.

The next step (rules 6–26) is to design the details of the filter unit, first specifying functionality like sliding versus rotating filters, cone versus flat filters, and whether or not there is a drip stop in the filter. Then the form of the filter is designed to meet the functional specifications; see, for example, Figure 3.9(a). Next the details of the base design are specified (rules 27–37), enabling a polygonal or oval–circular base design; see Figure 3.9(b). Included in the base is the specification of the appropriate heater.

The process next turns to the water storage unit, the main differentiator among different form designs. Initially the cross-section details are specified (rules 38–42), including the placement of the water tubing (Figure 3.10). Rules 43–100 then enable the form design of the water storage unit and its integration with the filter and base units. Here four planes are specified (top and bottom of the filter and top and bottom of the base), and at each plane the shape of the top view section is specified; the final three-dimensional form is then blended between the four planes. Almost any cross-sectional shape can be specified by selecting any number of circles or squares in sequence and then sweeping each in a straight line or circular arc, possibly changing the size or aspects of the circles and square through the sweep. Each shape is then blended together. Figure 3.11 shows the top view of such a sequence, including its merging with the circular filter unit.

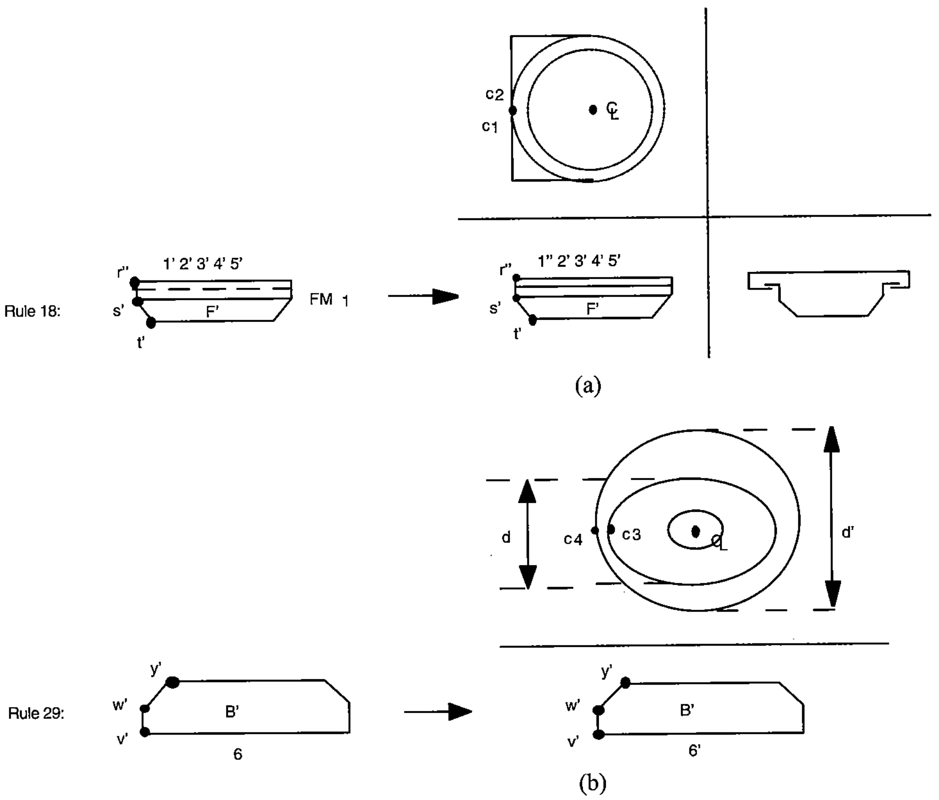


Figure 3.9. Example rules for the (a) filter and (b) base design from the coffee maker grammar; different views are shown (Agarwal and Cagan, 1998).

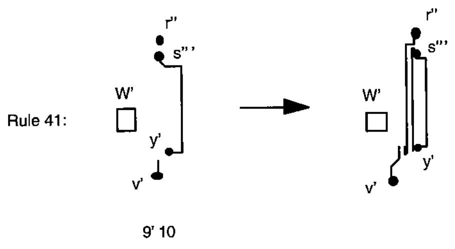


Figure 3.10. Placement of the water tubing in the water storage unit (Agarwal and Cagan, 1998).

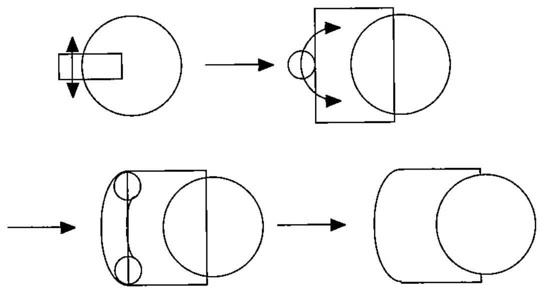


Figure 3.11. Generation and blending of the top view of a water storage unit (Agarwal and Cagan, 1997).

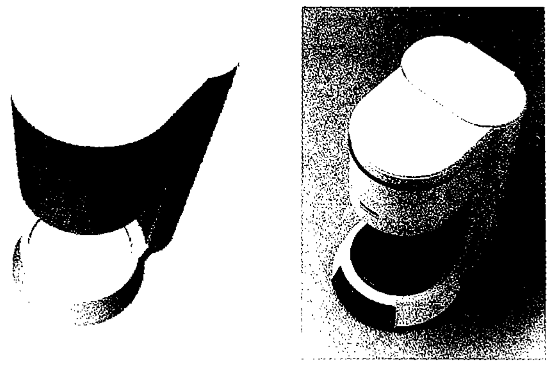


Figure 3.12. Coffee maker (left) generated by the grammar and (right) a Rowenta FK26-S (Agarwal, Cagan, and Constantine, 1999).

By following the grammar, starting with the initial shape of the coffee pot, both traditional and novel coffee makers can be designed. Figure 3.12 shows a coffee maker generated by the grammar and a Rowenta FK26-S coffee maker of the same basic form; Figure 3.13 shows two novel designs. Agarwal and Cagan point out that by modifying any one rule or parameter during the synthesis of a given design, one can generate a different coffee maker, defining an infinite number of such products that follow the basic function–form decomposition. Further, by recreating known coffee maker forms, one can identify certain defining characteristics of a product in terms of key rules; these rules can then be applied during other design sequences, resulting in products that have similar characteristics.

Although the form is in itself interesting, from an engineering perspective, the evaluation of any product is critical, both in providing an evaluation of the final design and also in providing feedback to aid in choosing from various rules during the generation process. Agarwal, Cagan, and Constantine (1999) argue that using performance metrics along with a grammar-based generative system will create a powerful feedback mechanism for the designer during the design generation process. To explore the ability of shape grammars to support the analysis of the resulting designs, they have incorporated costing evaluation into the coffee maker grammar. As any of the 100 rules is selected and its parameter values defined, the current cost of the coffee maker is evaluated. A manufacturing cost structure is defined for injection-molded parts, metal stamped parts, and product assembly, and it is incorporated into the grammar to obtain cost estimates during the generation process.

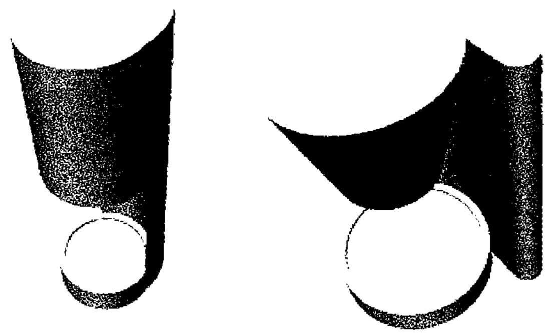


Figure 3.13. Two novel coffee makers generated by the grammar (Agarwal and Cagan, 1998).

The manufacturing structure is incorporated into the grammar by relating the part geometry and rule semantics to the various manufacturing processes. The cost of the coffee maker generated by the grammar shown in Figure 3.12 is estimated as \$7.32 for the cost of manufactured and purchased parts and the cost of assembly; the same coffee maker is estimated as costing within 3% of this estimate within the literature.

A MEMS GRAMMAR

This next shape grammar requires a deeper exploration into the coupling that can exist between function and form. Agarwal, Cagan, and Stiny have developed a shape grammar to generate microelectromechanical systems (MEMS) resonators (Agarwal and Cagan, 1999; Agarwal, Cagan, and Stiny, 2000). One aspect of the design of MEMS devices that makes the problem difficult is the strong form–function coupling; a small change in the topology of a design may result in significant performance changes. The grammar is designed to first satisfy the minimum required functionality of the device (by including at least one actuator and one spring) and then modify the device to obtain the desired specifications. The grammar is a parametric labeled $U_{2\,2}$ grammar augmented by weights, that is, a $U_{2\,2} \times$ {weights, labels} grammar, to differentiate and bring precedence to various types of components. Note that a two-dimensional representation is sufficient because MEMS resonators are 2-1/2 dimensional devices. The four main elements of a resonator are the central mass, actuators, anchors, and springs; each of these is assigned a different weight. The original grammar was a nonweighted labeled $U_{1\,2}$ grammar of 50 rules. The weighted labeled $U_{2\,2}$ grammar consists of only 15 rules to accomplish the same design characteristics.

Example rules are found in Figure 3.14, illustrating the addition of a comb drive actuator, the addition of a spring, and the modification of a spring. Figure 3.15 shows a scanning electron microscopy (SEM) image of an actual MEMS device, a similar MEMS device generated by the grammar, and a novel MEMS device also generated by the grammar.

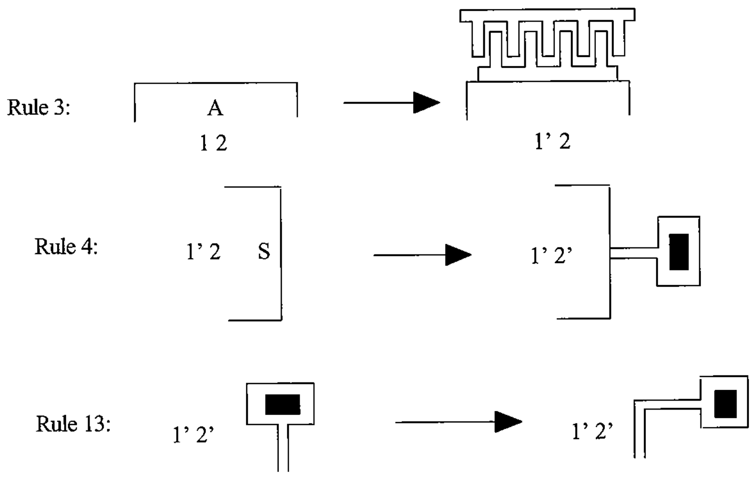


Figure 3.14. Example MEMS grammar rules (from Agarwal et al., 2000).

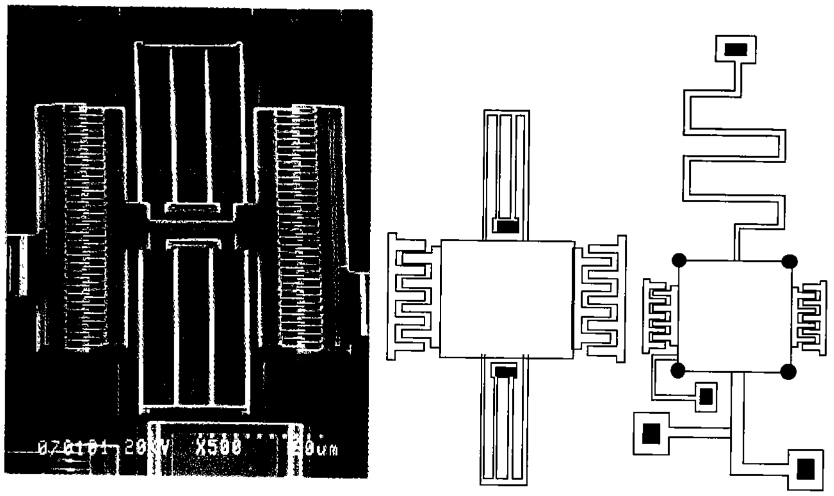


Figure 3.15. SEM image of a MEMS device, a similar device generated by the MEMS grammar, and a novel MEMS device generated by the grammar (from Agarwal et al., 2000).

The idea is not to just create a grammar, but rather a generative system – a designer assist tool – thus there is a need to align the grammar with a directed search algorithm such as shape annealing. Regardless of the final technique employed, the project emphasizes that in engineering design some form of directed generative design in conjunction with the shape grammar is what will lead to an effective design tool.

DESIGNING ARTIFICIAL HEARTS

McCormack and Cagan at Carnegie Mellon, and Antaki at the University of Pittsburgh's McGowan Center for Artificial Organs (McCormack et al., 1999) have developed a novel shape grammar[1] for the design of an artificial heart. The group at the McGowan Center have invented a new class of turbine-based artificial hearts. This grammar models the class of hearts and all variations, an infinite number of them. The grammar demonstrates a strong coupling between form and function and requires a detailed analysis to evaluate its performance, based on fluid dynamic, rotodynamic, and electromagnetic criteria. The application is certainly appealing, but it is also very pertinent in that the grammar shows potential to be used in future design generations of these devices.

The grammar is a $V_{0\,2} \times V_{1\,2} \times V_{2\,2}$ grammar currently composed of 51 rules. The grammar does not control the order of the design of the different subsystems of the heart, although there is precise termination. Labels drive the design and guarantee that all parts are eventually designed, even though any subsystem can be fully or partially defined in any order. The grammar is set up to define boundaries between subsystems that are defined by variables represented through labels; the four major

[1] The grammar was developed with additional input from Bradley Paden from University of California at Santa Barbara.

subsystems are the design of bearings, impellers, motors, and stators. Figure 3.16 illustrates two artificial hearts generated from the grammar. The first, Figure 3.16(b), is a re-creation of one of the current designs from the University of Pittsburgh, Figure 3.16(a), whereas the second, Figure 3.16(c), is a novel design generated by the grammar; in these figures the half-plane of a rotationally symmetric design is shown.

In addition to the shape grammar, engineering analyses can be associated with the rules to perform approximate evaluations of the devices generated. The vision is to first create a tool to assist designers in conceptualizing, visualizing, and gaining immediate feedback on new designs. Next the optimal selection of parameters and shape rules can be explored to help identify superior concepts and instantiations from the infinite set of devices modeled by the grammar.

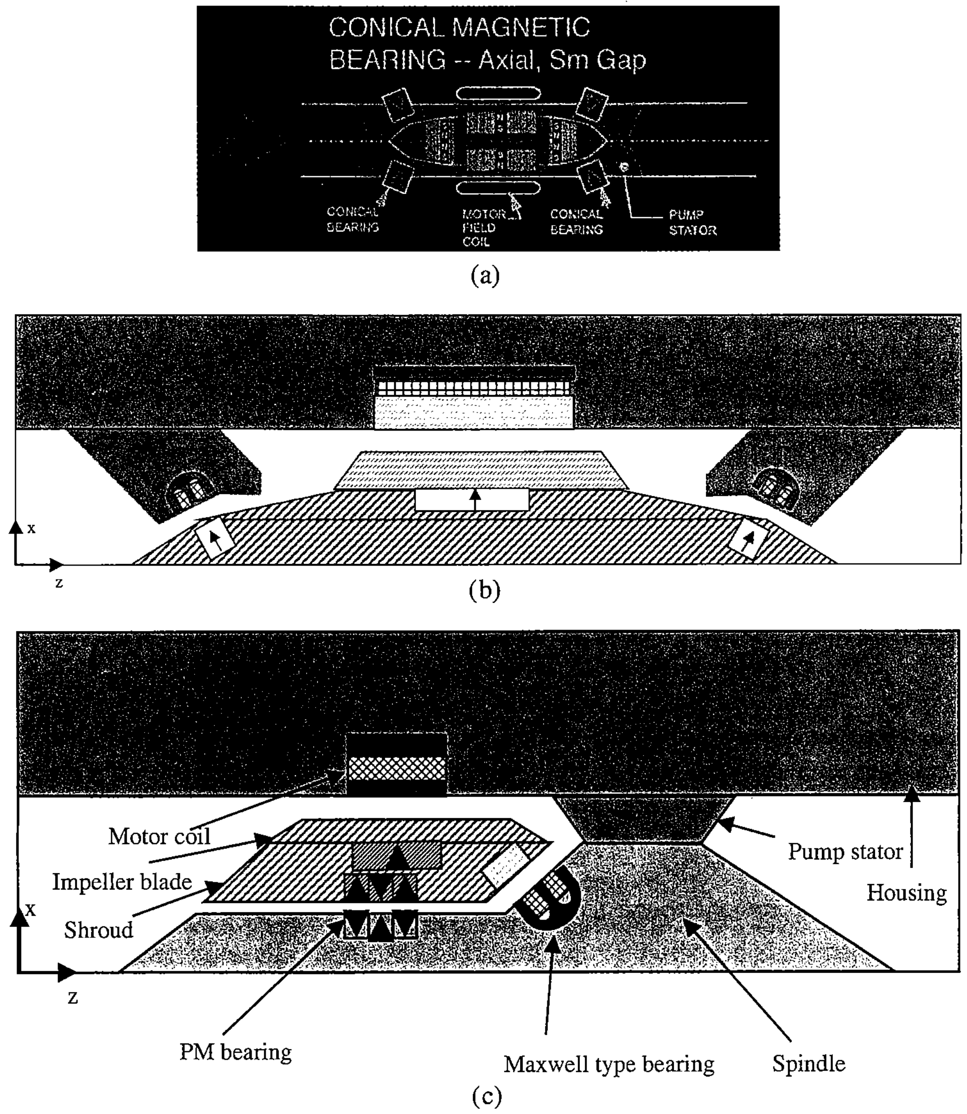


Figure 3.16. Artificial hearts generated from the artificial heart grammar: (a) actual design (reprinted with permission of J. Antaki), (b) similar design from the grammar, and (c) new design concept.

AIRPLANE TUBE ROUTING GRAMMAR

There is one example of a shape grammar being used in industry today. Heisserman led an effort at Boeing that has developed a shape grammar to route systems tubing through an airplane by using their Genesis generative design system (Heisserman, 1991, 1994; Heisserman and Callahan, 1996). The design representation in Genesis includes geometric models (boundary representation solid models), parts and assemblies, part classifications, part interfaces, and functional schematics. Genesis is used to interactively generate complete CAD geometry of aircraft tubing, including the associated fittings, clamps, and mounting brackets. The rules for this application include the manufacturing constraints for the geometry of the tubes, constructive rules for tube routing, fittings types and compatibilities, materials, and clearance constraints between different components and systems. Genesis was used to design several hundred tube assemblies on the 767-400ER. Details of the grammar are not available, yet the recognition that this technology has moved to the point of being applicable to industry is important to motivate future development of engineering shape grammars.

A COMMENT ON 1stPRINCE

From a personal perspective, it is interesting to review the work on 1stPRINCE (Cagan and Agogino, 1987, 1991a, 1991b; Aelion, Cagan, and Powers, 1991, 1992), which introduced mathematical expansion techniques to modify a design state by transforming geometries with the goal of creating innovative, better performing designs. All transformations were based on first principle formulations and reasoning, and they were directed by symbolic optimization techniques. One technique, called dimensional variable expansion (DVE), was formulated by noting that an integral can be divided into a series of integrals over subranges; if the properties within each subrange are allowed to be independent, then the effect is to expand the problem formulation. DVE applied over the radius to a rod under torsion load transformed the design into a hollow tube to minimize weight; when it was applied over volume to a mixed flow reactor, a series of mixed flow reactors were generated that, through inductive techniques, transformed into a plug flow reactor to maximize conversion rate. Note that those expansion techniques could be viewed as shape grammar rules, associated with detailed analytical expressions, driven by a highly constrained and intricate label system, and directed by an external optimizing search mechanism.

IMPLEMENTATION ISSUES

Most shape grammars found in the literature have not been implemented. Such implementations are of course required if the grammar is to be part of an automated generative system, and most of the more recent grammars discussed in this paper are either implemented or are being implemented. Our $V_{1\,2}$ truss grammar is implemented as a labeled, directed graph, based loosely on a winged-edge solid boundary representation (Baumgart, 1975; Heisserman and Woodbury, 1994). $U_{2\,2}$ grammars carry the difficulty of reasoning and manipulating planes directly (and not just the line and points that define them). One approach used in the MEMS grammar was to formalize the grammar as a $U_{2\,2}$ grammar with associated labels and weights yet implement the system through an equivalent $V_{1\,2}$ grammar; although not a pure one-to-one mapping of rules, there is a clear mapping of one to several between the two grammars. The artificial heart $V_{2\,2}$ grammar was implemented as a $V_{2\,2}$ grammar directly. The coffee maker grammar has been implemented in Java and, at this time can be accessed via the author's research web page.

All of these grammars have been driven by labels rather than emergent properties; except for the MEMS grammar, all of the them are set grammars, a subset of shape grammars. So long as emergence will not be required within an implementation, there is no need to implement the grammar using maximal lines (see Krishnamurti, 1980, 1981) or other means to detect emergent shapes; however, as emergence becomes more relevant, such fundamental work will become more pertinent. The difficulty in writing a grammar interpreter for engineering application is that shapes are not simple orthogonal lines or blocks; but tend to be more complex. McCormack and Cagan (2000) have introduced a parametric grammar interpreter able to recognize parametric emergent shapes by introducing a decomposition of shape through a hierarchy based on engineering properties. The result is that any $U_{1\,2}$ or $V_{1\,2}$ grammar (in the current implementation, though the theory extends to other algebras) can be quickly implemented. Further, the interpreter enables grammars to be written that utilize emergent shapes; such a feature is important for creative design. This interpreter should open up the scope of realizable engineering shape grammars. For example, we are using the interpreter to develop and implement a shape grammar for General Motors for the design of vehicle panels that uses emergent properties of shapes.

CREATING AN ENGINEERING SHAPE GRAMMAR

The current state of engineering shape grammars has been reviewed. Recent progress has demonstrated their use in very real settings, hopefully motivating others to think about applications. Below the issues that need to be considered in creating a shape grammar and the research issues that still must be addressed are considered.

DESIGNING A NEW GRAMMAR: WHAT ARE THE ISSUES?

There are eight fundamental issues that need to be discussed in relation to designing engineering shape grammars:

Simple Grammar Versus Knowledge Intensive Grammar. A simple grammar is one like Shea and Cagan's truss grammar, quite basic in that only a half-dozen rules are able to describe all legal geometries. A knowledge-intensive grammar, on the other side, is one like Agarwal and Cagan's coffee maker or McCormack et al.'s artificial heart grammar; these grammars act as a sophisticated expert system, one rich in geometric reasoning where creativity can be supported (as discussed below). The knowledge-intensive grammars generate feasible, functional designs, while the simple, more naïve grammars generate topologically valid but not necessarily feasible or functional solutions – a truss can be generated which protrudes through an obstacle and is overstressed. The knowledge-intensive grammars can be directed through search to generate designs of certain characteristics, but the simple grammars *require* such directed search.

Generation Versus Search. In the generation mode, use of a grammar produces designs that are feasible but are not likely to best meet certain design criteria. In the search mode, the grammar is used to search the design space for designs of certain characteristics and performance; in particular, directed search techniques such as shape annealing seek out optimal solutions among the many feasible designs within the language of the grammar.

These two issues, that of simple versus knowledge intensive and generate versus search, are interconnected as seen in Figure 3.17, which plots the amount of directedness versus the level of knowledge. The structural shape annealing method, combining a simple grammar with directed search, appears in the lower right. The robot arm grammar, because of its relative simplicity, lies in the lower left. The coffee maker grammar, in its current form a knowledge-intensive grammar that can generate designs without search, appears in the upper left. If the artificial heart grammar were to be used within an optimization framework, it would move from the upper left to the upper right. The same happened to the lathe grammar when merged with shape annealing. Knowledge tends to improve the efficiency of the use of a grammar but also restricts the amount of exploration done and novelty within the designs generated. The MEMS grammar, if used within an optimizing search strategy, would likely show a balance between how much knowledge is put into the grammar and the amount of pure exploration that can occur.

Independence Versus Coupling of Form and Function. The coffee maker grammar clearly was designed based on the decomposition (a.k.a. independence) of the functions of the different units. The MEMS grammar, though still identifying discrete function, is dependent on the coupling of the form and function, with the addition and modification of springs profoundly effecting the performance of the device; this effect is due to the continuum nature of the spring structures. The need for coupling over independence comes not from any notion of a more or less pure engineering representation, but rather from the need of the particular class of designs. That said, there is still much research to be done in modeling the potentially tight coupling between the function and form (and resulting behavior) of engineering systems within shape grammars.

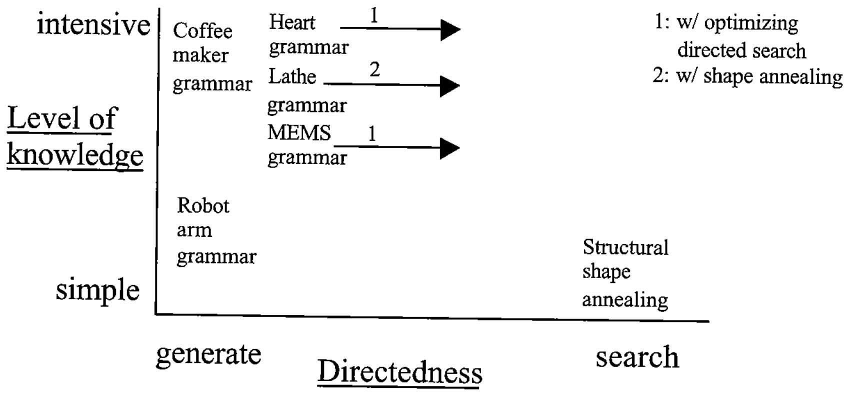


Figure 3.17. Connection between knowledge level and search level of grammars.

Symbolic–Atomic Grammar Versus Emergent Grammar. To date, emergence has not occurred in the pure sense in implementations of an engineering grammar. Some of the early truss shape annealing work showed trapezoids in the final design generated only from triangles; however, this came from the conscious decision to remove a truss member. For configuration problems of atomic components, emergence can happen only through chunking of components into subsystems; however, pure form emergence could result from models of continua or from nonsymbolic representations (e.g., maximal lines). Current work on applications of shape grammars in industry are indicating the benefit of supporting such emergence.

What makes these emergent forms interesting from an engineering perspective is really the implied function within that form; in engineering design, form often is designed to follow function. Thus the consideration of functional emergence becomes even more important. Chunked components effectively represent functional emergence such as a gear on top of a rack being grouped into a rack and pinion. Less obvious functional emergence might occur if a rib is used for structural support but also improves heat transfer capabilities. It may be that functional emergence stems from changing perspectives on the same representation.

Fixed Versus Parametric Grammar. Generally speaking, parametric grammars have more flexibility that is appropriate for most engineering applications. All of the shape grammars since Brown's lathe grammar have been parametric. The parametric nature of shape grammars enables the grammar to concisely represent large (sometimes infinite) variations within a class of designs. It is that characteristic that makes them most attractive for application to intricate engineering domains. However, fixed grammars may have use in engineering design when combinatoric enumeration of symbolic–atomic components is desired (such as with the robot arm grammar).

Type of Algebra that Will Best Model a Given Application. At a minimum, lines (a $U_{1\,2}$ grammar) will be necessary to represent any two-dimensional grammar, and thus any interesting engineering system. A $U_{2\,2}$ grammar may have more concise and powerful expressive properties if significant transformation occurs in the plane; however, with the $U_{2\,2}$ grammar comes complexities in implementation. Because the third dimension is important to many engineering artifacts, $U_{i\,3}$ grammars will become important as the maturity of engineering shape grammars continue to develop. To date we have been able to ignore the third dimension directly by considering symmetry (e.g., the heart grammar) or working only in planes with simple abstractions into the third dimension (e.g., the coffee maker grammar). Note that label ($V_{i\,j}$) and weight ($W_{i\,j}$) grammars each play an important role as well; the issues discussed in this chapter for shape ($U_{i\,j}$) grammars relate as well to label and weight grammars.

Including Evaluation with the Grammar. Evaluation is time intensive, and the effort to associate the proper level of evaluation with the grammar rules is high. It may be that from an education perspective, or when exploring the form space, evaluation is not necessary. However, engineering systems are driven by performance, and without an evaluation system design generation will occur in a vacuum. Thus evaluation of engineering designs is, in practice, necessary. The issue of when to evaluate, however, is dependent on the application and intended use of the grammar. If one wishes only to explore the design space and then observe behavior and performance, then postgeneration evaluation is acceptable. Further, if the cost of generation and evaluation is small then it may be acceptable to generate complete designs and then evaluate them. However, the power of shape grammars can be better illustrated if evaluation is provided during the generation process itself to provide feedback to the design engine (human or computer). The problem becomes *how* to evaluate intermediate designs. With the coffee maker grammar, intermediate evaluation was straightforward because manufacturing cost was directly dependent on the current geometry and context. With the lathe grammar, intermediate evaluation was much more difficult because the cost of the manufacturing plan was dependent on all of the steps toward creating the part; there an approximation was used as an intermediate evaluation.

Routine Versus Creative Design. Within bounds, both can be supported by shape grammars. Clearly shape grammars, as a form of production systems, can support routine design. A question (and potential criticism) of shape grammars is whether they support creative solutions. Brown and Cagan (1996), described the space for creativity within a shape grammar formalism as *bounded creativity*. They argue that, for a given design problem, there might exist a body of standard techniques, taking a designer through a series of actions, specifying the order in which subsolutions should be specified, and prescribing methods for achieving certain objectives, all specifying a routine design problem. Creativity, in their definition, occurs when a designer diverges from these standard procedures in the process of solving the problem. They contend that grammatical design is compatible with this definition, and, in particular, with bounded creativity. Realistically, a grammar does restrict the world of possible designs that can be generated. However, it also supports more focused designs within a world of legal configurations, and exploration beyond the few standard design instantiations often considered; see Figure 3.13 for novel coffee makers, or Figure 3.15(c) for a novel MEMS device. If leaps of technologies or new fundamental ways to look at a class of problems are desired, then a given shape grammar is unlikely to be useful and is unlikely to support that level of creativity. If, however, within an open but reasonably well-defined space novel configurations are desired, then a shape grammar is an ideal choice of representation. The idea of bounded creativity is that the grammar helps the designer move out of the traditional way of looking at the design problem.

WHAT SHOULD A GRAMMAR MODEL?

Next we explore what a grammar should model and how to determine whether it meets the needs specified. The issue becomes how expressive is the language of the grammar and how large is the design space it models. According to the discussion on bounded creativity, the larger the design space the more likely we are to discover novel designs; the smaller the design space, the more efficiently we can explore it. Figure 3.18 shows the space of all topologically valid designs ($\mathcal{S}$) and the space of all feasible designs ($\mathcal{F} : \mathcal{F} \subseteq \mathcal{S}$), where feasibility here is defined based on the satisfaction of design constraints. How much of these spaces should the grammar model? The Language A (L_A), which generates designs D_A, illustrates a grammar that describes a

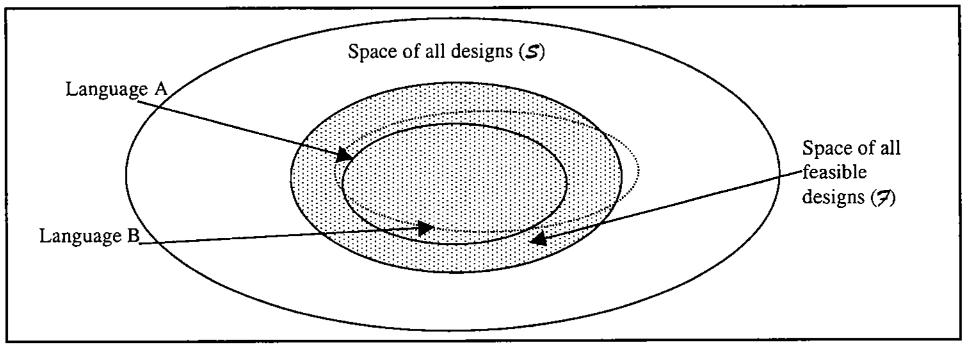

Space of all designs ($\mathcal{S}$)
Language A
Language B
Space of all feasible designs ($\mathcal{F}$)

Figure 3.18. Mapping of the feasibility of the space generated by various types of grammars.

proportion of the space of feasible designs ($D_A : \forall D_A \in \mathcal{F}$); all designs in the language are feasible. The dotted Language B (L_B), which generates designs D_B, illustrates a grammar that describes a portion of the feasible designs but also some designs that are not feasible ($D_B : \exists D_B \in \mathcal{F}, \exists D_B \notin \mathcal{F}$). If one wishes to model all feasible designs, then A would be the desirable grammar where the size of A is as large as possible *within* the feasible space. However, assume that a grammar within the feasible space covers only a small portion of that space, whereas a different grammar includes a large portion of the feasible space but also a few designs that are not feasible. In this case, which is better?

An argument could be made that the shape grammar should model only feasible designs – in this case Language A would be preferred. However, for automated design generation, if the cost of generating designs is low and the cost of evaluating those designs is also low, then one may wish a more complete exploration of the space with an external evaluator pruning out the infeasible, as well as suboptimal, designs – in which case Language B would be preferred. As the cost of searching the space or evaluating designs increases, the desire for a more compact language (and thus grammar) is desired. Along with this argument is an understanding of the characteristics of the resulting design space and where the preferred solutions (e.g., optima) lie; if the grammar can concisely pinpoint an optimal solution then the compact language would be desirable – why search the space if the one solution desired is known? If the optima, or good solutions, or at least those solutions that one wishes to explore, are distributed throughout the feasible space, then a more comprehensive grammar would be desired, possibly even if several infeasible designs were modeled.

The ideal grammar would be comprehensive yet model only feasible designs. However, we require the ability to explore the design space and seek out novel solutions. In engineering applications, evaluation is critical, and thus a good evaluator will help one search the space. If the only way to search a larger feasible space is to include some infeasible designs, then the best solution is to allow infeasible designs in the grammar and let the evaluator filter them out. All of the knowledge-intensive grammars that have been created have modeled only feasible designs based on the set of constraints imposed (those that dealt with function rather than performance), yet all of them have represented very large design spaces in which novel designs have been found. With simpler grammars as used in structural shape annealing, the

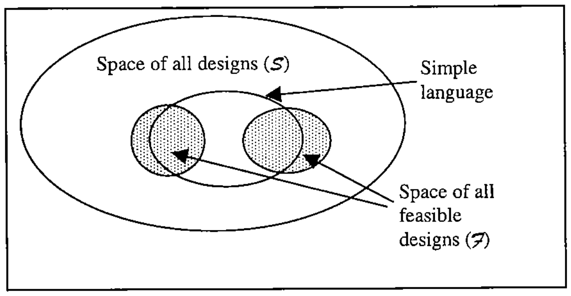

Space of all designs ($\mathcal{S}$)
Simple language
Space of all feasible designs ($\mathcal{F}$)

Figure 3.19. Illustration of disjointed feasible space.

grammars modeled only legal truss configurations (topologies), but they did allow for designs that were structurally infeasible (too large a stress in an element) or penetrated a geometric obstacle; those "infeasibilities" were easily ascertained from the evaluator and penalized until they became feasible; the annealer allowed for inequality constraint violations and then pushed the design into the feasible region as the annealing progressed with the idea that moving through an infeasible design could lead to a better feasible solution. This is seen in Figure 3.19, where feasible designs are formed by disjoint clusters. A simple grammar might include both clusters but also the infeasible designs in between (e.g., with the truss grammar the infeasible designs may be mapped by an obstacle). Here an external evaluator and directed search mechanism (shape annealing with finite-element analysis) would push the designs from one feasible cluster through the infeasible region into the other feasible cluster. Without the external evaluator and search strategy the grammar would have had limited use, but with the evaluator and directed search method the combination of the grammar and optimizer became a powerful design tool.

HOW TO CREATE AN ENGINEERING SHAPE GRAMMAR

With an understanding of each of the issues laid out above, we can now focus on the design of the grammar itself. Only directions and insights are given here, as each grammar will be dependent on its context of knowledge and use based on the results of the above analyses. Both the creation of the grammar and the evaluation of the generated designs are briefly addressed.

Creating the Grammar. The creation of each of our grammars began with observation. The truss grammar used basic definitions of truss kinematics. The coffee maker grammar began by reverse engineering several coffee makers and viewing their form and function. The MEMS grammar was developed by reading through the MEMS resonator literature, understanding what types of designs had been generated to date, and what the purpose of a resonator is; this last point led to extended capabilities of the grammar to generate designs beyond those under current consideration.

Once an understanding of functionality, functional decomposition, and basic form requirements are understood, the next focus is on defining form and function subsystems. The preference is for a consistency between the decompositions of form and function, but this may not be possible depending on the application. This function and form decomposition of the coffee makers into the central coffee pot, the filter unit, the base unit, and the water storage unit made the breakdown straightforward. The grammar begins with the coffee pot and builds the other units off of it; integration occurs by parameter instantiation and the merging the different units together. The artificial heart grammar required an understanding of overall integration as well as the intended function of each individual subsystem; the bearings, impellers, motors, and stators are each discrete subsystems, but their performance is tightly coupled as is highlighted through its evaluation. The MEMS grammar builds from the central mass and requires minimal functionality prior to more detailed design options.

For the knowledge-intensive grammars, labels typically control the generation process. Modifying, adding, and removing labels directs the design engine to access different parts of the grammar based on what parts of the device have been designed. For example, the form details of the water storage unit of the coffee maker can only be designed once the parameter values of the filter unit have been defined; this is necessary in order to guarantee that the water storage unit will properly blend with the core unit. Labels also direct the generation of each subsystem within the heart grammar. Although the grammar is designed so that the design engine can jump back and forth between the different subsystems, because each part does not have to be completed at one time, labels become the critical mechanism to guarantee that only those parts for which there is enough information can be designed (i.e., only those rules with sufficient left-hand-side information can be applied). As general parametric shape interpreters evolve, the dependence on labels may give way to direct properties of the shape geometry.

In designing a grammar a certain number of known designs should be removed from the observation process to act as a control group. The grammar can be judged to be effective if it recreates not only the observed designs but also those in the control group; of course, it has to generate new designs as well. In the coffee maker grammar we took apart three coffee makers, but we re-created six (four of which were not included in the three that were reverse engineered).

Design Evaluation. In designing an engineering grammar, various levels of evaluation may be necessary to provide feedback to the design engine that uses the grammar. As discussed, this evaluation could occur after completion of the design as a check on performance, or during the generation process itself. For evaluation to be included during the generation process, criteria that can be evaluated must be associated with individual grammar rules. When the evaluation is dependent only on the current geometry and context, the appropriate analyses can be performed at any point in the design generation process. Each perturbation to the design caused by any rule must be directly linked to a set of analyses or simulations.

If the intermediate stages are less directly associated with the current geometry state, then the issue of how to evaluate the state of a design based on the application of a rule becomes a much more challenging problem. Often an intermediate evaluation is impossible to know accurately because it is dependent on downstream design details. For example, a configuration problem must have the complete configuration to evaluate overall performance or cost. This was the case with an attempt to use shape annealing to perform component placement (Szykman and Cagan, 1993). Although it was able to perform simple component packing in three dimensions, as more realistic product layout problems were considered, the ability to evaluate the quality of a layout required all of the components to be present. In this case an approximation has to be used. The creation of the approximation may in itself be a difficult task, requiring insight into the domain of application and engineering judgment as to what details can be sacrificed to deliver an appropriate approximation. In the case of the component placement problem, the solution was to move away from shape grammars and focus more on optimizing perturbation methods by using move sets (e.g., Szykman and Cagan, 1995). For the lathe process planning with shape annealing, a reasonable approximation was created based on a detailed description of the operation of the lathe, including material to be removed, the different tools used, the speed of rotation of a workpiece, and the rate at which a tool cuts material.

CONCLUDING REMARKS: WHERE WE ARE GOING

This chapter has discussed the state of the art in engineering shape grammars, recognizing that there has been limited work on applying the technology to engineering domains. By examining the evolution of work in this field, we hope the reader recognizes that the time has come; significant applications can be created for many classes of artifacts and are beginning to be used in practice. The last part of this exploration is designed to help the reader consider the motivation for developing grammars and the remaining research issues that will further advance the field.

In general why should researchers, teachers, and design practitioners care about shape grammars? There are four basic reasons.

The first is representation and control. A shape grammar is a logical way to concisely model, generate, and search large design spaces. For many applications it is an ideal way to do so. It also supports analysis that is critical to the design of engineering systems, completing the synthesize–evaluate loop.

The second is design to order. As industry moves toward one-of-a-kind design and manufacture within a class of products, the means to rapidly model and generate that class of products for any given set of parameters will be needed. Shape grammars have demonstrated that ability, as with the coffee maker grammar. Other applications require exploration into many alternatives within a class to find the best configuration for a given design situation. This is the intended use of the artificial heart shape grammar by the designers at the McGowan Center at the University of Pittsburgh. It is the repeatable nature of the design of classes of artifacts that makes shape grammars an appropriate representation for these types of design activity.

Another aspect of consumer products that makes shape grammars interesting and potentially important is in modeling corporate identity or brand. Once grammars for classes of products are modeled, certain features that can be identified within a set of products can be mapped into a set of rules from the grammar; whenever that set of rules is invoked, those features are included in the generated design. Those features can include basic forms. This concept is useful in modeling and identifying corporate identity. Companies often attempt to have a consistent style across their product lines. By identifying grammar rules that lead to features that are associated with the corporate style and requiring those rules and associated parameter values to be applied to all designs generated, the grammar can be used to generate products all of which map to the corporate identity. This idea was discussed in the coffee maker grammar paper (Agarwal and Cagan, 1998), in which Black & Decker, Braun, Krups, and Proctor Silex coffee makers were all generated and differentiated based on the rules included in their design.

The third reason is logic of engineering. Fundamentally there is a logical structure to the synthesis of engineering design. Much like the logic of architecture that has been articulated through shape grammars, shape grammars have the ability to capture and illuminate the logic of designing classes of engineering artifacts. Some of the theoretical and practical issues that shape grammar designers face have been discussed in this paper. Shape grammars support the representation, generation, analysis, and search of the design space. Do shape grammars model human design thinking? It may be that shape grammars provide a theory of engineering design. This question is open, and only future exploration will begin to answer it.

The fourth reason is education. Modeling a class of designs and the accompanying logic of engineering may better position educators to teach the design process to students and practitioners. When the methodological framework of a grammatical representation is used, a design process can be articulated in a very clear, usable way. For example, shape grammars are compatible with methods such as reverse engineering and other observation techniques; it is through observation and patterns that can be identified through reverse engineering that shape grammars for product classes are themselves created. Thus the result of a reverse engineering process can very well be a formal shape grammar that models the space of products of interest. Such exercises would enable students and practitioners to dissect a design process, model it, and then use the results to explore alternatives.

Finally, this chapter concludes with a summary of what the difficult issues are that still must be solved to make the application of shape grammars transparent to the user and creator.

1. There must be a better understanding and modeling of the coupling between function and form.
2. There must be a better ability to demonstrate emergence, be it form or functional; like formal theories for form emergence, is there a similar mathematical formulation for the theory of functional emergence?
3. Grammars must be able to be adaptable to changes (such as through the introduction of new technologies) that will allow for the dynamic expansion of the grammar space.
4. There must be the ability to rapidly implement grammars and have the flexibility to add new rules, most likely through the development of robust grammar interpreters.
5. The ability must exist to couple the implementation of the grammar with simulators used to evaluate their incremental performance.
6. Means must be constantly improved to perform optimizing directed search on a design language.
7. There must be the ability to explore many applications; they are out there and require engineers with an understanding of shape grammars to seek them out and model them!

ACKNOWLEDGMENTS

The author is grateful to Manish Agarwal, Matthew Campbell, Jay McCormack, and George Stiny for their discussions on these ideas and for comments on this manuscript. This manuscript was originally prepared for the National Science Foundation Workshop on Shape Grammars, held at the Massachusetts Institute of Technology in 1999.

REFERENCES

Aelion, V., Cagan, J., and Powers, G. (1991). "Inducing optimally directed innovative designs from chemical engineering first principles," *Computers and Chemical Engineering*, **15**(9):619–627.

Aelion, V., Cagan, J., and Powers, G. (1992). "Input variable expansion – an algorithmic design generation technique," *Research in Engineering Design*, **4**:101–113.

Agarwal, M. and Cagan, J. (1997). "Shape grammars and their languages – a methodology for product design and product representation." In *Proceedings of the 1997 ASME Design Engineering Technical Conferences and Computers in Engineering Conference: Design Theory and Methodology Conference*, ASME, New York, DETC97/DTM-3867.

Agarwal, M. and 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. and Cagan, J. (1999). "Systematic form and function design of MEMS resonators using shape grammars." In *ICED'99*, Technische Universität München.

Agarwal, M. and Cagan, J. (2000). "On the use of shape grammars as expert systems for geometry based engineering design,"*Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, **14**:431–439.

Agarwal, M., Cagan, J., and Constantine, K. (1999). "Influencing generative design through continuous evaluation: associating costs with the coffee maker shape grammar," *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, Special Issue, **13**:253–275.

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

Baumgart, B. G. (1975). "A polyhedron representation for computer vision," *AFIPS Conference Proceedings*, **44**:589–596.

Brown, K. N. and Cagan, J. (1996). "Grammatical design and bounded creativity," EDRC Report 24-124-96, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

Brown, K. N. and Cagan, J. (1997). "Optimized process planning by generative simulated annealing," *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, **11**: 219–235.

Brown, K. N., McMahon, C. A., and Sims-Williams, J. H. (1994). "A formal language for the design of manufacturable objects." In *Formal Design Methods for CAD (B-18)*, J. S. Gero and E. Tyugu (eds.), North-Holland, Amsterdam, pp. 135–155.

Cagan, J. and Agogino, A. M. (1987). "Innovative design of mechanical structures from first principles," *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, **1**(3):169–189.

Cagan, J. and Agogino, A. M. (1991a). "Inducing constraint activity in innovative design," *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, **5**(1):47–61.

Cagan, J. and Agogino, A. M. (1991b). "Dimensional variable expansion – a formal approach to innovative design," *Research in Engineering Design*, **3**:75–85.

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

Cagan, J. and Mitchell, W. J. (1994). "A grammatical approach to network flow synthesis." In *Formal Design Methods for CAD (B-18)*, J. S. Gero and E. Tyugu (eds.), North-Holland, Amsterdam, pp. 173–189.

Fenves, S. and Baker, N. (1987). "Spatial and functional representation language for structural design." In *Expert Systems in Computer-Aided Design*, Elsevier, New York.

Finger, S. and Rinderele, J. R. (1989). "A transformational approach to mechanical design using a bond graph grammar." In *Proceedings of the First ASME Design Theory and Methodology Conference*, ASME, New York.

Fitzhorn, P. (1990). "Formal graph languages of shape," *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, **4**(3):151–163.

Flemming, U. (1987). "More than the sum of the parts: the grammar of Queen Anne houses," *Environment and Planning B*, **14**:323–350.

Heisserman, J. (1991). "Generative geometric design and boundary solid grammars," Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Heisserman, J. (1994). "Generative geometric design," *IEEE Computer Graphics and Applications*, **14**(2):37–45.

Heisserman, J. and Callahan, S. (1996). "Interactive grammatical design." Presented at AI in Design '96, Workshop Notes on Grammatical Design, Stanford, CA, June 24–27.

Heisserman, J. and Woodbury, R. (1994). "Geometric design with boundary solid grammars." In *Formal Design Methods for CAD (B-18)*, J. S. Gero and E. Tyugu (eds.), North-Holland, Amsterdam, pp. 85–105.

Knight, T. W. (1986). "Transformations of the meander motif on greek geometric pottery," *Design Computing*, **1**:29–67.

Kirkpatrick, S., Gelatt, Jr., C. D., and Vecchi, M. P. (1983). "Optimization by simulated annealing," *Science*, **220**(4598):671–679.

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

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

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

Longenecker, S. N. and Fitzhorn, P. A. (1991). "A shape grammar for non-manifold modeling," *Research in Engineering Design*, **2**:159–170.

McCormack, J. and Cagan, J. (2000). "Enabling the use of shape grammars: shape grammar interpretation through general shape recognition." In *Proceedings of the 2000 ASME Design Engineering Technical Conferences: Design Theory and Methodology Conference*, ASME, New York, DETC2000/DTM-14555.

McCormack, J., Cagan, J., and Antaki, J. (1999). "A shape grammar for the streamliner artificial heart," (working paper).

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). "Equation of state calculations by fast computing machines," *Journal of Chemical Physics*, **21**:1087–1092.

Mitchell, W. J. (1991). "Functional grammars: an introduction." In *Computer Aided Design in Architecture '91: Reality and Virtual Reality*, G. Goldman and M. Zdepski (eds.), New York: New Jersey Institute of Technology, pp. 167–176.

Reddy, G. and Cagan, J. (1995a). "Optimally directed truss topology generation using shape annealing," *ASME Journal of Mechanical Design*, **117**(1):206–209.

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

Rinderle, J. (1991). "Grammatical approaches to engineering design, Part II: melding configuration and parametric design using attribute grammars," *Research in Engineering Design*, **2**:137–146.

Rollo, J. (1995), "Triangle and T-square: the windows of Frank Lloyd Wright," *Environment and Planning B: Planning and Design*, **22**:75–92.

Schmidt, L. C. (1995)."An implementation using grammars of an abstraction-based model of mechanical design for design optimization and design space characterization," Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Schmidt, L. and Cagan, J. (1992). "A recursive shape annealing approach to machine design." Preprints of *The Second International Round-Table Conference on Computational Models of Creative Design*, Heron Island, Queensland, December 7–11, pp. 145–171.

Schmidt, L. C. and Cagan, J. (1995). "Recursive annealing: a computational model for machine design," *Research in Engineering Design*, **7**:102–125.

Schmidt, L. C. and Cagan, J. (1997). "GGREADA: a graph grammar-based machine design algorithm," *Research in Engineering Design*, **9**(4):195–213.

Schmidt, L. C. and Cagan, J. (1998). "Optimal configuration design: an integrated approach using grammars," *ASME Journal of Mechanical Design*, **120**(1):2–9.

Schmidt, L. C. Shetty, H., and Chase, S. C. (1998). "A graph grammar approach for structure synthesis of mechanisms." In *Proceedings of the 1998 ASME Design Engineering Technical Conference*, ASME, New York, DETC98/DTM-5668.

Shea, K. (1997). "Essays of discrete structures: purposeful design of grammatical structures by directed stochastic search," Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Shea, K. and Cagan, J. (1997). "Innovative dome design: applying geodesic patterns with shape annealing," *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, **11**:379–394.

Shea, K. and Cagan, J. (1999a). "The design of novel roof trusses with shape annealing: assessing the ability of a computational method in aiding structural designers with varying design intent," *Design Studies*, **20**:3–23.

Shea, K. and Cagan, J. (1999b). "Languages and semantics of grammatical discrete structures," *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, Special Issue, **13**:241–251.

Shea, K., Cagan, J., and Fenves, S. J. (1997). "A shape annealing approach to optimal truss design with dynamic grouping of members," *ASME Journal of Mechanical Design*, **119**(3): 388–394.

Spillers, W. (1985). "Shape Optimization of Structures." In *Design Optimization*, Academic Press Inc., Orlando, pp. 41–70.

Stiny, G. (1977). "Ice-ray: a note on the generation of Chinese lattice designs," *Environment and Planning B: Planning and Design*, **4**:89–98.

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

Stiny, G. (1981). "A note on the description of designs," *Environment and Planning B: Planning and Design*, **8**:257–267.

Stiny, G. (1982). "Spatial relations and grammars," *Environment and Planning B: Planning and Design*, **9**:113–114.

Stiny, G. (1991). "The algebras of design," *Research in Engineering Design*, **2**:171–181

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

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

Stiny, G. and Mitchell, W. J. (1978). "The Palladian grammar," *Environment and Planning B: Planning and Design*, **5**:5–18.

Stiny, G. and Mitchell, W. J. (1980). "The grammar of paradise: on the generation of mughul gardens," *Environment and Planning B: Planning and Design*, **7**:209–226.

Szykman, S. and Cagan, J. (1993). "Automated generation of optimally directed three dimensional component layouts." In *Advances in Design Automation*, ASME, New York, **65**(1):527–537.

Szykman, S. and Cagan, J. (1995). "A simulated annealing-based approach to three-dimensional component packing," *ASME Journal of Mechanical Design*, **117**(2A):308–314.

Wells, A. B. (1994). "Grammars for engineering design," Ph.D. Dissertation, California Institute of Technology, Pasadena, CA.

Formal Engineering Design Synthesis

Edited by

ERIK K. ANTONSSON
California Institute of Technology

JONATHAN CAGAN
Carnegie Mellon University

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011-4211, USA
10 Stamford Road, Oakleigh, VIC 3166, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First published 2001

Printed in the United States of America

Typefaces Times Ten 10/13 pt. and Helvetica Neue *System* LaTeX 2_ε [TB]

A catalog record for this book is available from the British Library.

Library of Congress Cataloging in Publication Data
Formal engineering design synthesis / edited by Erik K. Antonsson,
Jonathan Cagan.
p. cm.
Includes bibliographical references and index.
ISBN 0-521-79247-9
1. Engineering design. I. Antonsson, Erik K. II. Cagan, Jonathan, 1961–
TA174 .F67 2001
620′.0042 – dc21 2001025164

ISBN 0 521 79247 9 hardback