Shape Machine: A Primer for Visual Computation

Athanassios Economou, Tzu-Chieh (Kurt) Hong, Heather Ligler, and James Park

1 Introduction

Imagine working on a paper with your familiar word processor in your computer. As you work, you decide to edit a word only to realize that the word cannot be selected as intended, but only a part of it, few letters in no apparent order. You try to edit another occurrence of the same word in the text, and you realize that this time your selection included some fragments of other words adjacent to the one you started with. Frustrated you erase the complete selected mess determined to retype the correct version of the text, only to realize that there are other letters and fragments of letters underneath the text you deleted! In the end, you decide that the only way to modify the text is to retype everything.

Clearly, this scenario is imaginative, but it is used here as a prompt to illustrate the everyday frustration of designers, engineers, scientists, and all users working with lines in 2D and 3D geometric models. Every designer, engineer, and scientist who works with geometry models knows how difficult is to work with existing files—especially if they are done by others—and how difficult or often impossible, simple operations with lines like the *Find* and *Replace* command discussed above, can be.

This odd situation has of course not been unnoticed. As early as the early seventies, Stiny and Gips called out the need for a uniform representation of shape and over the years Stiny and a growing number of scholars, educators, designers, and scientists relentlessly built a school of thought starting at the Open University and the Royal College of Art at UK, later at UCLA and CMU, and since the mid-nineties at MIT, put together the formidable theoretical formalism of shape grammars with the promise of revolutionizing computer-aided design (CAD) (see, e.g., Stiny 1980, 2006; Krishnamurti 1981; Knight 1994; Earl 1997). Shape grammars' foregrounding

A. Economou (✉) · T.-C. Hong · H. Ligler · J. Park
School of Architecture, College of Design, Georgia Institute of Technology, Atlanta 30332, USA
e-mail: economou@gatech.edu

J.-H. Lee (ed.), *A New Perspective of Cultural DNA*, KAIST Research Series,
https://doi.org/10.1007/978-981-15-7707-9_6

of shape rules drawn in a 2D or 3D modeling system over instructions defined in some programming language have provided a robust theory for designers to believe in but nevertheless a formidable challenge to implement. In fact, up-to-date, there are more than 50 shape grammar interpreters that have been designed since the mid-seventies all claiming they can address a particular aspect of shape mapping replacement, the paramount characteristic of this technology (Eloy et al. 2018)—and still none of them has solved conclusively the problem of *Find* and *Replace,* casting thus a shadow of doubt on whether this task is feasible.

The work here takes on this specific problem and proposes a new computational technology, the *Shape Machine*, a new software built for scratch that fundamentally redefines the way shapes are represented, indexed, queried, and operated upon (Hong and Economou, forthcoming). Its foregrounding of visual rules (shape rules drawn in a 2D or 3D modeling system) over symbolic rules (instructions defined in some programming language) provides a robust and disruptive technology for engineers, computer scientists, designers, students, and educators and in general academics and professionals who use drawings and visual models to develop and communicate their ideas.

2 Identity Rules

Seeing in design can be modeled by identity rules—rules that pick-up parts without necessarily doing anything to them (Stiny 1996). The identity rules have identical left-hand sides (LHS) and right-hand sides (RHS) and apply under any given transformation to pick-up parts in a shape seamlessly reorganizing the underlying structure of the shape. The application of a rule follows the general format of a shape production outlined in Stiny (1980). Technically, for shapes A and B, the shape rule $A \rightarrow A$ can apply to a shape B whenever there is a transformation T that makes the shape $T(A)$ part of the shape B. If the shape $T(A)$ is part of the shape B, the rule subtracts the shape $T(A)$ from the shape B and replaces it with the very same shape $T(A)$. The resulting shape B' and the corresponding computation are given in (1).

$$B' = [B - T(A)] + T(A) \tag{1}$$

The computation in (1) captures the notion of a visual query: It restructures a shape in terms of what is the element (shape) that is queried. A series of visual queries are outlined below structured around three conditions: (a) The types of lines that make up a shape, limited here to straight lines, arcs and their combinations; (b) the types of transformations T under which a rule applies, namely isometries and similarities; and (c) the determinacy or indeterminacy of a rule application. Useful overviews of the types of shapes used in shape grammar interpreters are given in Chau et al. (2004) and McKay et al. (2012). A recent discussion on the algebraic language of the group of transformations and its applications in spatial design is given in Pottman et al. (2013). A preliminary discussion on the determinate or indeterminate application of

a rule and its relation on the types of intersections between pairs of lines is given in Stiny (2006) and Hong and Economou (forthcoming). A definitive account of the types of shapes based on the possible types of intersections between pairs of lines or arcs is given in Economou and Yu (forthcoming) and a catalogue of n-line shapes for $n \leq 4$ in Economou and Park (forthcoming). Clearly, the examples given below do not exhaust all possible permutations of queries but are selected to give a sense of the expressive power of the search engine of the Shape Machine. For brevity—and to foreground the intuitive aspects of the visual search, these queries are given here in a single shape representation foregrounding the shape that is searched.

2.1 Find Shapes Consisting of Lines Under Isometric Transformations

The first examples of visual queries in Shape Machine are constructed around shapes that consist of straight lines and are searched under isometric transformations in a determinate way. Any shape consisting of straight lines with more or equal than two registration points would do and here the computations shown are based on existing ones in the literature to make the transition from the theory to the application as clear as possible (see, e.g., Stiny 2006). All examples of searches here are based on isometric transformations, that is, transformations that keep the shape and size of a shape but alter its position and/or handedness. Here, this search means that it is confined to congruent instances of the shape that is searched.

The most straightforward shapes consisting of lines to search for are the closed polygons, regular or not. These shapes have typically a name, i.e., triangles, squares, rectangles, rhombi, quadrilaterals, pentagons, hexagons, and so forth, and because of their straightforward geometric structure consisting of well-defined circuits of vertices and edges, they are the very first shapes that have been used to demonstrate the desired ability of a shape grammar interpreter to identify different shapes from the ones used in a two-dimensional model in a visual search (Grasl and Economou 2013). In these cases, the shape grammar interpreter is able to query a specific design in terms of any well-defined polygons. Shape Machine is able to find any closed polygons in a shape even though these polygons may have not been explicitly registered in the database of a CAD system. The example in Fig. 1 showcases one of the possible ways that an initial design consisting of two squares can be queried to reveal possible embedded triangles, squares, pentagons, hexagons, and so forth. The process is straightforward: A pictorial query of a triangle drawn on top of the initial model identifies three more triangles in the model for a total of four. A pictorial query of a square drawn on top of the initial model identifies no more isometric instances of the square in the model. A pictorial query of a pentagon drawn on top of the initial model identifies three additional isometric pentagons in the model for a total of four. The example below shows that a pictorial query of a hexagon in Fig. 1a, in the design

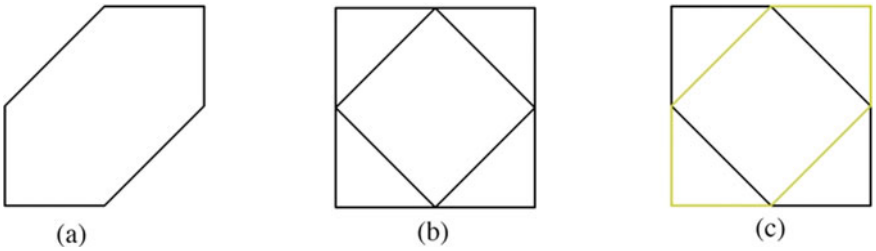

Fig. 1 A query of a closed polyline in Shape Machine under isometric transformations

consisting of two squares in Fig. 1b, identifies two embedded isometric hexagons, one of them shown in Fig. 1c.

The search for polygons can be extended for arrangements of polygons even though these arrangements are not registered in the database of a CAD system. The example in Fig. 2 shows a pictorial query of a spatial relation between two isosceles right triangles reflected along the leading diagonal of an underlying square upon which the triangles are embedded in. Note that this very specific spatial relation can be described in a number of alternative and equivalent ways (Economou and Kotsopoulos 2014). Even more, this spatial relation is just one of several other examples of spatial relations between closed polygons that may be observed in this design, including arrangements of triangles, squares, pentagons, hexagons, and so forth, all sharing a vertex or edge or just floating one to another. In all cases, the Shape Machine can calculate the symmetries of the matches and give the correct number of non-equivalent instances. Here the query shape x consisting of the two right triangles is shown in Fig. 2a, the design that is queried is given in Fig. 2b and one of the two isometric matches of the identity rule is given in Fig. 2c.

Shapes consisting of straight lines need not be well-defined closed polygons or arrangements of polygons. The pictorial query can be extended for continuous lines composed by one or more line segments evoking recognizable symbols such as the letters of the alphabet, for example, A, M, Σ or K, or other symbols from other notational systems. The example in Fig. 3 continues with a query of a three-line shape in the form of the lower-case letter k whose longer edge matches the edge

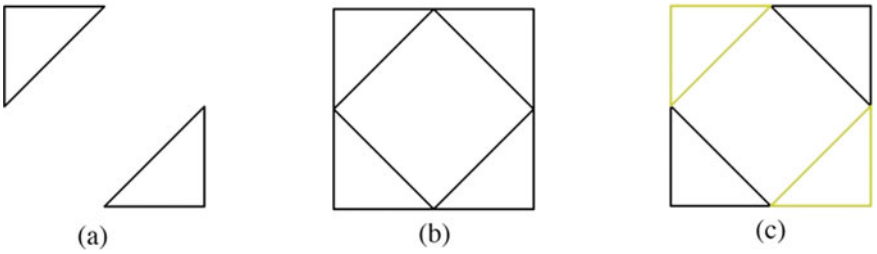

Fig. 2 A query of an arrangement of two polygons in Shape Machine under isometric transformations

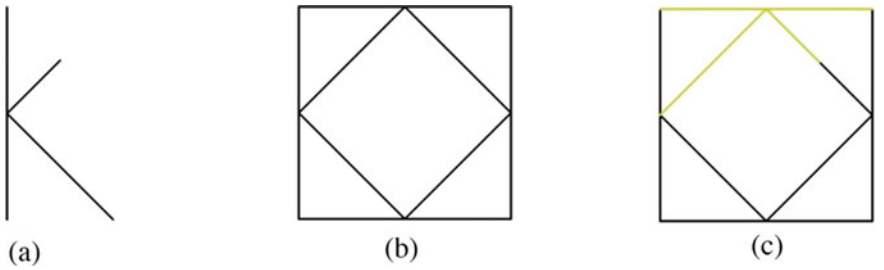

Fig. 3 A query of an arrangement of open polylines in Shape Machine under isometric transformations

of the square, one of its legs matches the edge of the smaller rotated square in the design and the third upright edge has an endpoint (boundary) that is embedded in the edge of the smaller rotated square. Note that the three of the endpoints of the shape *k* as well as its middle 3-valent intersection of its three legs are all registered in the database of the CAD system within the intersections of the lines making up the two underlying squares but the endpoint of the upper leg of the shape *k* in not. The query in Shape Machine yields, respectively, eight instances for each search because the symmetry of the lower-case *k* is equal to 1 and the symmetry of the overall design is 8, and by definition, there are no interactions between the symmetry elements of the symmetry groups of the two shapes—the *k*-shape and the overall design (Stiny 1986). As above, the Shape Machine can calculate the symmetries of the matches and give the correct number of non-equivalent instances. The query *x* of the lower-case *k* is shown in Fig. 3a, the design that is queried in Fig. 3b and one of the eight isometric matches of the identity rule is given in Fig. 3c.

2.2 Find Shapes Under Similarity Transformations

Visual queries need not confine to identical copies. Often designers want to search for similar copies of a shape in smaller or larger versions and in any location and/or possible enantiomorphic or handed versions in a model or series of models. The new unique transformation introduced in the visual query is the scale transformation that varies the size of the shape. Scale transformations combine with the isometric transformations to produce the similarity transformations that keep the shape of a shape but alter its size, position, and/or handedness. The next series of visual queries in Shape Machine are constructed around shapes that consist of straight lines and are searched under similarity transformations in a determinate way. Here, this search means that it is confined to similar instances of the shape that is searched. Note that when these searches are defined for shapes that have some conventional name, say squares, quadrilaterals, and so forth, they retain their semantics; i.e., the search for a particular shape, say a square, is extended for all possible squares in the design.

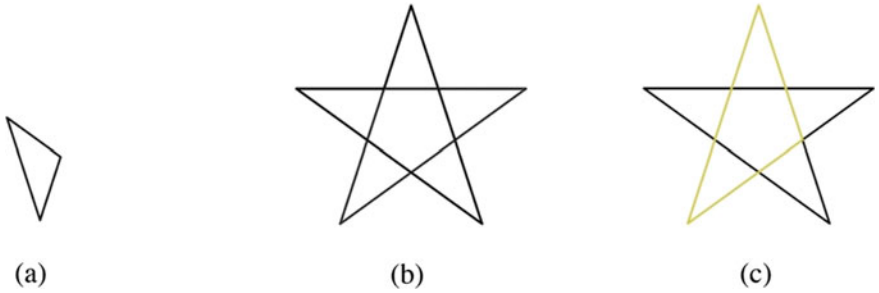

Fig. 4 A query of a polygon in Shape Machine under similarity transformations

The operations of visual queries can be extended for all sorts of shapes consisting of lines. All examples below are structured around the target shape of the 5-star polygon to start from the initial inquiry in Stiny (2006) and continue in a rising complexity from polygons and arrangements of polygons to arrangements of lines.

The most straightforward shapes consisting of lines to search for under similarities are the closed polygons, regular or not. An example of a visual query of a polygon under similarity transformations is given in Fig. 4. Here, the query is an isosceles triangle in with three angles 108°, 36°, and 36° respectively, and the shape that is queried is a regular 5-star polygon. Clearly, several more types of polygons could be searched in this design, for example, convex and concave quadrilaterals, regular pentagons, concave hexagons, concave heptagons, and several more. In all cases, the Shape Machine calculates the exact number of matches without any duplicates and the search here identifies five similar triangles in five different orientations. The query x of the isosceles triangle with angles 108°, 36°, and 36° is shown in Fig. 4a, the 5-star regular polygon design that is queried in Fig. 3b and one of the five similarity matches $T(x)$ of the identity rule is given in Fig. 4c.

The search for similar polygons can be extended for similar arrangements of polygons in any spatial relation without having them registered in the database of a CAD system. An example of a visual query of three polygons in a spatial relation is shown in Fig. 5. The specificity of this spatial relation is quite involved: All three polygons in the spatial relation are isosceles triangles having angles 36°, 72°, and 72°, respectively; two of them pivot around a shared vertex in an angle of 144° so that the endpoint of the short side of one triangle coincides with the endpoint of the long side of the other triangle and both lines forming one continuous line, while a third triangle lies on the middle reflection axis bisecting the angle 144° with its apex at a specific distance l. The discursive description of all this information is indifferent to the Shape Machine that is able to query the shape x in Fig. 5a, in the design in Fig. 5b and give the possible five matches $T(x)$, one of which is shown in Fig. 5c. In all cases, the Shape Machine calculates the exact number of matches without any duplicates: The symmetry of the arrangement of the three triangles is given by the dihedral symmetry group of order 1 and is equal to 2, and the symmetry of the star design is given by the dihedral symmetry group of order 5 and is equal to 10; the

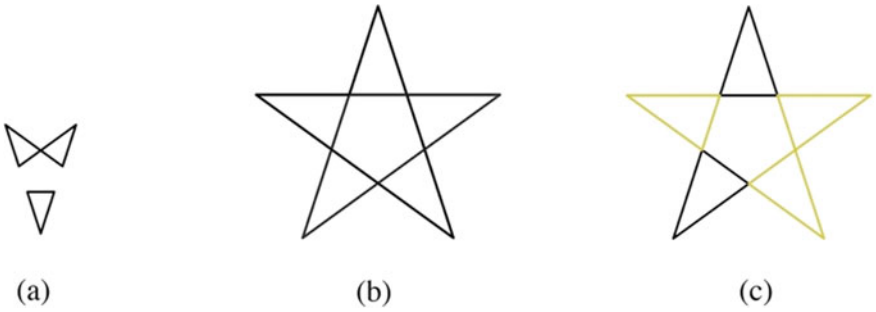

Fig. 5 A query of an arrangement of polygons in Shape Machine under similarity transformations

two shapes share one symmetry element—a mirror refection and the complete non-equivalent matches $T(x)$ are equal to the division of the orders of their symmetry groups equal to 5.

The visual query of shapes consisting of arrangements of lines open can quickly become complex engaging open-ended arrangements of lines that at any desired spatial relation without necessarily specifying some gestalt or conventionalized spatial arrangement. An example of such an arrangement of an L-shape consisting of two lines versus a simple line is shown in Fig. 6. The two lines, the composite one and the straight one, comprise a shape that consists of three lines that require an extensive discursive description to be precise: The three lines lie upon an underlying network of grid-lines forming an isosceles triangle whose inner three angles are 36°, 36°, and 108°, respectively; the projection of the single straight line meets the endpoint of the longest leg of the composite line; and the projection of the short leg of the composite shape meets the straight line in its body. Clearly, these spatial relations between the two lines—the composite and the simple—or the three lines, depending on how one sees it—need more features and parameters to be fully captured, including the lengths of the various parts of the lines in the relations. The structuring of all this information is indifferent though to the Shape Machine that is able to query the shape x in Fig. 6a in the design in Fig. 6b and give the possible ten matches $T(x)$, one of which is shown

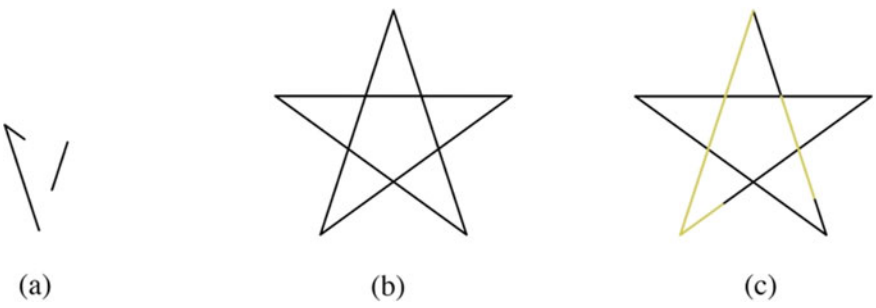

Fig. 6 A query of an arrangement of lines in Shape Machine under similarity transformations

in Fig. 6c. In all cases, the Shape Machine calculates the exact number of matches without any duplicates.

2.3 *Find Shapes Consisting of Arcs Under Isometric Transformations*

The second family of examples of visual queries in Shape Machine is constructed around shapes that consist of arcs that can be searched under isometric transformations. Any shape consisting of arcs would do, and here as before the computations shown are based on existing ones in the literature to make the transition from the theory to application as clear as possible (see, e.g., Jowers and Earl 2011).

A straightforward class of shapes consisting of arcs to search for under isometries is the closed lens shapes, constructed upon regular or irregular polygons substituting their straight edges with arcs of various lengths. An example of a visual query of a lens or *vesica piscis* shape, a classic figure in Euclid (Fletcher 2004)—and Carlo Scarpa's work for that matter—under isometric transformations is given in Fig. 7. The shape that is queried is a trefoil-like shape consisting of three identical arcs drawn from the vertices of an underlying regular triangle. The radii of the arcs of these shapes are typically equal to the length of the sides of the underlying regular triangle but here are equal to the two-thirds of the edge of the triangle. The symmetry of the *vesica piscis* shape is given by the dihedral symmetry group of order 1 and is equal to 2, and the symmetry of the trefoil design is given by the dihedral symmetry group of order 3 and is equal to 6. The two shapes share one symmetry element—a mirror refection, and the complete non-equivalent matches $T(x)$ are equal to the division of the orders of the two symmetry groups equal to 3. The query x of the *vesica piscis* shape is shown in Fig. 7a, the trefoil design that is queried in Fig. 7b and one of the three isometric matches $T(x)$ of the identity rule is given in Fig. 7c.

Shapes consisting of arcs need not be well-defined closed arrangements of sectors of circles or unions or intersections of sectors of circles. The pictorial query can be extended for continuous arcs composed by one or more arcs. The example in Fig. 8

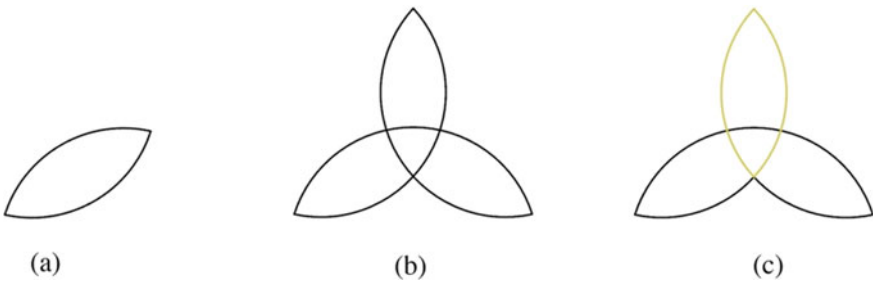

Fig. 7 A query of a closed arc shape in Shape Machine under isometric transformations

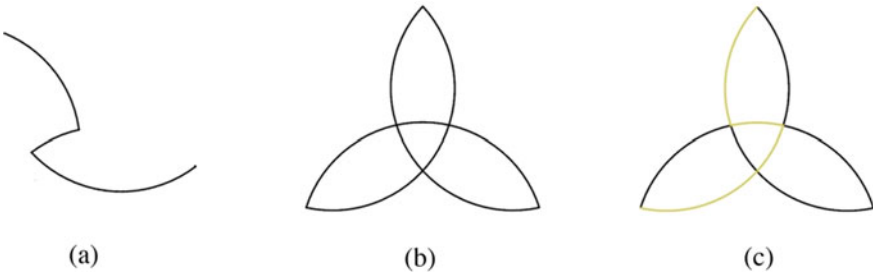

Fig. 8 A query of an arrangement of arcs in Shape Machine under isometric transformations

continues with a query of a three-arc shape in the style of the Mac logo. As before, the description of this shape can be formidable: Here, a chain of three arcs that share two endpoints with alternate convexities and concavities and specific radii and centers upon a regular triangle. Still, the query in Shape Machine is straightforward by simply pointing to the shape that will be queried. The query here yields six results calculating the threefold dihedral symmetry of the three-lens polygon of order 6 by the order of symmetry of the compound arc equal to 1. The query x of the compound arc is shown in Fig. 8a, the trefoil design that is queried in Fig. 8b and one of the six isometric matches $T(x)$ of the identity rule is given in Fig. 8c.

The previous example showed the query of a compound arc whose structural features, including centers of arcs and intersections of arcs, could be potentially retrieved from the database of CAD system. The next example in Fig. 9 showcases the search of a shape consisting of a rotational arrangement of three arcs such that each arc has only one endpoint registered in the database of a shape to be queried. Clearly, this query can be extended for a variety of shapes consisting of three or more arcs or combinations of arcs and straight lines without caring about whether their boundaries, endpoints, and intersections are registered in the database of the CAD system to be searched. The query here yields two results because the threefold dihedral symmetry of the three-lens polygon of order 6 is divided by the order of symmetry of the rotational symmetry of order 3 as long as both shapes share three symmetry elements, namely the rotations of order 0°, 120°, and 240° around their

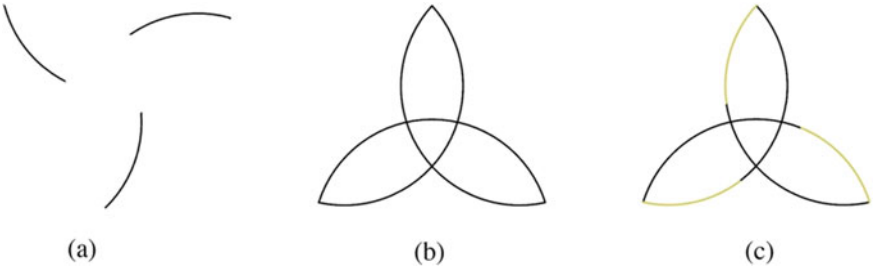

Fig. 9 A query of an arrangement of arcs in Shape Machine under isometric transformations

centers. The query x of the rotational arrangement of the three arcs is shown in Fig. 9a, the trefoil design that is queried in Fig. 9b and one of the two isometric matches $T(x)$ of the identity rule is given in Fig. 9c.

2.4 *Find Shapes Consisting of Arcs Under Similarity Transformations*

Visual queries of arcs need not confine to identical copies. As before, often designers search for similar copies of a shape in smaller or larger versions and in any location and/or possible enantiomorphic or handed versions in a model or series of models. Interestingly, scale transformations overall appear to affect the queries of shapes in similar ways with the queries of shapes consisting of straight lines but there are significant differences too—that will become apparent in the next section on the determinate and indeterminate application of recognition of identity rules.

The first family of examples of visual queries in Shape Machine of shapes that consist of arcs that can be searched under similarity transformations are again queries of the boundaries of well-defined unions, intersections, or symmetric differences of circles and/or closed polygons. One straightforward example of an intricate Boolean intersection of three circles constructed around an underlying equilateral triangle is shown in Fig. 10. The number of the three similar matches $T(x)$ of this shape is calculated by checking whether the symmetry elements of the query the queried shape intersect, and if yes, by dividing the order of symmetry of the queried shape by the order of symmetry of the query. The query x of the Boolean intersection is shown in Fig. 10a, the trefoil design that is queried in Fig. 10b and one of the three similarity matches $T(x)$ of the identity rule is given in a highlighted form in Fig. 10c.

A second family of examples of visual queries in Shape Machine of shapes that consist of arcs that can be searched under similarity transformations can be constructed by taking any arrangements of arcs irrelevant of whether they specify closed shapes or connected arcs in continuous circuits. Even more, this inquiry can be extended by asking that the boundaries of the arcs (endpoints) are not necessarily

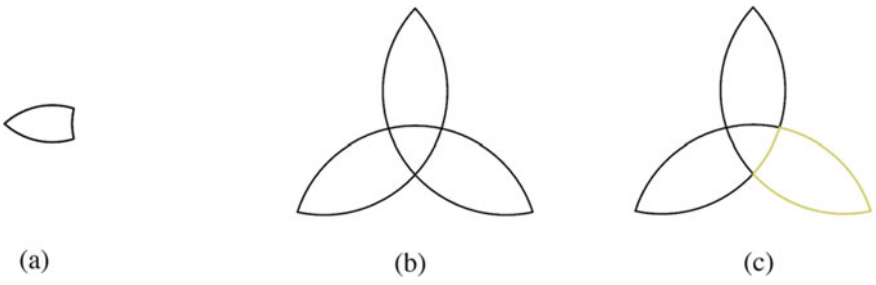

Fig. 10 A query of an arrangement of arcs in Shape Machine under similarity transformations

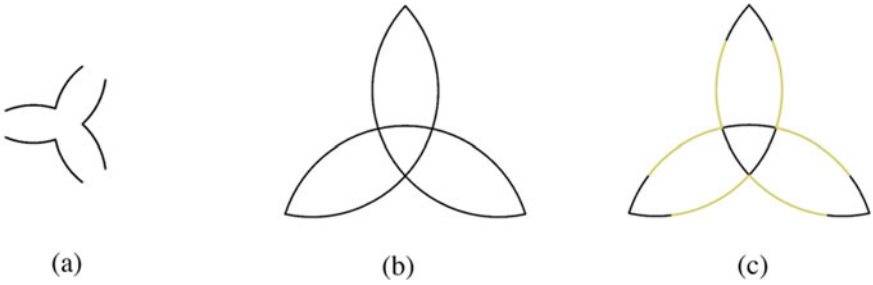

Fig. 11 A query of an arrangement of arcs in Shape Machine under similarity transformations

registered in the database of the CAD system that underlies the specification of a shape that will be searched. An example of a query of such a shape is given in Fig. 11. Here a shape consisting of six arcs is queried with three points defined as the intersections of three pairs of arcs while the other six endpoints are indefinitely defined upon the circumference of the arc. The lengths of the arcs are all the same, and they are identically disposed around a center giving to the shape a dihedral symmetry of order 3; that is, a total symmetry of order 6 equal to the order of symmetry of the trefoil shape that is queried. The query x of the six arcs is shown in Fig. 11a, the trefoil design that is queried in Fig. 11b and the single similarity match $T(x)$ of the identity rule is given in a highlighted form in Fig. 11c.

A last example of a visual query in Shape Machine of shapes that consist of arcs and that can be searched under similarity transformations is considered here by taking any arrangement of arcs irrelevant of specificities of spatial relations between them and to an underlying shape. Here this query shape x is constructed by eliminating parts of a trefoil shape yielding a highly expressive shape that defies a discursive description. Clearly, the symmetry of the emergent shape is trivial and equal to 1 and there are six $T(x)$ matches of this shape in the trefoil shape. The query x of the fragmented shape (arrangements of arcs) is shown in Fig. 12a, the trefoil design that is queried in Fig. 12b and one of the six similarity matches $T(x)$ of the identity rule is given in a highlighted form in Fig. 12c.

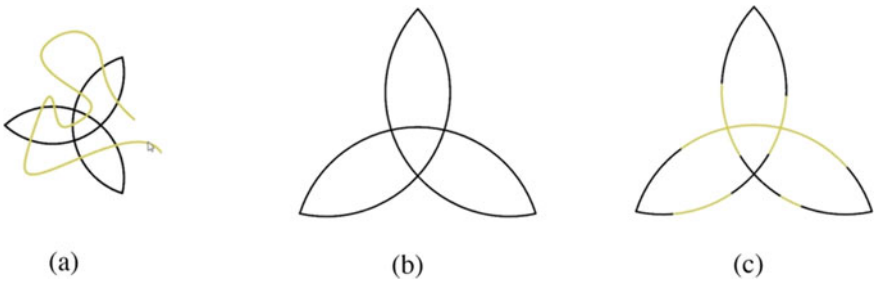

Fig. 12 A query of an arrangement of arcs in Shape Machine under similarity transformations

2.5 *Exceptions Noted: Indeterminate Applications of Rules*

The results of the queries of the shapes have been so far very successful for a good reason: Most rules apply in a determinate way, and the results can be enumerated and visually inspected. Still, in some cases, the application of the rules is not straightforward and some additional information may be needed to make the exact matches. In these cases, the queries rather than producing a set of instances of shapes $T(x)$ that match the query x, they produce families of sets of instances of shapes $T(x)$. A set of conditions specifying the determinacy or indeterminacy of a rule application is given in Stiny (2006). Any shape rule with a LHS shape consisting of lines in the plane with zero registration points cannot apply in a determinate way under any Euclidean, affine or linear transformation; and any such shape rule with one registration point cannot apply in a determinate way under any similarity, affine or linear transformation—or otherwise, it can apply in a determinate way only under isometric transformations; in either case, additional information is needed to determine the match of the rule.

An example of an indeterminate query can be given in the query of a lower-case k-shape under similarity transformations in Fig. 12. All three lines of the k-shape intersect in a singular point and match the condition above when the lines of the shape intersect in one registration point. The shape k can be matched (found) in eight different kinds of ways in a shape consisting of two squares because of the spatial relations of the elements of the symmetry groups of the two squares and the k-shape. Still, for any of these eight matchings there is an indefinite number of scale transformations that match the k-shape in the shape of the two squares having as a maximal extreme case the matching of the long line of the k-shape with the edge of the large square and as a minimum extreme the arbitrary screen resolution of the k-shape. The query x of the lower-case k-shape is shown in Fig. 13a, the nested square design that is queried in Fig. 13b and one instance $T(x)$ of the eight families of similarity matches with a scalar transformation of 1.5 of the identity rule is given in a highlighted form in Fig. 13c.

These two-step visual queries in Shape Machine need not be confined to the queries of well-structured shapes like the lower-case k-shape under similarity transformations. The queries x can be extended to any shapes consisting of arrangements

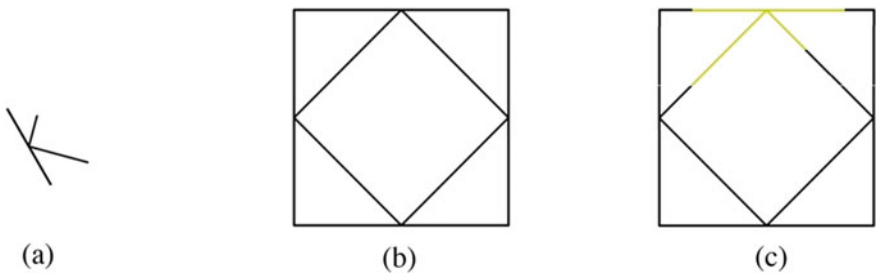

Fig. 13 A two-step query of an arrangement of lines in Shape Machine under similarity transformations

of lines drawn upon construction lines intersecting on a singular registration point. An example of a query of y-shape with an additional floating line emanating from its intersection point is given in Fig. 14. The query x of the y-shape is shown in Fig. 14a, the nested square design that is queried in Fig. 14b and one instance $T(x)$ of the eight families of similarity matches with a scalar transformation of 1.5 of the identity rule is given in a highlighted form in Fig. 14c.

Finally, these two-step visual queries in Shape Machine can be combined with symmetry calculations to provide the answers to queries that entail symmetrical matches and elimination of similar results. An example of such computation is given in Fig. 15. Here, the query is given in the form of a shape consisting of three separate lines, resembling a three-stroke symbol in some symbolic language, with a mirror symmetry of order 2. The query x of the symmetrical shape is shown in Fig. 15a, the nested square design that is queried in Fig. 15b and one instance $T(x)$ of the four families of similarity matches with a scalar transformation of 1.5 of the identity rule is given in a highlighted form in Fig. 15c.

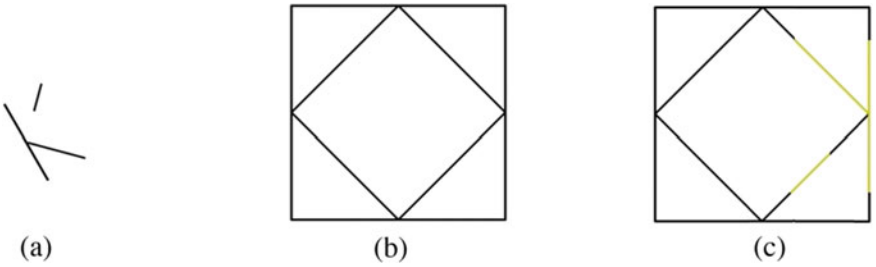

Fig. 14 A two-step query of an arrangement of lines in Shape Machine under similarity transformations

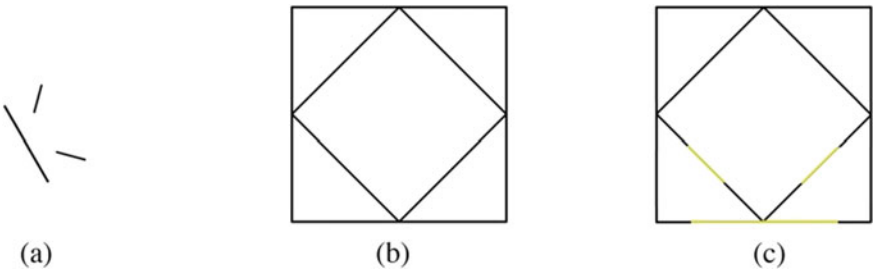

Fig. 15 A two-step query of an arrangement of lines in Shape Machine under similarity transformations

3 Replacement Rules

Doing in design can be modeled by replacement rules—rules that pick-up parts and replacing them with other parts. The replacement rules have different *LHS* shapes and *RHS* shapes and apply under any given transformation to pick-up parts in a shape reorganizing the underlying structure of the shape. The application of a replacement of production rule follows the general format of a shape production outlined in Stiny (1980). Technically, for shapes A and B, the shape rule $A \rightarrow B$ can apply to a shape C whenever there is a transformation T that makes the shape $T(A)$ part of the shape B. If the shape $T(A)$ is part of the shape C, the rule subtracts the shape $T(A)$ away from the shape B and replaces it by the shape $T(B)$. The resulting shape C' and the corresponding computation are given then as:

$$C' = [C - T(A)] + T(B) \tag{2}$$

The replacement rules (doing rules) in (2) are essentially rules in the replacement schema $x \rightarrow y$ (Stiny 2006; Economou and Grasl 2018). As before, the series of replacements outlined below exemplify the three conditions outlined in the seeing rules: the types of shapes involved in the computation (straight lines and arcs), the transformation under which the rules apply, and the types of rules involved in the computation—determinate or indeterminate—and as a consequence, the number of steps required to execute the rule (so far, one-step rules or two-step rules). After all, for a rule $x \rightarrow y$ to apply to a shape A the identity rule $x \rightarrow x$ has to apply under some transformation T and then the resulting shape $T(x)$ can be subtracted from the shape A and replaced by the corresponding transformation of $T(y)$ to produce the shape $A - T(x) + T(y)$.

Significantly, this series of rule applications (visual replacements) may be structured around one extra condition: the recursive definition of the schema y in terms of the schema x, so that the two parts of the rule are related in some way captured in distinct schemata rules (Stiny 2006; Economou and Kotsopoulos 2014). Here, a very brief discussion of these design actions (rule schemata) is given and the focus is given instead in summations of design actions to foreground the visual impact of the rule. Such fractal designs (Mandelbrot 1982) often provide the initial discourse for recursive geometric modeling because of the nature of the ordered repetition of given rules. All examples in this section are extracted from actual computations with the Shape Machine and are not edited in any way: The rules are represented in the classical shape grammar format with the two sides of the production system, *LHS* and *RHS*, the arrow (→) in-between and the two registration marks (+) on either side of the middle arrow (→) to fix the spatial relation between them. The shape rules in Shape Machine can be imported, or they can be defined from scratch.

3.1 *Substitute Shapes Consisting of Lines Under Isometry and Similarity Transformations*

An initial set of examples of replacement rules in Shape Machine is constructed within the rule schema $x \rightarrow x + t(x)$. These computations are samples of computations of shapes consisting of straight lines and searched under isometry and similarity transformations in a determinate or indeterminate way. The rule in Fig. 16 specifies that a right isosceles triangle should be substituted by the same triangle x plus a half size right isosceles triangle $t(x)$. Note that this specific action can be described equally well in a different of other schemata, for example, $x \rightarrow t_1(x) + t_2(\mathrm{x})$, for t_1 and t_2 similarity transformations of the original right isosceles triangle; or in terms of different operators, for example divisions and boundaries that may divide an initial shape into two different ones or even identical copies of itself as in rep-tile constructions (Gardner 2001). The series of parallel applications of the rule in the initial design comprised of four triangles produces a series of fractal-like designs comprised by $4 + 8 = 12$ triangles, $4 + 8 + 16 = 28$ triangles, $4 + 8 + 16 + 32 = 60$ triangles, and $4 + 8 + 16 + 32 + 64 = 124$ triangles, respectively. The shapes x in the *LHS* and $x + t(x)$ in the *RHS* of the rule are shown in Fig. 16a; one of the four possible matches of the shape $T(x)$ under a similarity transformation T and its corresponding replacement of the shape $T(x + t(x))$ is given in Fig. 16b; and a design after the execution of a series of rules in Fig. 16c.

Both parts of a shape rule, *LHS* and *RHS* are editable in Shape Machine. The example in Fig. 17 demonstrates the design of the *RHS* of a rule in the schema $x \rightarrow y$ or perhaps and more specifically, $x \rightarrow x - prt(x) + y$, for x an isosceles right triangle, $prt(\mathrm{x})$ the part of the hypotenuse of the right isosceles triangle that is erased, and y a new set of lines in some spatial arrangement with the leftover part of the right isosceles triangle. Clearly, this last action can be described in a number of alternative ways (Economou and Kotsopoulos 2014). A single application of the rule in the schema $x \rightarrow \sum T(y)$ in the initial design ends the computation providing a design with a dihedral symmetry of order 8. The shapes x in the *LHS* and y in the *RHS* of the rule are shown in Fig. 17a; one of the four possible matches of the shape $T(x)$ under a similarity transformation T and its corresponding replacement of the shape

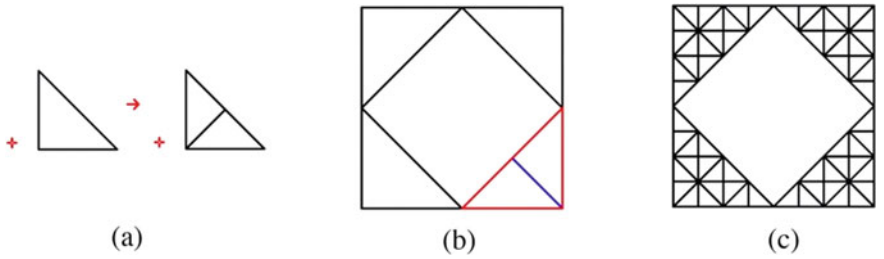

Fig. 16 A construction in Shape Machine using a single shape rule in the schema $x \rightarrow \sum T(x + t(x))$

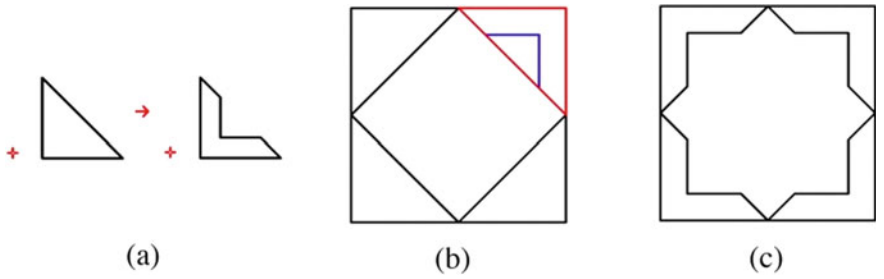

Fig. 17 A construction in Shape Machine using a single shape rule in the schema $x \rightarrow \sum T(y)$ in a determinate way

$T(y)$ is given in Fig. 17b; and a final design after the execution of a series of rules in Fig. 17c.

New shape rules can be defined from scratch in the Shape Machine and are editable right away. The construction of a design in Fig. 18 showcases the power of radical redescription Shape Machine can offer in design inquiry. The rule used here to illustrate exemplify this generative redescription is polemically built upon the k-shape, an otherwise rather esoteric illustration of the notion of a radical change of vocabularies in visual composition. Here, the rule specifies that the k-shape in the *LHS* should be replaced by a completely new shape in the *RHS* of the rule that is formed by connecting the endpoints of the k-shape with a set of two disjoint lines. Note that the application of the rule under isometry or similarity transformations provides a sense of surprise to the designer because of the emergent connections of previously disjointed lines to facilitate a reading of the overall design into one continuous folding line. The shapes x in the *LHS* and y in the *RHS* of the rule are shown in Fig. 18a; one of the eight possible matches of the shape $T(x)$ under a similarity transformation T and its corresponding replacement of the shape $T(y)$ is given in Fig. 18b; a final design after the execution of a series of rules is shown in Fig. 18c.

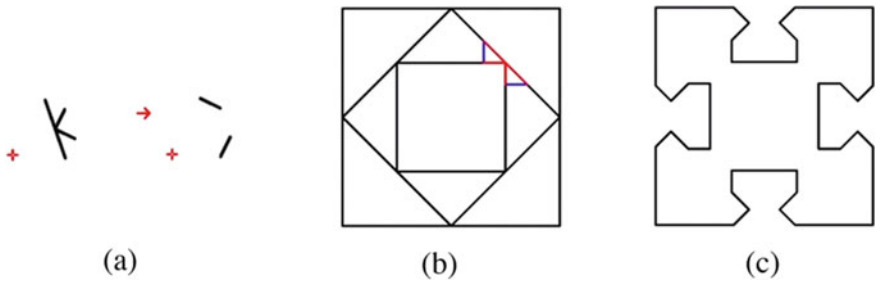

Fig. 18 A construction in Shape Machine using a single shape rule in the schema $x \rightarrow \sum T(y)$ in an indeterminate (two-step) way

3.2 *Substitute Shapes Consisting of Lines and Arcs Under Isometries and Similarities*

A second set of examples of replacement rules in Shape Machine may be constructed around shapes that consist of arcs and straight lines and searched under isometry and similarity transformations in a determinate or indeterminate way. The examples are built upon one of the studies of the original ice-ray grammars (Stiny 1977; Economou and Grasl 2018). The conventions of that analog grammar, including the specification of the initial labeled shape and the labeled rules, are exactly the same with the ones used in this digital grammar. However, the examples are extended here with the additional intent to generalize the design workflow from configuration to ornament demonstrating a seamless process in design. In this first example, the rule is cast in the general schema $x \rightarrow y$ or more specifically $x \rightarrow prt\ x + y$. In this example, the shape x in the *LHS* is a square with a circular label on the top left corner and the shape y in the *RHS* is an elongated hexagon with an internal set of diagonals to create two triangles and two quadrilaterals (and many more shapes as well)—all with a dihedral symmetry of order 4. The shape to be searched in terms of this rule is a 4 $\times$ 4 square grid with alternating labels ensuring a diagonal reflectional pattern. The shapes x in the *LHS* and y in the *RHS* of the rule are shown in Fig. 19a; one of the sixteen possible matches of the shape $T(x)$ under a similarity transformation T and its corresponding replacement of the shape $T(y)$ is given in Fig. 19b; a final design after the execution of a series of rules is shown in Fig. 19c.

A simple reworking of the same schema rule with a substitution of a shape made of straight lines in the *LHS* with a shape made up for arcs in the *RHS* can produce a radical redescription in a design. The substitution of the square shape x with a circular label on its top left corner in the *LHS* of a shape rule with a *vesica piscis* shape composed by two arcs with centers the two diagonal vertices of the square and radii the edges of the square in the *RHS* produces a rule that when it is applied in all sixteen parts of a 4 $\times$ 4 alternating square grid as described above, makes a completely new design that features the same underlying symmetry but with very different expressive qualities. Significantly, none of the underlying straight lies survive the parallel

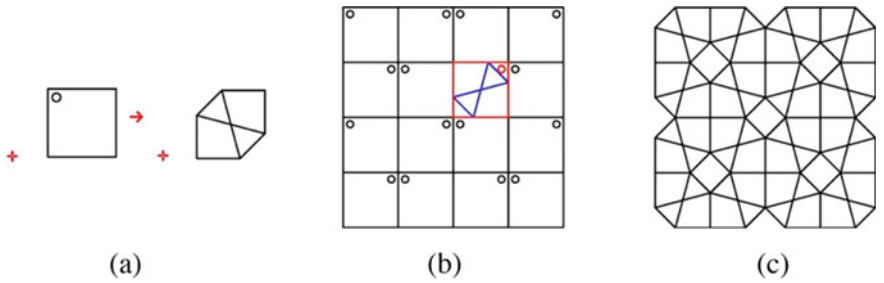

Fig. 19 A construction in Shape Machine using a single shape rule in the schema $x \rightarrow \sum T(y)$ in a determinate way

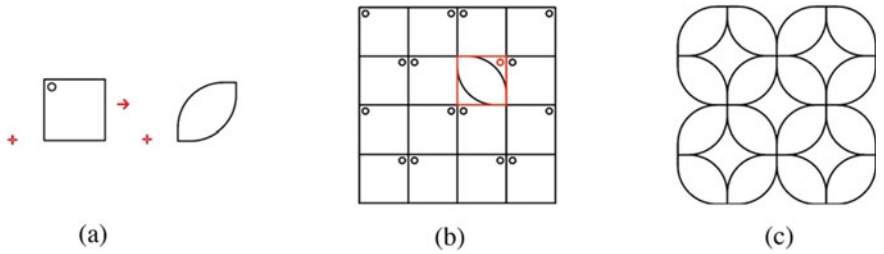

Fig. 20 A construction in Shape Machine using a single shape rule in the schema $x \rightarrow \sum T(y)$ in a determinate way

replacement of all the rule applications and the new design acquires a very different feel and look. The shapes x in the *LHS* and y in the *RHS* of the rule are shown in Fig. 20a; one of the 16 possible matches of the shape $T(x)$ under a similarity transformation T and its corresponding replacement of the shape $T(y)$ is given in Fig. 20b; a final design after the execution of a series of rules is shown in Fig. 20c.

These replacement rules need not confine to well-structured or conventionalized geometries as in the 4×4 alternating square grid discussed above. Shapes made of arcs can be transformed to straight lines on the spot and the other way around too if a shape rule that does so apply to them. Both translations are typical with the schema rule $x \rightarrow y$ and/or $x \rightarrow \sum T(y)$ if the rules are applied in a fractal way. An example illustrating this translation between geometries is given in Fig. 21. In this example, the part of an arc of a circle in the *LHS* of a rule is substituted by a three-line bracketed shape. When this rule applies in the design generated in the previous example, it generates a radically different lattice that keeps the topology of the previous pattern. The shapes x in the *LHS* and y in the *RHS* of the rule are shown in Fig. 21a; one of the 32 possible matches of the shape $T(x)$ under a similarity transformation T and its corresponding replacement of the shape $T(y)$ is given in Fig. 21b; a final design after the execution of all possible rule applications is shown in Fig. 21c.

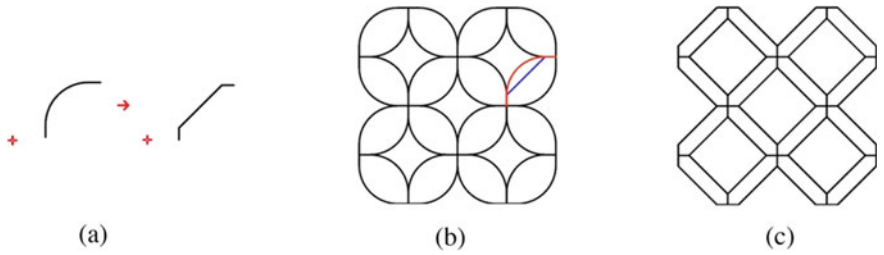

Fig. 21 A construction in Shape Machine using a single shape rule in the schema $x \rightarrow \sum T(y)$ in a determinate way

4 Applications

A very different discussion of rule replacements—and design actions—may be considered when the focus is directed instead in the calculation of specific examples in design domains such as ornamental design, product design, CAD drafting, architectural design, mechanical design, electrical design, and so forth. Few selected examples are given below to showcase the versatility and expressiveness of the Shape Machine in distinct design flows in an interactive mode as well as in a fully automated mode suggesting the promise of a complete visual programming language and/or a new paradigm in computing machinery.

4.1 Product Design: Celtic Knots

A product design application to generate a Celtic knot weave is presented in the form of a shape grammar implemented in the Shape Machine. Celtic knots and their spirals, step patterns, and key patterns are some of the most enduring stylized graphical representations of knots in ornamental art (Bain 1975). A shape grammar for Celtic knots has been proposed by Jowers and Earl (2011) and provides the blueprint for its implementation in Shape Machine. The grammar is modeled in three stages: the first stage fixes the central horizontal and vertical growth of the pattern. The second takes care of the corner conditions retaining the motion of the alternating loop throughout the pattern. The last stage modifies the intersections of the loops to disentangle the knot and begin to show how spirals, step patterns, and key patterns may be treated in the composition. The shape rules consist of shapes composed of arcs and straight lines and apply to the working model (design) under similarity transformations. Interestingly, the rules appear that work within the schema $x \rightarrow x + t(x)$, for x a loop and $x + t(x)$ a double loop, but the details are rendered somewhat differently to account for the alternating motion of the loop.

The first stage of Celtic knot grammar captures the underlying central horizontal and vertical growth of the pattern and consists of two shape rules that fix the horizontal and vertical growth of the pattern. Clearly, the rules can be applied several times to generate large cruciform configurations of weaves. Both shape rules have been designed directly in Shape Machine using an initial shape of the two loops as a construction shape, and they apply to the evolving design under similarity transformations. The open-loop shape x in the *LHS* and the open double-loop shape y in the *RHS* of the rule are shown in Fig. 22a; one of the two possible matches of the shape $T(x)$ under a similarity transformation T and its corresponding replacement of the shape $T(y)$ is given in Fig. 22b; a final design after the execution of two possible rule applications is shown in Fig. 22c.

The second stage of the Celtic knot grammar captures the field growth of the pattern from its central horizontal and vertical axes to its boundary extents. Two additional rules are introduced to capture the growth of the pattern along its minor

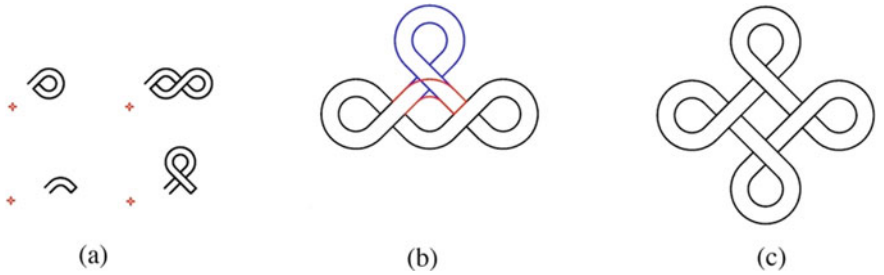

Fig. 22 Celtic knot grammar: Configuration stage

horizontal and vertical axes and resolve the corner conditions keeping throughout the alternating loops of the pattern. Note that the derivation of the pattern could terminate at the will of the designer to give a rotational symmetry to the overall design—in addition to the rotational symmetry of the inner pattern—or as above, to complete the boundaries and produce a composite symmetry: a square symmetry for the underlying configuration, a vertical symmetry for the loops and a rotational symmetry for the joints. As before, the shape rules apply to shapes composed of arcs and straight lines under similarity transformations. The two new rules are shown in Fig. 23a, a matching of one of the two rules in the evolving design is shown in Fig. 23b, and a design after the execution of four rules is shown in Fig. 23c.

The last stage of the Celtic knot grammar modifies the intersections of the loops to produce shapes that begin to capture the spirals, step patterns, and key patterns that are characteristic in the composition. Two additional rules are introduced here to show this possible editing of the grammar and the disentanglement of the knot. Both rules are symmetrical and reverse copies of each other each undoing what the other is doing, i.e., the one untying the knot and the second tying it back reversing its action. The lengths of the strings of the knots (the two pairs of parallel lines denoting the ends of the two strands of the knot) at the *LHS* of the rules are specified by an area selection action on the model itself to capture the area of intervention of the designer. These subshapes, once captured in the design model, are literally carried on the *LHS* of the rule space to be acted upon. The calculation, as before in this

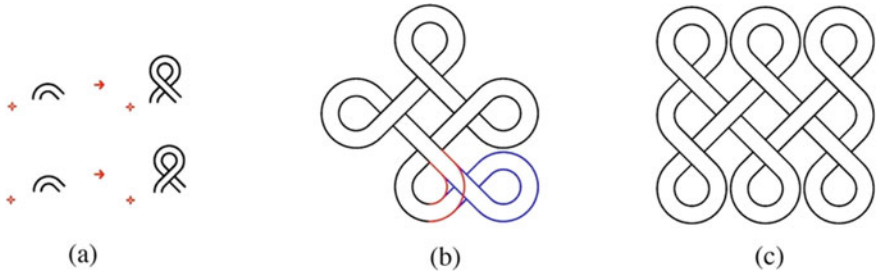

Fig. 23 Celtic knot grammar: Fabric stage

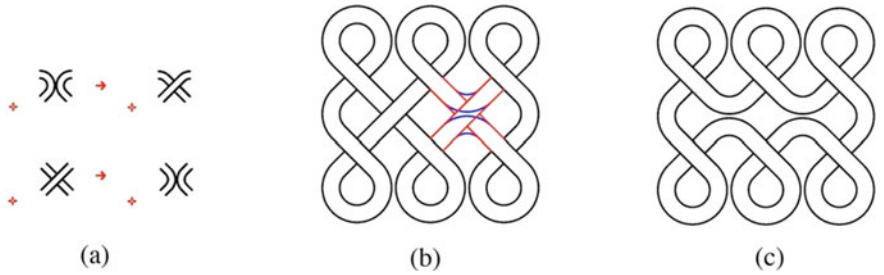

Fig. 24 A Celtic knot grammar: Modifying stage

series, is limited in matchings under similarity transformations. The two new rules are shown in Fig. 24a, a matching of one of the two rules in the evolving design is shown in Fig. 24b, and a design after the double execution of the untying rule is shown in Fig. 24c.

4.2 Mechanical Design: The Gear Grammar

A mechanical design application to generate a gear is presented in the form of a shape grammar implemented in the Shape Machine. The grammar consists of four rules: The first shape rule draws an initial circle defining the outer radius of the gear. The second shape rule offsets a 24-gon to define the inner radius of the gear. The third shape rule subdivides the outer radius to specify twelve wedged parts. Finally, the fourth shape rule specifies the design of a tooth at the outer boundary of the wedge. The mechanical gear is generated by the serial application of shape rules 1–3 one at a time, and the parallel application of shape rule 4 to all twelve parts of the design. Significantly, these rules are grouped and executed in the *DrawScript* mode of Shape Machine, an automated setting that executes rules under specifications given by the designer. The pictorial template allows for the visual specification of the shape rules along with numerical specification for the number of loops and the types of transformations under which the rules apply. When the visual program runs, the rules are applied sequentially to: (1) draw an initial circle defining the outer radius of the gear; (2) offset a second circle to define the inner radius of the gear; (3) subdivide the outer radius to specify twelve parts; and (4) generate teeth at each of the twelve parts. The fourth rule of the gear grammar is shown in the template of the *DrawScript* mode of the Shape Machine in Fig. 25a. One of the matchings of the fourth rule of the grammar that substitutes one of the twelve sectors of the circle with the pentagonal polyline that outlines the tooth of the gear is shown in Fig. 25b. The complete automated generation of the gear in Shape Machine is given in Fig. 25c.

At any time during the design process, the designer can intervene in the *DrawScript* mode to change a shape rule and create a variation of the design. Here, the last shape rule that specifies the end of the gear is slightly edited to substitute its straight end

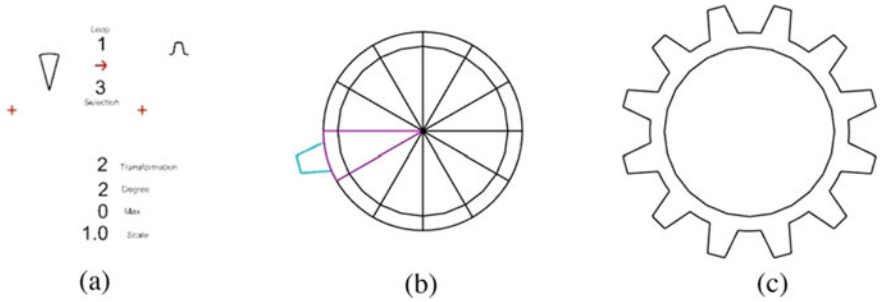

Fig. 25 An automated specification of a mechanical gear in the *DrawScript* mode of the Shape Machine

with an arc end to specify a second design variation. The editing of the profile takes place in real time with the same visual modeling tools of the application. Once the editing of the individual rules is done, the complete set of rules is selected and the program reruns to produce the final design. The edited rule that changes the profile of the tooth is shown in Fig. 26a. One of the twelve matches of the last rule of the gear grammar is shown in Fig. 26b. The new generated variation of the gear is given in Fig. 26c.

The variation on the design of the gear can be further expressed visually by adding new rules, subtracting rules or modifying rules. Here, a structural variation is given by editing the last rule of the template and the substitution of the pie-shaped part of the circle with a polyline composed of seven lines to produce the profile of two teeth. Note that these changes are all done visually, they do not require any programming skills and provide a parametric design agency to the author or user of the grammar. This edited rule is shown in Fig. 27a. One of the twelve matches of the last rule of the gear grammar is shown in Fig. 27b. The new generated variation of the gear with the double order of symmetry, a dihedral symmetry of order 24, is given in Fig. 27c.

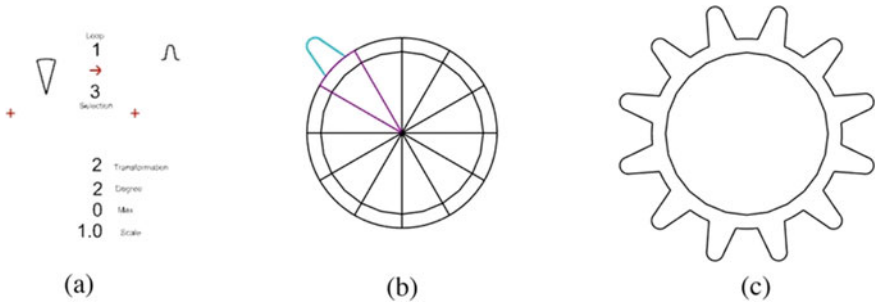

Fig. 26 Editing of the profile of the teeth of a gear in the *DrawScript* mode of the Shape Machine

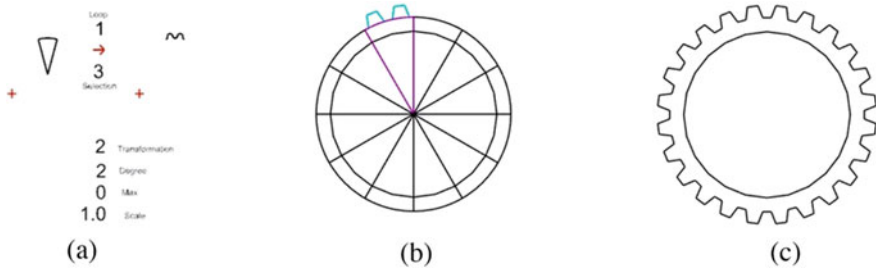

Fig. 27 Editing of the number of teeth of a gear in the *DrawScript* mode of the Shape Machine

4.3 Architectural Design: Portm-Ino

An architectural design application to generate the house plan of a unique private residence is presented in the form of a shape grammar implemented in the Shape Machine. The house, *Entelechy I*, designed by the architect John Portman back in the sixties, holds a unique position in the oeuvre of his office because of the belief of the architect that it encapsulated in some hybrid form the architectural elements of his language that were popularized in his immensely successful commercial architecture starting in the seventies and continued to this day (Portman and Barnett 1976). The generation of the plan of the house is presented in the form of a shape grammar in three stages that roughly correspond to three stages of design and the consolidation of an abstract idea into an architectural plan (Ligler and Economou 2019). The grammar is called Portm-Ino grammar to suggest how the house in Portman's conception is a systematic, residential configuration along the lines of Le Corbusier's Dom-Ino framework.

The first stage of the shape grammar, *Framework*, establishes a basic tartan grid that enables the underlying division of the house in major and minor spaces and the delineation of compositional enfilades to orchestrate the points of centrality within the house. A variety of $n \times m$ grids may be generated in the style and all can be used to be encoded with the layered tartan grid. One of the rules in this stage substituting a module of the underlying grid to a stellated aggregation of five cells with a major cell in the middle and four cells in the diagonals—the basic formation of the major–minor space—is given in Fig. 28a. One of the nine matches of this rule is shown in Fig. 28b. The new generated variation of the tartan grid plan with the nine major spaces is given in Fig. 28c.

The second stage of the grammar, *Configuration*, establishes the major organizational scaffold of the house with its functional division in public and private zones, entries, and its visual articulation in terms of the major and minor cells in terms in the form of an underlying grid punctuated by circular spaces. One of the rules in this stage substituting the boundary of a minor cell with a circular room—or in Portman's words, a hollow or exploded column—an architectural scaffold for lightwells, stairwells, closets, studies, libraries, bathrooms, and micro-galleries, is given in Fig. 29a. One of the sixteen matches of this rule is shown in Fig. 29b. The new generated

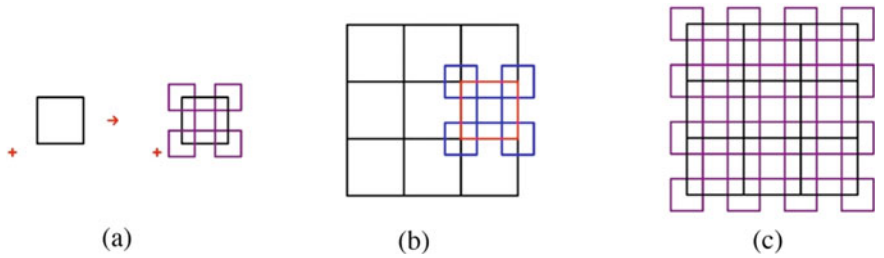

Fig. 28 Portm-Ino: Framework stage

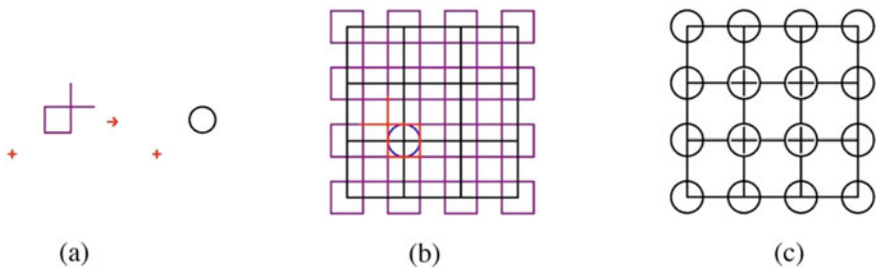

Fig. 29 Portm-Ino: Configuration stage

variation of the gridiron plan with its characteristic arrangement of an underlying grid and circular spaces is given in Fig. 29c.

The third stage of the grammar, *Architectonics*, completes the generation of the plan. This stage consists of shape rules that encode the conventions characterizing the articulation of an architectural plan and includes rules for the delineation of double-height living spaces (denoted by an x in the plans), walls, openings, staircases, bridges, and so forth, to fully disambiguate the design in terms of room of public rooms, living rooms, bedrooms, stairwells, closets, studies, libraries, half bathrooms, and micro-galleries. One of the rules in this stage showing the substitution of a hollowed column with a fully articulated staircase is shown in Fig. 30a. One of the three possible matches of the rule is shown in Fig. 30b. The complete automated production of a possible 3×3 gridiron plan in the language is shown in Fig. 30c.

5 Discussion

A brief survey of the expressive power of Shape Machine has been presented. The work was presented in two parts: A brief presentation on the implementation of the elusive shape computations that have been envisioned by the academic community since the seventies—and still used as the testbed for the tasks a shape grammar interpreter should be able to accomplish (Gips 1999; Stiny 2006). And a second

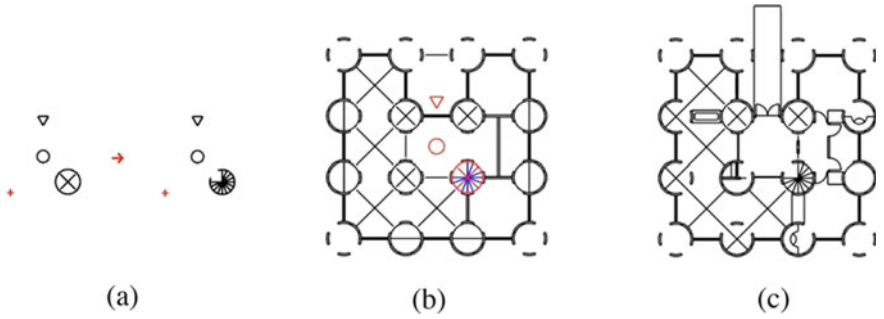

Fig. 30 Portm-Ino: Architectonics stage

part on the exploration of applications in various design domains, namely product modeling, mechanical and architectural design, to suggest possibilities for new design workflows. Clearly, these applications were only given as samples of transformative possibilities rather than as actual design applications. A more generous plateau of possible application domains including, and not limited to, origami structures, archaeology reconstructions, protein folding, virus recognition and more, is currently under development.

The future of the Shape Machine appears bright indeed: It has succeeded to provide a coherent framework to do the elusive computations that no other technology has been able to do so far; and it has managed to do so convincingly in run time promising the possibility of a real application that can indeed revolutionize design thinking and practice. The future steps are already within reach: three-dimensional geometry; *b*-rep representation; solid modeling; larger datasets of geometry primitives; implementation of general transformations for both identity and replacement rules. Some of these questions pertain to practical matters deploying the existing algorithms behind the Shape Machine into new tasks, scales, and dimensions. Others point to the solution of still new theoretical aspects of rule recognition and replacement for all sort of shapes under different transformations.

Current development in Shape Machine aims at the completion of shape rule recognition and implementation algorithms for lines and arcs under the complete Erlangen program, that is, under affine and projective transformations too. The unification is very promising: Existing modules of the Shape Machine, not presented here, suggest intuitive spatial searches in the *Napkin Draw* mode, whereas designers are able to query shapes under compress, shear, and perspectivity transformations inquiring and performing on shape type definitions according to essential characteristics defined on spot (Mitchell 1992). Queries including quick sketches of polygons, and/or arrangements of lines and arcs rather than precise constructions as outlined in this paper are within reach and the line and arc shapes that are used as queries or targets are expanded to include ellipses and conics at large. The work so far promises to completely revolutionize the current work on the Shape Machine and its rule implementations in the Euclidean space because shape computations under affine and projective transformations produce shapes that do not share any more the same

conventions with the ones that started the computation; after all, a square is a square no matter if it is rotated, reflected, diminished, or enlarged. But drawing a quadrilateral in the Identity mode (seeing) and having the Shape Machine finding, say, squares or rhombi under affine or projective transformations, respectively, requires some more reworking on behalf of the designer who works more with language (symbolic descriptions) rather than shapes.

But more than those fronts, the work on the Shape Machine front so far has opened two completely new trajectories not fully appreciated in the setting of the inquiry. The one is the work on visual scripting and the *DrawScript* mode. The ability to learn to script through drawing, the interface with loops, what-ifs, control structures and all, suggest an unprecedented environment for teaching computing and interfacing with computers. Sorting operations are important algorithmic constructs in programming, and they can be done in a straightforward manner in Shape Machine: Instead of writing code to sort binary digits, Shape Machine allows users to visually program this logic by considering the symbols 0 and 1 as shapes drawn in the shape rules: $10 \rightarrow 01$ or $01 \rightarrow 10$. The first rule when applied multiple times over a string of shapes made up of 0 s and 1 produces the so-called big-endian order consisting of all 0 s to the left and 1 s to the right. The second rule when applied multiple times over a string of shapes made up of 0 s and 1 s produces the so-called little-endian order consisting of all 1 s to the left and all 0 s to the right. But more than that, these shape rules can be used with additional shape rules to demonstrate that numerical calculation can be done visually! Figure 31 shows an example of such a visual calculation: In this example, a single-digit adder is implemented using 29 shape rules accomplishing three different main operations: (a) translating the numerals (1, 2, 3, …, 9) to numbers of 1 s (1, 11, 111, …, 111111111); (b) moving all the 1 s together; and (c) translating numbers of 1 s (1, 11, 111, …, 111111111) to numbers (1, 2, 3, …, 9, 10). The concept of addition, moving all the 1 s together, is commonly used in a digital environment. Shape Machine can process this operation visually with the simple shape rule discussed above in the visual sorting to achieve an adder. Two of the 29 rules are shown in Fig. 31a. The initial shape denoting an addition of two numbers (7 + 9) is shown in Fig. 31b. Two of the several visual substitutions of the operation and the final result are shown in Fig. 31c. Not that in all these computations, the symbol 7 is treated a shape consisting of two lines, the symbol + by two lines and the symbol 9 by a circle and a line.

Fig. 31 A visual calculator: Toward a new Turing Machine

This example is a demonstration that implementing a computer with a shape grammar is very possible in theory. And with that, a much deeper and farfetched question: Can we implement a visual computer and how different would it be? Wittgenstein came close to an arithmetic of shape in his remarks to the Foundation of Mathematics but in the end he discarded the possibility as an idle speculation (1956). We strongly believe that the insight this technology can bring will be invaluable to the things we do, but even more, to things we have not dreamed of yet.

Acknowledgements This work has been partially supported with grants by the NSF/I-Corps Site and the Digital Building Lab (DBL) at Georgia Institute of Technology. We would like to thank Pr. Josephine Yu and her students Cvetelina Hill, Nicholas Liao and May Cai at the School of Mathematics at Georgia Institute of Technology, for their valuable insights and advice on the project.

References

Bain, G. (1975). *Celtic art: The method of construction.* London: Lewis Reprints Ltd.

Chau, H. H., et al. (2004). Evaluation of a 3D shape grammar implementation. In J. S. Gero (Ed.), *Design computing and cognition'04* (pp. 357–376). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Earl, C. F. (1997). Shape boundaries. *Environment and Planning B: Planning and Design, 24,* 669–687.

Economou, A., & Kotsopoulos, S. (2014). From shape rules to rule schemata and back. In J. S. Gero & S. Hanna (Eds.), *Design computing and cognition DCC'14* (pp. 419–438). Springer.

Economou, A., & Grasl, T. (2018). Paperless Grammars. In J.-H. Lee (Ed.), *Computational studies on cultural variation and heredity* (pp. 139–160). Springer, Singapore: KAIST Research Series.

Economou, A, & Park, J. (Forthcoming). A pictorial catalogue of *n*-line shapes for *n.*

Economou, A., & Yu, J. (Forthcoming). Shape signature.

Eloy, S., Pauwels, P., & Economou, A. (eds). (2018). Advances in implemented shape grammars: solutions and applications. *AIEDAM Special Issue: Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 32*(2).

Fletcher, R. (2004). Musings on the Vesica Piscis. *Nexus Network Journal, 6*(2), 95–110.

Grasl, T., & Economou, A. (2013). From topologies to shapes: Parametric shape grammars implemented by graphs. *Environment and Planning B: Planning and Design, 40*(5), 905–922.

Gardner, M. (2001). *"Rep-tiles", The colossal book of mathematics: Classic puzzles, paradoxes, and problems* (pp. 46–58). New York: W. W. Norton.

Gips, J. (1999). Computer implementation of shape grammars. *Workshop on Shape Computation*, MIT.

Hong, T. K., & Economou, A. (Forthcoming). Shape machine.

Hong, T. K., & Economou A. (Forthcoming). Five criteria for shape grammars interpreters.

Jowers, J., & Earl, C. (2011). Implementation of curved shape grammars. *Environment and Planning B: Planning and Design, 38*, 616–635.

Knight, T. (1994*). Transformations in design: A formal approach to stylistic change and innovation in the visual arts*. Cambridge University Press.

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

Ligler, H., & Economou, A. (2019). From drawing shapes to scripting shapes: Architectural theory mediated by shape machine. In *Conference Proceedings of the Symposium on Simulation for Architecture and Urban Design (SimAUD)* (pp. 278–286).

McKay, A., Chase, S., Shea, K., & Chau, H. H. (2012). Spatial grammar implementation: From theory to useable software. *AI EDAM-Artificial Intelligence Engineering Design Analysis and Manufacturing, 26*(2), 143–159.

Mandelbrot, B. (1982). *The fractal geometry of nature*. Times Books.

Mitchell, W. (1992). *The logic of architecture: Design, computation, and cognition.* Cambridge: MIT Press.

Portman, J., & Barnett, J. (1976). *The architect as developer*. New York, NY: McGraw-Hill.

Pottmann, H., Asperl, A., Hofer, M., & Kilian, A. (2013). *Architectural geometry*. Bentley Institute Press.

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. (1986). A new line on drafting systems. *Design Computing, 1,* 5–19.

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

Stiny, G. (2006). *Shape: Talking about seeing and doing*. Cambridge, MA, USA: MIT Press.

Wittgenstein, L. (1956). *Remarks on the foundations of mathematics.* Oxford: Basil Blackwell.