The Fifth International Conference on Design Creativity (ICDC2018)
Bath, UK, January 31st – February 2nd 2018

SOME ISSUES CONCERNING THE USE OF DESIGN SPACE FORMALISMS IN CREATIVE DOMAINS

Alexandros Charidis

Department of Architecture, Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract: In this paper, an analytical examination of two types of classical design space formalisms is presented. Compositional aspects in each type of design space are examined in terms of the units and the rules given to synthesize descriptions of designs. A uniform treatment of the technical concepts underlying both types is developed based on shapes and on computations defined in terms of shapes. Issues concerning the use of both types of design space formalisms for synthesis tasks in creative domains are discussed. Alternative directions are outlined that reframe classical conceptualizations of design spaces in ways that emphasize the open-ended, improvisational nature of creative work in design.

Keywords: *Design space, shape grammars, creative reasoning, design description*

1. Introduction

The characterization of the *act of synthesis* in design as a process of search within a *design space* of alternative possibilities is long established. Elements of the idea originate in psychology but were popularized in design theory mostly due to the pioneering work of Newell & Simon (1972) in artificial intelligence. Much of subsequent work on the notion of a design space follows trajectories they established; the study of representation spaces for architectural configurations and their transformations (March & Steadman, 1978); the development of computational theories of scientific discovery and creative invention based on mechanisms of heuristic search in problem spaces (Simon, 1977; Langley et al., 1992); computer tools for automated exploration of design spaces in architecture, engineering spatial design and like areas, summarized by Cagan et al. (2005) and discussed in the collection of papers in Stouffs (Eds., 2006). Despite the central place that the notion of a design space has for computation in design, little investigation exists on design spaces themselves. What types of design spaces (as formal, mathematical objects) can be distinguished? Where do the atoms and constituent units that make up descriptions of alternative designs in each type come from? How do they change or stay untouched as a consequence of the rules given to synthesize alternative designs? Under what terms is a designer participating in the computations of the alternatives in each type of design space? Such questions are of significant importance for studies aiming at understanding aspects of creativity, invention, or discovery in design through the notion of a design space.

In this paper, I present an analytical examination of two types of classical design space formalisms widely used in architecture, engineering design and associated areas of spatial design. In Section 2, I develop a prototypical exposition of each type. The technical concepts underlying both types are presented in a uniform manner, following the mathematical treatment of shapes in the latest monograph on the subject of *shape grammars* (Stiny, 2006). Both types of design space formalisms present a number of issues when used for synthesis tasks in creative domains. I discuss these issues in Section 3 and I outline alternative directions that reframe classical conceptualizations of design spaces in ways that emphasize the open-ended, improvisational nature of creative work in design.

2. Two types of classical design space formalisms

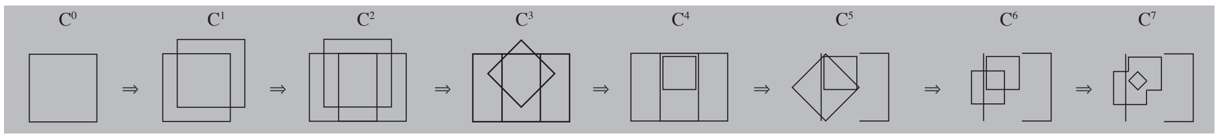
Consider the computation in Figure 1 (a) and the highlighted shape "C^3". It should suffice for the moment to ignore the precise rules that generate the designs in the figure and to instead imagine that this computation represents the work of an architect, artist, or composer – in the broadest sense of the word – who creates one design out of another in a fluid, improvisational manner. In the following two expositions, a design space is specified in which the selected shape C^3 is a closed member. A design space is considered here in a technical sense; a design space is a certain formal, mathematical construct. The two expositions differ in terms of the *type* of the design space examined. In the first type, a design space is a multidimensional, Cartesian space subject to geometric interpretation; descriptions of designs in the space (set) are represented as vectors over a fixed set of numerical variables. In the second type, a design space is a set of discrete states (a state is a composition of logical atoms) and a set of compositional rules over the states. For simplicity, I call *Type 1* and *Type 2* the design spaces constructed with the first and second approach respectively.

2.1 Type 1

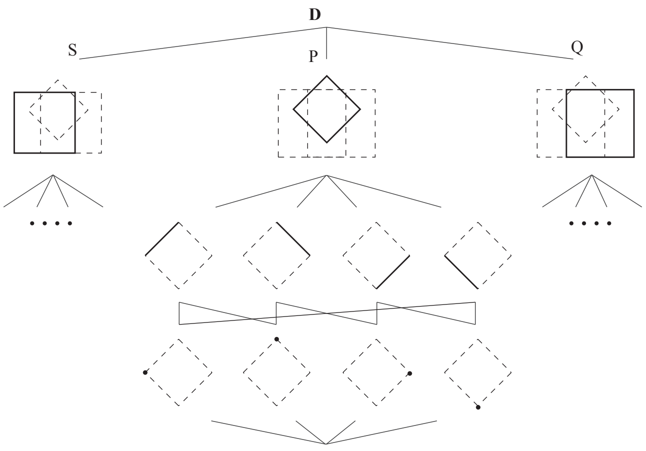
"What is the description of shape C^3?" At first, C^3 is only a *picture* without analysis into definite parts. The following verbal description of C^3 establishes a *spatial relation* between basic elements: "*A design made of three squares where one is rotated by 45 degrees around its centre*; *each square is made of four lines and each line is made of two (end) points*." The lattice diagram in Figure 1 (b) is the visual companion of this verbal description. Three basic elements make up shape C^3, namely, the squares labelled with letters S, P and Q (see figure for exact correspondences). Each square is represented as a set of four lines and each line is represented as a set of two points (The lattice shows the hierarchical decomposition of P; the decompositions for S and Q follow in a similar way). Let the following set D recapitulate these *atomic* elements in a set-theoretic format: $D = \{(S, L), P, Q\} = \{((s_0, s_1), (s_1, s_2), (s_2, s_3), (s_3, s_0), \{s_3, L\}), ((p_0, p_1), (p_1, p_2), (p_2, p_3), (p_3, p_0)), ((q_0, q_1), (q_1, q_2), (q_2, q_3), (q_3, q_0))\}$, where s_i, p_i, q_i are two-parameter points that mark the boundaries of lines, where the couple $\{s_3, L\}$ is a labelled point and works as a coordinate frame for the shape (see Figure 1, c). Shape C^3, initially given as a picture, is now casted into a collection of three ordered sets of points – the shape has become merely what its description tells.

The construction of a space in which C^3 – or more precisely, its description – is a closed member follows. Let the highlighted points in Figure 1 (c) represent six *design* or *decision variables.* Each variable represents an axis of a geometric, Cartesian space where each point (or vector) in the space is a particular combination of values for the variables. This space, sometimes called a parametric design space, is one of the basic mathematical objects used in any task of optimization in economics and engineering (Simon, 1977; Fensel, 2000; Cagan et al., 2005; Radford & Gero, 1988). In optimization, the sole purpose is to find points (designs) in the space with combinations of values that maximize, or nearly maximize, the numerical index of a utility function of the design variables. Informally, a designer (a human designer or an automated one) goes through the alternatives in the space in some order, evaluates each alternative in terms of a utility function and either selects the alternative(s) if it happens that it maximizes the index of the function, or continues to search for other alternatives. Fensel (2000) provides a combined treatment of parametric design spaces and the "generate-and-test" paradigm in artificial intelligence; Radford & Gero (1988) cover parametric design spaces for structural engineering of buildings.

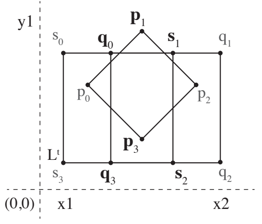
In this exposition, however, my interest is neither in rational choice among given alternatives, nor in algorithms that would make such choices practicable. Instead, the goal is to examine the *compositional terms* under which a designer participates in the calculations of the alternatives in a design space of Type 1 – what would a designer do to see "what designs are there in the space?" Designs in the space are represented in a uniform manner; each description of design is composed out



a.



b.



c.

Description

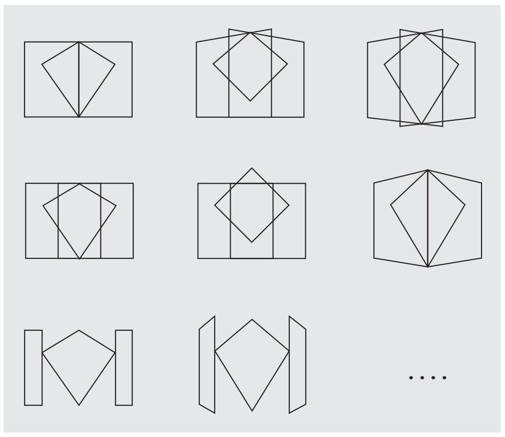
“A design made of three squares where one is rotate by 45 degrees around its center; each square is made of four lines; each line is made of two (end) points”

$$\mathbf{D} := \{\{S,\ L\},\ P,\ Q\} := \{\ \{((s_0,\ s_1),(s_1,\ s_2),(s_2,\ s_3),(s_3,\ s_0)),\ (s_3,\ L^t)\},\ ((p_0,\ p_1),(p_1,\ p_2),(p_2,\ p_3),\ (p_3,\ p_0)),\ ((q_0,\ q_1),(q_1,\ q_2),(q_2,\ q_3),(q_3,\ q_0))\ \}$$

Design Variables	Constraints			
$\mathbf{p}_{1y} \longleftarrow y$	$\mathbf{s}_{1x} \geq \mathbf{p}_{0x}$	$0 \leq \mathbf{p}_{1y} \leq y1$	$x1 \leq \mathbf{q}_{0x} \leq x2$	$\mathbf{q}_{0x} == -\mathbf{s}_{1x}$
$\mathbf{p}_{3y} \longleftarrow y$	$\mathbf{s}_{2x} \geq \mathbf{p}_{0x}$	$0 \leq \mathbf{p}_{3y} \leq y1$	$x1 \leq \mathbf{q}_{3x} \leq x2$	$\mathbf{q}_{0y} == \mathbf{s}_{1y}$
$\mathbf{q}_0 \longleftarrow (x,y)$	$\mathbf{q}_{0x} \leq \mathbf{p}_{2x}$	$0 \leq \mathbf{q}_{0y} \leq y1$	$x1 \leq \mathbf{s}_{1x} \leq x2$	$\mathbf{q}_{3x} == -\mathbf{s}_{2x}$
$\mathbf{q}_3 \longleftarrow (x,y)$	$\mathbf{q}_{3x} \leq \mathbf{p}_{2x}$	$0 \leq \mathbf{q}_{3y} \leq y1$	$x1 \leq \mathbf{s}_{2x} \leq x2$	$\mathbf{q}_{3y} == \mathbf{s}_{2y}$
$\mathbf{s}_1 \longleftarrow (x,y)$	$\mathbf{p}_{3y} \leq \mathbf{q}_{3y}$	$0 \leq \mathbf{s}_{1y} \leq y1$		
$\mathbf{s}_2 \longleftarrow (x,y)$	$\mathbf{p}_{3y} \leq \mathbf{s}_{2y}$	$0 \leq \mathbf{s}_{2y} \leq y1$		

d.

Set of alternative designs



e.

Figure 1. Definition of descriptions of designs in a design space of Type 1.

of a common set of design variables. The participation of a designer in the calculations of the alternatives is solely restricted to what changes can be performed to the values of the design variables. Thus, given a design C in the space, in order to create (calculate) another design C’ also in the space, a designer would perform one and only action: assign new values to the variables of C. Let r1: A → *g*(A) be a *value assignment* rule, shown in Figure 2, where A is a parametric shape represented as a tuple of four labelled points (a_i | i = 0, …3). In addition, a series of constraints are defined for each design variable, summarized in Figure 1 (d). The following are specified for each variable: (i) free and fixed axes; (ii) symmetry relations with other variables; and (iii) ranges of possible values as inequality expressions. Function *g*, in the definition of rule r1, is an *assignment function*. It works in the following manner. Say that r1 is applied to square P in Figure 1. First, labelled points in the left side of the rule are *binded* to labelled points in P as follows: a_0–p_0, a_1–p_1, a_2–p_2, and a_3–p_3. Then, new values are chosen that satisfy constraints (Figure 1, d). A constraint can be expressed as a *predicate*, for example, in the following form: "p_{1y} and p_{3y} must be between 0 and y_1, and p_0 and p_2 must be fixed in place." On values that satisfy this predicate, *g* changes the points in P accordingly and a new shape is obtained. A subtle point worth mentioning is that rule r1 applies to labelled points whenever the labels are from the *same* alphabet. It cannot apply to a shape that is labelled with symbols coming from two different alphabets, such as the shape labelled with symbols s_0, q_1, q_2, s_3 (see Figure 1 (c)). This condition makes clear that rule r1 can be applied to either one of the three individual shapes in the set {S, P, Q} (under an identity relation) but to no other shape or subshape perceived otherwise.

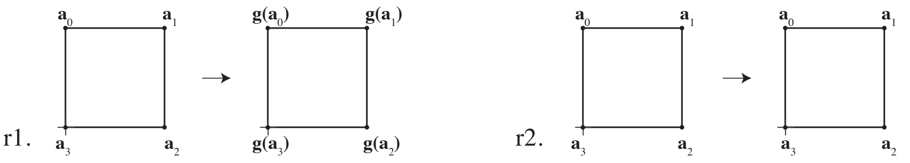


Figure 2. Rule r1 is an assignment rule A → *g*(A) and rule r2 is an identity rule A → A, where A is a parametric shape made of four variables and *g* a function of these variables.

A computation with rule r1 is shown in Figure 3. In each step: a transformation *t* makes the left side of r1 a *subset* of the current design, new values are assigned to change the shape "matched" in the rule application, and a new shape C' is obtained in return. We go from one combination of values to another combination of values in mechanical fashion:

$$\{c_1, c_2, c_3\} \Rightarrow \{g(c_1), g(c_2), g(c_3)\} \Rightarrow \{ g(g(c_1)), g(g(c_2)), g(g(c_3))\} \Rightarrow \ldots$$

A catalogue of designs created in this manner is shown in Figure 1 (e). All designs calculated with rule r1 share a common underlying compositional structure. Notice in Figure 3 that this structure corresponds precisely to the description of shape C^3 given in the beginning of this exposition – this structure is carried throughout untouched, with no new units added or existing units removed. The way this computation works should be reminiscent of parametric modelling software where one indirectly controls the generation of designs via preselected structures, which are well thought in advance.

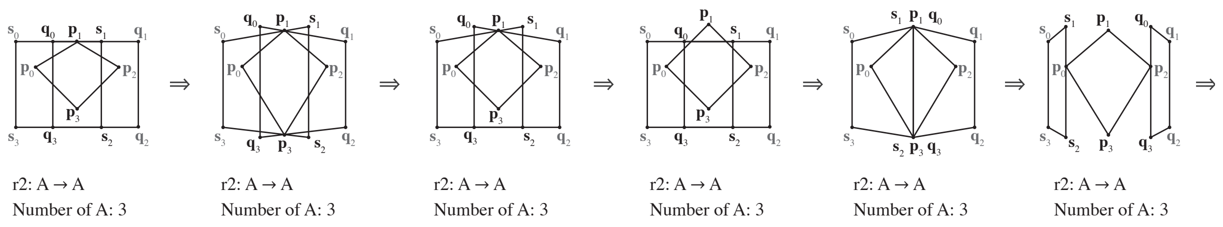


Figure 3. State (shape) transitions and counting of atoms (occurrences of left shape A in rule r2) in the description of each state using identity rule r2.

The numerical index at the bottom of each shape in Figure 3 explains the "conservation" of units of description from one step of the computation to another in a simple way. Let r2: A → A be the identity rule shown in Figure 2. When rule r1 is applicable to a shape, rule r2 is applicable too, under the same transformation *t* (in consequence of having the same left shape). But rule r2 is used as an *observational device*, to monitor the structural changes caused by rule r1: when r1 is applied to C to create C', rule r2 is applied exhaustively to both C and C' to count the number of occurrences of the left shape A in the structure of both shapes. As the figure shows, all shapes have equal counts.

Two properties of design spaces of Type 1 are important to emphasize. Given a design C in the space, rule r1 generates another design r1(C) = C', which is also in the space. This forms a *closure property* of the space with respect to rule r1. From a designer's perspective, that is to say from a creational standpoint, the notion of "closure" is meant to capture an intuitive assumption: what can or cannot be calculated (created) by a designer is implicitly decided by the given rules and the descriptions of designs to which these rules strictly apply. When we use rule r1 to alternate between different alternatives, we are merely "discovering" what is already "closed" in the space. The second property is a *continuity property* between descriptions of designs in the space. The design variables established earlier act as the parameters of function *g* in the definition of rule r1. This function and the generated set of designs form together a continuous transformation group (see Seshadri & Na (1985) for a presentation of such groups): the set of designs generated by r1 forms a group (i.e. the rule acts as the "operation" defined between the members of the set), the members of the group can be identified by the same set of continuous parameters, and *g* is a continuous function of these parameters.

One could argue that the two properties, closure and continuity, are different versions of the same concept, or that one somehow implies the other. However, the distinction is especially important. For in the following paragraphs we explore computations where, although the property of closure is assumed in the formal systems employed, continuity not only acquires a different flavour from the one found here, but in certain cases, continuity is not even exhibited in the first place.

2.2 Type 2

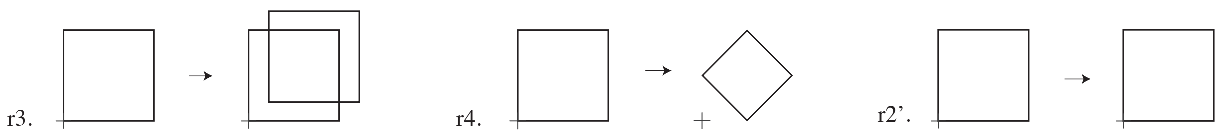


Figure 4. Three rules.

A design space in which shape C^3 is a closed member is constructed by following a very different line of thought. Consider the three compositional rules of Figure 4. Rule r3 is of the form $A \rightarrow A +_s g_1(A)$ and rule r4 is of the form $A \rightarrow g_2(A)$, where g_1 is a translation and g_2 a rotation augmented with uniform scale. Rule r2' is an identity rule very similar to rule r2 (Figure 2) with the difference that its left side is an *atom* – a unit which cannot be divided into parts – while the left side of r2 is a shape made of points with coordinate values[1]. To obtain a description for shape C^3, rule r2' is applied to C^3 exhaustively in the manner shown in Figure 5. This forms a decomposition of C^3 into the following set of shapes $D = (\emptyset, S, P, Q, \{P, Q\}, C^3)$, which includes the empty set and the shape itself as closed members. The bottom shapes S, P, and Q in Figure 5, are the atoms of this decomposition for C^3, the constituent units out of which all other members are composed of.

In the literature of artificial intelligence, there is a large family of formalisms known as rule-based systems, adapted from logic for knowledge representation and reasoning (Newell & Simon, 1972; Nilsson, 1982). Rule-based systems have two core components, namely, a set of compositional rules and an initial set of true assertions, sometimes called a database or "knowledge base." Such systems are meant to imitate logical implication in the most general way for the widest possible spectrum of applications (Ligeza, 2006). The setup we have so far with the system of rules in Figure 4 works very much along this tradition of rule-based systems. Although rules are defined in terms of shapes they are treated, as is shown below, as symbols and logical conjunctions of them imitating in effect the behaviour of classical rule systems for symbols coming from specified alphabets. For a more detailed account of the relationship between rule-based systems and grammar formalisms, including the shape grammar formalism, see Gips & Stiny (1980).

[1] The two identity rules are in a sense equivalent. Shapes represented as point sets and shapes as atoms belong to the zero-dimensional algebra U_0 and behave like symbols and strings (Stiny, 2006).

A design space of Type 2 is generated by the system of rules given in Figure 4, starting from an initial shape, namely, the shape C^3. Rules r3 and r4 are applied recursively to synthesize designs. Three series of computations are shown in Figure 6 where, in each series, shape C^3 is the start shape and rule r3 is applied first. In parallel to rules r3 and r4, identity rule r2' is used as an observational device to monitor the structural changes that both rules cause to the compositional structure of designs. Whenever either one of the two rules is applied to shape a C to create a new shape C', rule r2' is applied exhaustively in both C and C' to count the number of occurrences of its left shape in the structure of both shapes. We define the following method for counting. If r a rule then a *rule evaluation* r_C is a mapping r_C: $C \rightarrow \{0, 1\}$ from parts of C to labels 0 (or "false" in some way) and 1 (or "true" in some way). When the rule is applicable to some part c_i of C, then r_{Ci} evaluates to 1, otherwise r_{Ci} evaluates to 0. The number of times a rule applies to shape C is the sum of rule evaluations for which $r_{Ci} = 1$ that is: $\sum (r_{Ci} = 1 \mid \text{for every part } c_i \text{ of } C)$. This formula is used in combination with rule r2' to count the number of occurrences of its left shape in the structure of the shapes of Figure 5.

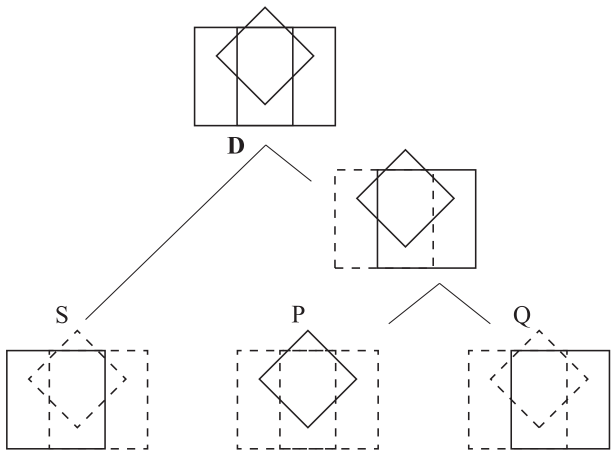


Figure 5. Repeated applications of identity rule r2'. The resulting set of parts D = {S, P, Q} is a decomposition of the given shape into three mutually exclusive shapes.

On a more detailed level, in computations (1) and (3) the counts stay constant, matching precisely the situation we encountered previously in Figure 3 where the structure assigned to the initial shape C^3 is carried throughout untouched; the rules merely transform existing parts without adding or removing anything. In computation (2), the counts increase monotonically; if C and C' are two consecutive shapes, then |C| is strictly less than |C'| as a consequence of applying an additive rule, namely, rule r3 ($| \cdot |$ represents the number of parts in the description of a shape). Informally, this implies that when going from shape C to the newly created shape C', new parts are added to C while no existing part is removed; if new parts are identical to existing ones they merge, in all other cases the two groups of parts remain mutually exclusive to one another. More generally, suppose when r is applied to C, *k* new parts are added to obtain C'. Then the following mapping describes this additive relation between C and C':

$$f\colon (c_1, c_2, \ldots c_n) \rightarrow (c_1, c_2, \ldots c_n) +_s (c_{n+1}, b_{n+2}, \ldots b_{n+k}),$$ where $+_s$ stands for union of sets.

(Complementary to this, we could have a similar situation where the action of a rule removes parts from the shape it applies to. Suppose when r is applied to C, *k* parts are removed to obtain C'. Then the following mapping describes the *subtractive* relation between the parts of C and C':

$$f\colon (c_1, c_2, \ldots c_n) +_s (c_{n+1}, c_{n+2}, \ldots c_{n+k}) \rightarrow (c_1, c_2, \ldots c_n)$$

The two mappings correspond to additive and subtractive *spatial relations* 1 and 2 in Stiny, 1980).

We now turn attention to the properties of closure and continuity in the computations underlying the construction of design spaces of Type 2. Rules r3 and r4 are for this new type of design space the "operations" defined over the members of the set. In a similar way as before, a closure property of the space is defined with respect to rules r3 and r4 in the sense that if C is a shape then r3(C) = C' is another shape also in the space (same with r4). Conversely, if a shape C' is in the space, there exists a sequence of rule applications (combining r3 and r4) that transforms the starting shape C^3 to C', or in notation, $C^3 \Rightarrow \ldots \Rightarrow$ C'. The continuity property of the space is exhibited in the way descriptions of shapes change as a consequence of rule applications. In computation (1) when going from a shape C to another shape C', the two shapes have equal number of parts in their respective descriptions. The rule used to calculate C' out of C explains the "change," that is, rule r4 merely replaces a part of C with its transformed version without adding or removing anything. Thus, the effects obtained *follow* from the atoms and the structure of the rule applied. Similarly, in computation (2) when going from a shape C to another shape C', the number of parts increase. The rule used to calculate C' out of C explains the change, that is, rule r3 is an additive rule – again, the effects obtained follow from the rule used. In all three series, newly created designs have descriptions that *follow logically* from descriptions of past ones: for computations (1) and (3), descriptions of designs follow logically from past ones by correspondence; in computation (2), descriptions of designs follow logically from past ones by addition[2]. We call this concept of following logically as *logical continuity*. In the previous case, in a design space of Type 1, the space was formulated as a geometric, Cartesian space where the notion of continuity had a numerical flavour – numbers changing continuously to other numbers on fixed descriptions. By logical continuity an analogous concept is exhibited where descriptions of shapes change from one step of the computation to another in a causal manner: a description is logically derived from a previous description based on the atoms and the structure of the rule used to do so.

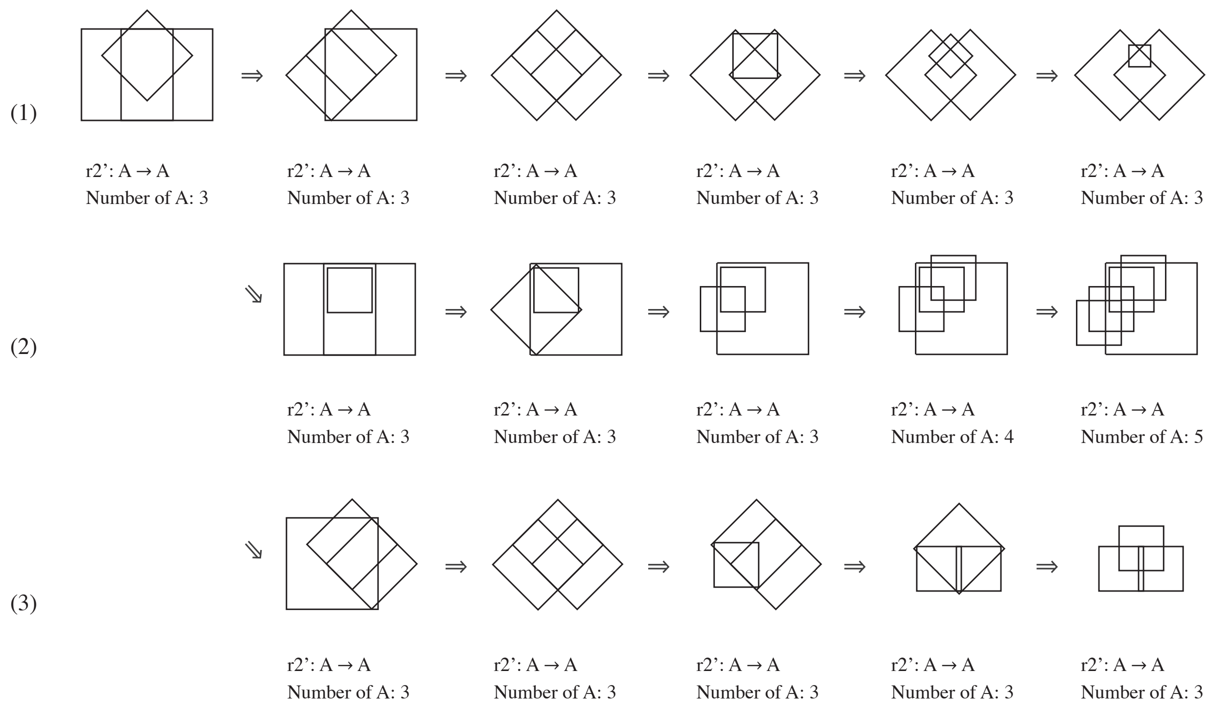


Figure 6. Designs created by applying rules r3 and r4. Rule r2' is applied in each shape repeatedly to count the number of occurrences of A in the structure of each shape. In all three cases, newly created

[2] This principle of following logically also applies to computations in which two consecutive shapes are related with the subtractive relation.

designs have descriptions that follow logically from descriptions of past ones. Paths (1) and (3) lead to consistent descriptions by correspondence; path (2) leads to consistent descriptions by addition.

3. Issues concerning the two types

Design spaces of Type 1 and 2, although based on different trajectories when we consider their technical details, express at their basis the very same idea: they both represent "containers" of end-results foreclosed in the terms of some specification given in advance. A specification for a design space of Type 1 is based on a priori definition of a set of parameters with associated numerical constraints; Type 2 is based on a set of fixed atoms and a set of compositional rules specified precisely in terms of these atoms. Hence, it is implicit in both Types of design spaces that in order for a designer to calculate designs, descriptions of alternative designs – the objects of invention – must be already closed in terms of the specification of the space. The natural question to ask, especially if we consider the design process in areas of design, such as architecture and the visual arts, is where does all this foreknowledge for "capturing" the future originate? Are there things to discover in the moment of the creative action that aren't captured by an antecedent specification?

3.1 New designs do not follow logically from past ones

In both types of design spaces, shapes were treated as symbols. What if shapes are treated as visual entities, as drawings on paper, without fixing their descriptions before calculations start? Consider once again the set of rules in Figure 4. A computation with rules r3 and r4 is shown in Figure 7 where rules apply *not* to descriptions of shapes but to visible-perceptible parts. With *visual shapes*, any part distinguished in observation is open to a rule application – there are no hidden or layered parts (the mathematical formalization of visual shapes, that is to say of shapes of dimension higher than zero, is discussed in full detail in Stiny, 2006).

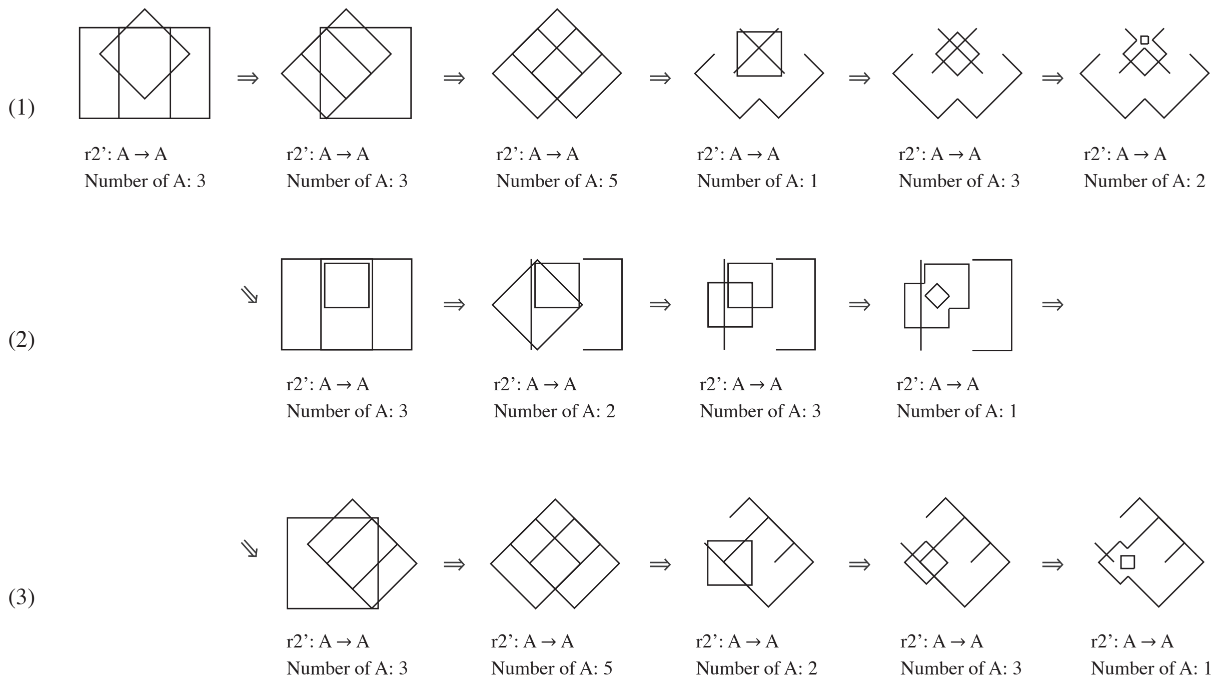


Figure 7. Designs created by applying rules r3 and r4. Rule r2' is applied in each state exhaustively to count the number of occurrences of A in the structure of the created designs. Descriptions of designs do not follow logically from descriptions of past ones; parts change from one step to another independently of the atoms and the structure of the rule that caused the change.

The new series of computations behave differently from the series in Figure 6. To see these differences, rule r2' (see Figure 4) is employed once more as an observational device along with the formula for counting defined in subsection 2.2 to monitor the changes caused in the compositional structure of designs. It is easy to see by the numbers and the visuals provided that the continuity property of design spaces of Type 2 is no longer exhibited: descriptions of newly created designs do not follow logically from descriptions of past ones. In particular, in computation (1), when going from a shape C to another shape C' the number of parts decrease or increase independently of the structure of the rule applied. The same holds for computations (2) and (3). Regardless if shapes were initially made by the very same parts (three squares when identity rule r2' is applied repeatedly), the rules do not comply only with these parts; different (emergent) parts appear manipulable at different moments in time, discovered by a participating designer as the computation goes by. *Where do the constituent units that make up descriptions of designs originate if not in the creative process itself?* The benefits should be clear, by shifting the "materials" of computation from symbols to visual shapes, we go from mere counting to "seeing" without marks of past specifications influencing in any way whatsoever our present – *improvisational* so to say – actions.

3.2 Notes on alternative characterizations of design spaces

An alternative, improvisational specification for design spaces is presented in Charidis (2017), where the open endedness offered with shape grammars when visual shapes are used for calculating is reconciled with the classical, formalistic pursuit for "closing" descriptions of designs in terms of some specification of a design space. In particular, as shapes are created improvisationally, the specification of the space that explains the descriptions of the generated shapes is worked out *backwards*, after recording the shapes unanalysed and specifying afterwards the atoms and the precise structures of the rules that yield them. By looking the history of generated designs backwards, multiple *computation histories* – and consequently, multiple specifications of design spaces – are crafted that explain the descriptions of the generated designs. The notion of a design space becomes in this manner a device for recapitulating what a designer chooses to do on the spot, as opposed to a device for prescribing a future of closed possibilities, as in the classical meaning of the term.

Acknowledgements

A graduate fellowship from the Department of Architecture, MIT (2015-2016) and a Scholarship from the AG Leventis Foundation (2016) were granted to the author in support of this work. I would like to thank Professor George Stiny for his comments on an early draft of this paper and the anonymous reviewers for their helpful feedback.

References

Cagan, J., Campbell, M. I., Finger, S., & Tomiyama, T. (2005). A framework for computational design synthesis: model and applications. Journal of Computing and Information Science in Engineering 5 (3), 171-181.

Charidis, A. (2017). Improvisational specification of design spaces. MSc Thesis, submitted to the Departments of Architecture and Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA.

Fensel, D. (2000). Problem solving methods: understanding, description, development and reuse. In Lecture Notes in Artificial Intelligence 1971. Springer-Verlag.

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

Langley, P. W., Simon, H. A., Bradshaw, G. & Zytkow, J. M. (1992). Scientific discovery: computational explorations of the creative process. The MIT Press.

Ligeza, A. (2006). Logical foundations for rule-based systems, 2nd ed. In Studies in Computational Intelligence, Volume 11, Springer-Verlag.

March, L. & Steadman, J. P. (1979). From descriptive geometry to configurational engineering. In Proceedings of the International Conference on Descriptive Geometry, Vancouver, B. C., 21-24.

Newell, A. & Simon, H. A. (1972). Human problem solving. Prentice-Hall, Englewood Cliffs.

Nilsson, N. J. (1982). Principles of artificial intelligence. Springer-Verlag.

Radford, D. & Gero, J. (1988). Design by optimization in architecture, building, and construction. John Wiley & Sons.
Seshadri, R. & Na, T. Y. (1985). Group invariance in engineering boundary value problems. Springer-Verlag.
Simon, H. A. (1977). Models of discovery: and other topics in the methods of science. Springer.
Stiny, G. (1980). Kindergarten grammars: designing with Froebel's building gifts, Environment and Planning B: Planning and Design 7, 409-462.
Stiny, G. (2006). Shape: talking about seeing and doing. The MIT Press.
Stouffs, R. Ed. (2006). Design spaces: the explicit representation of spaces of alternatives. Artificial Intelligence in Engineering Design, Analysis and Manufacturing, Special Issue 20 (2).