CAAD Futures '93
U.Flemming and S. Van Wyk (Editors)
© 1993 Elsevier Science Publishers B.V. All rights reserved

# Spatial Grammars: Motivation, Comparison, and New Results

*Ramesh Krishnamurti*
*Rudi Stouffs*

Department of Architecture
Carnegie Mellon University
Pittsburgh, PA 15213 USA

*The paper starts by giving a motivation for studying grammars in design and is based on considerations of style, discovery, and constructive techniques. This paper goes on to survey a variety of spatial grammar formalisms from an implementation standpoint. For each formalism, the salient computational issues pertaining to rule application are discussed. Two aspects of shape grammars are considered in detail: (a) the conditions for reversibility of shape rules, and (b) the recognition of planar shapes. An outline of subshape detection in $U_{23}$ is given.*

*Keywords: spatial grammars, shape rules, shape recognition.*

## 1 Introduction

Spatial grammars excite some and appall others.

Spatial grammars have been in vogue for some time now, though its adherents are few in number. This may be due, in part, to a different understanding of the role and usefulness of grammars in design research, and in part, to the difficulty of the computational issues that arise in grammars. In this paper, we suggest an alternative motivation for grammars in design, briefly survey some of the spatial grammar formalisms and present some new results on shape grammar computation.

Several authors have provided sound reasons for the usefulness of grammars in design research (e.g., Stiny, 1980b; Earl, 1986; Flemming, 1987; Coyne et al., 1990; Mitchell, 1990). Still, there remain those who are skeptical (e.g., Snodgrass and Coyne, 1990; Fleisher, 1992), though for misplaced reasons.

One source of the misconceptions associated with spatial grammars can be attributed directly to the word 'grammar,' which conveys a linguistic association in that spatial grammars ought to somehow be able to act as a reference by means of which designs can be analyzed, understood and specified in lexical, semantic, pragmatic, epistemic terms accompanied by all the ambiguity, superfluidity and indirection that one associates with common language. The unsurprising failure of spatial grammars in succeeding in this endeavour is chalked up as a 'win' for opponents of grammar systems. Grammars are just formal devices first used in logic by Post (1943); later, in natural language processing by Chomsky (1957), who coined the term; and more recently and originally, by Stiny (1980a) in design. Grammars should only be interpreted in their strict technical sense.

A second source of misconception has been, unfortunately, provided by some proponents and relates to the generative nature of grammars. The argument they advance is that by applying a given set of rules within some formal grammar one can crank out ‘new’ designs; a similar argument is raised by opponents in describing spatial grammars as being against creativity.

It is not the purpose of this paper to debate fully the merits and demerits of spatial grammars or to discuss other misconceptions. The above observations serve merely to illustrate a point. An answer to such criticisms might be better understood if one first distinguishes between design and designing. A design is simply the end of some process and may well be the beginning of another. An architect produces a plan, that a builder constructs, that an interior designer decorates, that a child by placing a toy or two gives character.

Designing, on the other hand, is the means to an end. The choice of any particular approach to designing is personal, though this choice may be influenced by the environment in which one is situated. The mistake is to presume that a characterization of the means extends naturally to a characterization of the ends or vice versa. Thus, one finds statements in the literature such as design is predictive, design is reflective action, design is search, design is constraint resolution, design is puzzle-making, design is hermeneutical, design is dialogue, design is grammatical, design is geometrical, design is object-oriented, design is optimization, design is knowledge-based … we could go on *ad infinitum* and *ad nauseam*. The plain fact is that none of these statements are true and all of them are true. Unlike theories in the natural sciences, approaches to designing are not competing theories though some may complement others.[1] Each approach has its pros and cons. The role of computer-aided design, seems to us, is not to vilify any particular theory of designing, but rather to question the assumptions that each makes, the restrictions that each imposes, and the ease or difficulty with which each facilitates the very act of designing. As Ömer Akin (1986) rightly remarked: “There are probably as many definitions of design as there are designers… ,” each merely posits a particular, perhaps shared, view of *designing*. Equally, it is the responsibility of proponents and not just opponents, to question any theory that is put forward to an audience of design researchers.

So why do we like grammars? What motivates us?

There are several reasons. Firstly, spatial grammars have been successful in analyzing styles of designs. It is conceivable that as humans we are inclined to repeatedly rely on our experience and our familiarity with certain known concepts and metaphors and apply them to our way of doing things. Through corpora of spatial designs, grammarians have clearly demonstrated that designers tend to employ a limited set of spatial relationships to produce seemingly distinctive designs. In other cases, grammarians have demonstrated the existence of spatial transformations from one style to another. This is not unlike the architectural historian who attempts to discover the principles underlying a given building or styles of buildings. In our own teaching, we encourage students to play with known styles using architectural history as the basis of their experiment. It is equally valid to experiment with different spatial relations on a grammatical basis. This is not to say that the grammarians’ explanations of style are any

[1] This is not to say that designs themselves are not subject to the laws of the natural sciences; that is an altogether different issue.

more accurate than those of architectural historians; nevertheless, the value of grammars for pedagogy cannot be disclaimed.[2]

Secondly, the act of designing is sometimes seen as an act of 'discovery.' It has been remarked by some researchers that they have been surprised by the forms and arrangements that they come across by playing with spatial forms and relationships through using interactive grammar systems. It is perhaps this same sense of discovery that motivates other researchers to look at emerging technologies such as hypermedia, multimedia and virtual reality.

Thirdly, if one accepts that spatial designs are represented by drawings, one can then examine the techniques by which drawings can be constructed. Euclid was the first to influence constructive techniques through his invention of the compass and straight edge. One has to just examine early architectural drawings for the truth of this remark. A history of Euclidean construction and constructibility can be found in (Eves, 1963). Briefly, towards the latter half of seventeenth century geometers such as Georg Mohr began to question Euclidean constructibility using just a compass. Mohr's result, originally in Danish and discovered in 1928 by Johannes Hjelmselev, was independently discovered by the Italian poet, Lorenzo Mascheroni in 1797. This work was complemented by Poncelot and Steiner to give their famous rusty compass construction theorem. All of this culminated in August Adler's famous turn of the century theorem on Euclidean constructibility using two straight edges be they parallel or intersecting. It is not an exaggeration to claim that Adler's work directly impacted on modern architectural drafting tools such as the Mayline and the architect's triangle. With the advent of computer graphics and Sutherland's (1963) work on Sketchpad, the use of geometric transformations have become part of the draftsman's arsenal. The story does not end here. If one accepts a pencil and eraser metaphor to drawings, then spatial rules are simply extensions of the metaphor. One uses the leaded side to add 'marks' onto paper and one uses the eraser to remove marks. A rule application is a simultaneous use of lead, eraser and geometrical transformations. One point to note is that all of these techniques have their basis in formal geometry. We believe that the next generation of constructive tools have their basis in formal grammar theory.

Lastly, grammar systems that are really useful are difficult to implement. This is a legitimate criticism of grammar systems. We should know; one of the authors was among the first to implement a spatial grammar system. Part of the difficulty stems from the technical considerations of implementing grammars, which we address in this paper. Part of the difficulty is developing ways of enabling designers to employ grammatical rules in a manner that does not impede their act of designing, which we do not address in this paper. However, the latter is not merely a matter of developing appropriate human-computer interfaces. The problem runs deeper and relates to the appropriateness of a given grammar formalism for a given design problem, and to the representational demands that grammar systems impose on users. Unfortunately, systems that are comparatively easy to use such as *discoverForm* (Carlson, 1989), and to some extent, *Tartan Worlds* (Woodbury et al., 1992) do not provide for rich classes of spatial designs; systems that are not so easy to use such as *SGI* (Krishnamurti, 1982) and *GENESIS* (Heisserman, 1991) do provide for richer classes of spatial designs. In recent articles,

[2] In a private communication, Ömer Akin offered another more powerful reason for studying grammars, especially in connection with issues of style and of the design process, namely, the question of *where do grammars come from?* This, in itself, is a design problem and relates to issues in cognitive psychology, a discussion of which is beyond our competence at present.

Woodbury and Griffiths (1993) describe their experiences with *GENESIS* in developing a grammar for fire-station layouts, and Carlson (1993) explores this problem from a programming standpoint. While the problem of bringing grammars to a wider audience of designers remains high on our priority, there are other issues that need to be tackled first.

In this paper we examine issues relating to grammar implementations. Spatial grammar formalisms share certain characteristics in common, which we explore from a computational standpoint. Section 2 introduces the necessary terminology. Section 3 briefly surveys three spatial grammar formalisms. Sections 4 through 6 describe shape grammars and provide new results on shape recognition.

## 2 Grammars: A Uniform Treatment

Grammars are formal devices for specifying ‘languages’ and just that. Grammars for generation and analysis have found wide ranging use in a variety of fields.

All grammars share certain definitions and characteristics. Grammars are defined over an algebra of objects, $U$, closed under the operations of ‘+’ and ‘–’ and a set of transformations $F$. In other words, if $u$ and $v$ are members of $U$, then so are $u + f(v)$ and $u - f(v)$ where $f$ is a member of $F$. In addition, we can define a match relation ‘$\leq$’ on $U$ such that $f(u) \leq v$ whenever $u$ occurs in $v$, for some member $f$ of $F$.

In general, there are four parts to a grammar $G = (N, T, R, I)$:

- nonterminal vocabulary $N$
- terminal vocabulary $T \subset U$
- finite set of rewriting rules (or *productions*) $R \subset \psi_1(T, N) \times \psi_2(T, N)$
- (set of) initial objects $I \subset \psi_1(T, N)$

$\psi_1$ and $\psi_2$ are set defining functions on $T$ and $N$. In general, the elements of the terminal and nonterminal vocabularies are members of $U$, though certain restrictions may apply.

A rewriting rule has the form $lhs \rightarrow rhs$. The *lhs* (left hand side) of the rule contains elements from $T$ and $N$, but cannot be empty. The *rhs* (right hand side) of the rule, in addition, may be empty. A rule applies to a particular object if the *lhs* of the rule ‘matches’ a part of the object under some allowable transformation $f$. Rule application consists of replacing the matching part by the *rhs* of the rule under the same transformation. A rule $a \rightarrow b$ is applicable to $u$ whenever there is a transformation $f$ such that $f(a) \leq u$, in which case, under rule application, the object $v$ is produced, given by the expression:

$$v = [u - f(a)] + f(b)$$

We denote a rule application by $u \Rightarrow v$ and say that *u directly derives v. A* sequence of rule applications from object $u$ to object $v$ is denoted by $u \Rightarrow^* v$; that is, u $\Rightarrow \ldots \Rightarrow v$. In this case, we say that *u derives v.*

The central problem in implementing grammars is the *matching problem*, that of determining the transformation $f$ under which the relation $\leq$ holds for $f(a)$. Clearly, this problem depends upon on the representation of the elements of $U$.

A grammar defines a *language*; that is, a set containing all objects generated by the grammar, where each generation starts with an initial object and uses rules to achieve an object that contains only terminals.

A grammar is *serial* if a rule is applied to just one instance of a matching object, and *parallel* if a rule is applied to all such instances. Most grammars are used serially.

A uniform characterization for a variety of grammar systems is given in (Gips and Stiny, 1980).

## 3 Spatial Grammars

In this section, we distinguish between three kinds of spatial grammars: string grammars, set grammars, and graph grammars.

### *3.1 String Grammars*

The algebra $U = \Sigma^*$, the least set of all strings over a set of symbols $\Sigma$, and includes $\varepsilon$, the string of length 0.

A common example of a string grammar systems is a phrase-structure grammar (Sudkamp, 1988), which is defined as the four-tuple $G_p = (N, T, R, I)$ over a vocabulary $V$, where $V = N \cup T$, $N \cap T = \emptyset$, $T \subset U$. The *lhs* and *rhs* of a rule are strings over $V$ with the *lhs* involving at least one symbol of $N$.

The operations of '+', '–' and the transformation $f$ have the following interpretation: Let $|u|$ denote the length of string $u$. $f$ takes a string $u$ and gives a pair $(u, i)$, where $i$ is an integer. Then,

| | |
|---|---|
| $f(u) \leq v$ | whenever $v = xuy$ and $i = \|x\|$ |
| $v + f(u) = xuy$ | whenever $v = xy$ and $i = \|x\|$ |
| $v - f(u) = xy$ | whenever $v = xuy$ and $i = \|x\|$ |

The integer $i$ is used as position index to indicate where the string $u$ starts. The operators, + and –, are respectively string insertion and deletion operators. $\leq$ is the substring relation. Under rule application, $u$ directly derives $v$, $u \Rightarrow v$, if $u = xay$, $v = xby$ and $a \rightarrow b$ is a member of $R$. The language generated by grammar $G_p$ is $L(G_p) = \{ v \mid v \in T^* \text{ such that } I \Rightarrow^* v\}$, where $T^*$ is the least set of the terminal vocabulary closed under string concatenation.

String grammars lend themselves naturally to classification according to the form of the rules. A *regular* string grammar only allows rules of the forms $A \rightarrow aB$ and $A \rightarrow b$, where $A, B \in N$ and $a, b \in T$. A *context-free* string grammar has rules of the form $A \rightarrow b$, where $A \in N$ and $b \in V^* - \{\varepsilon\}$. A *context-sensitive* grammar has rules that take the form $xAy \rightarrow xby$, where $A \in N$ and $x, y, b \in T^*$ with $b \neq \varepsilon$. Otherwise a string grammar is *unrestrictive*. A similar classification can be specified for certain non-string grammars.

*Example*

The *Tartan Worlds* (Woodbury et al., 1992) is an example that bestrides string and set grammars, which are considered in section 3.2, in which each symbol corresponds to a geometrical entity represented as a graphical icon. The grammar can be used either serially or in parallel. We consider a simplified string grammar version of the *Tartan Worlds.*

The *Simple Tartan World* is represented by a grid of cells $(x, y)$, $x \geq 0$, $y \geq 0$. For any string $v$, we can define an assignment function $g$ that gives a list of triples $<x, y, icon>$ as follows:

$g(v) = g(a), (x(i), y(i), icon(u))$ whenever $v = au$, $|u| = 1$ and $|a| = i$.

$g(\varepsilon)$ gives an empty triple. The comma ‘,’ operator separates the different elements in a list. For each symbol in a string, $g$ identifies its tartan grid location and associated graphical icon.

Each rule in the *Simple Tartan Worlds* consists of one symbol on the *lhs* and symbols on the *rhs* given in their spatial relation. For each rule $l \rightarrow r$, we can define $g$ as $g(l) \rightarrow g(r)$, where the $(x, y)$ coordinates of the elements in $r$ are relative to the $(x, y)$ coordinate of the symbol $l$. Thus, under rule application the new string, $w$, is given by

$$w = [v - f(l)] + f(r)$$

and its $g$ value, $g(w)$, updated to,

$$g(w) = g_u, \{(x+x_a-x_l, y+y_a-y_l, i) \mid (x,y,i) \in g(r), g(l) = (x_l, y_l, \_), g(a) = (x_a, y_a, \_)\}, g_t$$

where $v = uat$ and $g(v) = (g_u=g(u)), g(a), (g_t=g(t))$.

Note that, in the *Tartan Worlds*, a grid cell may be occupied by several symbols, the order of display depending on the order in the rule. Note too that different symbols may map onto the same graphical icon. In its full generality, the *Tartan Worlds* allows many symbols on the left hand side of a rule. This situation is better considered as a set grammar.

### 3.2 *Set Grammars*

As the name implies, set grammars deal with objects expressed as sets of entities.

#### 3.2.1 *Structure Grammars*

Structure grammar is one such example (Carlson et al., 1991). Here an object, referred to as a *structure*, is represented as a set of pairs $\{ (a, h), \ldots \}$, where $a$ is drawn from a set of symbols $V$ and $h$ is drawn from a group of transformations $F$. Thus, the algebra, $U$, under consideration is the power set, $\wp(V \times F)$. The nonterminals are drawn from a similar algebra of symbols and the same group of transformations, with the provision that the nonterminal and terminal symbols are not shared.

The operations of ‘+’ and ‘–’ correspond to the familiar set operations of union, $\cup$, and difference, $\setminus$ . The relation, $\leq$, corresponds to the subset relation. A transformation $f$ on a structure $\{ (a, g), (b, h), \ldots \}$ produces the structure $\{ (a, fg), (b, fh), \ldots \}$, where $fg$, $fh$, … are members of $F$, by virtue of the group property of $F$. Thus, rule application in a structure grammar yields the set:

$$[u \setminus f(a)] \cup f(b) \text{ provided } f(a) \subset u.$$

A structure grammar $G_s$ is the 4-tuple $(T, N, R, I)$, where $I$ is the initial structure drawn from the power set, $\wp((T \cup N) \times F)$. The language $L(G_s)$ is the set of all structures that do not contain nonterminals.

Structure grammars are especially useful when one can establish a 1-1 correspondence between symbols and spatial icons, though the attractive feature of structure grammars is that, in its full generality, it is independent of any spatial connotations associated with either the symbols or the group of transformations and thus, can also be applied to problems without spatial content. Another feature of set grammars that distinguishes them from string grammars is that the symbols are order independent.

Note that the *Tartan Worlds* considered previously as a string grammar extension can be considered as a structure grammar where $F$ is the infinite group of translations.

### *3.2.2 Solid Grammars*

Solid grammars (Heisserman, 1991) may be considered as constrained set grammars. Basically, a solid is represented as a collection of faces, each of which is described by its edges and vertices that are mapped onto ($x$, $y$, $z$) coordinates. Thus, a solid is given by a set of triples $\{( f_i, E_i, V_i)\}$, where

$$\sum_i [|V_i| - |E_i| + 1] = 2 \ .$$

In addition, there are other constraints on the topology and geometry that have to be satisfied in order for the set to represent a solid (see Mäntylä (1988) on boundary representations). A face may be augmented with labels, which are treated as nonterminals. In principle, a solid rule $a \rightarrow b$ serves to replace one or more faces by a collection of faces with proviso that the resulting set satisfies the requirements of a solid. The operations of '+' and '–' do not have unique interpretations but correspond to collections of Euler operators (Mäntylä, 1988) such that the resulting object is a proper solid. Thus, '+' and '–' are elements of sets of functions made up of sequences of Euler operators. Let $\tau$ denote a truth value functional that holds whenever the set represents a proper solid. Whence, rule application is given by $+(-(u, f(a)), f(b))$, where $f$ is a geometrical transformation such that $\tau(+(-(u, f(a)), f(b)))$ is TRUE.

It should be noted that a solid grammar manipulates data structures that represent the faces, edges, and vertices of a solid. In his original formulation, Heisserman describes solid grammars in terms of the graph of the boundary representation augmented by labels and states. A solid rule then translates to one of a predicated rule, where the matching condition is captured by truth value functionals on the elements of the graph and the replacement is effected by a sequence of primitive Euler operations such that the resulting graph is a boundary representation of some solid possibly augmented with labels and states. The matching condition reflects the existence of some subgraph of the boundary representation and replacement is given by a sequence of data structure operations. In other words, a solid grammar can be treated as a graph grammar, which is considered in the next section.

### *3.2.3 Computational Considerations*

In its simplest form, rule application involves set operations under the identity transformation. In structure grammars, the transformation is an element of a group. The matching relation is the subset relation and can be determined in linear time. For complex extensions of set grammars such as solid grammars, rule application is described in different terms. Here, the matching relation is absorbed into the *lhs* of a rule that serves as the 'antecedent' for some action. The *rhs* of the rule is then the 'consequent' action and is expressed as a sequence of operations on the representation of the solid. The complexity of rule application depends on the complexity of determining the antecedent condition.

## *3.3 Graph Grammars*

Graph grammars are particularly useful if connectivity or incidence between elements is the dominant feature of the design problem. Graph grammars can be defined in a number of ways depending on the application; here, we follow Pavlidis (1972).

A nonterminal graph structure of the *m*th order is an entity that is connected to the rest of the graph by *m* nodes. A graph structure may be termed a *node structure* (1st order), a *branch structure* (2nd order), a *triangle structure* (3rd order) or, in general, a *polygon structure*. The vocabulary of terminals consists of nodes and branches.

An *m*th order context-free graph grammar is given by $G_g = (N, T, R, I)$, where $N$ is a set of *m*th order nonterminal graph structures. The *lhs* of a rule is a single nonterminal graph structure, the *rhs* is a graph containing possibly both terminals and nonterminals, and when a rule is applied, all the nodes connecting the structure with the rest of the graph are treated uniformly. An *m*th order context-free graph grammar is an *m*th order linear graph grammar if the *rhs*'s of the rules each contain at most one nonterminal structure.

A rule applies to a graph if its *lhs* is isomorphic to a nonterminal structure in the graph. The problem of finding a subgraph isomorphic to the *lhs* of the rule can be accomplished by some guided graph traversal such as depth- or breadth-first search. In the application, the matched nonterminal structure is replaced by the *rhs* of the rule, which is connected to the rest of the graph through exactly the same *m* nodes as the nonterminal structure was. The transformation *f* under which the rule applies is simply an assignment of coordinates to the nodes of the *lhs* and *rhs*.

*Example*

The trestlelike graphs illustrated in Figure 1 can be generated by the following linear graph grammar $G_g$. Let $n$ denote a terminal node, $b$ a terminal branch and $B$ a nonterminal branch structure. $K_4$ denotes the complete graph on four nodes with just one nonterminal branch (Figure 1a).

$$G_g = (N, T, R, I)$$
$$N = \{B\},\ T = \{b, n\},\ R = \{B \rightarrow b,\ B \rightarrow K_4\},\ I = \{K_4\}$$

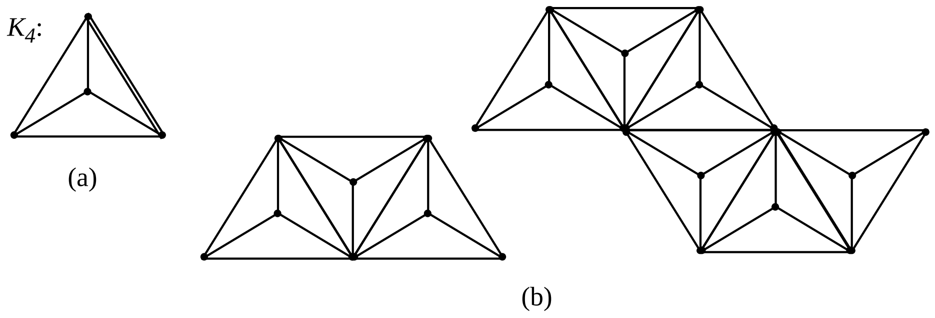

Figure 1. Objects in the grammar $G_g$: (a) the initial graph $K_4$ (the double edge denotes the nonterminal branch structure $B$); (b) two sample graphs generated with the grammar.

Other variations of graph grammars employ similar matching and replacement principles, though for general graphs the subgraph isomorphism problem is *NP*-complete, even for special kinds of graphs (Garey and Johnson, 1979). Despite this, the matching algorithm might be made simpler if the graphs do possess specific features.

## 4 Shape Grammars

For the remainder of this paper we discuss shape grammars in detail and present results on two aspects: (a) the reversibility of shape rules, and (b) the recognition of shapes made up of planar segments. A *shape grammar* is a formal rewriting system for producing languages of shapes (Stiny, 1980a). Unlike the other spatial grammars, shape grammars operate directly on spatial forms.

### *4.1 Shape*

A *shape* is defined as a finite arrangement of spatial elements from among points, lines, planes, volumes, or higher dimensional hyperplanes, of limited but non-zero measure. A shape is considered an element of an algebra $U$ that is ordered by a part relation and closed under the operations of sum and difference, and the Euclidean transformations augmented with scale (Stiny, 1991). If the shapes are defined in a $k$-dimensional space, $k \geq n$, $U_{n,k}$ denotes the set of all finite arrangements of $n$-dimensional hyperplanes of limited but nonzero measure in $k$-dimensional space. If $k$ is unambiguously understood, it may be dropped and $U_{n,k}$ can be referred to as $U_n$. Thus, $U_0$ refers to the algebra of points, $U_1$ the algebra of lines, $U_2$ the algebra of planes and $U_3$ the algebra of volumes. A shape may consist of more than one type of spatial element, in which case it belongs to the algebra given by the Cartesian product of the algebras of its spatial element types.

A shape is a part of another shape if it is embedded in the other shape as a smaller or equal element. A part of a shape is also called a *subshape*, and specifies the relation, $\leq$. The operation of *sum* combines two shapes, and the operation of *difference* takes the relative complement of one shape with respect to another. These operations can be defined in terms of the part relation as follows:

The sum of two shapes $a$ and $b$ is a shape $c = a + b = b + a$, such that $a$ and $b$ are parts of $c$ and any part of $c$ is either a part of $a$ or $b$, or can be divided into two parts, of which one belongs to $a$ and the other belongs to $b$.

The difference of two shapes $a$ and $b$, in that order, is a shape $c = a - b$, such that $c$ is a part of $a$, any part of $c$ is not a part of $b$, and the sum of $b$ and $c$ equals the sum of $a$ and $b$. The algebra has a zero given by the empty shape.

The definitions of sum and difference can also be expressed in terms of set operations on point sets. The operation of sum is equivalent to the point set operation of union and the operation of difference to the point set operation of difference or relative complement. The *product* $a \cdot b$ of two shapes $a$ and $b$ is equivalent to the point set operation of intersection; $a \cdot b = b \cdot a = a - (a - b) = b - (b - a)$. We define the *symmetric difference* of two shapes $a$ and $b$ as $a \oplus b = b \oplus a = (a - b) + (b - a)$. Figure 2 illustrates the specification of shapes in $U_3$.

The algebra $U_n$ satisfies the axioms of a boolean ring under symmetric difference and product. Shapes in algebra $U_n$ have their boundaries in algebra $U_{n-1}$ (Stiny, 1991; Krishnamurti, 1992). A shape $a$ is said to *contain* a shape $b$ if $b$ is a part of $a$. Two shapes *overlap* if they have a common part, that is there exists a nonzero element of $U_n$ that is a part of both shapes, and neither shape contains the other. Two shapes *share boundary* if they do not overlap, but their boundaries overlap in $U_{n-1}$. Otherwise, the two shapes are known to be *disjoint*.

### *4.2 Augmenting Shapes*

A shape can be augmented by distinguishing certain points, which introduce additional spatial relations. This is usually done by labelling these points. Let $L$ be a set

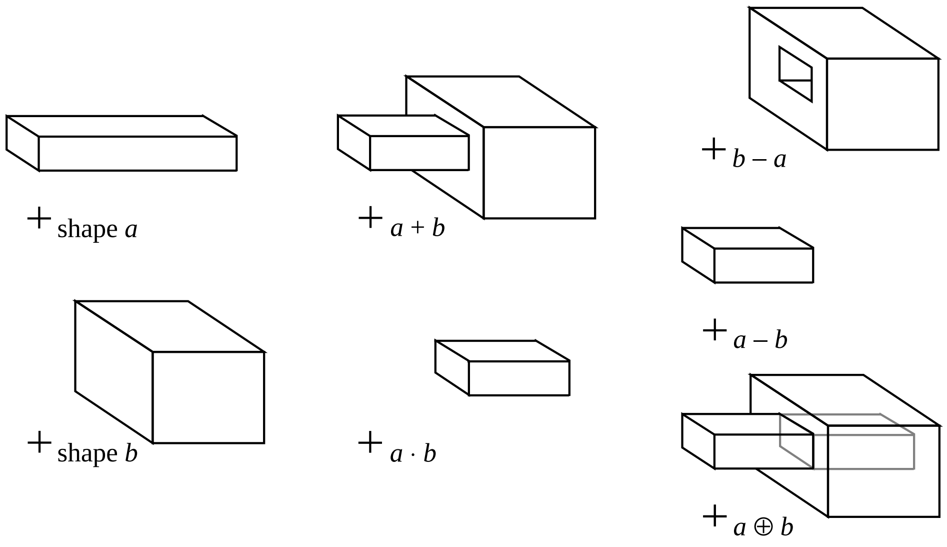

Figure 2. Two shapes $a$ and $b$, their sum, differences, product and symmetric difference; all shapes in $U_3$.

of symbols, which may be empty. Then, we can define a set $V_0 = U_0 \times 2^L$. Thus, a point is labelled if it has a set of symbols associated with it.

A *labelled shape* is an element of $V = U \times V_0$. The algebra $V$ has the same property as $U$. A labelled shape is made up of a shape and a finite, but possibly empty, set of labelled points. The set $(S, L)^+$ is the set of all labelled shapes made up of shapes in the set $S \subset U$ and labels in the set $L$. If the empty shape is included, then the set is denoted $(S, L)^*$.

A shape grammar $G_s$ is defined as a four-tuple $G_s = (S, L, R, I)$; $S$ represents the terminal vocabulary; the set of symbols $L$ specifies the nonterminal vocabulary; $R$ is a finite set of shape rules in the form $a \rightarrow b$, where $a \in (S, L)^+$ and $b \in (S, L)^*$; and $I \in (S, L)^+$ is the initial shape. The vocabulary of $G_s$ equals $(S, L)^*$.

A shape rule $a \rightarrow b$ is a spatial relation between shapes $a$ and $b$; it applies to a shape $s$ if we can find a transformation $f$ such that $f(a) \leq s$, in which case $f(b)$ replaces $f(a)$ in $s$ under rule application. That is, when the shape rule is applied to the shape $s$ it produces the shape, $s - f(a) + f(b)$. A shape grammar $G_s$ defines a language $L(G_s)$, the set containing all shapes generated by the grammar $G_s$ that have no symbols associated with them.

$$L(G_s) = \{ x \mid x \in U \text{ such that } I \Rightarrow^* x \}.$$

## 5 Reversible and Irreversible Rules

String grammars and graph grammars share a common characteristic in that their rules are reversible. That is, for any rule in such a grammar, a reverse rule can be constructed that, when applying, consecutively, the original and the reverse rule to any element of the algebra over which the grammar is defined, the result is identical to the original element. Furthermore, the reverse of a rule $a \rightarrow b$ is the inverse rule $b \rightarrow a$.

That such is the case is easily demonstrated for string grammars. A rule $a \rightarrow b$ applies to a string $u = xay$ and forms the string $v = xby$. The inverse rule $b \rightarrow a$ then applies to the string $v$ to form the original string $u$. A similar proof can be developed for graph grammars. However, this characteristic does not apply to either set grammars or shape grammars. A set grammar rule or shape rule may be either reversible or irreversible, depending on whether *rhs* ≤ *lhs* or not, respectively. We prove this for shape rules below.

Intuitively, we note that when combining two sets or shapes under the respective operation of '+', identical elements 'merge.' That is, only a single occurrence of the element appears in the resulting set or shape. Formally, this means that in the case of set grammars, if $|u|$ denotes the cardinality of a set $u$, $|u + v| \leq |u| + |v|$. In the case of string or graph grammars, given the appropriate definition of the size of a graph, this would constitute strict equality. No comparable measure exists for shapes. The irreversibility of shape rules, in general, is proven below.

*Theorem 1: Given any shape rule $a \rightarrow b$ with $b$ not a part of $a$, no shape rule $x \rightarrow y$ exists such that, for any shape $u$, if $u \Rightarrow v$ under rule $a \rightarrow b$ and a transformation $f$, then $v \Rightarrow w$ under rule $x \rightarrow y$ and the same transformation $f$.*

*Proof:* Assume that, given a rule $a \rightarrow b$ and a transformation $f$, there exists a rule $x \rightarrow y$, which may be equal to $b \rightarrow a$, such that for any shape $u$ with $f(a) \leq u$, the shape that results from applying to $u$ the rules $a \rightarrow b$ and $x \rightarrow y$, in that order and under the same transformation $f$, equals $u$. Note that two shapes are considered equal if they are a part of each other, that is, if one is embedded in the other as an equal element. We may assume that the rule $x \rightarrow y$ applies under the same transformation: if otherwise, we can always transform the rule such that it applies under the same transformation, without changing the rule application nor its scope. Thus,

$$u \Rightarrow v \Rightarrow w \text{ with } v = u - f(a) + f(b) \text{ and}$$
$$w = v - f(x) + f(y) = u - f(a) + f(b) - f(x) + f(y) = u.$$

Firstly, take $u \leftarrow f(a)$. Then, $u - f(a) = \varnothing$ and $f(b) - f(x) + f(y) = f(a)$ or, identically, $b - x + y = a$. From the definition of sum, it follows that $y \leq a$ and $(b - x) \leq a$. Since $b \nleq a$, it must be that $x \nleq a$.

Secondly, take $u \leftarrow f(a) + f(b)$. Then, because sum and difference define a distributive lattice on $U$,

$$\begin{aligned} u - f(a) &= (f(a) + f(b)) - f(a) \\ &= (f(a) - f(a)) + (f(b) - f(a)) \\ &= \varnothing + (f(b) - f(a)) \\ &= f(b) - f(a). \end{aligned}$$

From the definition of difference ($s - t \leq s$) it follows that

$$u - f(a) + f(b) = f(b) - f(a) + f(b) = f(b).$$

Thus, $u = f(b) - f(x) + f(y) = f(a) + f(b)$ or $b - x + y = a + b = b + a$. If we add $a$ to both sides of the equation we obtain $b - x + y + a = b + a + a$ or $b - x + a = b + a$ ($y \leq a$), which is impossible, unless $x \leq a$. But we know from above that $x \nleq a$.❑

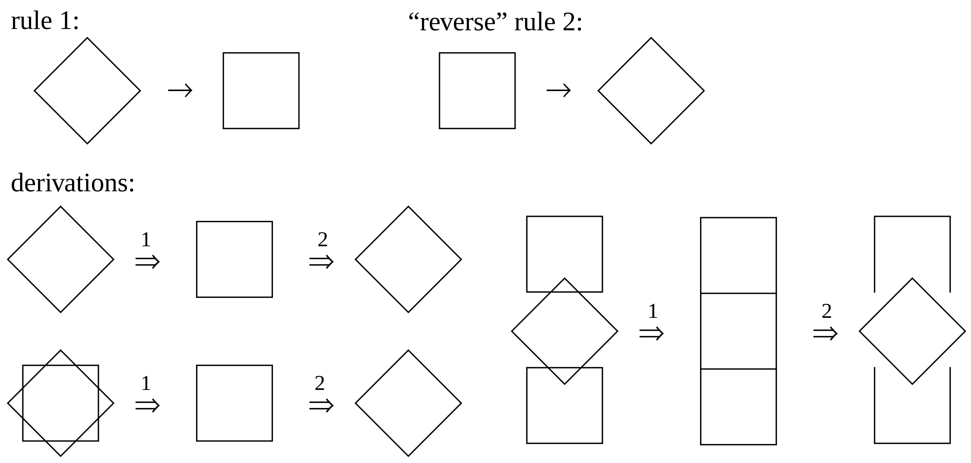

Figure 3. An example of an irreversible rule $a \rightarrow b$ (rule 1). We observe from the exemplar derivations that when applying the rules $a \rightarrow b$ and $b \rightarrow a$ (rule 2) subsequently and under the same transformation, the resulting shape may not equal the original shape.

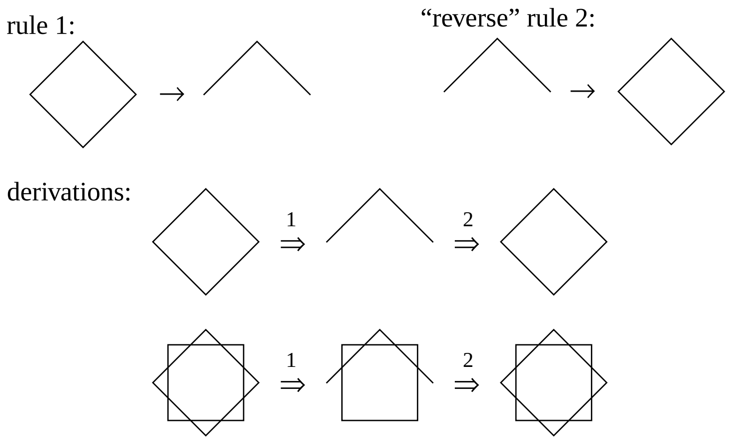

Figure 4. An example of a reversible rule $a \rightarrow b$ (rule 1). We observe from the exemplar derivations that when applying the rules $a \rightarrow b$ and $b \rightarrow a$ (rule 2) subsequently and under the same transformation, the resulting shape equals the original shape.

*Theorem 2: A shape rule $a \rightarrow b$ is reversible if and only if $b \leq a$; the reverse rule is $b \rightarrow a$, or $f(b) \rightarrow f(a)$ for any transformation $f$.*

*Proof:* We already know that a rule $a \rightarrow b$ is irreversible if $b \nleq a$. It follows that, if $a \rightarrow b$ is reversible, $b$ must be a part of $a$.

The following proves the converse, that is, if $b \leq a$ then $a \rightarrow b$ is reversible and the reverse rule under the same transformation is $b \rightarrow a$. Given the shape rules $a \rightarrow b$ and $b \rightarrow a$, and given any shape $u$ with $f(a) \leq u$, the shape that results from applying to $u$ the rules $a \rightarrow b$ and $b \rightarrow a$, in that order and under the same transformation $f$, is

$$w = u - f(a) + f(b) - f(b) + f(a).$$

We have that $b \leq a$ and, therefore, $f(b) \leq f(a)$. Then, for any shape $s$, $s - f(b) + f(a) = s + f(a) = s + f(b) + f(a)$. Thus,

$$\begin{aligned} w &= u - f(a) + f(b) \underline{- f(b) + f(a)} \\ &= u - f(a) \underline{+ f(b) + f(a)} \\ &= u - f(a) + f(a) = u + f(a) = u.❑ \end{aligned}$$

The theorem above gives a sufficient condition for the reversibility of a shape rule independent of the shape to which it is applied. We can state a weaker condition for reversibility that is dependent on the shape under application.

*Theorem 3: A shape rule $a \rightarrow b$ applied to a shape $u$ under a transformation $f$ is reversible if and only if $f(b{-}a) \cdot u = \emptyset$.*

*Proof:* Given the shape $u$ and transformation $f$, the shape rule $a \rightarrow b$ and its inverse rule $b \rightarrow a$, with $f(a) \leq u$, the shape that results from the application of rules $a \rightarrow b$ and $b \rightarrow a$, in that order, is $w = u - f(a) + f(b) - f(b) + f(a)$. We may assume that $b \not\leq a$, and therefore, $b = (b{-}a) + (b \cdot a) = (b \cdot a) + (b{-}a)$. Hence,

$$\begin{aligned} w &= u - f(a) + f(b) - f(b) + f(a) \\ &= u - f(a) + f(b \cdot a) + f(b{-}a) - f(b{-}a) - f(b \cdot a) + f(a) \\ &= u - f(a) + f(b \cdot a) + f(b{-}a) - f(b{-}a) + f(a) \text{ since } f(b \cdot a) \leq f(a) \\ &= u - f(a) + f(b \cdot a) - f(b{-}a) + f(a) \\ &\qquad\qquad \text{since } s + f(b{-}a) - f(b{-}a) = s - f(b{-}a) \text{ for any shape } s \\ &= u - f(a) + f(b \cdot a) + f(a) - f(b{-}a) \text{ since } f(b{-}a) \cdot f(a) = \emptyset \\ &= u - f(a) + f(a) - f(b{-}a) \text{ since } f(b \cdot a) \leq f(a) \\ &= u + f(a) - f(b{-}a) \text{ since } f(a) \leq f(a) \\ &= u - f(b{-}a) \text{ since for rule application } f(a) \leq u. \end{aligned}$$

Thus, $w = u$ if and only if $f(b{-}a) \cdot u = \emptyset$.❑

Irreversibility of shape rules distinguishes shape grammars from most other spatial formalisms. The significance from a design standpoint, to us, at any rate, stems from the fact that designing is a temporal activity. The irreversibility of a rule has the effect of time stamping each rule application and, thus, capturing design 'intent' at any given time.

## 6 Subshape Detection

A rule in a grammar applies to a member of the algebra over which the grammar is defined, if an 'occurrence' of the *lhs* exists in the member under a valid transformation.

This relation, ≤, is termed substring, subset, subgraph or subshape, respectively, for the grammars systems presented in this paper. Substring detection consists of a linear search in a given string for a matching pattern string. Similarly, subset and subgraph detection consist of a search of either a single entity (in the case of context-free grammars) or a group of entities within a set or a graph. Such a search is straightforward, i.e., it requires a one-to-one matching of entities that are identical under a certain transformation, even though not always computationally efficient, e.g., the subgraph isomorphism problem is *NP*-complete. Therefore, these grammars may be termed object-oriented; not so shape grammars.

A prime requirement for shape grammar computation is that *any* subshape of a shape is spatially replaceable. That is, a shape, even though with definite description, has indefinitely many 'touchable' parts; a shape is an *individual*[3] (Stiny, 1982), and this is reflected in the part relation defined on shapes. Subshape detection consists of finding one or all valid transformations under which the *lhs* of the rule is a part of a given shape. This problem is composed of finding a correspondence between the spatial elements in the *lhs* and elements of the given shape, and determining the transformation that represents this correspondence.

For $U_0$ and $U_1$ *distinguishable points* serve as the basis for reducing the problem (Krishnamurti, 1981; Chase, 1989; Krishnamurti and Earl, 1992). The application of this concept to $U_k$, for $k > 1$, is highly probable. In $n$ dimensions, a correspondence between $n$+1, not co-hyperplanar, distinguishable points uniquely determines a linear transformation, that is a valid transformation if the corresponding 'point figures' are similar. Otherwise, if no $n$+1, not co-hyperplanar, distinguishable points can be determined in the *lhs*, then, an indeterminate number of valid transformations may exist.

### *6.1 Shape Recognition in $U_{23}$*

We present new results on shape recognition for shapes in $U_{23}$. Shape recognition problems in $U_{23}$ can be reduced to corresponding problems in $U_{13}$; consequently, we can use recently published results (Krishnamurti and Earl, 1992). We outline the cases.

*Case 1: There are four planes, at most two are parallel and they do not intersect in a single point.*

We can determine the intersecting lines for each pair of planes, of which there are at least five (one pair may be parallel). Since the planes do not intersect in a single point, neither do the lines. Thus, we can find two skew lines and reduce the problem consequently to the case of two skew lines in $U_{13}$, for which there exists a determinate number of valid transformations (see case 5(a) in Krishnamurti and Earl (1992)).

*Case 2: There are three planes, and not all the lines of intersection are parallel.*

Another way to formulate this is the following: *There are three planes, and their normal vectors are linearly independent.* As a consequence, no two planes are parallel. Then, all the lines of intersection intersect in a single point, and the problem reduces to the case in $U_{13}$ when all lines are coincident at a common point and not all are colinear, for which there exists an indeterminate number of valid transformations (under scaling)

[3] The concept of individuals differs from the generally accepted concept of classes (or sets) in that no subdivision into subclasses or members is established or suggested *a priori* (Leonard and Goodman, 1940).

(see case 5(c) in Krishnamurti and Earl (1992)).

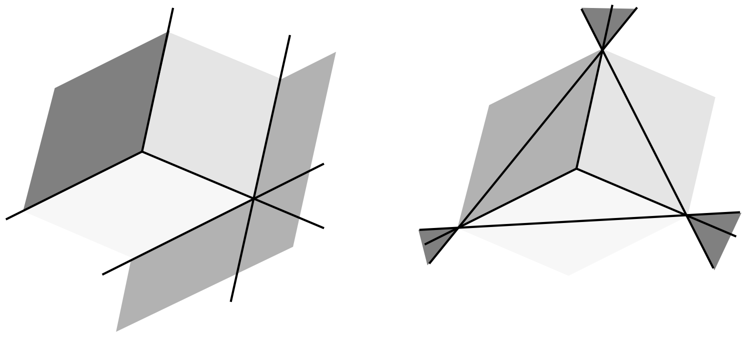

Figure 5. Examples illustring the determinate Case 1.

*Case 3: There are three planes and no two are coplanar.*

This problem reduces to the case in $U_{13}$ when all lines are parallel and not all are colinear, for which there exists either a determinate or an indeterminate number of valid transformations (under translation) (see case 5(d) in Krishnamurti and Earl (1992)).

*Case 4: There are two non-parallel planes.*

This problem reduces to the case in $U_{13}$ when all lines are colinear, for which there exists an indeterminate number of valid transformations (under translation and scaling) (see case 5(e) in Krishnamurti and Earl (1992)).

*Case 5: All planes are parallel.*

There exist an indeterminate number of valid transformations (under rotation and translation). The base transformation maps the carriers of two planes in the *lhs* onto the respective carriers of two planes in the given shape.

*Case 6: All planes are coplanar.*

There exist an indeterminate number of valid transformations (under rotation, translation, and scaling). The base transformation maps the carrier of a plane in the *lhs* onto the carrier of a plane in the given shape.

Cases 2 through 6 are illustrated in Figure 6. We can reduce the indeterminacy inherent in shape recognition by considering the problem in the algebras of labelled shapes, $V_0$, $V_1$ and $V_2$. Following similar arguments to those outlined above, shape recognition in $V_{23}$ can be reduced to the problem in $V_{13}$, which, in turn, can be reduced to the problem in $V_{03}$. We are still working on the cases for $V_{23}$. The cases for $V_{03}$ and $V_{13}$ have been discussed elsewhere (Krishnamurti, 1981; Krishnamurti and Earl, 1992).

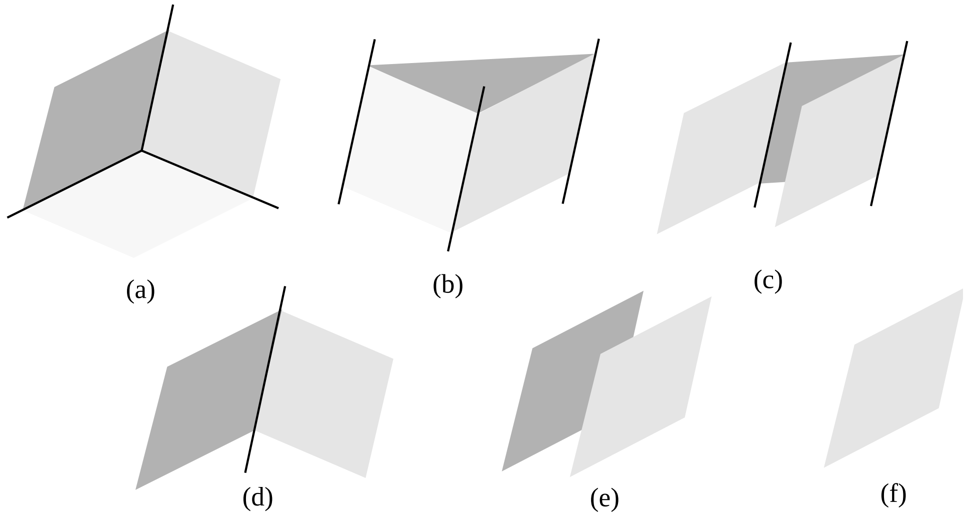

Figure 6. Examples illustrating the indeterminate cases: (a) Case 2, (b, c) Case 3, (d) Case 4, (e) Case 5 and (f) Case 6.

## 7 Conclusion

A framework for grammar implementations for a variety of formalisms has been presented and the main computational issues that arise in each discussed. An outline of the rule application is given for each formalism, though details of actual implementations are omitted. These would depend on the specific representation used, algorithms for which, in most cases, can be found in standard computer science texts on data structures. The formalisms can be used in at least two ways: as a set of *a priori* rules that can be used to produce spatial designs, or as a set of rules that are dynamically specified. In either case, rule application is identical.

In our introduction, a case was made for a new theory of ‘shape editing,’ the ability to interactively apply constructive techniques for the creation and manipulation of drawing documents. We believe that spatial rules, based on the precepts of formal grammars, provide the proper basis for work in this direction. We further believe that the research reported here lends credible support for this belief.

**Acknowledgements**

We would like to thank the anonymous referees for their constructive comments.

## References

Akin, Ö., 1986. “A Formalism for Problem Restructuring and Resolution in Design,” *Environment and Planning B: Planning and Design* 13, pp. 223-232.
Carlson, C., 1989. *The Explorer’s Guide to discoverForm*™. Manual, Center for Art and Technology, Carnegie Mellon University, Pittsburgh, PA.

Carlson, C., 1993. “Describing Spaces of Rectangular Dissections via Grammatical Programming,” in U. Flemming and S. Van Wyk (eds.) *Proceedings of CAAD Futures ’93*.

Carlson, C., McKelvey, R., and Woodbury, R.F., 1991. “Introduction to Structure and Structure Grammars,” *Environment and Planning B: Planning and Design* 18, pp. 417-426.

Chase, S.C., 1989. “Shapes and Shape Grammars: From Mathematical Model to Computer Implementation,” *Environment and Planning B: Planning and Design* 16, pp. 215-242.

Chomsky, N., 1957. *Syntactic Structures*. The Hague: Mouton.

Coyne, R.D., Rosenman, M.A., Radford, A.D., Balachandran, M., and Gero, J.S., 1990. *Knowledge-Based Design Systems*. Reading, MA: Addison-Wesley.

Earl, C.F., 1986. “Creating Design Worlds,” *Environment and Planning B: Planning and Design* 13, pp. 177-188.

Eves, H., 1963. *Survey of Geometry: Volumes 1 and 2*. Boston: Allyn and Bacon.

Fleisher, A., 1992. “Grammatical Architecture?” *Environment and Planning B: Planning and Design* 19, pp. 221-226.

Flemming, U., 1987. “The Role of Shape Grammars in the Analysis and Creatiuon of Design,” in Y.E. Kalay (ed.) *Computability of Design*. New York: John Wiley, pp. 245-272.

Garey, M.R., and Johnson, D.S., 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman.

Gips, J., and Stiny, G., 1980. “Production Systems and Grammars: A Uniform Characterization,” *Environment and Planning B: Planning and Design* 7, pp. 399-408.

Heisserman, J., 1991. *Generative Geometrical Design and Boundary Solid Grammars*. Ph.D Dissertation, Department of Architecture, Carnegie Mellon University, Pittsburgh, PA.

Krishnamurti, R., 1981. “The Construction of Shapes,” *Environment and Planning B: Planning and Design* 8, pp. 5-40.

Krishnamurti, R., 1982. “SGI: An Interpreter for Shape Grammars,” Research report, Centre for Configurational Studies, The Open University, Milton Keynes, England.

Krishnamurti, R., 1992. “The Maximal Representation of a Shape,” *Environment and Planning B: Planning and Design* 19, pp. 267-288.

Krishnamurti, R., and Earl, C.F., 1992. “Shape Recognition in Three Dimensions,” *Environment and Planning B: Planning and Design* 19, pp. 585-603.

Leonard, H.S., and Goodman, N., 1940. “The Calculus of Individuals and its Uses,” *The Journal of Symbolic Logic* 5, pp. 45-55.

Mäntylä, M., 1988. *An Introduction to Solid Modeling*. Rockville, MD: Computer Science Press.

Mitchell, W.M., 1990. *The Logic of Architecture*. Cambridge, MA: MIT Press.

Pavlidis, T., 1972. “Linear and Context-Free Graph Grammars,” *Journal of the Association for Computing Machinery* 19, pp. 11-22.

Post, E., 1943. “Formal Reductions of the General Combinatorial Decision Problems,” *American Journal of Mathematics* 65, pp. 197-268.

Snodgrass, A., and Coyne, R.D., 1990. “Is Designing Hermeneutical?” Working paper, Faculty of Architecture, University of Sydney, Australia.

Stiny, G., 1980a. “Introduction to Shape and Shape Grammars,” *Environment and Planning B: Planning and Design* 7, pp. 343-351.

Stiny, G., 1980b. “Kindergarten Grammars: Designing with Froebel’s Building Gifts,” *Environment and Planning B: Planning and Design* 7, pp. 343-351.

Stiny, G., 1982. “Shapes are Individuals,” *Environment and Planning B: Planning and Design* 9, pp. 359-367.

Stiny, G., 1991. “The Algebras of Design,” *Research in Engineering Design* 2, pp. 171-181.

Sudkamp, T.A., 1988. *Languages and Machines: An Introduction to the Theory of Computer Science*. Reading, MA: Addison-Wesley.

Sutherland, I.E., 1963. *SKETCHPAD: A Man-Machine Graphical Communication System*. Baltimore, Md.: Spartan Books.

Woodbury, R.F., and Griffiths, E., 1993. “Layouts, Solids, Grammar Interpreters and Firestations: Some Experience from the Trenches,” in U. Flemming and S. Van Wyk (eds.) *Proceedings of CAAD Futures ’93.*

Woodbury, R.F., Radford, A.D., Taplin, P.N., and Coppins, S.A, 1992. “Tartan Worlds: A Generative Symbol Grammar System,” in D. Noble and K. Kensek (eds.) *ACADIA ’92.*