Emergence and Continuity in Shape Grammars

George Stiny

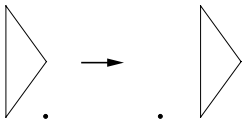
Graduate School of Architecture and Urban Planning
University of California
Los Angeles, CA 90024 USA

Standard topological devices can be adapted for shapes, and then used to establish the continuity of computations in a shape grammar. This explains how shapes are structured in relation to one another as rules are applied in computations, even when emergence plays a crucial role.

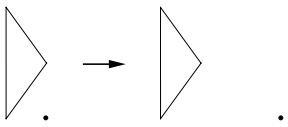
1 Emergence

By themselves, shapes are unanalyzed and without apparent structure; they are divided into parts as rules are applied in computations in a shape grammar. These divisions are independent from one rule application to another, allowing for emergent properties of shapes to be recognized and exploited freely. This is especially clear in a marvelous series of examples.

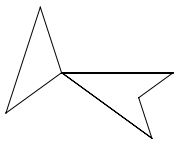
Consider the rule



that translates an isosceles triangle along its central axis, and the inverse of the rule

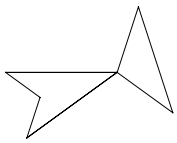


that translates the triangle in the opposite direction to its original position. The two rules apply in turn to connect the shape



U. Flemming and S. Van Wyk (eds.), ***CAAD Futures '93,*** *pp. 37-54.*

and the shape



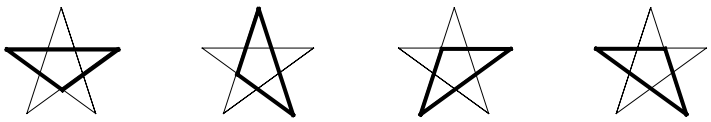
in a computation.

At first sight, this appears to be impossible. The rules divide the two shapes into the same components: a triangle and a quadrilateral apiece. But the final shape in the computation is a rotation or a reflection of the initial one. How can rules that merely move a triangle back and forth along a common axis produce this change in orientation, with the paradoxical implication that translation includes rotation and reflection? The axes of the triangles in the two shapes are not the same, and alone the translations determined by the rules are not enough to explain the difference.

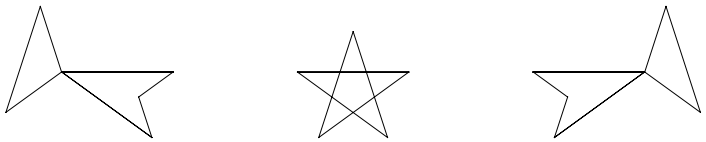
The answer to the question is easy to see after the first rule is applied to translate the triangle in the initial shape. This five pointed star

is produced. The star can be described in five different ways with respect to the second rule and the transformations that make the triangle in the left side of the rule a part of the star. These alternative descriptions compete with one another, as they are incompatible for the separate applications of the rule. One of the descriptions is expected. In it, the translation of the triangle in the initial shape is distinguished.

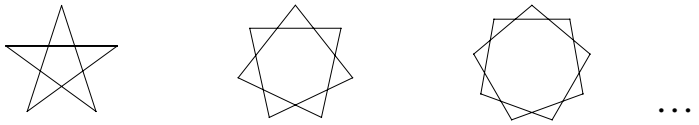
This triangle, however, is not recognized in the other descriptions



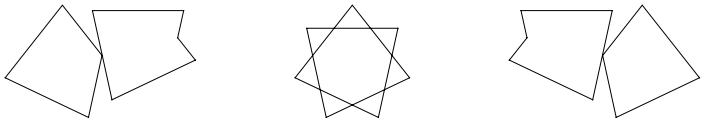
in which four emergent triangles are distinguished one at a time. If the second rule is applied to the star to translate the triangle recognized in the second of these new descriptions, then the final shape is produced. The complete computation is shown here.



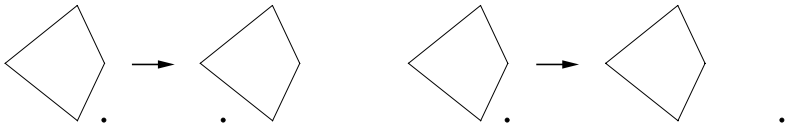
This computation is not unique, but begins a series of like computations in which, for $n = 3, 4, 5, \cdots$, stars with $2n - 1$ points



can be described in $2n - 1$ different ways that are mutually incompatible with respect to a rule and its possible applications. In each of these computations, a shape made up of a convex polygon with n sides and a concave polygon with $n + 1$ sides is transformed in one of $2n - 1$ possible ways. Two rules are defined for this purpose to translate the convex polygon back and forth along a common axis. The first rule applies to the initial shape to produce a star. The second rule applies to the star to produce the initial shape again, or a rotation or a reflection of it. For example, a seven pointed star is the center of this computation

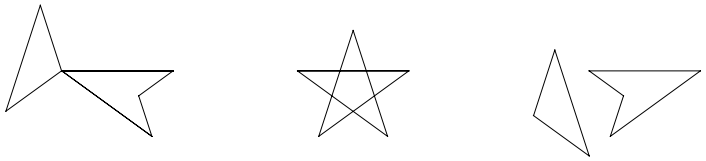


in which the two rules

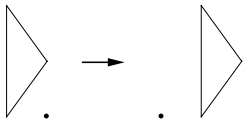


are applied once apiece. Emergence is not an isolated phenomenon, but one that occurs naturally in myriad ways.

It is worth a short digression to notice that the rules used to produce stars in the computations in the series can by themselves produce some surprising results. For example, in this computation



for $n = 3$, the rule



that translates a triangle in one direction only along its central axis is applied twice. The triangle and the quadrilateral in the initial shape in the computation are separated from one another and reversed in position, paradoxically keeping the triangle on the left and rotating or reflecting the two polygons.

Ironically, the fluid relationship between shapes and rules in the computations in the series and in other computations like them is recommended enthusiastically in the canonical formulas of recent, anti-formalist literary theory that explore the links between text and reader. Stars are constructed in one way and deconstructed in other ways to produce a variety of contradictory effects. More appositely, though, these effects result not because shapes announce their divisions conclusively, but because they receive them provisionally as rules are applied.

Stanley Fish, "Chairman (sic) of the Department of English . . . Duke University," makes precisely this point over and over and over again for text and reader:

> . . . an interpreting entity, endowed with purposes and concerns, is, by virtue of its very operation, determining what counts as the facts to be observed.
>
> . . . the properties of the text (whether they be literary or "ordinary" properties) are the product of certain ways of paying attention.
>
> . . . the very features of the text emerge into being in a reciprocal relationship with the reader's activities . . .

Applying rules is merely the special way "purposes and concerns," "certain ways of paying attention," and "the reader's activities" are manifested in a shape grammar to recognize "facts," "properties," and "features." The division of shapes into parts and the reading of a text are really not very dissimilar. Both are interpretative pursuits, but the one is more obviously algorithmic than the other. And in fact, this is exactly the kind of methodological advance that Fish values highly, ". . . the research accomplishments, methodological techniques, powerful interpretations, pedagogical innovations, etc., that bring some men and women to the 'summit' . . . " of their profession, whether it be literary criticism or design.

But rules do more; they change shapes to produce new ones, and thereby connect shapes in computations in a variety of interesting relationships that may be unexpected or even appear paradoxical. Rules are doubly instrumental, determining in the course of their use what parts of shapes are recognized, and what actions to take to alter shapes in different ways as different parts emerge. This correspondence between seeing in certain ways and doing in certain ways is what rules are really all about. It is how ideas are expressed in a shape grammar to allow for observation and action to work together in computations. And it is how these computations become just another kind of inquiry.

Whenever a rule applies to a shape in a computation, the rule implicitly provides a description of the shape that guides the action of the rule. This description, however, does not impose restrictions on other applications of rules either to the shape itself or to

successive shapes in the computation. There is nothing whatsoever to keep rules from being defined or applied to respond to change and circumstance expediently and opportunistically, because the necessity to know how shapes have been described or have come to be never intrudes. The history of shapes may play a role by choice, or may be ignored completely. Anyone is welcome to enter the computation at any time with different ideas given in rules; and others, working independently or in concert with different rules, expressing alternative points of view, may also contribute to the computation.

One does not usually find computations of this kind in information processing models that rely on structure and semantics to produce their results. No one denies the importance of structure and semantics, especially when they originate contingently, influenced by what happens inside computations with shapes, as rules are applied to divide shapes into parts and these parts are classified, possibly in different ways at different times. But if structure and semantics are prefigured outside of computations before they begin, because permanent units of information that come in standard packages are already in place, then there is ample cause for concern. Now, descriptions represent things in computations, and nothing is ever more than its description anticipates explicitly. Difficulties with this approach arise almost at once.

For example, ambiguity is a persistent problem. On the one hand, there may be no ambiguity, so that each thing has a single, correct description. This is neat and tidy, but precludes, at least in their present form, all of the computations in the series in which transformations of shapes are produced that are not the identity. And on the other hand, there may be ambiguity, except with no means to decide whether or not different descriptions represent the same thing, or things that are some parts of others. Either way, the prospects for useful inquiry are dismal. Without ambiguity, there is no appeal to different descriptions, and to the novel opportunities for action these descriptions may entail. And with ambiguity, there is only the possibility of chaos and confusion.

Information processing models are really not as bad as they may look. In fact, shapes are normally represented by finite sets of basic elements–points, lines, planes, or solids–that are maximal with respect to one another. This allows for shapes to be identified uniquely. But there are subtleties, too, to avoid the kinds of difficulties mentioned above. These involve embedding relations on basic elements and reduction rules that ensure that shapes are without definite parts, and that determine all of the other properties of the algebras to which shapes belong.

Computations with shapes are defined in the algebras in Table 1. The shapes in an algebra U_{ij} are made up of basic elements of dimension i that are arranged in dimension j. Shapes are partially ordered by a part relation ($\leq$), and combined by Boolean operations of sum (+), product ($\cdot$), and difference (-). This forms a relatively complemented, distributive lattice that is equivalent to a Boolean ring with the empty shape for zero and no unit. The shapes in the algebra U_{ij} can also be changed one into another via the Euclidean transformations of dimension j, that together form a group among themselves. The part relation, the operations of sum and difference, and the transformations fix the mechanism used in every shape grammar to perform computations with shapes.

Table 1

U_{00}	U_{01}	U_{02}	U_{03}

$$U_{11} \quad U_{12} \quad U_{13}$$

$$U_{22} \quad U_{23}$$

$$U_{33}$$

A few more rudimentary details are needed for later. Any pair of shapes A and B in an algebra U_{ij} defines a rule $A \rightarrow B$. The rule applies to a shape C also in U_{ij} in a two stage process involving a transformation t. The transformation is used in both stages, once with the relation $\leq$ to distinguish some part of C, and then again with the operations + and - to replace the part that has been picked out. More precisely, the rule $A \rightarrow B$ applies to the shape C if the formula

there is a t such that $t(A) \leq C$

can be satisfied. In this case, a new shape C' in U_{ij} may be produced according to the formula

$$C' = (C - t(A)) + t(B).$$

The rules in a shape grammar apply recursively in compliance with these two formulas beginning with a given initial shape C_0 in U_{ij} to determine the shapes $C_0, \cdots, C_n$ that comprise each of the computations in the grammar. The full resources of the algebra U_{ij} thus come into play as computations unfold.

The way rules work in the algebras U_{ij} is the same in more complex algebras, allowing for computations with shapes made up of basic elements of various kinds and other things. These new algebras are derived from the algebras U_{ij} in two main ways. First, labels or weights may be associated directly with basic elements to define the algebras of shapes V_{ij} and W_{ij}. Labels classify basic elements, keeping ones in different categories separate when shapes are compared and combined, and do not interact themselves. In contrast, weights give basic elements properties and interact as basic elements do. Weights may add formal features, or anything else, from physical properties like mass to intentional properties like function and use. And second, after the algebras U_{ij}, V_{ij}, and W_{ij} have been defined, they can be combined in different operations, including, for example, appropriate sums and products, to obtain other algebras. These algebras typically contain compound shapes with one or more components in which basic elements of various kinds, labels, and weights are mixed.

The algebras U_{ij}, V_{ij}, and W_{ij}, and their combinations contribute to an ever expanding repertoire of spatial and symbolic devices that promise as much scope for creative expression as architects and designers enjoy in their professional activities. Even so, there are undoubtedly many concerns about the chances for computations in these algebras to match the impressive results of intuition and imagination in actual practice. But these concerns really amount to very little. The opportunities to use shapes in practice are no wider than those offered freely in computations. In both cases, shapes come unanalyzed and undivided. And as the computations in the series show convincingly, this can be used without difficulty to wonderful advantage, when rules are defined in a shape grammar.

2 Continuity

After computations with shapes are completed, it is time for explanations. These come in a variety of ways. However, they are meant mostly to rationalize computations, so that rules appear to apply in a continuous fashion without discernible break or inconsistency in the way shapes are described. One way to fabricate these explanations–that itself involves computations–builds neatly on a few of the leading ideas of topology.

The outline of the approach is simple. Functions of a certain kind are used to describe the applications of rules in a computation, and topologies are used to describe the shapes, so that the functions are continuous for the topologies. The functions respect the divisions in the shapes recognized in the topologies, as the shapes are changed one into another in the course of the computation.

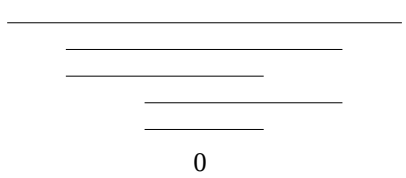
The approach is developed directly in terms of shapes. This simple expediency requires the use of heuristic devices in two places that diverge from the conventional topological account of continuity that relies entirely on sets. The results obtained in a few easy steps show clearly how shapes are structured in relation to one another as rules are applied in computations. If rigor is not to be sacrificed, however, then the approach can be developed with a little more complication albeit with a little less intuitive appeal, so that it adhers strictly to convention. The additional details are sketched in the penultimate section of this paper.

A topology for a shape C is determined by any set of shapes that are all parts of C whenever these three conditions are met:

(1) the empty shape and C are both in the set;
(2) if the shapes x and y are in the set, then so are the shapes formed in the sum $x + y$, and in the product $x \cdot y$; and
(3) for every part x of C, there is a smallest shape in the set that includes x as a part, that is, there is a shape y in the set such that $x \leq y$, and for any shape z in the set, if $x \leq z$, then $y \leq z$.

There may be finitely or infinitely many shapes in a topology for C if it has lines or planes or solids for its basic elements. There are only finitely many shapes in any topology for C, however, if it just has points.

For example, this set of six shapes



0

is a topology for a shape made up of a single line. The shapes in the topology are nicely conceived of as the pieces of the shape that can themselves stand alone as independent shapes; they are the individual components and the partial assemblies that are meant to be identified and manipulated separately.

The relationship between topologies for shapes and topologies for sets deserves a closer look. The first two conditions in the previous definition match like conditions for sets. The third condition, however, is framed especially for shapes to modify the

stronger requirement for sets that the infinite subsets of a topology have intersections that belong to the topology as well. This change is necessary because only finite sets of shapes are guaranteed to have products that are shapes. It is easy to see that a topology for a shape can have infinite subsets that do not have products. In particular, suppose the shape has lines or planes or solids for its basic elements. Then, the set of all parts of the shape is a topology—the discrete topology—for the shape. And furthermore, where for every part x of the shape, there is a smallest shape in the topology that has x as a part, namely, x itself, there are infinite subsets of the topology without products.

Every topology for a shape C determines a function $\Gamma: C \rightarrow C$ that associates every part x of C with the smallest shape in the topology that includes x as a part. This function forms a closure relation. For every part x of C, $\Gamma(x)$ is the closure of x, and if $\Gamma(x) = x$, then x is closed.

A closure relation for a shape C can be defined in its own right, thereby allowing for topologies and closure relations to be determined reciprocally. A function $\Gamma: C \rightarrow C$ that associates every part of C with another part of C forms a closure relation whenever five axioms are satisfied:

(1) $\Gamma(0) = 0$;
(2) $x \leq \Gamma(x)$;
(3) $\Gamma(\Gamma(x)) \leq \Gamma(x)$;
(4) if $x \leq y$, then $\Gamma(x) \leq \Gamma(y)$; and
(5) $\Gamma(x + y) \leq \Gamma(x) + \Gamma(y)$;

where the empty shape is given by 0, and x and y are any parts of C. The set of closed parts of C obtained from Γ is then a topology for C.

Intuitively, a function associating the parts of two shapes is continuous if the divisions it respects in the first shape are compatible with the divisions recognized in the second shape. This idea has a precise formulation in terms of the topologies that fix the divisions in the shapes. Let $g: C \rightarrow C'$ be a function from the parts of a shape C to the parts of a shape C'. Then, the function is continuous for the topologies for C and C' with the closure relations Γ and Γ' if this formula is satisfied

$$g(\Gamma(x)) \leq \Gamma'(g(x))$$

for every part x of C.

Here, another modification is introduced that moves away from the topological account of continuity for sets. Where sets have members and subsets, shapes only have parts. So, the usual practice of defining a function $g: C \rightarrow C'$ from the members of one set C to the members of another set C', and then extending g to subsets of C and C' cannot be followed for shapes. Rather, a function $g: C \rightarrow C'$ is defined to associate the parts of one shape C directly with the parts of another shape C'. This has implications for what can be assumed about functions for shapes. For example, if the function g is extended to subsets of the sets C and C', with Ø the empty set, and x and y sets, then these five formulas hold: (1) $g(\text{Ø}) = \text{Ø}$; (2) if $x \neq \text{Ø}$, then $g(x) \neq \text{Ø}$; (3) if $x \leq y$, then $g(x) \leq g(y)$; (4) $g(x + y) = g(x) + g(y)$; and (5) $g(x \cdot y) \leq g(x) \cdot g(y)$. In contrast, however, if the function g is defined for the parts of the shapes C and C', then the corresponding formulas in which the empty shape 0 replaces the empty set Ø, and x and y are shapes cannot be used without qualification.

The definition of continuity can now be extended to computations in a shape grammar. For this purpose, every application of a rule $A \rightarrow B$ is described by a function. Suppose the rule is applied to a shape C under a transformation t to produce the shape C'. Then, the function g: $C \rightarrow C'$ from the parts of C to the parts of C' is defined in the formula

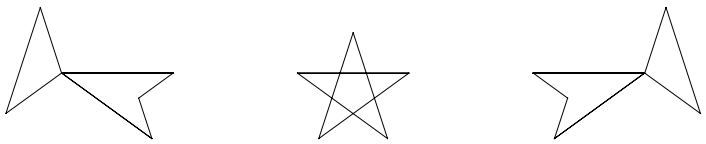
$$g(x) = x - t(A).$$

The function shows how the rule changes every part x of C before adding $t(B)$. And in so doing, it identifies those parts that are guaranteed to stay the same.

The function g is not the only function that can be defined to describe the application of the rule. It is, however, one way of paying attention to the distinctions the rule makes that gives informative results. Where this is as good a reason as any to choose the function instead of another, it need not discourage further investigation. The classification and comparison of different functions to describe applications of rules will likely provide additional insights.

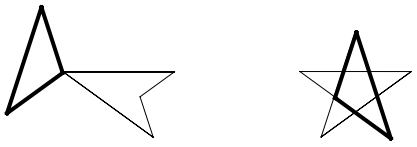
If the rule $A \rightarrow B$ applies to a shape C to produce the shape C', then the application of the rule is continuous just in case the function g is continuous for the topologies for the shapes C and C'. And by extension, a computation $C_0, \cdots, C_n$ is continuous for the topologies for the shapes $C_0, \cdots, C_n$ whenever every application of a rule is continuous.

The first thing to notice about a computation with shapes, especially when emergence plays a role, is that the computation is not automatically continuous in terms of the descriptions that are formed naturally for the shapes as rules are applied. For example, in the computation

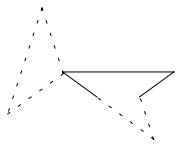


the initial shape and the star both have descriptions in which a triangle and a quadrilateral are distinguished when the two rules

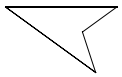
are applied, the one and then the other. The divisions made in these descriptions are indicated in these figures



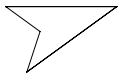
and they determine two topologies. In both topologies, a triangle and a quadrilateral are the only closed, nonempty, proper parts either of the initial shape or of the star. However, the application of the first rule that changes the initial shape into the star is not thereby continuous. In particular, for this part



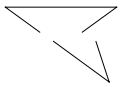
of the initial shape, the quadrilateral



is formed in the left side of the inequality that defines continuity, and the quadrilateral



is defined in the right side of the inequality. The quadrilateral in the initial shape must be torn apart in this way



if the topology for the star is to contain the triangle and the quadrilateral distinguished by the application of the rule in the computation. This division of the quadrilateral in the initial shape cuts the quadrilateral into two separate pieces. The piece comprised of two line segments combines with two segments in the sides of the triangle produced by the rule applied to the initial shape to form the emergent triangle in the star that is contained in its topology

Are computations with shapes ever continuous? And what topologies for shapes make them so?

There are two answers to these questions that fix the extremes for topologies for shapes. On the one hand, an application of a rule $A \rightarrow B$ to a shape C is continuous whenever C has the discrete topology with the closure relation $\Gamma(x) = x$, for every part x of C. Every part of C is recognized in this topology, and every division is thereby

allowed. But the topology really cuts C too finely -- infinitely so, at least if C has lines or planes or solids for its basic elements -- making no serious division because it makes every division. On the other hand, an application of the rule is continuous if the shape C' produced from C has the trivial topology with the closure relation $\Gamma'(x) = C'$, for every nonempty part x of C'. No part of C' is recognized in this topology, except the empty shape and C' itself. The topology does not cut C' at all, keeping it monolithic and without useful structure. There are, however, other topologies between these extremes that always make computations continuous, and at the same time, provide useful descriptions of shapes.

Once again, consider a rule $A \rightarrow B$. Suppose that the rule applies to a shape C under a transformation t to produce the shape C'. Furthermore, let the topology for C contain at least the two parts of C that are resolved as the rule is applied, namely, the shapes $t(A)$ and $C - t(A)$. Then, the application of the rule is continuous if and only if the closure relations Γ and Γ' obtained from the topologies for C and C' are related in the inequality

$$\Gamma(x) \leq \Gamma'(x)$$

for every part x of $C - t(A)$. Now, if any two parts of C have the same closure, then the corresponding two parts of C' have the same closure, too. There is no division in C' that is not anticipated in C.

There is an easy way to take advantage of this relationship to make the application of the rule continuous. Simply let

$$\Gamma(x) = (C - t(A)) \cdot \Gamma'(x)$$

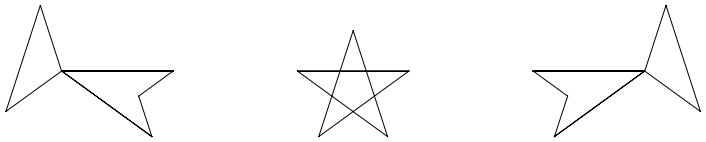
for every part x of $C - t(A)$. And let

$$\Gamma(x) = t(A) + (C - t(A)) \cdot \Gamma'(x - t(A))$$

otherwise.

These formulas are defined for individual applications of rules, but they can be used recursively, beginning with the final shape in any given computation and working backwards to the initial shape, to obtain topologies for the shapes in the computation that make the computation continuous. The use of the formulas in this way is clear in two simple examples.

Again, consider the computation



in which the two rules

are applied once apiece. If the final shape is given the trivial topology, then the three shapes in the computation have the topologies shown in Figure 1. As noted earlier, the divisions in the initial shape that cut the quadrilateral into pieces allow for an emergent triangle to be formed (recognized) in the star, so that the computation can go forward continuously. This same process whereby divisions are made to anticipate emergent shapes unfolds more elaborately in longer computations.

The rule

rotates an equilateral triangle about its center, so that this point is permanently fixed.

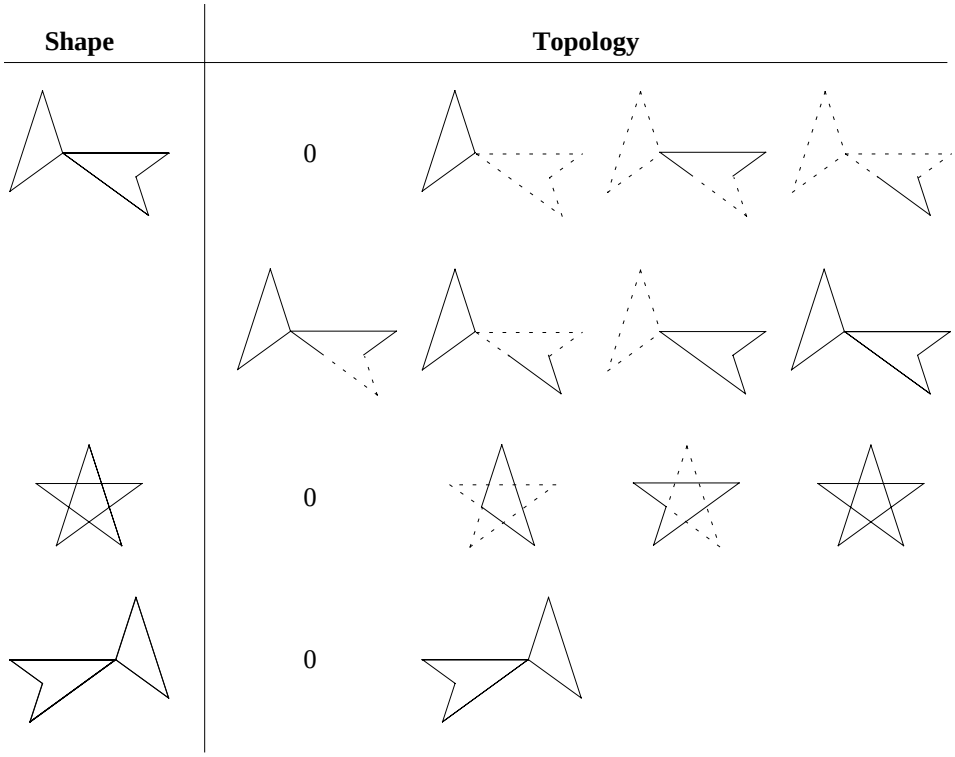


Figure 1.

The rule is used repeatedly in this computation

in which the final shape is a rotation of the initial shape about its center. The transformation is surprising because the centers of the triangles in the initial shape change position when the initial shape is moved into the final one. How can a rule that leaves the centers of triangles alone make these changes?

Emergence plays a crucial role in the computation in the separate applications of the rule to the fourth and sixth shapes. The fourth shape in the computation can be described in two ways with respect to the rule and different transformations. In one way, the three triangles in the fourth shape that correspond to the three triangles in the initial shape are distinguished separately. None of these triangles, however, is recognized in the other way. Instead, the fourth shape is divided into two triangles—the large, outside one and the small, inside one—that have sides formed from sides of the three triangles obtained from those in the initial shape. The rule is applied to rotate each of these emergent triangles in turn to produce the fifth and sixth shapes in the computation.

Now, there are alternative ways to describe the sixth shape with respect to the rule and different transformations. In one way, the two triangles that correspond to the two triangles in the fourth shape are distinguished. But in the other way, three triangles—one at each corner of the sixth shape—that have sides formed from segments of sides of the two triangles recognized in the fourth shape serve to divide the sixth shape. The rule rotates each of these emergent triangles to produce three new shapes, completing the computation to connect the initial shape and the final one.

The recursive application of the two formulas given above defines topologies for the shapes in the computation that take account of these twin shifts in perspective to make the computation continuous. If the final shape is given the trivial topology, then the nine shapes in the computation have the topologies shown in Figure 2.

It is instructive to keep a record of the consequent divisions in the individual triangles that are produced as the rule is applied repeatedly in the eight steps of the computation

Each triangle is cut, so that separate parts of it combine with different shapes in a corresponding topology, anticipating the subsequent emergence of triangles and providing for the sequenced assembly of the final shape piece by piece. The triangle in the right side of the rule so assumes a special structure with respect to a topology every time the rule is used. This shows in one way why shapes are so hard to handle in terms of fixed descriptions that do not allow for this kind of contingent variability: these descriptions usually leave out too much, each one imposing a single scheme of division

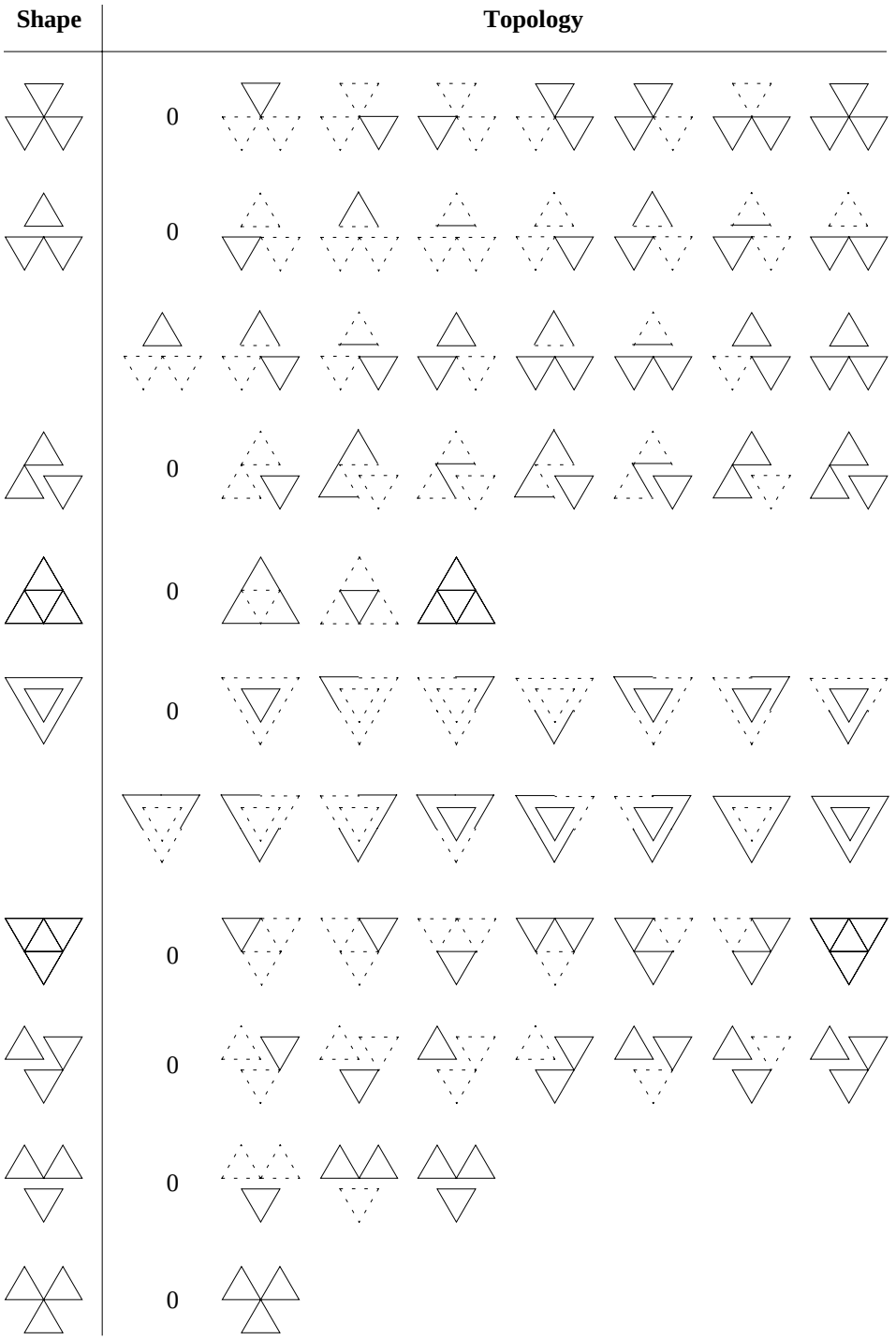


Figure 2.

that excludes all others. And it serves further to highlight the difference between computations in a shape grammar, and computations of a more conventional kind, say, in a set grammar, in which symbolic devices are marshalled to manipulate shapes indirectly in selected descriptions. There are two things about the way topologies for shapes are defined to make computations continuous that deserve a little more elaboration. Both engage central themes in this paper.

First, the trivial topology is used adjunctively in two places to form topologies for shapes: once explicitly when it is given to structure the final shape in a computation, and once implicitly when it is assumed to structure the shape $t(A)$ in the second formula given to define the topologies for the other shapes in the computation. In both cases, the use of the trivial topology is arbitrary. And in fact, other kinds of topologies work equally well. Nonetheless, the choice of these topologies is not without importance, because it affects the topologies for the shapes in the computation. For example, if the topologies are Boolean algebras—and the trivial topology is one—then the topologies for the shapes in the computation are Boolean algebras, too. More importantly, though, the way the computation is ultimately explained as a continuous process is neither unique nor inevitable. What happens inside the computation contributes essentially to its explanation, but always proves inconclusive. Other things outside of the computation that arise circumstantially influence understanding as well.

Second, the topologies for the shapes in a computation are defined retroactively, after the computation has been completed. The main reason for this is that there is no general, effective way of knowing how things will turn out in any given computation until rules are actually applied and their subsequent effects are observed. Sometimes this uncertainty about how shapes are finally divided makes people uncomfortable. There are those who want immediate answers, and who make sure that they get them by circumscribing topologies for shapes before a computation begins.

The definition of a set grammar shows the preferred way of doing this with symbolic devices. Some technical details aside, simply assume that every shape is a finite set of components—possibly even overlapping ones—from a given vocabulary, and that this set has the discrete topology in which all of its subsets are closed. This approach provides counterparts for computations in a shape grammar if shapes only have points for basic elements, and in many other cases, too, if it is apparent how shapes will be divided by looking at rules alone. In fact, the approach even works to recast the computations considered above in which emergence plays a crucial role. But this accomplishes very little, because the computations must be explained first.

The assumption that shapes are described completely by finite sets of given components restricts the way shapes can be structured in a computation, and limits the freedom to manipulate shapes offered in a shape grammar. This freedom comes by viewing the topologies for shapes as a way of explaining a computation as a continuous process after it has been completed, and not as a prerequisite for shapes that must be satisfied before there can be any computation at all.

Stanley Fish with his penchant for interpretative freedom and anti-formalist rhetoric provides a nicely symmetric account of how precedent ensures continuity in judicial decisions:

> . . . rather than the past controlling the present, the present controls the past by providing the perspective from which the two must be brought into line. The truth about precedent, then, is the opposite of the story we tell about it; precedent is the process by which the past gets produced by the

> present so that it can then be cited as the producer of the present. It is in this way that the law achieves what Ronald Dworkin calls 'articulate consistency,' a way of thinking and talking about itself which creates and re-creates the continuity that is so crucial to its largest claim, the claim to have an unchanging center that founds its authority. Articulate consistency is not a fact, but an achievement, something that is forever being wrested out of diverse materials which are then retroactively declared always to have had its shape.

The topologies for the shapes in a computation, and the shape of articulate consistency that gives the law authority are each formed to explain a process by structuring its objects, so that they are connected continuously. And each is fabricated in the same way. If continuity is wanted retroactively in computations with shapes and in judicial decisions -- and in fact, it may be all there is of interest that can be obtained with certainty -- then, once more, what happens when a shape grammar is put to work need not be too far away from what happens when the diverse activities of professional practice unfold.

3 Making It Look More Like Real Topology

The function $g\colon C \rightarrow C'$ describes the application of a rule $A \rightarrow B$ under a transformation t, as it changes the shape C into the shape C'. Strictly speaking, g is continuous with respect to closure relations Γ^* and Γ'^* for the sets PC and PC', where, for any shape x, Px is the set of all parts of x. In this case, the formula

$$g(\Gamma^*(X)) \subseteq \Gamma'^*(g(X))$$

is satisfied for every subset X of PC.

Now, let Γ be a closure relation for the shape C. Then, the corresponding closure relation Γ^* for the set PC is defined in this way. If X is a nonempty subset of PC, then the value of $\Gamma^*(X)$ is given in the formula

$$\Gamma^*(X) = \bigcup_{x \in X} P\Gamma(x).$$

Otherwise, X is the empty set Ø, and $\Gamma^*(\text{Ø}) = \text{Ø}$. Every closed, nonempty subset of the set PC thus contains all of the parts of one or more closed parts of the shape C.

Putting this all together, if the function g is continuous with respect to the closure relations Γ and Γ' for the shapes C and C', then it is also continuous with respect to the corresponding closure relations Γ^* and Γ'^* for the sets PC and PC'.

4 Background

The ideas in this paper touch almost everything that has been said about shape grammars since they were invented more than twenty years ago (Stiny and Gips, 1972). In fact, a complete bibliography with appropriate annotation would not be misplaced. The work involved, however, would require resources of space and time that are not available. Instead, a brief survey of some highlights will have to suffice.

The first completly formal account of shape grammars that provides for emergence in computations with shapes is given in Stiny (1975). More recent definitions and extentions to algebras of various kinds, including the algebras U_{ij}, V_{ij}, and W_{ij}, and their combinations, are developed in Stiny (1990a; 1991; 1992a; 1992b). Some differences between shapes and sets, and why infinite sets of shapes do not always have sums and products are also discussed in Stiny (1992b). Emergence and ambiguity are considered explicitly in Stiny (1989; 1990b; 1992c). Set grammars are defined in Stiny (1982), and their relationships to shape grammars are explored in Stiny (1986; 1987). In particular, the kinds of difficulties encountered when set grammars are formed as counterparts for shape grammars are described.

The characterization of computations with shapes as a kind of inquiry owes much to the ideas of the American pragmatists, especially Charles Saunders Peirce and John Dewey. A sketch of some of the details is presented in Stiny (1992c).

The role played by shape grammars in a more general algorithmic approach to design is examined in Stiny and Gips (1978), and Stiny and March (1981). The variability of interpretation in design and criticism, and its roots in convention, are stressed in both of these studies.

Stanley Fish is very repetitious—maybe he needs to be to make his colleagues understand—but always entertaining. His academic title and affiliation come from the back cover of the paperback edition of *Is There a Text in This Class?* (Fish, 1980). And his first two remarks on the origin of facts and properties in texts are found in the introduction (p. 8 and p. 12). The later discussion of ambiguity (pp. 281-292) in the chapter "Normal Circumstances, Literal Language, Direct Speech Acts, the Ordinary, the Everyday, the Obvious, What Goes without Saying, and Other Special Cases" also relates nicely to themes in this paper. Fish's third observation on how the features of a text emerge is found in the essay "Why No One's Afraid of Wolfgang Iser" (Fish, 1989, p. 75) at a place where he is trying to agree with Iser. And his remarks on professionalism occur in the essay "Anti-Professionalism (Fish, 1989, p. 238). Finally, Fish's discussion of continuity in judicial decisions is taken from the essay "Force" (Fish, 1989, pp. 514-515). Ronald Dworkin introduces the phrase "articulate consistency" in his essay "Hard Cases" (Dworkin, 1977, p. 88). The discussion of jurisprudence in which the phrase is embedded (pp. 85-88) is of interest in its own right, contrasting with Fish's views. Where Dworkin takes the hallmark of articulate consistency to be distributional equality in the application of principles to decide cases involving different people at different times, Fish, perhaps less stringently, settles simply for continuity.

The topological devices used in this paper are standard. The book by John D. Baum (1964), *Elements of Point Set Topology*, provides a good introduction.

Finally, the formal results developed in this paper are presented without proofs. This omission in the interest of brevity is corrected in the long version of the paper (Stiny, 1993). Other elaborations are included, too.

References

Baum, J. D., 1964. *Elements of Point Set Topology*. New York: Dover.
Dworkin, R., 1977. *Taking Rights Seriously*. Cambridge: Harvard.
Fish, S., 1980. *Is There a Text in This Class?* Cambridge: Harvard.
Fish, S., 1989. *Doing What Comes Naturally*. Durham: Duke University.

Stiny, G., 1975. *Pictorial and Formal Aspetcs of Shape and Shape Grammars*. Basel: Birkhauser.

Stiny, G., 1982. "Spatial Relations and Grammars," *Environment and Planning B*, Vol. 9, pp. 113-114.

Stiny, G., 1986. "A New Line on Drafting Systems," *Design Computing*, Vol. 1, pp. 5-19.

Stiny, G., 1987. "Composition Counts: $A + E = AE$," *Planning and Design: Environment and Planning B*, Vol. 14, pp. 167-182.

Stiny, G., 1989. "Formal Devices in Design," in S. L. Newsome, W. R. Spillers, and S. Finger (eds.), *Design Theory 88*, New York: Springer-Verlag, pp. 173-188.

Stiny, G., 1990a. "What Is a Design?," *Planning and Design: Environment and Planning B*, Vol. 17, pp. 97-103.

Stiny, G., 1990b. "What Designers Do That Computers Should," in M. MuCullough, W.J. Mitchell, and P. Purcell (eds.), *The Electronic Design Studio: Architectural Knowledge and Media in the Computer Era*. Cambridge: MIT Press, pp. 17-30.

Stiny, G., 1991. "The Algebras of Design," *Research in Engineering Design*, Vol. 2, No. 3, pp. 171-181.

Stiny, G., 1992a. "Weights," *Planning and Design: Environment and Planning B*, Vol. 19, pp. 413-430.

Stiny, G., 1992b. "Boolean Algebras for Shapes and Individuals," to appear in *Planning and Design: Environment and Planning B*.

Stiny, G. 1992c. "The Pedagogical Grammar," to appear in A. Tzonis and I. White (eds.), *ABCD-Current Issues in Computing and Architecture*. Amsterdam: Elsevier.

Stiny, G., 1993. "Emergence and Continuity in Shape Grammars," manuscript, UCLA, Graduate School of Architecture and Urban Planning.

Stiny, G. and Gips, J., 1972. "Shape Grammars and the Generative Specification of Painting and Sculpture," in C.V. Frieman (ed.), *Information Processing 71*. Amsterdam: North-Holland, pp. 1460-1465.

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

Stiny, G. and March, L., 1981. "Design Machines," *Environment and Planning B*, Vol. 8, pp. 213-238.