Environment and Planning B: Planning and Design, 1983, volume 10, pages 129 - 154

Transformations of languages of designs: part 2

T Weissman Knight¶
Royal College of Art, Kensington Gore, London SW7 2EU, England
Received 1 June 1983

Abstract. A formal model for defining transformations of languages of designs in terms of the grammars which generate them is described in detail. First, a *normal form* for grammars is presented which distinguishes two basic determinants of the compositional structure of designs in a language: spatial relations and the order in which they are employed to generate designs. These two constructive mechanisms are used to specify each rule in a normal form grammar. An internal formal property of a normal form grammar called its *recursive structure* is also characterized. Rules of transformation are then defined which map the rules of a given normal form grammar onto rules of new grammars by changing independently the two components of rules in the original grammar. The new grammars produced specify new languages of designs. Of particular interest are those transformations which preserve the recursive structure of the original grammar.

In part 1 of this paper (Knight, 1983a), a model for defining transformations of languages of designs was sketched informally and considered in relation to other approaches to transformation in design. It was proposed that descriptions of stylistic change and innovation could be treated by formulating rules of transformation which change a grammar defining one style into a grammar (or grammars) defining another style (or styles). Those ideas, along with some preliminary formalisms, are now developed in detail.

A normal form for shape grammars

To facilitate the comparison and construction of languages of designs via the grammars which define them, a standard format or *normal form* for expressing the rules of grammars is defined. This normal form makes explicit (1) the spatial relations used to construct designs in a language and (2) external (nonspatial) controls on the structure of rule applications in the generation of these designs. Both determine the characteristic spatial features and organization of designs in a language.

The specification of a grammar in normal form follows the programme outlined by Stiny (1980b) with some modifications and refinements of the stages involved. Each stage of this new programme requires a more detailed definition of shapes and rules than the corresponding stages of the earlier, more basic programme. For this reason, a thorough understanding of the earlier programme and its applications or of the more general formalism (Stiny, 1980a) and its applications is essential to understanding the programme given here. The sequential ordering of stages in the programme does not imply that stages are obligatory, sequential steps in the specification of a normal form grammar. Each stage entails a description of some component of the final grammar. If any stage is omitted, it can be reconstructed from information given in some later stage.

Stage 1 A vocabulary of labelled (and possibly parameterized) shapes[1] is defined. A labelled shape σ is given by $\sigma = s_P$, where s is a (parameterized) shape and P is a

¶ Present address: School of Architecture and Urban Planning, University of California, Los Angeles, CA 90024, USA

[1] In this and following sections, the term 'labelled shape' is generally understood to include parameterized labelled shapes whenever it refers to a labelled shape in the vocabulary, spatial relations, or rules which specify a grammar.

set of labelled (parameterized) points associated with s. Labelled shapes can be defined either in two or in three dimensions. The definition of a labelled shape here corresponds exactly to the earlier definition of a labelled shape (Stiny, 1980a) given by the ordered pair $\langle s, P\rangle$.

Stage 2 A set of spatial relations between labelled shapes in the vocabulary is defined. Each spatial relation S is specified by a set $\{(s_P)_1, (s_P)_2, ..., (s_P)_n\}$, $n \geqslant 1$, where each $(s_P)_i$ is a labelled shape. In the usual case and in most of the examples given in this paper, a spatial relation S consists of a pair of shapes, that is, $S = \{s_P, t_Q\}$, where s_P and t_Q are labelled shapes.

Stage 3a Shape rules are defined in terms of the spatial relations given in the previous stage. Similar to type 1 and type 2 shape rules (Stiny, 1980b), these rules allow labelled shapes, as distinct from shapes, to be added to or subtracted from a design being generated. Because shapes in spatial relations are labelled, the rules given in terms of them will immediately restrict the language of designs produced by analogous rules given in terms of spatial relations between unlabelled shapes.

The two types of shape rules, corresponding to the operations of addition and subtraction of labelled shapes, are:

type I: $(s_P)_1 + (s_P)_2 + ... + (s_P)_m \rightarrow (s_P)_1 + (s_P)_2 + ... + (s_P)_n$,

type II: $(s_P)_1 + (s_P)_2 + ... + (s_P)_n \rightarrow (s_P)_1 + (s_P)_2 + ... + (s_P)_m$,

where $\{(s_P)_1, (s_P)_2, ..., (s_P)_n\}$ is a set of labelled shapes specifying a spatial relation and $\{(s_P)_1, (s_P)_2, ..., (s_P)_m\}$ is a subset of $\{(s_P)_1, (s_P)_2, ..., (s_P)_n\}$. In the typical case, rules of type I and type II are of the form:

type I: $\sigma \rightarrow s_P + t_Q$,

type II: $s_P + t_Q \rightarrow \sigma$,

where $\sigma = s_P$ or t_Q, and $\{s_P, t_Q\}$ is a set of labelled shapes specifying a spatial relation.

Stage 3b It turns out that rules of types I and II are not powerful enough to describe any language of designs, definable in terms of other shape grammar formalisms. This is because a type I or II rule only allows labelled shapes to be added to or subtracted from a design; labelled shapes cannot be both added to and subtracted from a design in a single step. If this were permitted, additional controls over the order and frequency of rule applications would be made possible. With type 3 and 4 rules (Stiny, 1980b), for example, this is handled by allowing labels to be defined independently of shapes. Thus, although rules do not permit the simultaneous addition and subtraction of shapes, they do permit the simultaneous addition and subtraction of the labels associated with these shapes. The net result is that labelled shapes can be added to and subtracted from a design in just one step. However, rules now no longer correspond to simple spatial relations between labelled shapes. Nor, as will be shown, do they adequately isolate important relationships between labels, shapes, and rules.

With type I and II rules, an additional mechanism is needed which performs the same function as the simultaneous addition and subtraction of labelled shapes, but which operates independently of spatial relations. For this purpose, additional labels, called *state labels* are introduced. An ordered pair of state labels (q_1, q_2) is associated with each shape rule of type I and type II to determine shape rules of types III and IV, respectively:

type III: $\langle (s_P)_1 + (s_P)_2 + ... + (s_P)_m, q_1\rangle \rightarrow \langle (s_P)_1 + (s_P)_2 + ... + (s_P)_n, q_2\rangle$,

type IV: $\langle (s_P)_1 + (s_P)_2 + ... + (s_P)_n, q_1\rangle \rightarrow \langle (s_P)_1 + (s_P)_2 + ... + (s_P)_m, q_2\rangle$,

or more typically:

type III: $\langle \sigma, q_1 \rangle \rightarrow \langle s_P + t_Q, q_2 \rangle$,

type IV: $\langle s_P + t_Q, q_1 \rangle \rightarrow \langle \sigma, q_2 \rangle$.

Notice that the formalism for type III and IV rules (and for type I and II rules) allows for rules to be defined where $(s_P)_1 + (s_P)_2 + \ldots + (s_P)_m$ is equal to $(s_P)_1 + (s_P)_2 + \ldots + (s_P)_n$. This occurs when the subset $\{(s_P)_1, (s_P)_2, \ldots, (s_P)_m\}$ of the set $\{(s_P)_1, (s_P)_2, \ldots, (s_P)_n\}$ specifying a spatial relation is also equal to it. Rules like this may be viewed as either type III or type IV (or, in stage 3a, as either type I or type II). In the typical case, such rules are of the form $\langle s_P, q_1 \rangle \rightarrow \langle s_P, q_2 \rangle$ where the set $\{s_P\}$ is the spatial relation from which a rule is defined.

For the purposes of later transformations, each type III and type IV rule is always associated with, and given in terms of, the unique spatial relation and the unique pair of state labels which define it.

Formally, the state labels q_1 and q_2 are attached to unrestricted parameterized points associated with the labelled shapes on the left-hand and right-hand sides of these rules. This allows rules to be applied in the usual way. As far as their function, however, state labels may be considered to be associated with rules and not with shapes. To explain this point, a short digression on the role of labels in grammars is necessary.

Labels in a shape rule normally supply additional information not provided by the shapes themselves as to (1) how, (2) where, and (3) when a shape rule may be applied to a design being generated. Let the symbol γ denote such a design and let $\alpha \rightarrow \beta$ denote a shape rule. The function of a label or set of labels is determined by its occurrence in both α and γ and, less directly, by its occurrence in β since this decides its subsequent occurrence in designs generated from γ.

In case (1), labels specify under which euclidean transformations a rule can be applied to a subshape of γ similar to α. These labels alter the symmetry of α and of the subshape of γ similar to it. Numerous examples of this kind of labelling are given by Stiny (1980b).

In case (2), labels specify to which subshape or subshapes of γ similar to α a rule may be applied. These labels identify, but do not alter the symmetries of, α and the subshape of γ similar to it. Thus, they do not restrict the euclidean transformations under which the rule can be applied.

In case (3), labels specify that a rule may be applied to a subshape of γ similar to α simply by being associated with the same point or points relative to α and to the subshape of γ similar to it. These labels are distinguished from the previous two kinds because they do not need to be associated with any particular point or points in α and γ so long as the preceding criterion is satisfied. If they are removed both from α and γ, the rule would apply in exactly the same way or ways. Thus, they do not restrict the subshapes to which the rule can be applied or the euclidean transformations under which it can be applied. This kind of labelling is most frequently used to indicate successive stages in the generation of a design. [For example, see the grammar for generating Palladian villas (Stiny and Mitchell, 1978a) and the grammar for generating Japanese tearooms (Knight, 1981a).] Here, labels regulate the sequence and repetition of rule applications.

In the first two cases, labels are *spatial*. Their locations relative to the shapes with which they are associated are essential to their function. In the third case, labels are *nonspatial*. Their locations are not important; only their presence is. They simply indicate the stage or *state* a shape must be in for a rule to apply.

Normally, the rules of a grammar do not express distinctions between spatial and non-spatial labels. In a type 3 or type 4 shape rule (Stiny, 1980b), for instance, both kinds

of labels are included in the single sets of labels associated with the shapes on each side of a rule. When rules are given in the normal form presented here, however, spatial labels are clearly differentiated from nonspatial labels. Spatial labels are labels which are associated with shapes to form labelled shapes in the vocabulary, in spatial relations, and in type I and II shape rules. Nonspatial or state labels are labels associated with type I and II shape rules to define type III and IV shape rules.

To avoid confusing spatial labels with state labels in graphic representations of rules and in rule applications, the sets of symbols which comprise the two must be disjoint. In the examples given in this paper, all state labels are denoted by natural numbers or by the symbol F. A special symbol # is used to denote a variable state label. If the symbol # appears on either the right-hand or left-hand side of a rule, any state label may be substituted for it. When the symbol # appears on both sides of a rule, the same state label must be substituted for it on each side. The rule is then applied in the usual manner. All spatial labels are denoted by symbols other than natural numbers, F, and #. In graphic illustrations of rules, state labels are normally located above the labelled shapes in these rules. If no state label appears on either side of a rule, the symbol # is assumed. Of course, any other conventions for depicting labels, either graphically or symbolically, which are well defined and do not conflict with the basic formalism for type III and IV rules would be equally acceptable.

The different results of defining normal form rules using different kinds of labels are illustrated in figures 1, 2, and 3. In each of these figures, rules are applied recursively to the grid design on the left to produce any one of the grids with embedded triangles on the right as well as other designs not shown[2].

Figure 1 shows three sets of rules which use only nonspatial or state labels. In figure 1(a), the rule can be applied to place a triangle in any square in the grid any number of times. In figure 1(b), the rule can be applied to place a triangle in any square in the grid only once. In figure 1(c), the rules apply to place at least one and at most two triangles in the grid. Notice that in each of these sets of rules, labels do not restrict the euclidean transformations under which a rule can be applied or the subshapes (squares) of the grid to which a rule can be applied.

Figure 2 shows two sets of rules which use only spatial labels. Rules in figure 2(a) contain 'where' labels. These rules can be applied under any appropriate euclidean transformations to place one or more triangles in any square labelled by the symbol •. Rules in figure 2(b) contain 'how' labels. These rules can be applied under restricted euclidean transformations to place at most one triangle for any symbol • labelling a square in the grid. Notice that in both sets of rules there are no restrictions on the repetition or order of rule applications other than those given by the labelled shapes in these rules.

Figure 3 shows two sets of rules which use both state labels and spatial labels. State labels are added to the rules of figure 2(a) to define the rules of figure 3(a). Here, rules apply under any appropriate euclidean transformation to place only one triangle in any square labelled by the symbol •. State labels are added to the rules of figure 2(b) to define the rules of figure 3(b). Here, rules apply under restricted euclidean transformations to place a triangle in just one square labelled by the symbol •.

Stage 4a An initial shape is defined. An initial shape I of a grammar is given by $I = \langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_k,\ q_0 \rangle$, $k \geqslant 1$, where $\{(s_P)_1, (s_P)_2, \ldots, (s_P)_k\}$ is a set of labelled shapes specifying a spatial relation and q_0 is the second coordinate of a pair of state labels $(\#, q_0)$. The first coordinate of this pair is a dummy label which

[2] Readers uncomfortable with the spatial ambiguities of the grid shape may wish to consider rules to apply only under isometric transformations.

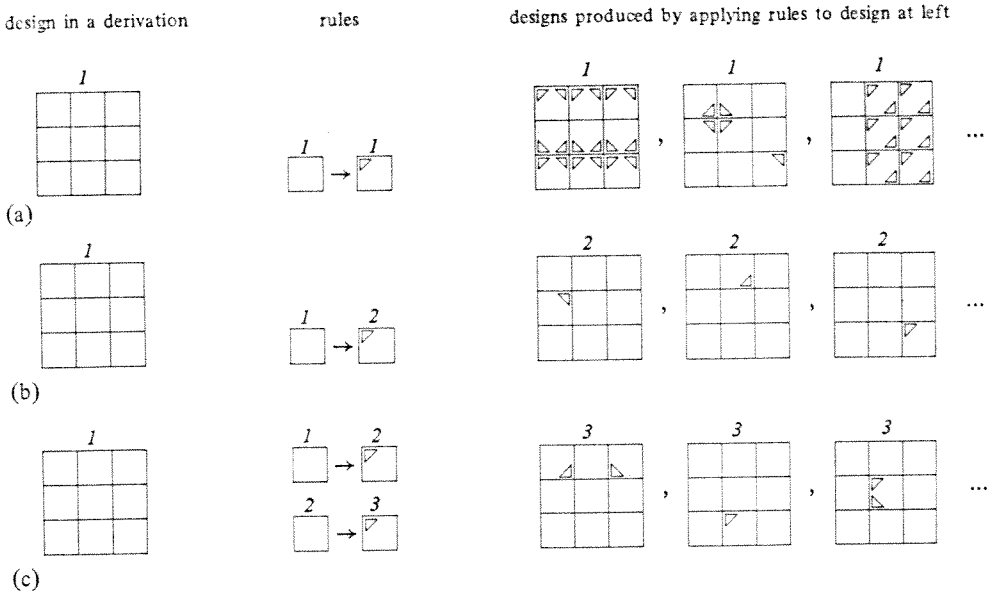


Figure 1. Normal form rules which use only nonspatial or state labels.

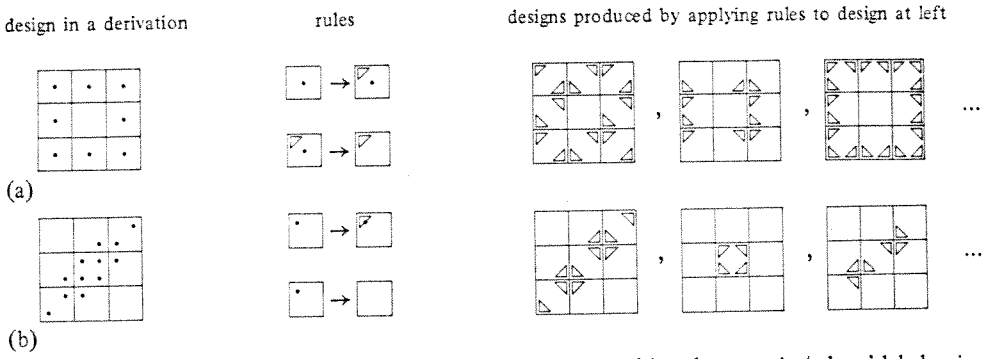


Figure 2. Normal form rules which use only spatial labels. In (a), rules contain 'where' labels; in (b), rules contain 'how' labels.

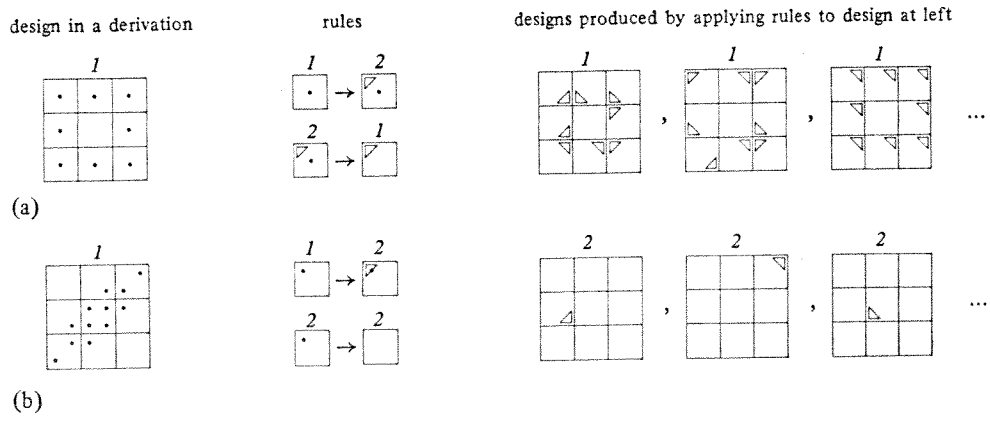


Figure 3. Normal form rules which use state labels and spatial labels. Rules in (a) are defined by adding state labels to the rules in figure 2(a); rules in (b) are defined by adding state labels to the rules in figure 2(b).

has no function in the grammar, but which facilitates subsequent definitions of transformation rules. Usually, the initial shape consists of a single labelled shape together with a state label, that is, $I = \langle s_P, q_0 \rangle$. Like type III and IV shape rules, the initial shape is always associated with, and given in terms of, the unique spatial relation and the unique pair of state labels which define it.

Stage 4b A set of final states is defined. Each final state is denoted by a state label. In the examples given here and in part 3, the set of final states consists of one final state denoted by the symbol F. In the standard shape grammar formalism, a final state is always indicated by a blank or empty state label—a label formally equivalent to any other final state label.

Stage 5 A shape grammar is defined in terms of type III and IV shape rules, the initial shape, and the set of final states. Any design in a final state and with no other (spatial) labels associated with it is a member of the language of designs defined by the grammar.

In figure 4(a), a very simple example of a normal form grammar is shown. The spatial relations and ordered pairs of state labels associated with the initial shape and rules of this grammar are shown in figures 4(b) and 4(c), respectively. The grammar generates an infinite language of designs. Four designs in this language are illustrated in figure 4(d).

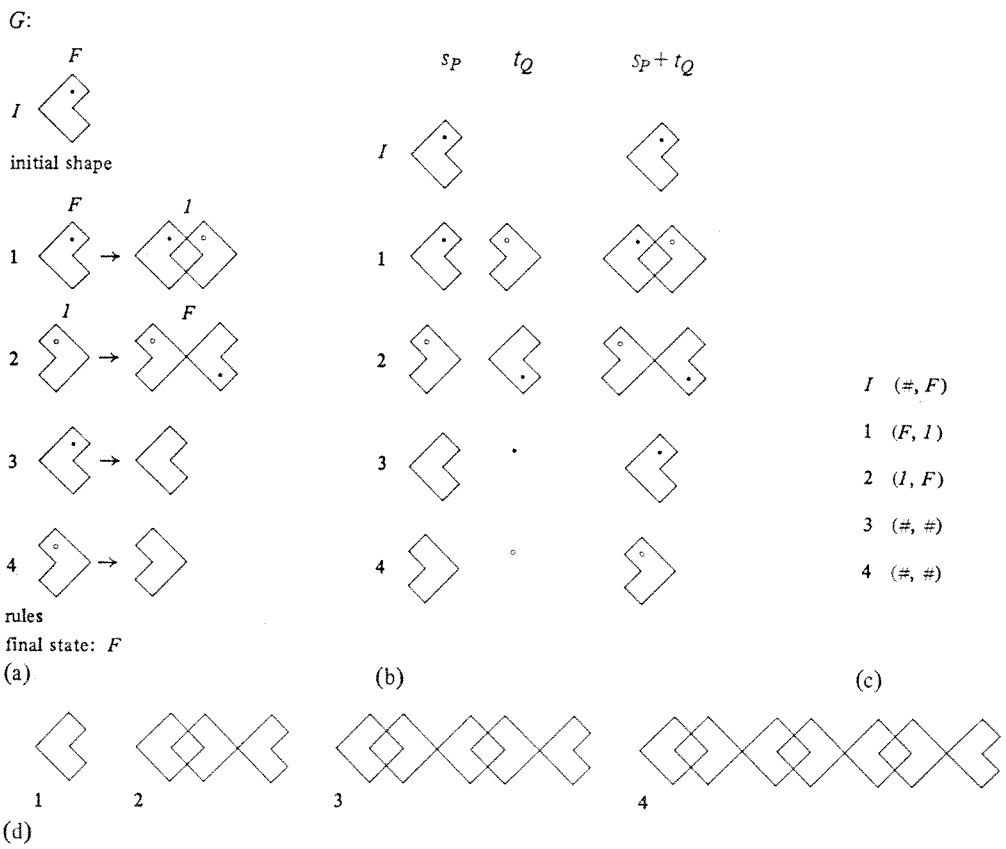


Figure 4. (a) A normal form grammar G. (b) The spatial relations associated with the grammar in (a). The labelled shapes s_P and t_Q in each spatial relation as well as their shape union $s_P + t_Q$ are shown. (c) The ordered pairs of state labels associated with the grammar in (a). (d) Four designs in the language generated by the grammar in (a).

Clearly, a normal form shape grammar will not usually be the shortest or most concise one for defining a particular language. Rules must be broken down into discrete steps involving the elementary operations of addition and subtraction of labelled shapes and changes in states. However, by demanding that shape rules take this simple form, two fundamental determinants of the compositional structure of the designs they generate are isolated: spatial relations and nonspatial mechanisms for ordering spatial relations in rules. Transformations of grammars and the languages of designs they generate can now be defined in a straightforward way. By changing either the spatial relations or the state labels which specify the rules and initial shapes of grammars, new rules and initial shapes of new grammars, and from them, new languages of designs are created.

Nonetheless, changes to spatial relations and state labels should not be made in a completely arbitrary and unrestricted way. This would lead to equally arbitrary and unrestricted transformations. For a transformation to have any innovative value—to produce grammars which yield new but nonrandom languages—and for a transformation to have any explanatory value—to demonstrate convincingly a relationship between two grammars and the languages they generate—more information is needed about the structures of the grammars themselves and thereby about the languages they define. If it can be shown that a transformation preserves some fundamental formal property of a grammar and if it can be shown as a result that individual changes to the grammar are connected in some way, then there are adequate grounds for asserting that a grammar and a transformation of it are related and that the transformation accounts for this relationship in a meaningful way.

Recursive structures of shape grammars

A very basic property of a grammar, one that imposes a meaningful constraint on possible transformations of it, is its *recursive structure*. The recursive structure of a normal form grammar G is given by a relation $R(G)$ on the set of rules and initial shape of which it is comprised. $R(G)$ is defined in terms of the way these rules are linked to each other and to the initial shape in selected, typical derivations of designs in G.

Informally, an ordered pair (rule x, rule y) is a member of $R(G)$ whenever
(1) rule x is a type III (addition) rule in G or the initial shape I of G,
(2) rule y is a type III or type IV (addition or subtraction) rule in G, and
(3) in a derivation of a design, rule y is applied to that part of the design which includes the labelled shape or subshape of the labelled shape which was added by a previous application of rule x.
Thus, when generating a design, applying rule x makes it possible to apply rule y subsequently.

Because it may be impossible to have a complete knowledge of how the rules of a grammar apply to produce designs or, more specifically, to know all possible designs and derivations of designs in the grammar, a recursive structure is defined relative to a limited understanding of the language of designs produced by a grammar, namely, in terms of a finite subset of the language which we know the grammar generates. Thus, whenever multiple understandings of a grammar and its language are possible, multiple definitions of its recursive structure may also be possible. This appeal to a known part of a language as a way of fixing our knowledge of it, however, is not unusual. In fact, the standard method for describing the language generated by a grammar is to give a finite catalogue of representative designs in the language. [For example, see the catalogues given for the language of Hepplewhite chair-back designs (Knight, 1980) and the language of Terragni-style architecture (Flemming, 1981).] The recursive structure of a grammar is similarly specified by means of a finite catalogue of designs

and their derivations. Given a grammar G and a set $D = \{D_1, D_2, ..., D_n\}$, $n \geqslant 1$, of derivations of designs in G [3], $R(G)$ is formally defined as follows.

Let $R_i(G)$ be the recursive structure of G defined on the basis of a single derivation D_i in the set D. $R_i(G)$ is defined recursively in parallel with the derivation D_i with the procedure given below. $R(G)$ is then determined by taking the union of every $R_i(G)$ defined from a corresponding derivation D_i in D. More precisely, $R(G) = R_1(G) \cup R_2(G) \cup ... \cup R_n(G)$.

A procedure for defining $R_i(G)$ from a derivation D_i of a design is now given. In each step x of this procedure, $R_i(G)$ is specified in terms of a labelled shape in a state q called $\Sigma(x)$ and a labelled shape called $\Sigma'(x)$. $\Sigma(x)$ is simply the design in step x of the derivation D_i. $\Sigma'(x)$ is a labelled shape in step x of a derivation parallel to D_i. $\Sigma'(x)$ is always made up of and given in terms of labelled shapes in the vocabulary of labelled shapes which make up the initial shape and rules in G. $\Sigma'(x)$ represents the substructure of $\Sigma(x)$. If no subtraction rules are used, $\Sigma(x)$ without its state label is always identical to $\Sigma'(x)$. Otherwise, $\Sigma(x)$ without its state label is always a subshape of $\Sigma'(x)$.

Recursive definition of $R_i(G)$:
Base:

$$\Sigma(0) = \langle (s_P)_1 + (s_P)_2 + ... + (s_P)_k, q_0 \rangle$$
$$= \text{initial shape of } G,$$
$$\Sigma'(0) = (s_P)_1 + (s_P)_2 + ... + (s_P)_k,$$
$$R_i(G) = \emptyset.$$

Recursion:
A rule y is applied in the standard way to a previously generated design $\Sigma(x)$.
(1) If rule y is a type III rule $\langle s_P, q_1 \rangle \rightarrow \langle s_P + t_Q, q_2 \rangle$ [4], then

$$\Sigma(x+1) = [\Sigma(x) - \tau(\langle s_P, q_1 \rangle)] + \tau(\langle s_P + t_Q, q_2 \rangle).$$

Let μ be a subshape of $\Sigma'(x)$ made up of some of the labelled shapes in it such that $\tau(s_P)$ is a subshape of μ, and for every labelled shape σ in μ, $\tau(s_P)$ is not a subshape of $\mu - \sigma$. In other words, μ is a least collection $\sigma_1 + \sigma_2 + ... + \sigma_n$, $n \geqslant 1$, of labelled shapes of which $\tau(s_P)$ is a subshape.

Let {rule x_1, rule x_2, ..., rule x_m}, $m \geqslant 1$, be a set of type III addition rules (possibly including the initial shape) in G. A rule is a member of this set if and only if it added a labelled shape σ_i in the collection specifying μ to a design in a previous step of the derivation of $\Sigma(x)$. Then,

$$\Sigma'(x+1) = \Sigma'(x) + \tau(t_Q),$$

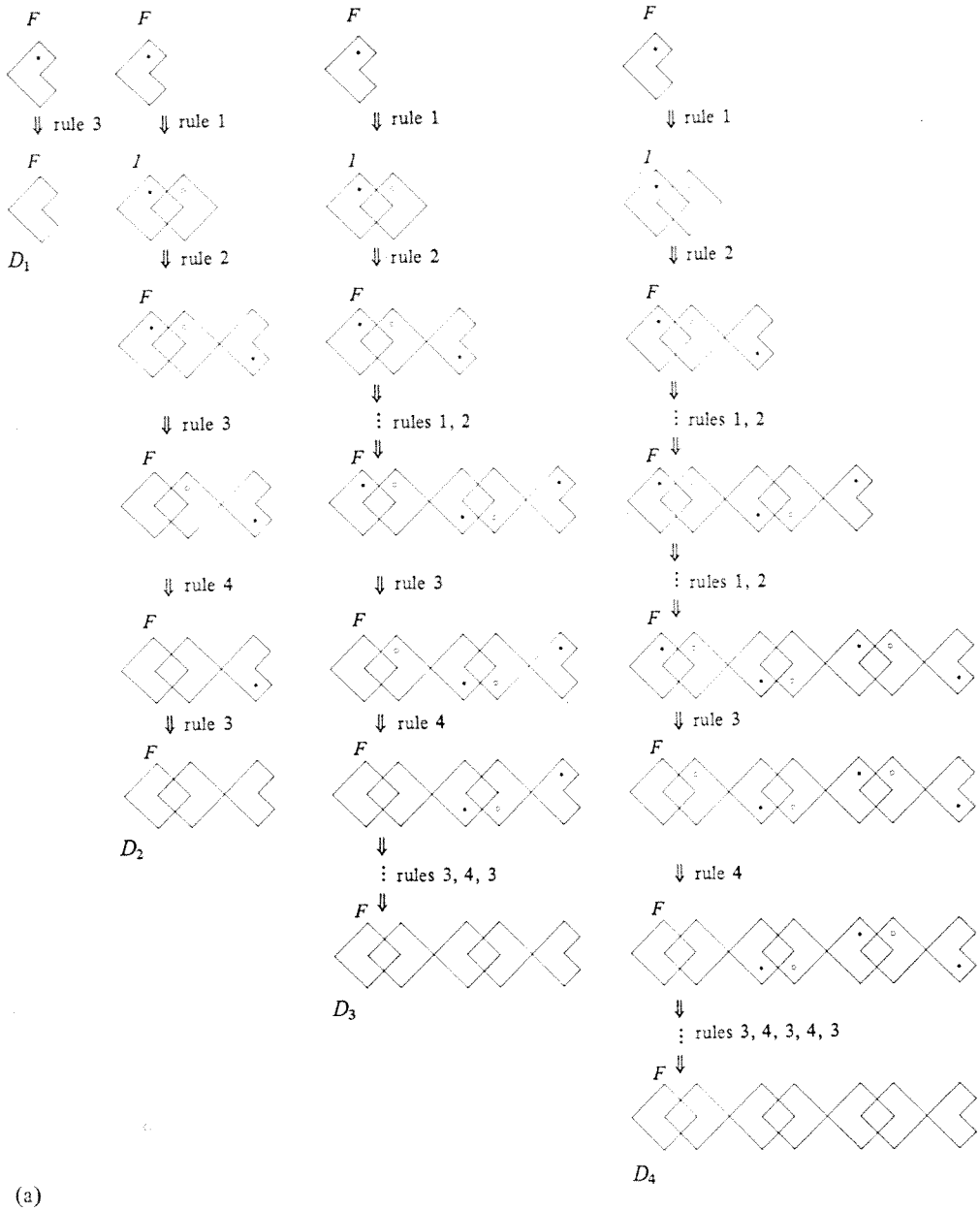
and

$$R_i(G) = R_i(G) \cup \{(\text{rule } x_1, \text{ rule } y), (\text{rule } x_2, \text{ rule } y), ..., (\text{rule } x_m, \text{ rule } y)\}.$$

In the simplest case, μ consists of a single labelled shape and one ordered pair of rules is added to $R_i(G)$. When μ consists of more than one labelled shape, it is an emergent shape—a shape which results from several rule applications. More than one pair of rules may then be added to $R_i(G)$.

[3] When a design can be generated in more than one way by rules in G, multiple derivations of this design can be included in D.

[4] For ease of exposition, type III and IV shape rules are given here in terms of the typical forms of rules (see preceding section). The procedure can easily be generalized to include any type III or type IV rule.



(a)

$R_1(G)$ = {(I, rule 3)}

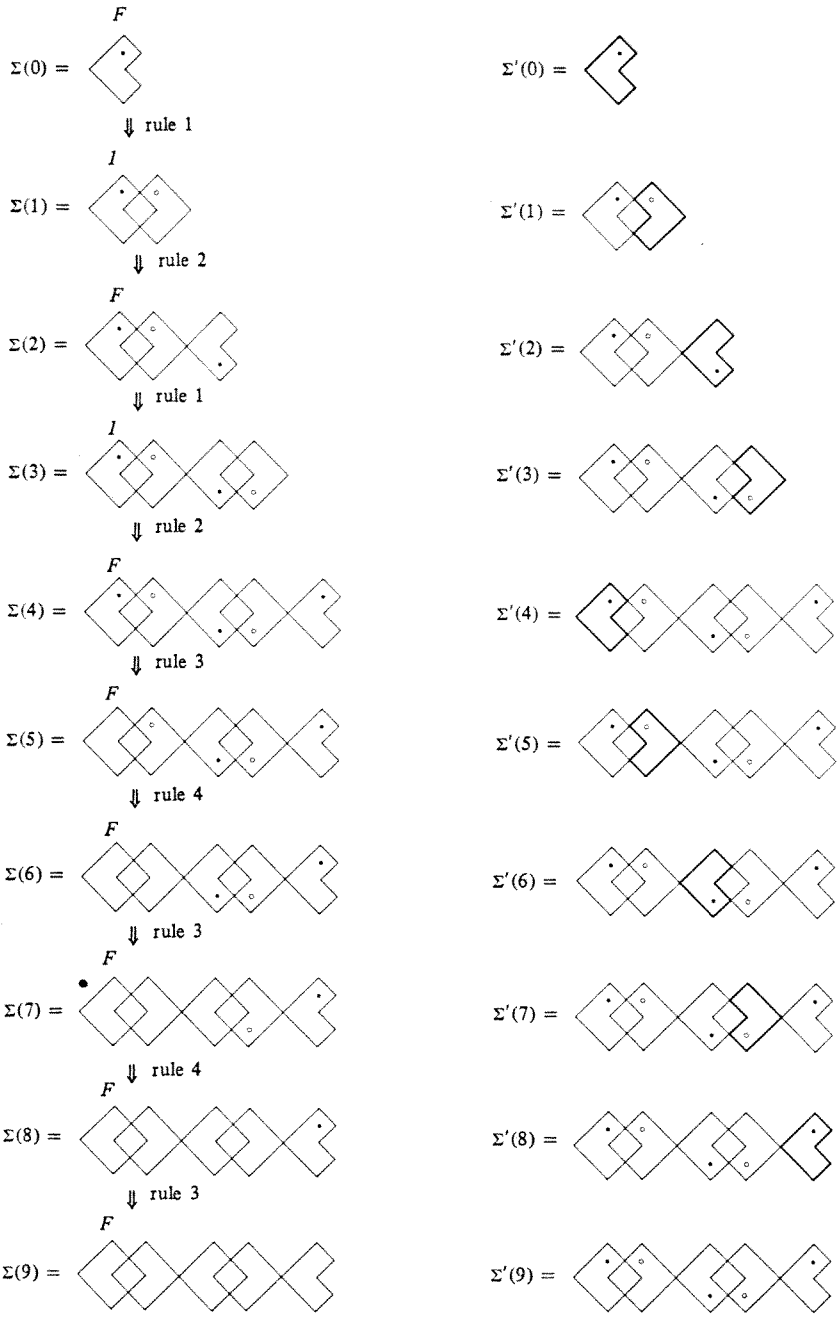
$R_2(G)$ = {(I, rule 1), (rule 1, rule 2), (rule 2, rule 1), (I, rule 3), (rule 1, rule 4), (rule 2, rule 3)}

$R_3(G)$ = {(I, rule 1), (rule 1, rule 2), (rule 2, rule 1), (I, rule 3), (rule 1, rule 4), (rule 2, rule 3)}

$R_4(G)$ = {(I, rule 1), (rule 1, rule 2), (rule 2, rule 1), (I, rule 3), (rule 1, rule 4), (rule 2, rule 3)}

(b)

Figure 5. Defining the recursive structure of the grammar of figure 4(a). (a) A set D of derivations of designs. D_1, D_2, D_3, and D_4 are derivations of designs 1, 2, 3, and 4, respectively, in figure 4(d). (b) The recursive structures $R_1(G)$, $R_2(G)$, $R_3(G)$, and $R_4(G)$. (c) The complete computation of $R_3(G)$. In each step x of the computation, $\Sigma(x)$, $\Sigma'(x)$, and $R_3(G)$ are shown. The subshape μ of $\Sigma'(x)$ is shown by thick lines. Labels inside μ are also part of μ. (d) Definition of $R(G)$.



(c)

$R(G) = R_1(G) \cup R_2(G) \cup R_3(G) \cup R_4(G)$
$= \{(I, \text{rule 1}), (\text{rule 1}, \text{rule 2}), (\text{rule 2}, \text{rule 1}), (I, \text{rule 3}), (\text{rule 1}, \text{rule 4}), (\text{rule 2}, \text{rule 3})\}$

(d)

Figure 5 (continued)

$R_3(G) = \emptyset$

$R_3(G) = \emptyset \cup \{(I, \text{rule } 1)\}$
$= \{(I, \text{rule } 1)\}$

$R_3(G) = \{(I, \text{rule } 1)\} \cup \{(\text{rule } 1, \text{rule } 2)\}$
$= \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2)\}$

$R_3(G) = \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2)\} \cup \{(\text{rule } 2, \text{rule } 1)\}$
$= \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1)\}$

$R_3(G) = \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1)\} \cup \{(\text{rule } 1, \text{rule } 2)\}$
$= \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1)\}$

$R_3(G) = \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1)\} \cup \{(I, \text{rule } 3)\}$
$= \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1), (I, \text{rule } 3)\}$

$R_3(G) = \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1), (I, \text{rule } 3)\} \cup \{(\text{rule } 1, \text{rule } 4)\}$
$= \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1), (I, \text{rule } 3), (\text{rule } 1, \text{rule } 4)\}$

$R_3(G) = \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1), (I, \text{rule } 3), (\text{rule } 1, \text{rule } 4)\} \cup \{(\text{rule } 2, \text{rule } 3)\}$
$= \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1), (I, \text{rule } 3), (\text{rule } 1, \text{rule } 4), (\text{rule } 2, \text{rule } 3)\}$

$R_3(G) = \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1), (I, \text{rule } 3), (\text{rule } 1, \text{rule } 4), (\text{rule } 2, \text{rule } 3)\} \cup \{(\text{rule } 1, \text{rule } 4)\}$
$= \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1), (I, \text{rule } 3), (\text{rule } 1, \text{rule } 4), (\text{rule } 2, \text{rule } 3)\}$

$R_3(G) = \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1), (I, \text{rule } 3), (\text{rule } 1, \text{rule } 4), (\text{rule } 2, \text{rule } 3)\} \cup \{(\text{rule } 2, \text{rule } 3)\}$
$= \{(I, \text{rule } 1), (\text{rule } 1, \text{rule } 2), (\text{rule } 2, \text{rule } 1), (I, \text{rule } 3), (\text{rule } 1, \text{rule } 4), (\text{rule } 2, \text{rule } 3)\}$

Figure 5 (continued)

(2) If rule y is a type IV rule $\langle s_P + t_Q, q_1\rangle \rightarrow \langle s_P, q_2\rangle$, then

$$\Sigma(x+1) = [\Sigma(x) - \tau(\langle s_P + t_Q, q_1\rangle)] + \tau(\langle s_P, q_2\rangle) .$$

Let μ be a least collection of labelled shapes in $\Sigma'(x)$ [as in case (1)] such that $\tau(s_P + t_Q)$ is a subshape of μ.

Let a set of type III addition rules be defined as in case (1). Then,

$$\Sigma'(x+1) = \Sigma'(x) ,$$

and $R_i(G)$ is defined as in case (1).

In the simplest case, μ consists of two labelled shapes and two pairs of rules are added to $R_i(G)$.

(3) If rule y is a type III or IV rule $\langle s_P, q_1\rangle \rightarrow \langle s_P, q_2\rangle$, then

$$\Sigma(x+1) = [\Sigma(x) - \tau(\langle s_P, q_1\rangle)] + \tau(\langle s_P, q_2\rangle) .$$

Let μ be a least collection of labelled shapes in $\Sigma'(x)$ [as in case (1)] such that $\tau(s_P)$ is a subshape of μ.

Let a set of type III addition rules be defined as in case (1). Then,

$$\Sigma'(x+1) = \Sigma'(x)$$

[as in case (2) above because no labelled shapes are added to the design] and $R_i(G)$ is defined as in case (1).

The simplest case is the same as that given in case (1).

When labelled shapes in rules are parameterized, the definitions given here are easily extended by including wherever necessary the assignments under which rules are applied[5].

Figure 5 shows how the recursive structure of the grammar of figure 4(a) is defined. It is specified in terms of a set D containing derivations D_1, D_2, D_3, and D_4 of designs 1, 2, 3, and 4 shown in figure 4(d). D_1, D_2, D_3, and D_4 are illustrated in figure 5(a). These derivations are used to compute $R_1(G)$, $R_2(G)$, $R_3(G)$, and $R_4(G)$ given in figure 5(b). Other possible derivations of designs 1, 2, 3, and 4 define recursive structures identical to $R_1(G)$, $R_2(G)$, $R_3(G)$, and $R_4(G)$. The complete computation of $R_3(G)$ is shown in figure 5(c). Notice that the number of times a rule applies in this derivation has no bearing on the definition of $R_3(G)$. Once a connection between two rules is established and an ordered pair is added to $R_3(G)$, any repetition of this connection is not recorded. $R_1(G)$, $R_2(G)$, and $R_4(G)$ are computed in essentially the same way as $R_3(G)$.

$R(G)$ is defined in figure 5(d). Observe that $R_2(G) = R_3(G) = R_4(G) = R(G)$. Because all derivations are repetitions of a derivation of design 2, the inclusion of any other derivation of a design in D does not alter the definition of $R(G)$. Of course, the recursive structures of grammars more complex than the one given here will generally be dependent upon more than one derivation of a design.

Once the recursive structure of a grammar is defined, the recursive structure of any subset of the rules and the initial shape of the grammar can also be defined. The recursive structure of a subset G' of a grammar G is the subset of the recursive structure of G which contains all and only those pairs of rules (and the initial shape) in G'.

[5] A stricter, more detailed definition of recursive structure can be obtained on the basis of this same procedure, where $R(G)$ is a relation, not on the rules and initial shape of G, but on the labelled shapes in these rules and initial shape. Each labelled shape on the left side of a type III or type IV rule y is linked to labelled shapes in type III rules $x_1, x_2, \ldots, x_m$ which are added to a design, to form ordered pairs in $R(G)$. This definition of recursive structure, however, is slightly less intuitive and more difficult to define informally than the one presented above. Both definitions characterize the structure of rule applications in a grammar in essentially the same way.

Pairs in $R(G')$ are thus determined by only those derivations or parts of derivations included in the catalogue D, which use rules in G'. Derivations using rules in G' may terminate with finished designs or, if they are partial, they may terminate with unfinished, partial designs which correspond to the completion of a particular stage or stages in the grammar. [For instance, see the designs in the catalogues of Palladian room layouts (Stiny and Mitchell, 1978b), partial tearoom plans (Knight, 1981a), and basic compositions of Frank Lloyd Wright prairie houses (Koning and Eizenberg, 1981). All are partial designs derived using a subset of a grammar.] An illustration of the recursive structure of a subset of a grammar is given in part 3.

Despite the length and detail of the formal definition of recursive structure, the basic concept is not difficult to grasp. Once understood, a familiarity with the rules of a grammar and the designs it generates is generally sufficient to enumerate pairs in its recursive structure without following step-by-step the procedure given above.

Transformations of shape grammars

A transformation or, more generally, a *family* of transformations of a grammar in normal form is now defined. Two independent stages are involved. In the first stage, a set A of *shape change rules* is defined which transforms the set of spatial relations associated with the rules and initial shape of a grammar into new sets of spatial relations. For each transformation of a set of spatial relations defined by A, a corresponding transformation T_A of the rules and initial shape is defined.

In the second stage, a set B of *state change rules* is defined which transforms the set of pairs of state labels associated with the rules and initial shape of a grammar into new sets of pairs of state labels. For each transformation of a set of pairs of state labels defined by B, a corresponding transformation T_B of the set of rules and initial shape is defined.

Finally, each T_A and each T_B defined in stages 1 and 2 are combined to produce a complete transformation T of the set of rules and initial shape of a grammar. A transformation T is a member of a family Υ of transformations, whenever the recursive structure defined for the original grammar (or subset of the original grammar if T is partial) is *isomorphic* to the recursive structure defined for the transformed grammar.

In any transformation of a grammar, the set of final states in the original grammar is mapped unchanged to new grammars, since changes to final states can be made indirectly by changing state labels associated with rules.

Stage 1a A set A of shape change rules is specified. A defines one-to-one mappings or transformations of a set of spatial relations called the *initial set of spatial relations*, into new sets of spatial relations called *final sets of spatial relations*. The initial set of spatial relations is simply the set of spatial relations associated with the rules and initial shape of a grammar together with a set of ancillary labels described below. Shape change rules in A apply recursively to a spatial relation in the initial set to derive a new or *final* spatial relation. When every spatial relation in the initial set is, where possible, transformed using rules in A into a final spatial relation, a final set of spatial relations is determined. Different applications or sequences of applications of shape change rules in A to the same spatial relation in the initial set may produce different final spatial relations. For each such spatial relation, a different transformation of the initial set of spatial relations into a final set is defined.

To control the application of shape change rules to spatial relations, a set L of *nonterminal labels* is defined in conjunction with A. Nonterminal labels are symbols associated with labelled shapes in spatial relations in the initial set and with labelled shapes in shape change rules in A. Nonterminal labels which are attached to labelled shapes in the initial set of spatial relations are not shown, and have no function,

in the corresponding grammar. To avoid confusion, these labels should always be different from labels used in the grammar.

A shape change rule is a rule of the form $u_X \rightarrow v_Y$, where u_X and v_Y are labelled shapes (X and Y may include nonterminal labels) and u_X is equal to $(s_\phi)_\phi$ if and only if v_Y is equal to $(s_\phi)_\phi$. A shape change rule $u_X \rightarrow v_Y$ applies to a spatial relation S whenever there is a labelled shape s_P (P may include nonterminal labels) in S and a euclidean transformation τ such that $\tau(u_X)$ is a subshape of s_P. The spatial relation S' produced by applying the shape change rule to S is given by $S' = (S - \{s_P\}) \cup \{s_P'\}$, where s_P' is the labelled shape equal to $[s_P - \tau(u_X)] + \tau(v_Y)$. Thus, a new spatial relation is formed by replacing the labelled shape s_P in S with the labelled shape s_P'. [Shape change rules are a generalization of the earlier shape equivalence rules $\alpha \leftrightarrow \beta$ (Knight, 1981b) which replace a shape s, but not a labelled shape, in a spatial relation, whenever $\tau(\alpha)$ or $\tau(\beta)$ is equal to, not just a subshape of, s. Shape change rules apply indirectly to labelled shapes in a spatial relation in exactly the same way that shape rules apply to labelled shapes.]

A shape change rule $u_X \rightarrow v_Y$ also applies to a spatial relation S whenever there is a parameterized labelled shape s_P in S, a euclidean transformation τ, and an assignment h of values to variables in s_P such that $\tau(u_X)$ is a subshape of $h(s_P)$. The new spatial relation S' produced is given by $S' = (S - \{s_P\}) \cup \{s_P'\}$, where s_P' is a new parameterized labelled shape defined in terms of a representative labelled shape in its family given by $[h(s_P) - \tau(u_X)] + \tau(v_Y)$. In other words, the shape change rule replaces a parameterized labelled shape s_P in S with a labelled shape to which parameters are attached subsequently to define s_P'. Parameters are not given explicitly in the rule itself[6].

To specify no change for a labelled shape in a spatial relation in the initial set, an *identity* shape change rule $u_X \rightarrow u_X$ must be defined. For instance, if s_P is a labelled shape in a spatial relation S and there is a euclidean transformation τ such that $\tau(u_X)$ is a subshape of s_P, then applying the rule $u_X \rightarrow u_X$ to S simply replaces s_P in S with s_P.

New spatial relations are generated from a spatial relation in the initial set by recursively applying any applicable rule or rules in A to it. A new spatial relation is called a final spatial relation if and only if all labelled shapes in it have been replaced and no nonterminal labels are associated with it. A final set of spatial relations is derived from the initial set by deriving a final spatial relation from each spatial relation in the initial set, wherever it is possible to do so. When more than one final spatial relation can be derived from any spatial relation in the initial set, then more than one transformation of the initial set into a final set is determined. If there is any spatial relation in the initial set which cannot be transformed into a final spatial relation by rules in A, then A defines only partial transformations of the initial set.

Figures 6–9 illustrate how a set A of shape change rules is used to transform the set of spatial relations associated with the grammar of figure 4(a) into new (final) sets of spatial relations.

The initial set of spatial relations is shown in figure 6. Here, the set L of nonterminal labels is empty. The example given in part 3 (Knight, 1983b) will demonstrate how nonterminal labels can be employed in transformations.

A set A of shape change rules is shown in figure 7. To make clear the changes in labelled shapes, the cartesian coordinate system in which labelled shapes are located is represented by broken lines in these rules. These lines should not be interpreted as part of a labelled shape in a rule. Rules 1 and 2 in A change the position of an L-shape by reflecting it as shown. These rules can only apply to

[6] Definitions for shape change rules can be extended in the obvious way to define more general kinds of rules called shape change rule schemata which specify families of shape change rules. Definitions and examples of shape change rule schemata are omitted here only to simplify the discussion and illustrations of transformations.

spatial relations I, 1, or 2 in the initial set. Rules 3 – 5 are identity rules. Rule 3 can apply to any spatial relation in the initial set; rule 4 to spatial relations I, 1, 2, and 3; and rule 5 to spatial relations 1, 2, and 4.

Figure 8 shows how a final set of spatial relations is derived from the initial set using rules in A. Notice that rule 3 is applied to spatial relations I, 1, and 2 to produce identity changes. However, application of either rule 4 or 5 or recursive application of rule 1 or 2 would also define an identity change.

Fifteen other final sets of spatial relations can be generated by applying rules in A to the initial set. These, together with the final set shown in figure 8, are illustrated in figure 9.

Of course, there are many other possible ways to transform the initial set of spatial relations using different sets of shape change rules. Changing the vocabulary elements which form spatial relations or simply changing the labels associated with shapes in these

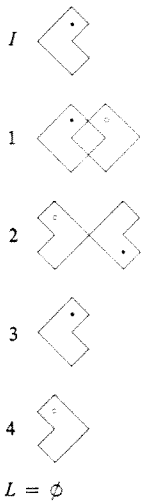


Figure 6. The initial set of spatial relations for the grammar of figure 4(a). To simplify illustrations of spatial relations in this and other figures, they are shown as the shape unions of labelled shapes in them.

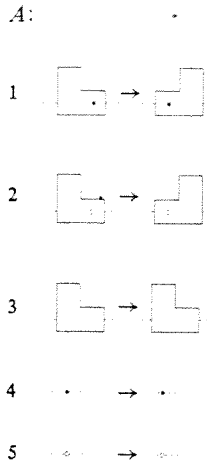


Figure 7. A set A of shape change rules.

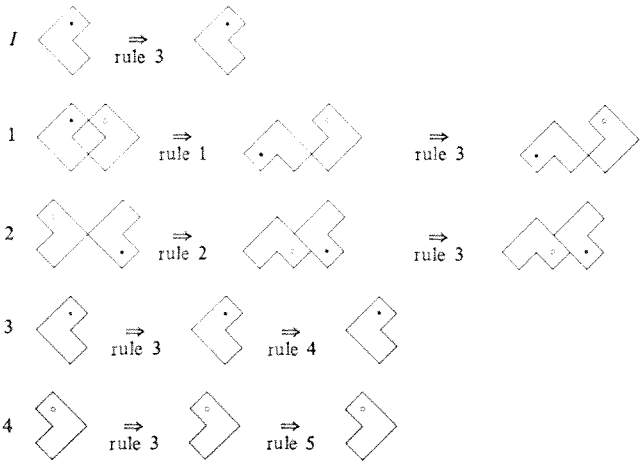


Figure 8. A derivation of a final set of spatial relations from the initial set of spatial relations (figure 6) using shape change rules in A (figure 7).

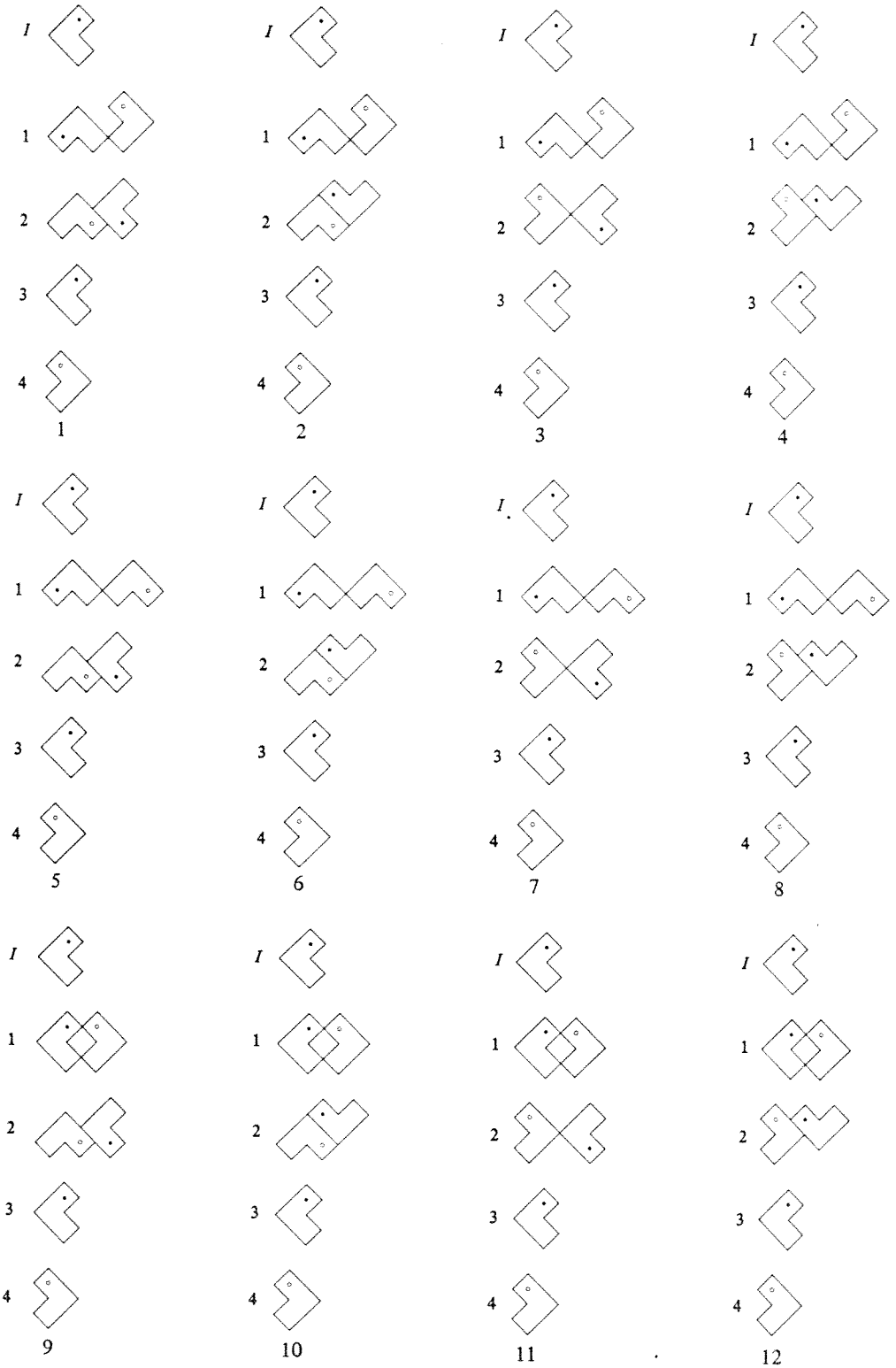


Figure 9. The sixteen final sets of spatial relations produced by applying rules in A (figure 7) to the initial set of spatial relations (figure 6).

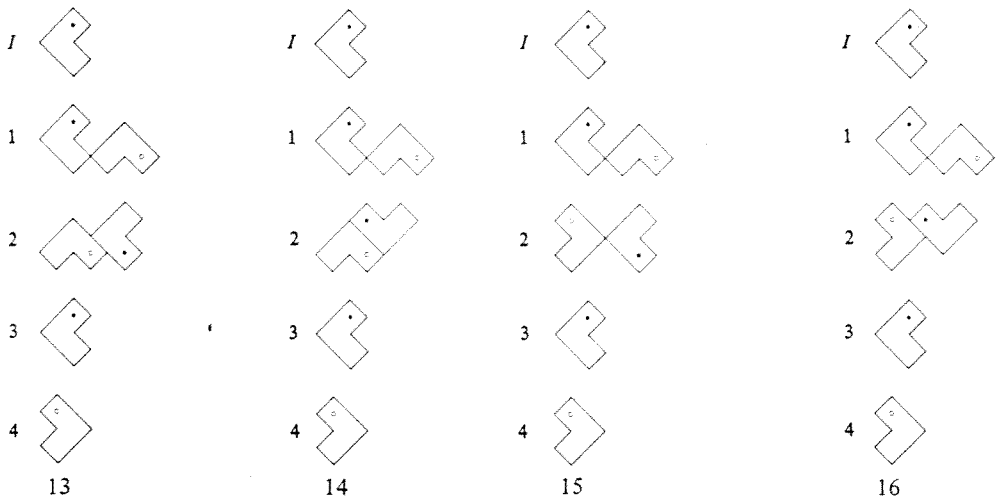


Figure 9 (continued)

relations would produce equally interesting results. Readers interested in exploring transformations of spatial relations should refer to Knight (1981b), where several examples of simple changes to spatial relations are given. These changes, which are limited to spatial relations between unlabelled shapes, are just a small sampling of the diversity of ways that new spatial relations can be created.

Stage 1b For each transformation of an initial set of spatial relations into a final set of spatial relations defined by A, a one-to-one mapping or transformation T_A of the corresponding grammar is defined. Let Υ_A be the set of all such transformations. A transformation T_A of a grammar is specified by a transformation of its associated set of spatial relations in an obvious way: T_A replaces each labelled shape s_P in a shape

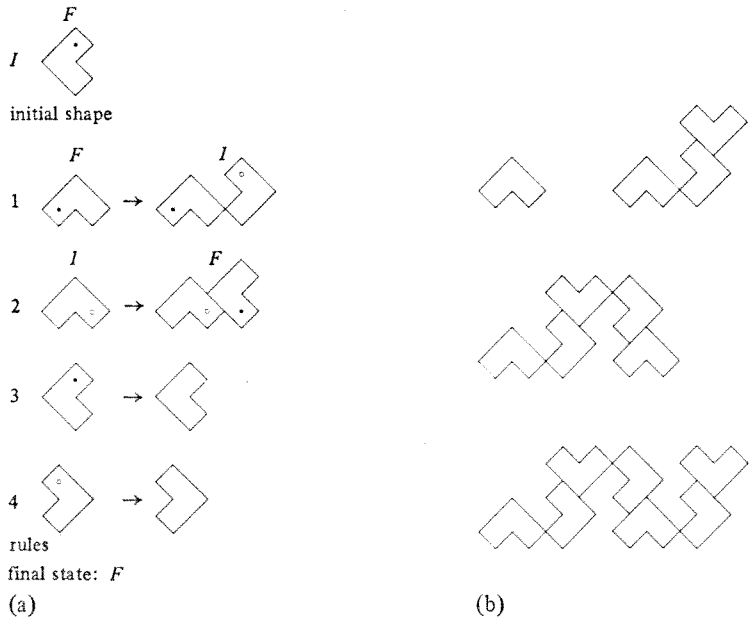


Figure 10. (a) A transformation T_A of the grammar G of figure 4(a). $T_A(G)$ is defined using the final set of spatial relations shown in figure 8. (b) Four designs in the language generated by $T_A(G)$.

rule or initial shape of the grammar with a new labelled shape s_P' if and only if s_P' replaces s_P in the transformation of the spatial relation associated with this rule or initial shape. More formally:

$$T_A(\langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_m, q_1\rangle \rightarrow \langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_n, q_2\rangle)$$
$$= \langle (s_P)_1' + (s_P)_2' + \ldots + (s_P)_m', q_1\rangle \rightarrow \langle (s_P)_1' + (s_P)_2' + \ldots + (s_P)_n', q_2\rangle ,$$

where $\langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_m, q_1\rangle \rightarrow \langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_n, q_2\rangle$ is a type III or type IV shape rule, and

$$T_A(\langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_k, q_0\rangle) = \langle (s_P)_1' + (s_P)_2' + \ldots + (s_P)_k', q_0\rangle ,$$

where $\langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_k, q_0\rangle$ is an initial shape.

Figure 10(a) shows a transformation T_A of the grammar G of figure 4(a). $T_A(G)$ is specified in terms of the final set of spatial relations shown in figure 8. Four members of the language defined by $T_A(G)$ are illustrated in figure 10(b). This transformation, along with transformations corresponding to each of the fifteen other final sets given in figure 9, determine a set Υ_A of sixteen transformations of the original grammar.

Stage 2a A set B of state change rules is specified. B defines one-to-one mappings or transformations of a set of pairs of state labels called the *initial set of pairs of state labels*, into new sets of pairs of state labels called *final sets of pairs of state labels*. The initial set of pairs of state labels is simply the set of pairs of state labels associated with the rules and initial shape of a grammar. A state change rule in B applies to a pair of state labels in the initial set to produce a new or *final* pair of state labels. When every pair of state labels in the initial set is transformed, where possible, into a final pair of state labels using rules in B, a final set of pairs of state labels is determined. Applications of different state change rules in B to the same pair of state labels may produce different pairs of new state labels. For each such pair, a different transformation of the initial set of pairs into a final set is defined.

A state change rule is a rule of the form r_i: $(q_1, q_2) \rightarrow (q_1', q_2')$, where r_i refers to the ith shape rule or the initial shape I of a grammar, and (q_1, q_2) and (q_1', q_2') are ordered pairs of state labels. A state change rule r_i: $(q_1, q_2) \rightarrow (q_1', q_2')$ applies to an ordered pair of state labels (q_1, q_2) in the initial set whenever (q_1, q_2) is the pair of state labels associated with the initial shape or shape rule i. The new or final pair of state labels produced is the pair (q_1', q_2'). To specify no change for a pair of state labels in the initial set, an *identity* state change rule r_i: $(q_1, q_2) \rightarrow (q_1, q_2)$ must be defined for that pair.

A final set of pairs of state labels is derived from the initial set by applying any applicable rule in B to a pair of state labels in the initial set, wherever it is possible to do so. If there is any pair of state labels in the initial set which cannot be transformed by a rule in B, then B defines only partial transformations of the initial set.

	B:	
I $(\#, F)$	r_I: $(\#, F) \rightarrow (\#, I)$	I $(\#, I)$
1 (F, I)	r_1: $(F, I) \rightarrow (I, F)$	1 (I, F)
2 (I, F)	r_2: $(I, F) \rightarrow (F, I)$	2 (F, I)
3 $(\#, \#)$	r_3: $(\#, \#) \rightarrow (\#, \#)$	3 $(\#, \#)$
4 $(\#, \#)$	r_4: $(\#, \#) \rightarrow (\#, \#)$	4 $(\#, \#)$
(a)	(b)	(c)

Figure 11. A set of state change rules is defined for and applied to the pairs of state labels associated with the grammar of figure 4(a). (a) The initial set of pairs of state labels; (b) a set B of state change rules; (c) the final set of pairs of state labels produced by applying rules in B to the initial set.

Like shape change rules, state change rules may be nondeterministic; different state change rules can be defined for the same pair of state labels. Unlike shape change rules, state change rules do not apply recursively. For any transformation of a pair of state labels in the initial set into a pair of state labels in a final set, one and only one state change rule is applied.

In figure 11, a set B of state change rules is defined for and applied to the set of pairs of state labels associated with the grammar of figure 4(a). The initial set of pairs of state labels is shown in figure 11(a), the set of state change rules in figure 11(b), and the final set of pairs of state labels it generates in figure 11(c). Here, only one final set can be produced by applying rules in B to the initial set.

Stage 2b For each transformation of an initial set of pairs of state labels into a final set defined by B, a one-to-one mapping or transformation T_B of the corresponding grammar is defined. Let Υ_B be the set of all such transformations. A transformation T_B of a grammar is specified by a transformation of its associated set of pairs of state labels in the expected way: T_B replaces each state label q in a shape rule or initial shape of the grammar with a new state label q' if and only if q' replaces q in the transformation of the pair of state labels associated with this rule or initial shape. More formally:

$$T_B(\langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_m, q_1\rangle \rightarrow \langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_n, q_2\rangle)$$
$$= \langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_m, q_1'\rangle \rightarrow \langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_n, q_2'\rangle ,$$

where $\langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_m, q_1\rangle \rightarrow \langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_n, q_2\rangle$ is a type III or IV shape rule, and

$$T_B(\langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_k, q_0\rangle) = \langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_k, q_0'\rangle ,$$

where $\langle (s_P)_1 + (s_P)_2 + \ldots + (s_P)_k, q_0\rangle$ is an initial shape.

Figure 12(a) shows a transformation T_B of the grammar G of figure 4(a). $T_B(G)$ is specified in terms of the final set of pairs of state labels shown in figure 11(c).

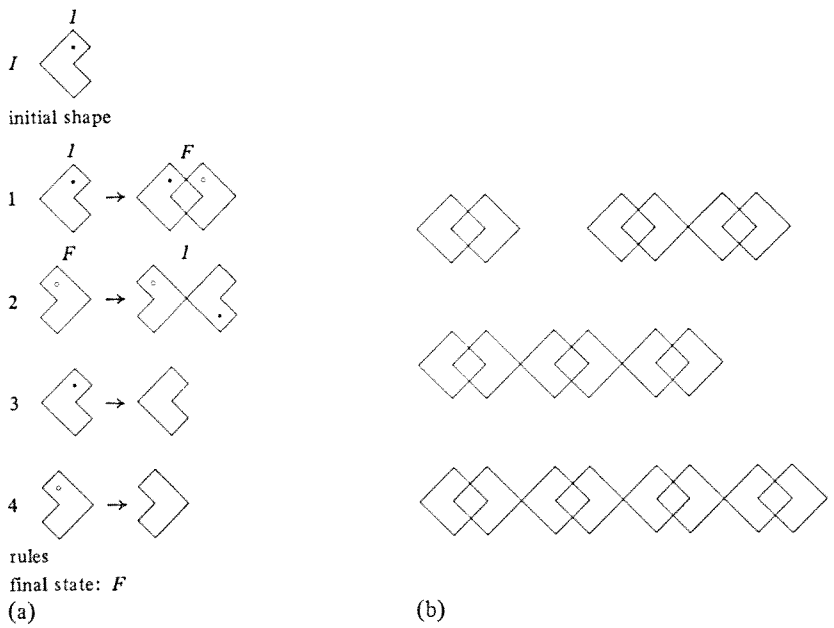


Figure 12. (a) A transformation T_B of the grammar G of figure 4(a). $T_B(G)$ is defined using the final set of pairs of state labels shown in figure 11(c). (b) Four designs in the language generated by $T_B(G)$.

Because only one final set is determined by rules in B, T_B is the only member of the set Υ_B of transformations of this grammar. T_B produces a small but nontrivial change in the ordering of the rules of the grammar simply by changing its final state. Figure 12(b) shows four designs in the language generated by the new grammar $T_B(G)$.

The composition of any T_A in Υ_A defined in stage 1 with any T_B in Υ_B defined in stage 2 produces a complete transformation T of the rules and initial shape of a grammar. T is defined for only those rules and initial shapes for which T_A and T_B are both defined. Thus, if either T_A or T_B is a partial transformation, then T is a partial transformation. Because T_A and T_B operate independently on different parts of shape rules and initial shapes, the order in which they are composed has no effect on the transformation T they determine. That is, $T_A \cdot T_B = T_B \cdot T_A$.

A transformation T is a member of a family Υ of transformations of a grammar G, whenever the recursive structure defined for the subset G' of G transformed is *isomorphic* to the recursive structure defined for $T(G)$ (G' is equal to G if T is total; G' is a proper subset of G if T is partial.) $R(G')$ is isomorphic to $R[T(G)]$ whenever

(rule x, rule y) is in $R(G') \Leftrightarrow [T(\text{rule } x), T(\text{rule } y)]$ is in $R[T(G)]$.

In other words, the recursive structure of a (subset of a) grammar is isomorphic to the recursive structure of another (subset of a) grammar whenever there is a one-to-one transformation which maps every rule and the initial shape in (the subset of) one grammar onto a rule and initial shape of (the subset of) the other grammar and which maps every ordered pair in the recursive structure of (the subset of) one grammar onto an ordered pair in the recursive structure of (the subset of) the other grammar.

To preserve the recursive structure of a grammar or subset of a grammar G under a transformation T, whenever changes are made to the labelled shapes and state labels in a rule x of G to define T (rule x) and (rule x, rule y) is in $R(G')$, then changes must be made to labelled shapes and state labels in rule y to define T(rule y) such that $[T(\text{rule } x), T(\text{rule } y)]$ is in $R[T(G)]$. Using the recursive structure this way as a guide for the definition of change rules ensures that the rules of new grammars operate in the same way as the rules of the original grammar. Knowledge of how a grammar works is thus transferred to new grammars whose structures are immediately understood, though the designs they produce may be new and unexpected.

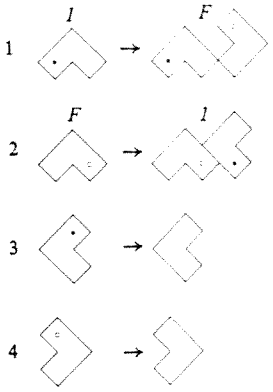
When defining rules for transformations which preserve recursive structures, shape change rules and state change rules may still be considered completely independently of each other. This is because each of the transformations T_A and T_B, restricted to the domain of the transformation T they determine, preserves the recursive structure of a (subset of a) grammar G whenever T preserves the recursive structure of (a subset of) G and vice versa. Thus, either of the grammars $T_A(G)$ or $T_B(G)$ can be viewed as an intermediate stage (logically, not historically) in the formation of $T(G)$.

Figure 13 (see pages 149–152) illustrates the family Υ of transformations of the grammar of figure 4(a) specified from the sets Υ_A and Υ_B defined previously. All transformations in Υ are total. A design in each of the languages generated by these new grammars is also shown. Some of the designs (as well as the languages) determined by different transformations of the original grammar are equivalent under euclidean transformations. This arises from symmetry properties of the spatial relations associated with the original grammar and is not a feature of transformations in general.

Figure 13 (pages 149–152). The family Υ of transformations of the grammar G of figure 4(a). Υ contains sixteen transformations of G producing the sixteen new grammars, $T_1(G)$–$T_{16}(G)$, shown here. A design in each of the languages generated by these grammars is also shown. The subscript i of each $T_i(G)$ corresponds to the number i given beneath a final set of spatial relations in figure 9 which, together with the final set of pairs of state labels shown in figure 11(c), determines $T_i(G)$. $T_1(G)$ is the composition of $T_A(G)$ and $T_B(G)$ shown in figures 10(a) and 12(a), respectively.

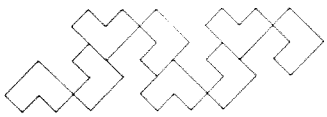
$T_1(G)$:

initial shape



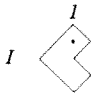
rules

final state: F

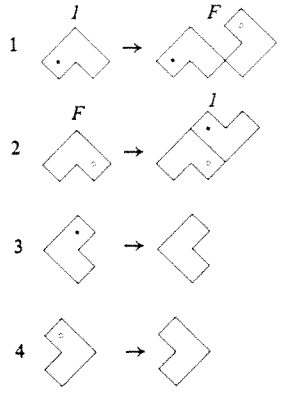


design in language

$T_2(G)$:

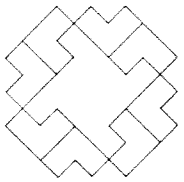


initial shape



rules

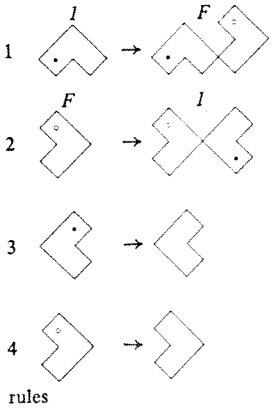
final state: F



design in language

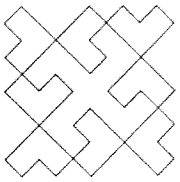
$T_3(G)$:

initial shape



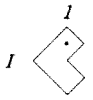
rules

final state: F

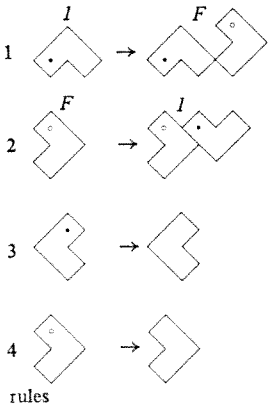


design in language

$T_4(G)$:

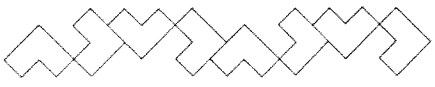


initial shape



rules

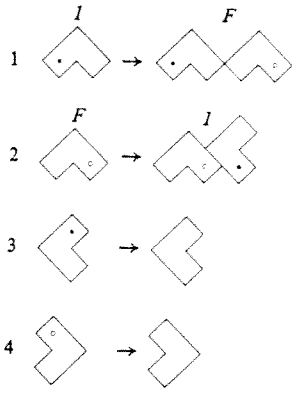
final state: F



design in language

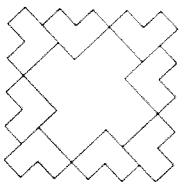
$T_5(G)$:

initial shape



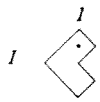
rules

final state: F

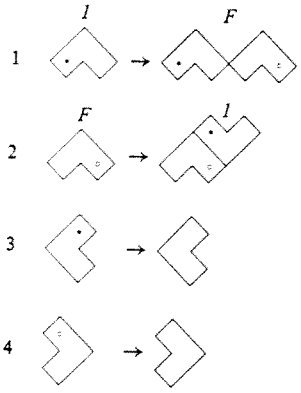


design in language

$T_6(G)$:



initial shape

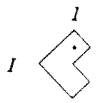


rules

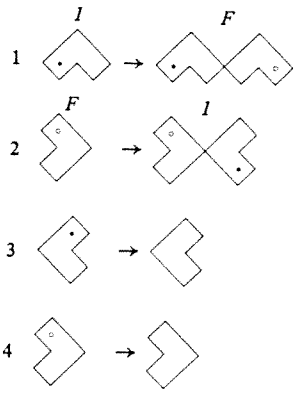
final state: F

design in language

$T_7(G)$:

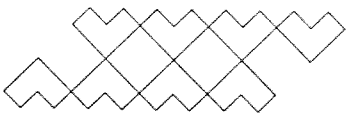


initial shape



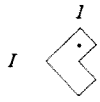
rules

final state: F

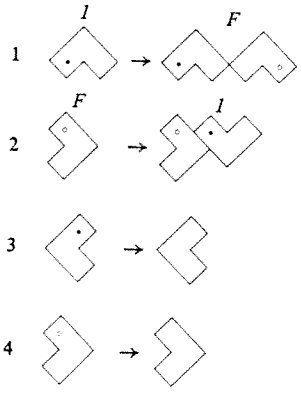


design in language

$T_8(G)$:

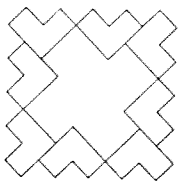


initial shape



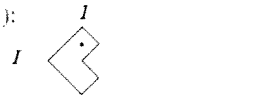
rules

final state: F

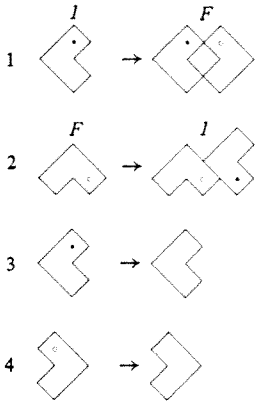


design in language

$T_9(G)$:

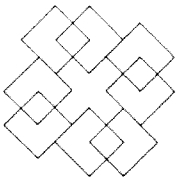


initial shape



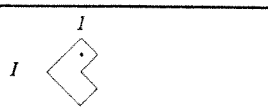
rules

final state: F

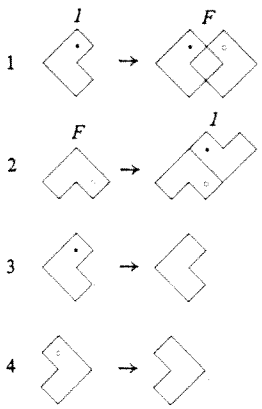


design in language

$T_{10}(G)$:

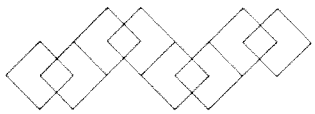


initial shape



rules

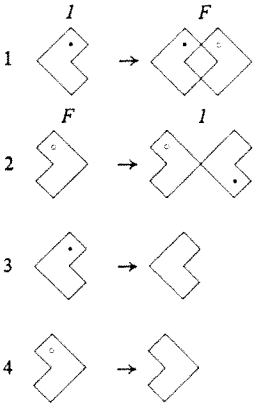
final state: F



design in language

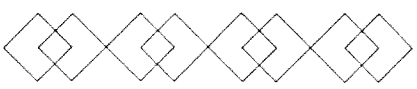
$T_{11}(G)$:

initial shape



rules

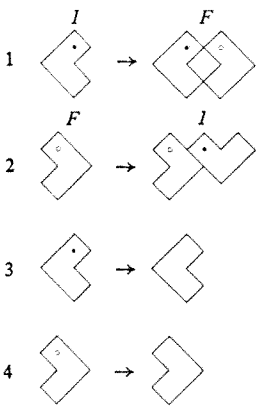
final state: F



design in language

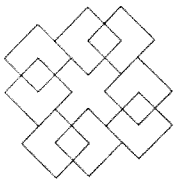
$T_{12}(G)$:

initial shape



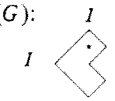
rules

final state: F

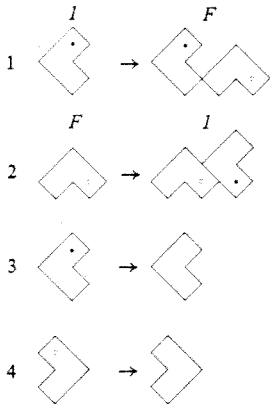


design in language

$T_{13}(G)$:



initial shape



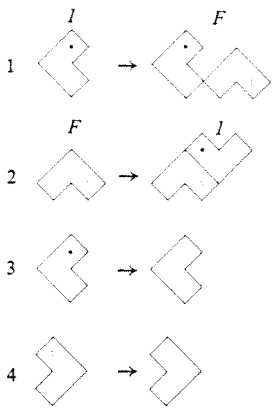
rules

final state: F

design in language

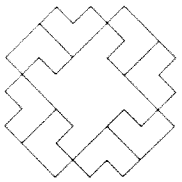
$T_{14}(G)$:

initial shape



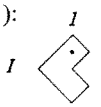
rules

final state: F

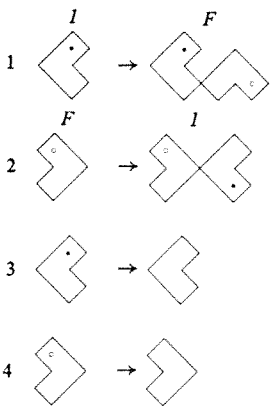


design in language

$T_{15}(G)$:

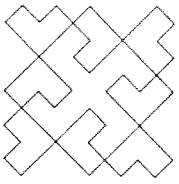


initial shape



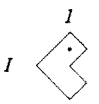
rules

final state: F

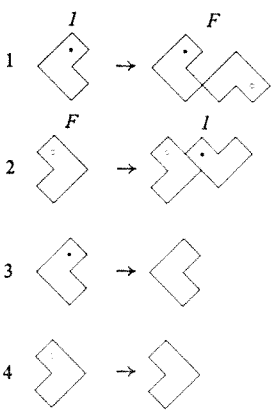


design in language

$T_{16}(G)$:

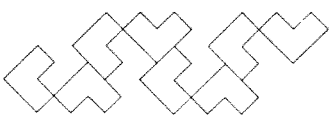


initial shape



rules

final state: F



design in language

The recursive structures of all the grammars shown in figure 13 are identical. Each is given by the set

{(*I*, rule 1), (rule 1, rule 2), (rule 2, rule 1), (*I*, rule 3), (rule 1, rule 4), (rule 2, rule 3)}.

The recursive structure of the original grammar is given by the same set (see figure 5). Clearly, the recursive structures of all new grammars are isomorphic to the recursive structure of the original one. This is illustrated graphically in figure 14 by representing the recursive structure of the original grammar and the recursive structure of any new grammar by directed graphs. Arrows from one structure to the other denote any transformation of rules in the original grammar into rules in a new grammar.

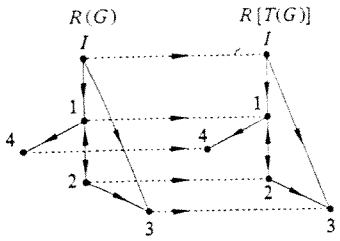


Figure 14. An isomorphism between the recursive structure of the grammar of figure 4(a) and the recursive structure of any of the transformations of it illustrated in figure 13. Both recursive structures are depicted as directed graphs. The number at each node in a graph is the number of a rule or initial shape of a grammar. Broken lines with arrowheads show the mapping from rules in the original grammar onto rules in a transformation of it.

Redundant rules in shape grammars

As demonstrated in the previous section, a state change rule or rules in A can apply to a spatial relation to generate more than one new spatial relation. Similarly, state change rules in B can apply to a pair of state labels to produce more than one new pair of state labels. Each of these different spatial relations and each of these different pairs of state labels determines a different transformation of a rule. However, only one transformation of a rule can be included in a transformation of a grammar.

In some circumstances, it may be desirable to include multiple transformations of a single rule in the same transformation of a grammar. Allowing this would permit greater freedom in defining the range of new languages of designs. To meet this end, multiple copies of a rule, or *redundant rules*, can be introduced into a grammar. Since they are exact duplicates of already-existing rules, redundant rules do not alter the language of designs which the grammar generates. Instead, they allow the same rule to be mapped onto different new rules in just one transformation. Thus, when redundant rules appear in a grammar, they represent diverse but coexistent ways of thinking about a single rule. Each copy of a rule corresponds to a different conception of how it can be changed.

A redundant rule is added to the grammar of figure 4(a) to produce the grammar $\bar{G}$ shown in figure 15(a). In figure 15(b), a transformation T of this grammar is illustrated. $T(\bar{G})$ is defined in terms of the set A given in figure 7, and the set B given in figure 11(b) together with a new rule $r_{2'}$: $(I, F) \rightarrow (F, I)$. A design in the language defined by $T(\bar{G})$ is also shown. The recursive structures of $\bar{G}$ and $T(\bar{G})$ are isomorphic. Both are given by the set

{(*I*, rule 1), (rule 1, rule 2), (rule 2, rule 1), (*I*, rule 3), (rule 1, rule 4),
(rule 2, rule 3), (rule 1, rule 2′), (rule 2′, rule 1), (rule 2′, rule 3)}.

Readers are invited to define other transformations of $\bar{G}$ in the family of transformations specified by the set A and the set B augmented as above. Sixty-four transformations are possible.

In general, a transformation of a grammar, either partial or total, with or without redundant rules, may be only one step towards the development of a new grammar. Transformations of the rules of different grammars as well as rules defined from scratch can be combined to produce the final grammar. The more elaborate example of a transformation given in part 3 of this paper (Knight, 1983b) is a good illustration of how a small but significant subset of a grammar is transformed to form the basis of a new one.

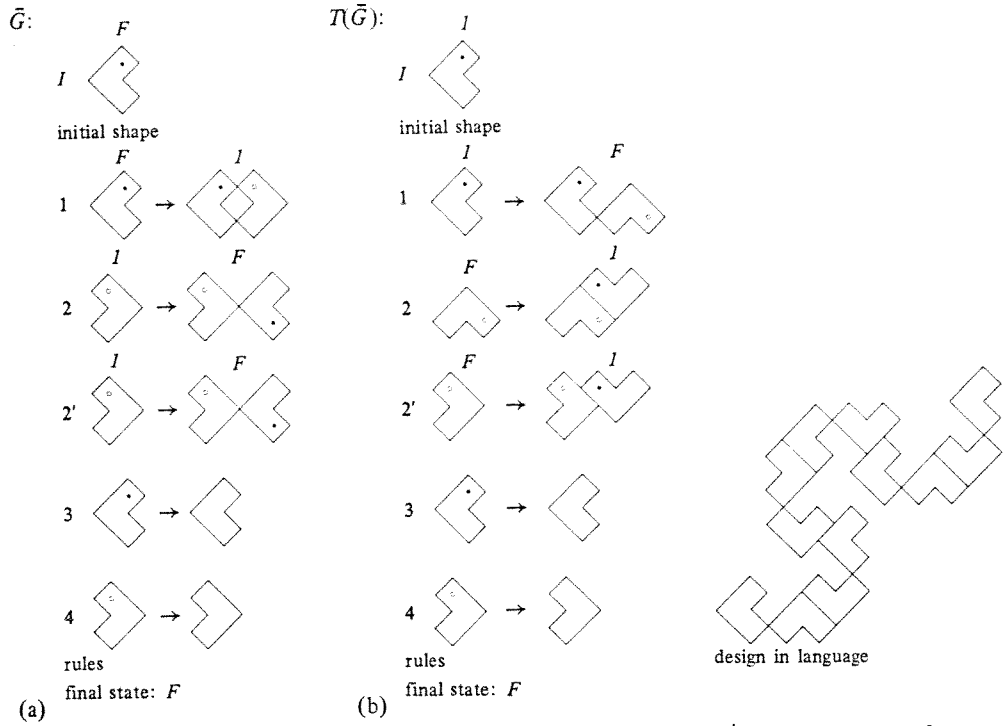


Figure 15. (a) A grammar $\bar{G}$ defined by adding a redundant rule, rule 2′, to the grammar of figure 4(a). (b) A transformation T of $\bar{G}$ determined by the set A given in figure 7, and the set B given in figure 11(b) together with a new rule $r_{2'}$: $(1, F) \rightarrow (F, 1)$. A design in the language generated by $T(\bar{G})$ is also shown.

References

Flemming U, 1981, "The secret of the Casa Giuliani Frigerio" *Environment and Planning B* **8** 87-96
Knight T W, 1980, "The generation of Hepplewhite-style chair-back designs" *Environment and Planning B* **7** 227-238
Knight T W, 1981a, "The forty-one steps" *Environment and Planning B* **8** 97-114
Knight T W, 1981b, "Languages of designs: from known to new" *Environment and Planning B* **8** 213-238
Knight T W, 1983a, "Transformations of languages of designs: part 1" *Environment and Planning B: Planning and Design* **10** 125-128
Knight T W, 1983b, "Transformations of languages of designs: part 3" *Environment and Planning B: Planning and Design* **10** 155-177
Koning H, Eizenberg J, 1981, "The language of the prairie: Frank Lloyd Wright's prairie houses" *Environment and Planning B* **8** 295-323
Stiny G, 1980a, "Introduction to shape and shape grammars" *Environment and Planning B* **7** 343-351
Stiny G, 1980b, "Kindergarten grammars: designing with Froebel's building gifts" *Environment and Planning B* **7** 409-462
Stiny G, Mitchell W J, 1978a, "The Palladian grammar" *Environment and Planning B* **5** 5-18
Stiny G, Mitchell W J, 1978b, "Counting Palladian plans" *Environment and Planning B* **5** 189-198