

## CHAPTER ONE

# Vitruvius Redux

## Formalized Design Synthesis in Architecture

William J. Mitchell

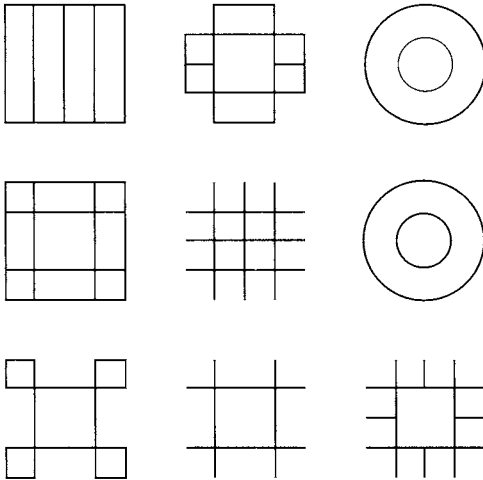
### PARTICULATE COMPOSITION

A quarter of a century ago I published a paper titled “Vitruvius Computatus” (Mitchell, 1975). In it I argued that, if computer systems achieved the capacity to execute nontrivial architectural design tasks, we could “expect the designs which emerge from them to be characterized by particular stylistic traits.” I went on to suggest, more specifically, “If computer systems become architects they may be academic classicists, heirs to Durand and Guadet.” Sadly, I turned out to be right.

Jean N. L. Durand and Julien Guadet were great French architectural pedagogues, and authors of highly influential texts that were used by generations of students.<sup>1</sup> They provided models to copy, and they promulgated a view of architectural composition that was essentially particulate and combinatorial. Within the tradition that Durand and Guadet established and formalized, designers relied heavily upon abstract ordering devices such as grids and axes, and they would typically begin a project by exploring various combinations of these devices to produce an organizing skeleton of construction lines (Figure 1.1). Guided by this skeleton, they would then consider alternative ways to arrange the major rooms and circulation spaces. Finally, they would develop the design by deploying elements from an established vocabulary of construction elements – columns, entablatures, doors, windows, and so on. As Durand’s famous plates illustrated (Figure 1.2), it was a recursive process of top-down substitution.

Thus, learning to design was much like learning Latin prose composition. There was a well-defined vocabulary of discrete construction elements, there were rules of syntax expressed by allowable combinations of grids and axes (the graphic equivalent of sentence schemata found in grammar textbooks), and you could produce acceptable designs by arranging the construction elements according to these rules. Many of Durand’s plates were devoted to illustrating alternative “horizontal combinations” (plans) and “vertical combinations” (elevations) that could be produced in

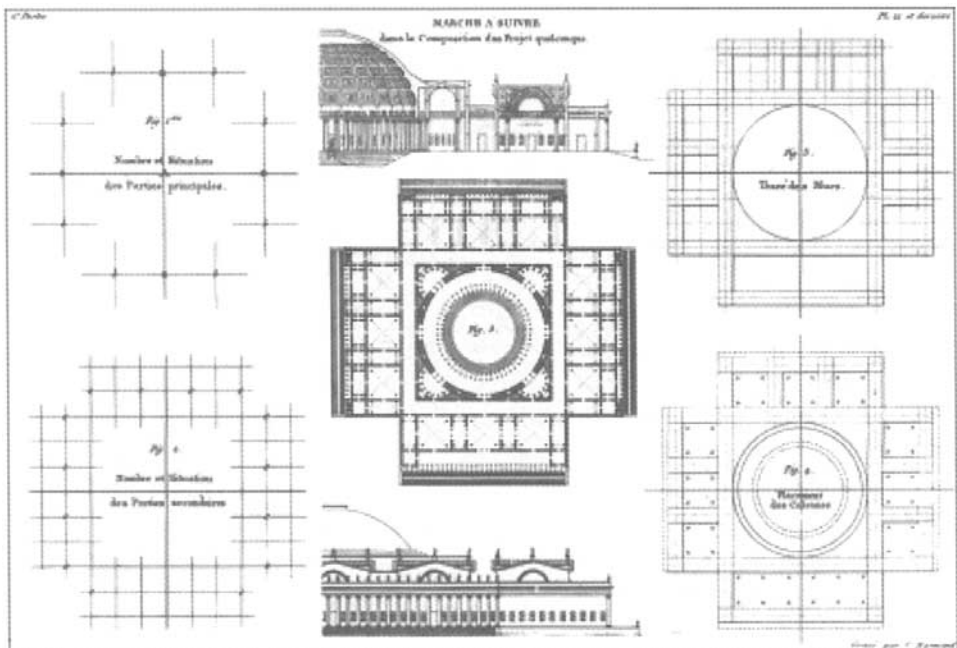
<sup>1</sup> See, in particular, Durand’s *Précis des Leçons d’Architecture* (Durand, 1819), his *Partie Graphique des Cours d’Architecture* (Durand, 1821), and Guadet’s *Éléments et Théories de l’Architecture* (Guadet, 1902).



**Figure 1.1.** Skeletons of construction lines as starting points for plan compositions, from Durand's *Précis* (1819).

this way. In addition to generating a design according to Durand's rules, you could run the process in reverse to parse a completed design into a hierarchy of definite elements and subsystems, just as you could parse a Latin sentence into phrases and parts of speech.

The great figures of early architectural modernism mostly represented themselves publicly as anticlassicists, but they did not forget the intellectual legacy of Durand and Guadet. As Reyner Banham noted, Guadet's *Éléments et Théories de l'Architecture* (1902) "formed the mental climate in which perhaps half the architects of the twentieth century grew up." The compositional strategy that they followed



**Figure 1.2.** Design as a process of top-down substitution, from Durand's *Précis* (1819).

could be characterized as follows:

The approach is particulate; small structural and functional members (elements of architecture) are assembled to make functional volumes, and these (elements of composition) are assembled to make whole buildings. To do this is to compose in the literal and derivational sense of the word, to put together.

R. Banham, *Theory and Design in the First Machine Age* (1967)

The new conditions of industrial mass production clearly strengthened this idea of composing a building from discrete standard parts. The difference, now, was that the drive to achieve economies of scale (rather than the pedagogical strategy of publishing models to copy) motivated standardization and repetition. The extreme response to these conditions was the creation of industrialized component building systems that provided complete, closed “kits of parts” for assembling entire buildings. At a smaller scale, construction toys such as Froebel and Lego blocks reproduced the idea of particulate composition at a smaller scale. Proponents of these systems frequently illustrated their capabilities by producing exhaustive enumerations of possible combinations of elements.

## THE BEGINNINGS OF ARCHITECTURAL CAD

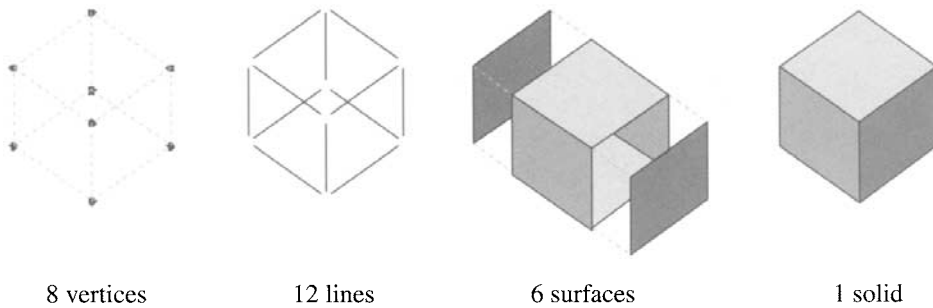
Not surprisingly, when the first computer-aided design (CAD) systems appeared, they reified these well-established design traditions.<sup>2</sup> They allowed drawings to be organized in “layers,” which reflected both the earlier practice of drafting on overlaid sheets of translucent paper and Durand’s notion of successive layers of abstraction. They provided grids, construction lines, and “snap” operations for positioning graphic elements in relation to the established skeleton of a composition. They also relied heavily upon the operations of selecting standard graphic elements, then copying, translating, and rotating them to assemble compositions from repeated instances.

At a deeper level – that of their underlying data structures – these systems also assumed definite, discrete elements. The basic geometric units were points, specified by their Cartesian coordinates. Lines were then described by their bounding points, polygons by their bounding lines, and closed solids by their bounding polygons.<sup>3</sup> More complex graphic constructions were treated as sets of these basic geometric entities, and nongeometric properties (color, mass, material properties, cost information, etc.) might be associated with such groupings. Subshapes were simply subsets, and complete drawings could unambiguously be parsed into hierarchies of subshapes and geometric entities – all the way down to the elementary points. There were many variants on this basic scheme – some of them very clever – but they were all grounded upon the idea of sets of geometric entities encoded as relations on sets of points (Figure 1.3).

Because the usual goal of early CAD systems was efficiency in the production of construction documents, many went a step further and provided higher-level vocabularies of architectural elements such as standard walls, doors, windows,

<sup>2</sup> For details of early architectural CAD systems, see Mitchell (1977).

<sup>3</sup> This strategy is grounded in the pioneering work of Steven Coons and Ivan Sutherland. See Stiny, Chapter 2, for a critical discussion of the theoretical foundations.



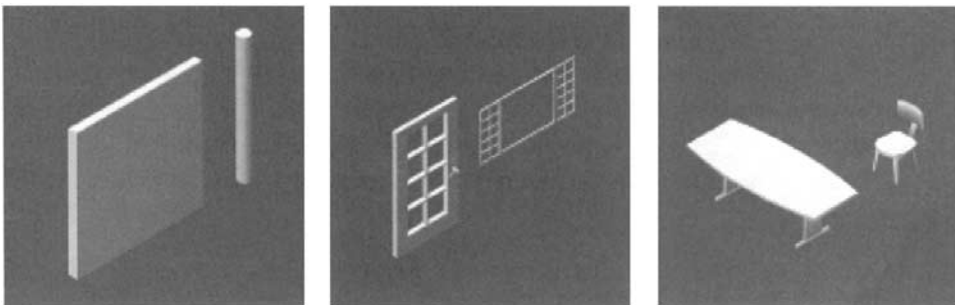
**Figure 1.3.** Selectable geometric entities in a typical CAD system.

columns, desks and chairs, mechanical and electrical equipment items, and so on (Figure 1.4). These became the elements that an architect could quickly select, transform, and assemble in a process that directly (though frequently unknowingly and uncritically) recapitulated Durand. In other words, these systems assumed and enforced a conventionalized architectural ontology. This was reasonable if you were prepared to accept the ontology, but not if you wanted to stretch the boundaries of architectural discourse. You were out of luck if you wanted to explore an architecture of free-form blobs, or of nonrepeating, nonstandard parts, for example.

At an implementation level, however, there were many practical advantages to this approach. It was conceptually simple, and it supported the creation of elegant and efficient graphics pipelines that took advantage of increasing computer resources. It lent itself to some useful elaborations, such as parametric CAD. It also benefited from new approaches to software engineering, such as object-oriented programming.

In contrast, the limitations were obvious enough from the beginning – though, as it turned out, little reflected upon or heeded in what became the headlong post-PC rush to commercialize and market standard computer graphics technology. In “Vitruvius Computatus” I warned:

Is this continuity of the academic classical tradition of elementary composition in a rather unexpected way merely a historical curiosity of little practical consequence? I suggest not. It indicates that computer systems cannot be regarded as formally neutral tools in the design process. On the contrary, their use can be expected to demand acquiescence to particular formal disciplines. This implies several things. Firstly it becomes clear that the design and development of satisfactory computer-aided design



**Figure 1.4.** Typical vocabulary elements of an architectural CAD system.

systems is not simply a technical task to be left to systems analysts, engineers, and programmers. It demands a high level of *architectural* skill and capacity for rigorous analysis of built form. Secondly, as architects begin increasingly to work with such systems they will need to devote as much attention to understanding the vocabulary and syntax of form implied by the data structures of those systems as the classical architects of old devoted to study of the orders.

W. Mitchell (1975)

The consequences of implicitly encoding stylistic presuppositions in software eventually showed up in built work. As one prominent and sophisticated designer recently remarked to me, “If you just look at the work of many young designers, you can immediately see what sort of CAD system they are using.”

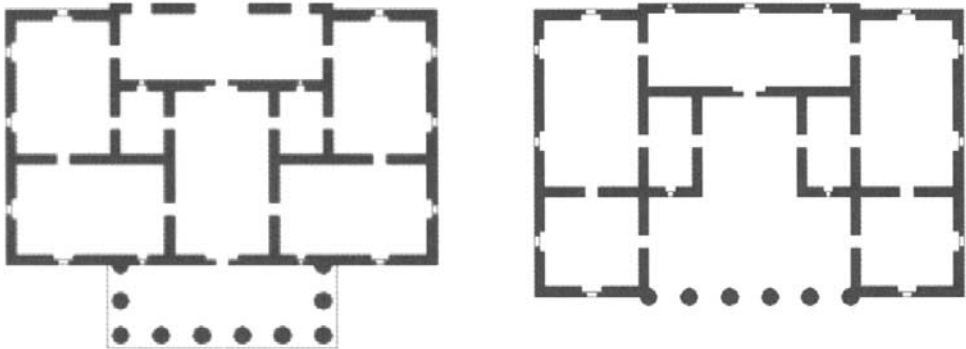
## LANGUAGES OF ARCHITECTURAL FORM

Just as Durand conceived of design as a process of recursive substitution of shapes terminated by the insertion of elements from an established architectural lexicon, formal linguistics taught us to understand sentence construction as the recursive substitution of symbols terminated by the insertion of words from a written language’s vocabulary. The frequently invoked analogy between classical architectural design and Latin prose composition turns out to be more than superficial!

With the development of the shape grammar formalism, Stiny and Gips (1972) made the idea of shape substitution precise, and they cast it in modern computational form. Whereas formal generative grammars (as the concept is normally understood in linguistics and computer science) generate one-dimensional strings of symbols, shape grammars generate two-dimensional or three-dimensional (or, indeed,  $n$ -dimensional) arrangements of shapes. They operate by recursively applying shape rewriting rules to a starting shape. A language of designs thus consists of all the shapes that can be derived in this way.

Stiny and Mitchell (1978a, 1978b) first applied shape grammars to nontrivial architectural design by writing a grammar that produced schematic villa plans in the style of the great Italian renaissance architect Andrea Palladio. In some sense, this grammar captured the essence of Palladio’s style, and it allowed others to design in that style – thus functioning pedagogically much as Palladio’s own *Four Books of Architecture* had been intended to do. Stiny and Mitchell demonstrated the grammar by exhaustively enumerating (at a certain level of abstraction) the plans in the language that it specified. The resulting collection of possibilities included the villa plans that were actually produced by Palladio, together with others that might have been if there had been the time and the opportunity (Figure 1.5).

Subsequently, there have been many other attempts to write grammars that produce particular classes of architectural designs and to enumerate the languages they specify. Numerous of these have been published, over the years, in the journal *Environment and Planning B*. In 1981, for example, Koning and Eizenberg (1981) demonstrated a grammar that produced convincing “prairie houses” in the style of Frank Lloyd Wright (Figure 1.6). A particularly compelling recent example is Duarte’s grammar (Duarte, 2000a, 2000b) of house designs in the style of Alvaro



Villa Hollywood

Villa Vine

**Figure 1.5.** Plans of the Villa Hollywood and the Villa Vine, two of the pseudo-Palladian villas generated by Stiny and Mitchell's grammar.

Siza's Malagueira project (Figure 1.7). Duarte's grammar is presented in the form of a Web site that allows house clients to derive designs that respond to their own particular needs and budgets.

## COMBINATORIAL SEARCH

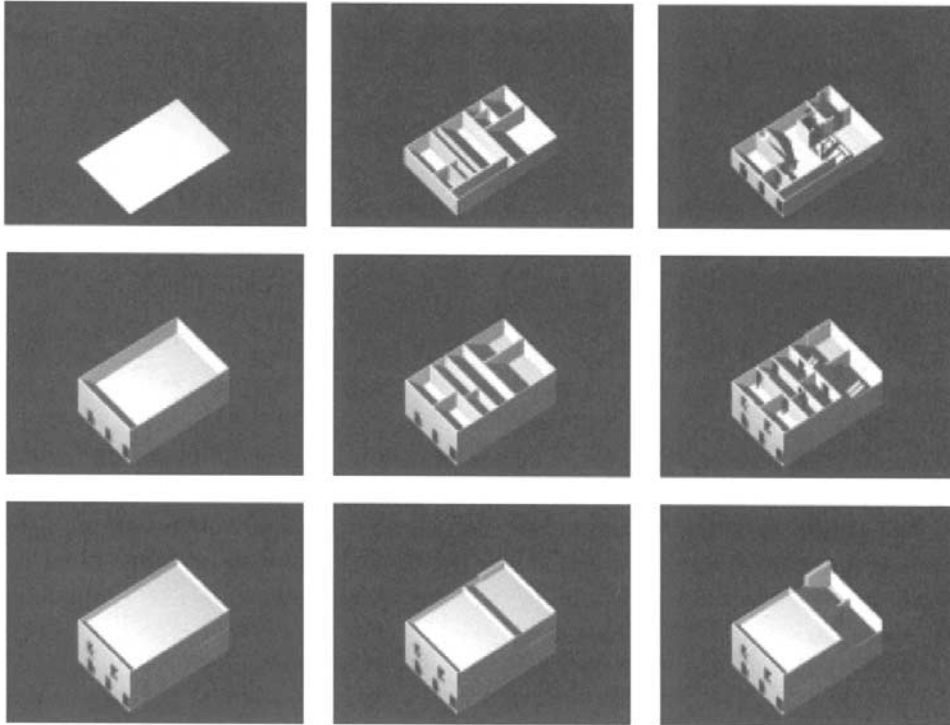
The most obvious way to implement a shape grammar (or some functionally equivalent set of generative rules expressed in a different way) is to begin with the sort of data structure common to CAD systems, as described earlier. Subshapes are thus treated as subsets of geometric elements, and rewriting rules replace one subset with another. This is all a fairly straightforward extension of the computer science tradition dealing with formal languages, automata, production systems, and the like. And it also allows architectural design problems to be assimilated into the intellectual



**Figure 1.6.** A prairie house in the style of Frank Lloyd Wright, generated by Koning and Eizenberg's grammar.

## VITRUVIUS REDUX

7



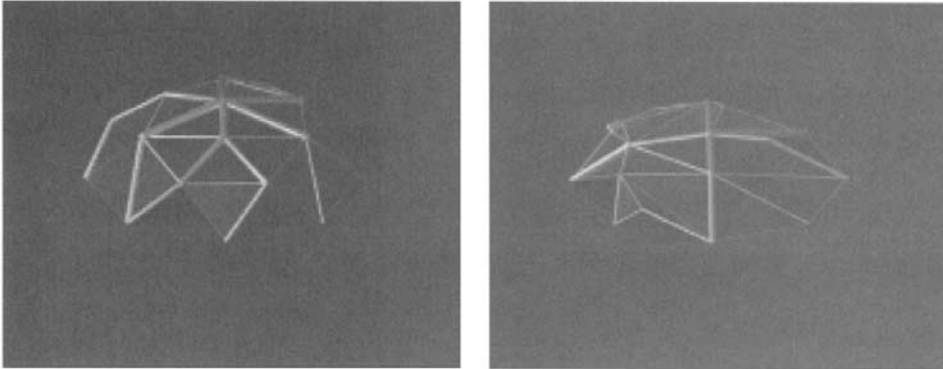
**Figure 1.7.** Step-by-step derivation, according to Duarte's grammar, of a Malagueira house.

traditions of AI problem-solving (as represented, in particular, by the pioneering work of Newell and Simon) and combinatorial optimization.<sup>4</sup>

Within this framework, design problems can be formulated as problems of searching discrete (or partially discrete) state-spaces to find designs in a language that satisfy specified criteria. (Recall Durand's "horizontal combinations" and "vertical combinations" – variants adapted to different site contexts, functional requirements, and budgets.) These criteria can be expressed, in the usual ways, in the form of constraints and objective functions. The search tasks that result are typically large scale and nastily *np* complete. Heuristic search, simulated annealing, evolutionary strategies, and other such techniques are thus employed to control the search process and produce acceptable results with a reasonable expenditure of computational resources.

This combinatorial approach had its early successes. For example, Mitchell, Steadman and Liggett (1976) produced an efficient and reliable system for designing minimal-cost house plans – much like Siza's Malagueira plans – that satisfied specified area and adjacency requirements for the constituent rooms. More recently, Shea and Cagan (1997, 1999) demonstrated the possibility of deriving unusual but practical and efficient structural designs by means of a shape grammar encoding rules of stability in pin-jointed structures, finite-element analysis of alternatives that are generated for consideration, and the use of simulated annealing to control the search

<sup>4</sup> My early architectural CAD text *Computer-Aided Architectural Design* (Mitchell, 1977) explicitly makes this connection and provides many examples.



**Figure 1.8.** Unusual dome designs generated by Shea's shape annealing procedure.

(Figure 1.8). However, there were several difficulties. One was the difficulty of capturing the subtleties and complexities of practical design goals in terms of constraints and objective functions. A second was the typical intractability of the search problems that result from such formulations. The third – and most profound – derived from the fundamental assumption of definite, discrete elements and treatment of subshapes as sets of such elements.

### LIMITATIONS OF COMBINATORIAL SEARCH

To see the nature of this last difficulty, consider the shape shown in Figure 1.9.<sup>5</sup> It is from master mason Mathias Roriczer's famous pamphlet on the design of cathedral pinnacles (Roriczer, 1486).

Roriczer described its construction, in detail, as follows.

Would you draw a plan for a pinnacle after the mason's art, by regular geometry? Then heave to and make a square as it is here designated by the letters A, B, C, D. Draw A to B, and B to D, and D to C, and from C to A, so it may be as in the given figure [Figure 1.10].

Then you make another square: divide A to B in two equal parts, and place E; likewise, divide BD and there make H; and from D to C and there make an F; similarly from C to A, and there make a G. After that draw a line from E to H, and from H to F, F to G, G to E. An example is in the following figure [Figure 1.11].

After that you make like the one made above another square: divide FH into two equal parts and there put a K. Similarly on HF place M; also on FG make L, similarly on GE place I. After that draw a line from K to M, from M to L, from L to I, from I to K, as in the following figure [Figure 1.12].

M. Roriczer, *On the Ordination of Pinnacles* (1486)

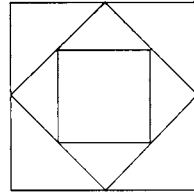
It is easy to execute this construction in a standard CAD system, such as AutoCad; the result is a shape represented as 12 lines. Any one of these lines may then be selected and transformed or deleted, to produce design variations of the kind shown in Figure 1.13. (To make this exercise interesting, of course, you have to restrict

<sup>5</sup> See also Stiny, Chapter 2.

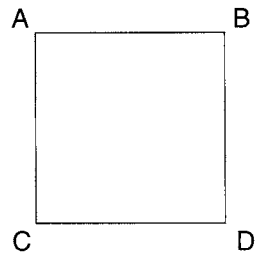


VITRUVIUS REDUX

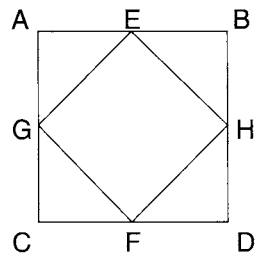
**Figure 1.9.** Mathias Roriczer's construction of nested squares.



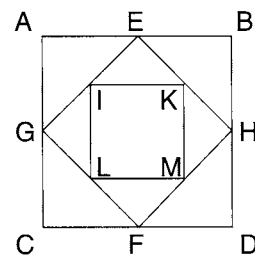
**Figure 1.10.** The first step in Roriczer's construction procedure.



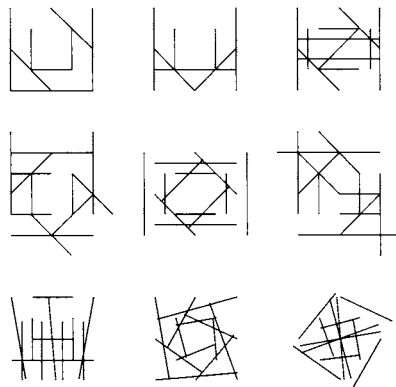
**Figure 1.11.** The second step in Roriczer's construction procedure.

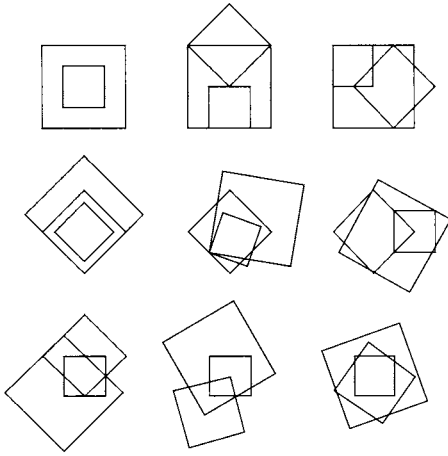


**Figure 1.12.** The third step in Roriczer's construction procedure.



**Figure 1.13.** Variations on Roriczer's shape produced by selecting and transforming straight lines.





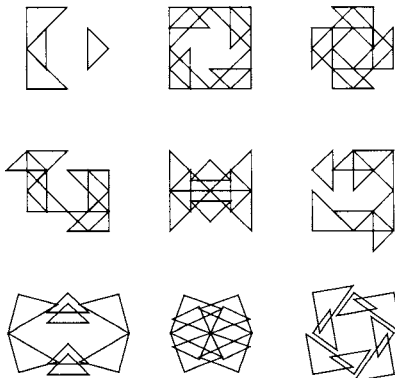
**Figure 1.14.** Variations on Roriczer's shape produced by selecting and transforming squares.

the allowable transformations such that you preserve some of the properties of the original figure. If you simply allow translation, rotation, reflection, and scaling in general, you end up – unhelpfully – with a universe of all possible figures made from 12 or fewer straight lines.) It is very straightforward to write search procedures that enumerate such designs.

Alternatively, using a standard CAD capability, the 12 lines might be grouped into three squares. Now, selection, transformation, and deletion operations produce variations of the kind shown in Figure 1.14. It is straightforward to produce extended ranges of variants by extending the repertoire of transformations – by allowing shear and stretch, for example.

So far so good; but what if you want to see the shape as a collection of eight triangles, and to produce variations of the kind shown in Figure 1.15? This is a perfectly natural and appropriate thing for a designer to do, but it is impossible to select and transform any of the triangles. As a consequence of the way the shape was constructed, and of the structure that was implicitly assigned in doing so, the triangles simply do not exist in the data structure. A whole universe of potentially interesting design variants has thereby been excluded from consideration.

Of course it is possible to reconstruct Roriczer's shape as a set of triangles, but then the squares are lost. And, as Stiny demonstrates in Chapter 2, the difficulty is



**Figure 1.15.** Variations on Roriczer's shape produced by selecting and transforming emergent triangles.