1989

Geometric design spaces

Robert F. Woodbury
Carnegie Mellon University

Christopher Carlson

Jeff A. Heisserman

Carnegie Mellon University.Engineering Design Research Center.

Geometric Design Spaces

by

Robert F. Woodbury,
Chris N. Carlson, Jeff A. Heisserman

EDRC 48-13-89

Geometric Design Spaces

Robert F. Woodbury,
Chris N. Carlson, Jeff A. Heisserman

Abstract:

One of the goals of design research is the creation of systems that can be active participants in the design process, in contrast to many current systems that are primarily passive repositories of information. A number of techniques have been employed for these purposes, including constraint satisfaction, synthesis by analogy, prototype refinement and search in a space of designs; the last of these is the subject of this paper. Two formalisms for search in a geometric design space, structure grammars and solid grammars, are introduced.

This work has been supported by the Engineering Design Research Center, an NSF Engineering Research Center,

Geometric Design Spaces

Robert F. Woodbury/ Chris N. Carlson, Jeff A, Heisserman
Department of Architecture
Carnegie-Mellon University
Pittsburgh, Pa. USA, 15213

One of the goals of CAD research is the creation of systems that can be active participants in the design process, in contrast to many current systems which are primarily passive repositories of information. This active role can occur in a variety of ways [Galle81, Woodbury87], from the automated evaluation of designs to complete design automation, with more elaborate tlieoretical foundations required as complete automation is approached. A significant boundary is crossed in this spectrum when a machine is used to suggest alternatives; to generate configurations that may satisfy a design problem. A number of techniques have been employed for these purposes, including constraint satisfaction, sysnthesis by analogy, prototype refinement and search in a space of designs. The last of these, search in a space of designs, is the subject of this paper.

This paper has three major parts in which are discussed: 1) search in a design space, 2) structure grammars, and 3) solid grammars.

1 Search in a design space

Design can be conceived of as search in a space of possible designs. In this paradigm an agent (human or machine) performs design by applying operators to representational states to produce new representational states, until some criterion for completion has been satisfied. The search is incremental, in the sense that the design agent makes only small changes to the representation content on any particular state transition. A *design space* consists of (in analogy to thejproblem space of Newell and Simon [Newell72]):

> A representation scheme for designed artifacts. This representation scheme ideally is syntactically well-formed (or valid), that is, every representation within the scheme denotes a realistic artifact and every realistic artifact may be denoted by a representation within the scheme. A representation scheme may also be based upon classes rather than instances. In this case, a representation within the scheme denotes not a single realistic artifact, but an entire family of *similar* artifacts.
>
> A set of constructive operators on the representation scheme. These operators should be both closed and complete, that is, every application of an operator produces a well-formed output (consistent with the representation scheme) if given

*** First author's telephone number (412) 268-8853, and email address rw@cad.cs.cmu.edu.**

a well-formed input and every well-formed representation may be produced by a series of operators. A less formal, but crucial, condition on the operators is that they have semantic relevance to the design process ar hand; that each operator corresponds to a meaningful state transition in the space of representations. Several other properties of operators are useful depending upon the type of search used. If the operators are *monotonic,* that is the application of an operator does not change any previously established facts in a representation, then a pruning strategy based on representational state can be used. If die operators are *non-redundant*$_y$ that is that any particular point in the design space can be reached by only one sequence of operators, then the efficiency of an exhaustive search strategy can be improved.

An initial state of a design problem. This is an instance (or set of instances) of a well-formed representation and is taken as a starting point for the search.

A design problem, which is specified as a statement of goals that a solution must satisfy if it is to be considered successful. In most real design problems these goals may be quite vague. Since the goals describe conditions that must be met in a design state, evaluating the goals against a state implies an ability to ask questions of a design representation. These take the form of a set of queries on the design representation. These queries, though limited by the information carried in the design representation, should be able to supply all information relevant to goal evaluation.

A set of knowledge which may be applied to a given design problem. This knowledge is used (and maintained) by the design agent in determining control of the search. Its use depends upon the same set of queries that is used to compare a design state against a set of goals.

Design spaces that display these properties are usually not simple to find, although a well-established paradgim for them has been developed in the domain of architectural floor layout (Flemming85). Those parts of the representation and operators that pertain to geometry usually present the largest difficulty in development of a design space. In this paper we present two approaches to the development of geometrically based design spaces: one based on entities we call *spatial structures,* and the other based upon the well-known paradigm of *solids modeling.* Spatial structures describe spatial relationships between point sets while solids models provide a means of representation for point sets in two and three dimensional space.

2 Spatial Structure Grammars

2.1 Prologue

The spatial structure of an object is embodied in the spatial relationships between its parts. Spatial structure grammars provide a formalism for representing structure as abstracted from the constitutions of component parts, and a production system mechanism for generating families of spatial structures.

A *spatial structure* is an unordered set of ordered pairs. The first member of each pair is a symbol that represents a part, and the second a transformation that specifies the orientation of the part with respect to a common reference. This representation, sufficient to represent an arbitrary N-dimensional spatial configuration of discrete parts, is closed with respect to the standard set operations such as union and set difference, and with respect to the transformation functions that relate substructures. Transformations may be broadly defined to include shape-distorting and non-Euclidean functions.

A *language* of structures is defined procedurally by the action of a set of *productions,* formulated as *rewrite rules,* on an initial structure. When the left-hand side of a production matches a substructure in a structure, the substructure is replaced by the production's right-hand structure . Any structure that can be derived by the application of some sequence of productions in the grammar to the initial structure is **a** member of the language, provided it contains only "terminal" parts. The structure grammar formalism provides a basis for characterizing classes of structures that is analogous to Chomsky's phrase structure grammars for symbol strings.

Much of the experience gained with implementations of production systems in other domains may be directly applicable to implementations of structure grammar interpreters. It is hoped that fast algorithms and symbolic processing hardware will eventually make interactive graphic tools for structure space explorations a practicaility.

2.2 Definitions and notation

Spatial Structures

A spatial structure a is a finite set of *structure elements:*

$$ex = \{(aj,/i), (a_2,/2) \ldots .(a_n,/n)J.$$

Each element in the structure is an ordered pair consisting of a symbol aj taken from an alphabet S, and a transformation/i which together express the identity and *orientation* of a part of the structure. A *realization* of a structure is obtained by mapping its symbols into point sets in some N-dimensional space, R^N. By taking **the** union of these point sets under their corresponding orientation transformations, a new point set is obtained that may be

interpreted as ink on a page, or the material makeup of a three-dimensional object, according to the dimensionality of the space.

Realization of structures is made precise with the following definitions. Let^ be a mapping of the symbols of S into their corresponding point sets. Then the realization of a structure a is the point set

$$\rho(\alpha) = \bigcup_{a\ 5\ (a/)} f(\hat{\rho}(a))$$

Since the symbol in a structure element may represent an arbitrary subset of R^N, the function that specifies the element's orientation must be defined for every point in that space. We further require that the function have a unique inverse, and that it map R^N onto itself. These conditions are satisfied by functions known as *transformations.* Any transformation may be employed as the orientation of a structure element, including the shape-preserving isometries and similarities, and also the shape-distorting affine and non-euclidean transformations.

The *composition* of transformations/and g, written/g, is defined by the equality /g(x) = /(gOO). If/and *g* are transformations, then the *spatial relationship off to* g, *r(f, g)* is/$g^{"1}$. Informally, if/and *g* are the orientations of two structure elements, then *r(f, g)* is the transformation that can be applied to the second element to give it the same orientation as the first element. The orientation of an element is its spatial relationship to the identity transformation, /. From this definition it follows that $r(f_y\ h)r(h_f\ g) = r(f, g)\backslash\ r(f_9 f) = /$; *r(f,* /)=/; and r(/,/)=/-!.

The orientations of a structure's elements are drawn from a group of transformations, F. By definition, F is closed with respect to composition of its elements; F contains the inverses of its elements; and F contains a unique identity transformation /. A transformation *g* in F may be applied to a structure element to produce a transformed element g((a,/)) = (a, *gf).* The application of a transformation to a structure a is the structure that results when the transformation is applied to each element of a individually.

A structure whose symbols are drawn from an alphabet S and whose functions are drawn from a group F is a member of P(S x F), the power set of S x F. Conversely, P(S x F) contains every valid representation of spatial structures with symbols in S and functions in F. It is easy to prove that this universe is closed with respect to transformation of structures by members of F, and with respect to set intersection, union, and complement of its members. The same holds if S is replaced by S, the set of all symbols, and F by F, the set of all transformations. P(S x F) defines the universe of valid spatial structure

representations. It follows that the representation is closed with respect to transformation, intersection, union, and complement.

Spatial structure grammars

A *production* is an ordered pair $p = (\alpha, \beta)$ which associates a left-hand structure α with a right-hand structure β. Productions may also be written $p: \alpha \rightarrow \beta$. A production $p: \alpha \rightarrow \beta$ is said to *apply* to a structure γ *with transformation* f iff $\gamma \supseteq f(\alpha)$. The production is applied to γ to produce a new structure γ' by the *production function*, Ψ:

$$\gamma' = \Psi(\gamma, p, f) = [\gamma - f(\alpha)] \cup f(\beta),$$

where $-$ and $\cup$ denote the conventional set difference and union operations.

A *structure grammar* $G = \{S, T, F, P, \gamma_0\}$ is an ordered quintuple consisting of S, an alphabet of symbols; T, a subset of S designated "terminal" symbols; F, a (possibly infinite) group of transformations; P, a set of productions; and an initial structure, γ_0. Terminal symbols are distinguished from non-terminal symbols in order that "finished" structures can be distinguished from "unfinished" structures. A structure that contains non-terminal symbols must undergo further refinement (by the application of productions to it) before it is complete.

If a production in a grammar G can be applied to a structure γ to obtain a structure δ, then γ *directly derives* δ in G, or $\gamma \overset{}{\Rightarrow}_G \delta$. This will also be written $\gamma \overset{}{\Rightarrow}_p \delta$ when it is necessary to specify the production involved. If there exists some sequence of direct derivations in a grammar G such that

$$\alpha_0 \overset{}{\Rightarrow}_G \alpha_1 \overset{}{\Rightarrow}_G \alpha_2 \overset{}{\Rightarrow}_G \cdots \overset{}{\Rightarrow}_G \alpha_{n-1} \overset{}{\Rightarrow}_G \alpha_n$$

then α_0 *derives* α_n in G, or $\alpha_0 \overset{*}{\Rightarrow}_G \alpha_n$.

The *language* L(G) generated by a grammar $G = \{S, T, F, P, \gamma_0\}$, is the set of all structures that can be derived from the initial structure and that contain only terminal symbols:

$$L(G) = \left\{ \gamma \mid \gamma_0 \overset{*}{\Rightarrow}_G \gamma \text{ and } \gamma \text{ is in } P(T \times F) \right\}$$

2.3 Examples

Binary trees

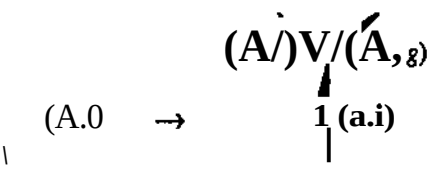
Let $G = \left\{ \{A, a\}, \{a\}, \langle f, g \rangle, P, \gamma_0 \right\}$ and let f and g each be a composite rotation, scale, and translation. $\langle f, g \rangle$ denotes the group generated by compositions of f and g. When P

consists of the two productions described below, L(G) is an infinite set of binary tree structures.

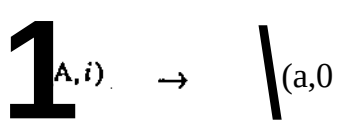
The symbols "A^M and "a" represent fertile and non-fertile branches. One production in P causes a fertile branch to fork into two fertile branches. The otlier production turns fertile branches into non-fertile branches. Realizations of these productions are shown below in which structure elements are represented by oriented arrows. It should be remembered, however, that the realizations are arbitrary, and not a part of the grammar.

P:

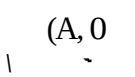
1: {(A,/)) -» {(a,i),(AA(A.«))



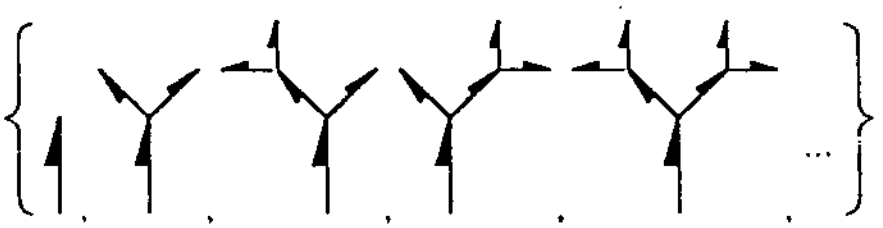
2: **{(A,/)) -» ((a,0)**



70: {(A.0)



UG):



Realizations:

Solving a maze

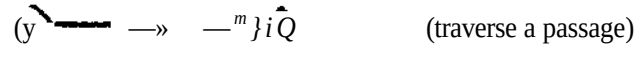
In addition to serving as compact representations of sets of spatial structures, spatial structure grammars may be construed as procedures for solving spatial problems. Navigation through a maze is an example. The initial structure, a maze, corresponds to the problem statement and the language generated by the grammar corresponds to the set of solutions to the problem.

The following grammar runs a mouse tlirough an acyclic maze, leaving a trail of footprints that mark a path from the starting point to die goal. Passages through the maze are represented by line segments, which are actually two overlapping, antiparallel, oriented segments that indicate that a passage can be traversed in both directions. Light gray line segments represent the absence of passages, a device that is necessary because productions can only match what is explicitly represented. It is not otherwise possible to write a production that tests for the absence of a passage.

The goal passage is represented by a thick black segment. When the mouse traverses the goal, it changes from a non-terminal to terminal symbol. The resulting structure contains only terminal symbols, and is therefore a member of L(G).

G: 

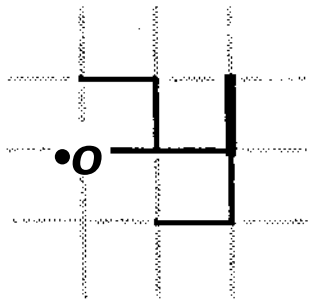
P:

1:  (traverse a passage)

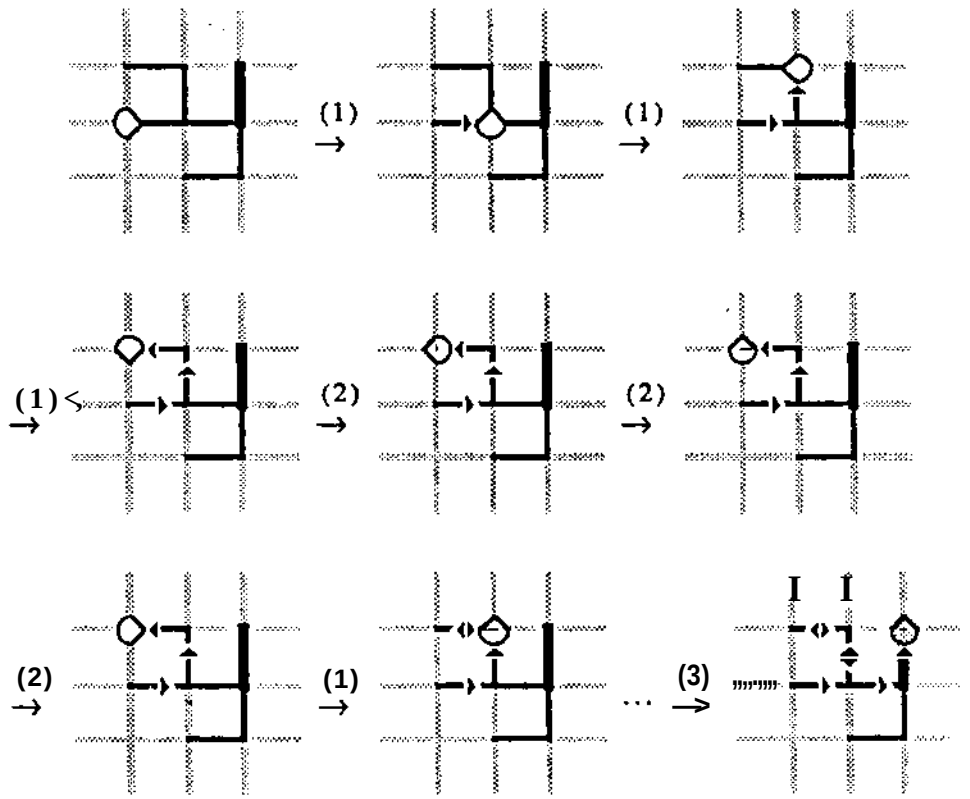
2:  **(look for a** passage **to** the right)

3:  **(traverse the goal** passage)

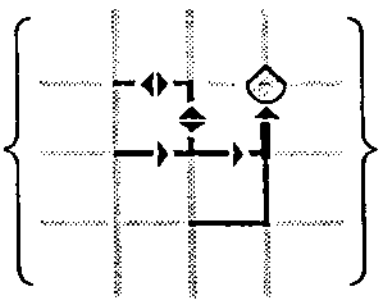
γ_0:



Derivation:



L(G):



2.4 Parameterized grammars

The generality of spatial structure grammars can be extended by using variables in die left-hand sides of productions. This permits the formulation of general rules that match any number of different substructures in a stiucture, depending on the values substituted for the variables. Variables may stand for symbols or parameters of transformations. When a set of values is found that makes the left-hand side of a production match a substructure, the same values are substituted for instances of the variables in the right-hand side, in effect yielding a new production which is then applied to the structure in the conventional way.

Most problems of practical interest require parameterized grammars. Prototype parameterized spatial structure grammars have been constructed to generate human figures in the style of Ottl Aicher's 1972 Olympic sports figures, and to enumerate rectangular partitionings of a rectangular area (an adaptation of a shape grammar by Ulrich Flemming).

2.5 Spatial structure grammars as search spaces

An "extended" language may be defined by relaxing the condition that members of a grammar's language may contain only terminal symbols:

$$L^{+}(G) = \left\{ \gamma \mid \gamma_0 \overset{*}{\Rightarrow}_G \gamma \right\}.$$

This set together with the productions of G constitute a search space on spatial structures. The elements of $L^{+}(G)$ are the "terrain" of the space, and the productions the means of moving through it. L+(G) defines the representation scheme of the search space, which is to say that "valid representation" and "member of $L^{+}(G)$" are equivalent statements. In the following discussion, "a representation" will be used to mean "member of $L^{+}(G)$" and "the representation" to mean "L+(G)'\

Given a realization p defined on the alphabet of G, every representation a in $L^{+}(G)$ corresponds to the physical object p(oc). Thus half of the condition for syntactic well-formedness is met for every grammar and realization. The other requirement, that every physical object have a representation, depends on an exact notion of which physical objects we intend to represent, and will vary according to the application. A general characterization of the families of point sets that have representations as spatial structure grammar languages is an open problem.

Productions in G are the operators for moving about in the search space. By definition, the representation is closed with respect to the application of productions. The set of productions is also complete, since every representation can be derived from the initial structure by some sequence of productions. The use of productions as operators provides considerable flexibility in matching search space operators to the semantics of a particular design problem. This is evident in the maze grammar presented above, whose productions

correspond in a straightforward manner to moving one step through the maze, turning to look for a passage, and recognizing the goal.

Goal-directed search in a space requires the ability to ask questions of a representation. Answers to the questions guide the search process. Spatial structures contain complete information about the spatial relationships among their elements, since the relationship between any two elements can be derived from their orientations. This is a convenient consequence of the transitivity and invertability of spatial relationships. We may therefore ask questions of spatial structures such as, Is a structure symmetric, or nearly so? Does it contain any local symmetries? Does it have realizations that avoid overlap of the component parts?

Questions about topological properties, on the other hand, cannot be answered without knowledge of, or assumptions about, a structure's realization. It cannot in general be said whether the parts represented by structure elements are adjacent, between, connected, intersect, or are even simply near one another. Some of these questions can be answered, however, when F is appropriately restricted. For example, when F is $<t_x, t_y>$ where t_x and *ty* are orthogonal translations, it is possible to answer questions of adjacency, nearness, and betweenness. Similarly, adjacency and betweenness can be determined when F is $<r>$ and *r* is a rotation.

The independence of structure and realization in spatial structure grammars can be an advantage when exploring design spaces. A design object may be transformed by varying relationships between parts, leaving the parts unchanged; or by varying the shapes of parts, leaving the relationships between parts unchanged. Spatial structure grammars provide formal basis for implementations that offer this modularity to the user.

3 Solid Grammars

3.1 Prologue

Solid grammars provide a formalism for generating complex models of rigid solid objects. They are a manifestation of die concepts of rule based systems into the realm of solid modeling, as well as an extension of shape grammars from line drawing to solid representation.

The representation of a solid used in solid grammars consists of three parts:

1. a topology model defined as a graph of faces, edges and vertices;
2. coordinate geometry associated with each vertex;
3. identification labels associated with the faces, edges and vertices of the solid, where desired.

An *initial solid* is defined in the above representation. Modifications of the initial solid are accomplished by the application of a set of *solid rules.* These solid rules are productions defined as rewrite rules. The left hand side of a production matches on a (whole or partial) solid representation, and the (whole or partial) solid specified in the right hand side of the production is substituted.

A solid grammar describes a language of solids. A language consisting of all possible topologically valid *genus zero* solids (solids that have no holes or handles) is produced with a set of unrestricted solid rules. These rules would allow modifications of topology corresponding to the Euler operators, would not restrict assignment of coordinate geometry of vertices, and would not (necessarily) assign any identification labeling except for termination purposes. Since the underlying representation is based on boundary models, the grammar would not guarantee geometric validity, nor would the arrangements be unique.

The solid grammar formalism is intended to provide a mechanism for generating subsets of this language that satisfy some design criteria, those criteria being described by the solid rules.

3.2 Research Precedents

Solid grammars have developed at the convergence of several distinct lines of research. Those contributions are outlined below.

3.2.1 Solid Modeling

Solid modeling emphasizes the general applicability of models and insists on creating only complete representations for physical solid objects [Man88]. Because of these qualities, they are proving their importance and usefulness in a broad range of domains, including engineering and architectural design, manufacturing, robotics, computer vision, and 3-D computer animation and graphics.

3.2.2 Shape Grammars

Shape grammars provide a representation for shapes and a method for their generation using shape specific rules [SG72]. This representation for shapes is defined as a limited arrangement of straight lines defined in a cartesian coordinate system with real axes and an associated euclidean metric [Sti8O].

Shape grammars have provided a powerful formalism within which to think about design, and have resulted in a significant body of design rules for a variety of applications, primarily in the architectural domain [DF81,Fle87,KE81,SM78,SM80].

3.z3 Graph Grammars

The correspondence of the topology of boundary representation solid models and a class of graph grammars has been shown for solids of genus zero [Fit87]. These graph grammars use rules analogous to Euler operators and maintains topological validity of the representation scheme, i.e. that it is syntactically complete and closed for all genus zero solids.

3.2.4 Rule-Based Expert Systems

A substantial amount of research has been devoted to the study of knowledge representation, and one the most widely used models of knowledge representation and application is the *production-system model* [B*85]. This model has proved successful in a variety of applications in the area of design and design synthesis [McD80,MF85].

3.3 Definitions & Notation

A *solid grammar* is defined to be a 4-tuple G = (XV, *EN,I,P),* where:

1. $\Sigma_T = T \cup G \cup L_T$ are the terminals of the grammar, T is a finite set of topology symbols, G is a finite set of symbols representing vertex coordinate geometry, and L_T is a finite set of terminal identifying labels
2. $\Sigma_N = L_N$ are the non-terminals of the grammar, L_N is a finite set of non-terminal identifying labels, and L is the set of all identifying labels such that $L_N = L - L_T$
3. I is a labeled solid in (T_i^*, G_i^*, L_i^*) called the *initial solid*
4. P is a finite set of *solid rules* of the form $\alpha \rightarrow \beta$, where $\alpha = (T_l^*, G_l^*, L_l^*)$, and $\beta = (T_r^*, G_r^*, L_r^*)$

3.3.1 Solids

The primary objects in this representation are the boundary representation solid models as defined in the solid modeling literature [Req80,Man88]. In this representation, a solid is described by its bounding surface. The surface is divided into faces, that are joined at edges, which are in turn joined at vertices. As in the solid modeling literature, we make a distinction between the topology and the geometry of a solid model. The topology of a solid is represented as a graph composed of faces, edges and vertices, and their adjacency relationships. The geometry of a solid is represented by assigning cartesian coordinates (in cartesian three space) to each vertex of the solid.

The solids of solid grammars have one additional part, identification labels, that will be discussed next.

3.3.2 Labels

The components of a solid can be distinguished by labeling them. Formally, A *labeled component* $c : A$ is a component c with a symbol A associated with it. Thus we can have *labeled vertices*, *labeled edges* and *labeled faces*. Two labeled components $c_1 : A_1$ and $c_2 : A_2$ are the same iff both the components c_1 and c_2, and the symbols A_1 and A_2 are the same.

Labels have several uses in solid grammars. They may be used to reduce search, by labeling components that will need to be located for later operations. Application of solid rules may be restricted by requiring labels as part of rules' left hand sides. A label may also carry information for users or for future processes. For example, a label may indicate the top face of an engine block, or the south face(s) of a building.

3.3.3 Rules

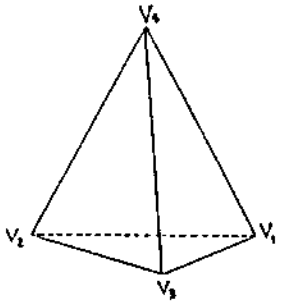
A solid rule is a rewrite rule that may add, modify or delete solids and parts of solids (complying with Eulers law), coordinate geometry associated with vertices, and identification labels.

The rules represent modifications on a solid, thus the modifications of solids must yield valid solids. These rules would allow matches on the faces, edges, vertices, the angles between edges and/or faces, lengths of and distance between components, and labels on solids and components in the left hand side.

3.4 Examples

A sample solid grammar is presented for the production of tetrahedïonal crystal forms. We begin with an initial shape which is a valid solid representation of the simplest crystal. It is represented as a solid tetrahedron, composed of four vertices connected by six edges, defining four faces. The four faces are labeled with non-terminal symbols *F:A.*

Intial Solid:



V_x = [2,0, Of 77,23:/I
V_2 = [-1,%/W $F_{X74}:A$
V_3 = [_i,_v%o]^r $F_{l3A}:A$
V_A = 10,0,2>/2]^r ^234 :A

A single generation rule in needed. It first matches on a labeled face $F : A$ and its partial topology graph of that face bordered by three edges and three vertices. It then modifies a face of the crystal by dividing it into six faces and adding four more vertices. The solid rule assigns of coordinate geometry for the new vertices, as well as attaches new labels to the faces.

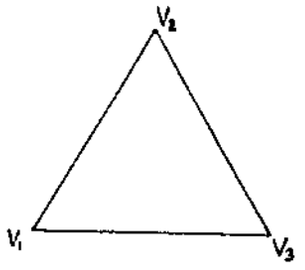
The topological modifications can be represented by the application of Euler operators, as follows:

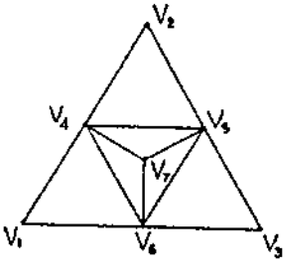
- apply MEV (three times) to each of the original edges, which divides each edge into two edges and creates a new vertex
- apply MEF (three times) to each pair of newly created vertices - this divides the original face into four faces

- apply MEV (once) to one of the new vertices to create the final (center) vertex
- apply MEF (twice) to the center vertex and each of the two new vertices not used in the last operation - this divides what was the center face (of four faces) into three faces

Thus the topological validity of our solid is guaranteed.

Rule 1 - Generation Rule:





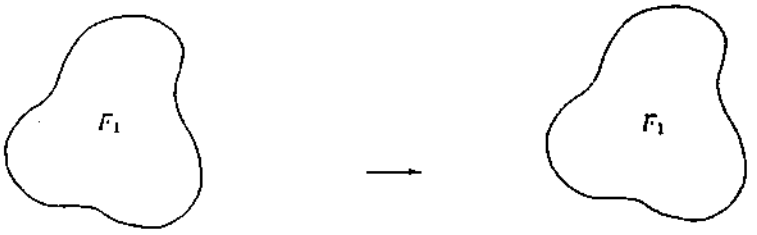
$$
\begin{aligned}
V_4 &= \tfrac{1}{2}(V_1 + V_2) \\
V_5 &= \tfrac{1}{2}(V_1 + V_3) \\
V_6 &= \tfrac{1}{2}(V_2 + V_3) \\
V_i &= \tfrac{1}{3}(V_1 + V_2 + V_3) \\
&\quad + j_S \lvert V_2 - V_1 \rvert \mathit{normal}(F_{123})
\end{aligned}
$$

$F_{123} : A$

$F_{123} :$
$F146 : A$
$F245 : A$
$F3S6 : A$
$F_{457} : A$
$F_{567} : A$
$F_{467} : A$

A clean-up rule is needed to remove the unwanted non-terminal symbols from the faces. When a solid rule removes the last non-terminal symbol, the grammar terminates and a valid crystal curve has been produced.

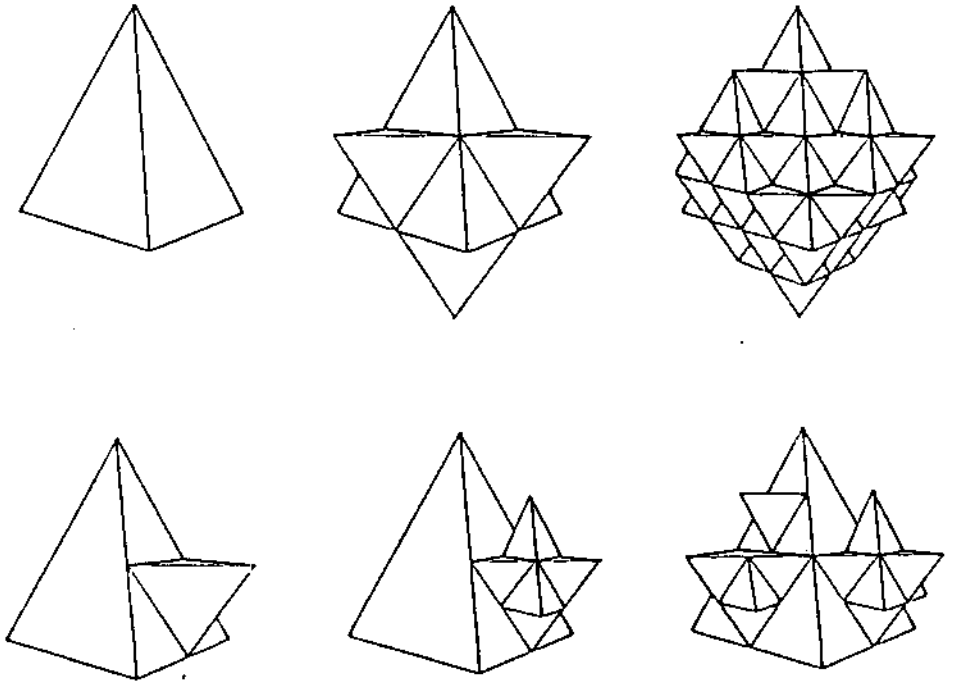
Rule 2 - Cleanup Rule:



F_x :A F_1 :

A language of crystals is produced by this solid grammar, including both regular and non-regular examples. A portion of this language is presented below.

Language of Crystals



3.5 Extensions

The representation, as presented, is limited to genus zero solids. This limitation can be removed in order to include solids of higher genus by adding *shell* and *loop* components to the topology graphs, as well as by expanding the Euler rules to operate on them.

The ability to represent multiple solids is needed for many applications. For example, the shape rules of Flemming's Queen Anne houses [Fle87J could be implemented in solid rules with this extension. A *solid* element would need to be added to our representation, with modifications to the Euler rules to allow joining two solids and spliling one into two.

These expansions of representational capabilities suggest additions to the identification labeling. Labels for *labeled solid, labeled shells* and *labeled loops* would maJce our grammars complete.

References

[AN72] H.A. Simon A. Newell. *Human Problem Solving.* Prentice-Hall Inc., Englewood Cliffs, N.J., 1972.

[B*85J L. Brownston et al. *Programming Expert Systems in OPS5.* Addision-Wesley, Reading, Massachusetts, 1985.

[DF81] F. Downing and U. Flemming. The bungalows of Buffalo. *Environment and Planning B,* 8:269-293, 1981.

[Fit87] Patrick Fitzhorn. A linguistic formalism for engineering solid modeling. In *Graph-Grammars and Their Application to Computer Science,* pages 202-215, Springer-Verlag, Berlin, 1987.

LFle85] U. Flemming. On the representation and generation of loosely-packed arrangements of rectangles. *Planning and Design,* December 1985.

[Fle87J U. Flemming. More that the sum of parts: the grammar of Queen Anne houses. *Environment and Planning B,* 14:323-350, 1987.

[Gal81] Per Galle. An algorithm for exhaustive generation of building floor plans. *Communications of the ACM,* 24:813-825, 1981.

[K£81] H. Koning and J. Eizenberg. The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and Planning B,* 8:295-323, 1981.

[Man88] Martii Mantyla. *An Introduction To Solid Modeling.* Computer Science Press, Rockville, Maryland, 1988.

[McD80J J. McDennott. *Rl: A Rule-Based Configurer Of Computer Systems.* Technical Report CMU-US-80-119, Department of Computer Science, Carnegie Mellon University, 1980.

[MF85] Mary Lou Maher and Steven J. Fenves. *HI-RISE: A Knowledge-Based Expert System for the Preliminary Design of High Rise Building.* Technical Report R-85-146, Department of Civil Engineering, Carnegie Institute of Technology, Carnegie Mellon University, January 1985.

[Req80J A. A. G. Requicha. Representation of rigid solids: theory, methods, and systems. *ACM Computing Surveys,* 12(4):437-464, 1980.

[SG72] George Stiny and James Gips. Shape grammars and the generative specification of painting and sculpture. In *IFIP-Congress 1971,* pages 1460-1465, North Holland, Amsterdam, 1972.

[SM78J George Stiny and William J. Mitchell. The Palladian grammar. *Environment and Planning B,* 5:5-18, 1978.

[SM80J George Stiny and William J. Milchell. The grammar of paradise: on the generation of Mughul gardens. *Environment and Planning B,* 7:209-226, 1980.

[Sti80] George Stiny. Introduction to shape and shape grammars. *Environment and Planning B,* 7:343-351, 1980.

[Woo87] Robert F. Woodbury. ***Strategics for Interactive Design Systems,*** **chapter 1, pages 11-36. Volume 2** ***of Principles of Computer-Aided Design,*** **John Wiley** and Sons, 1987.