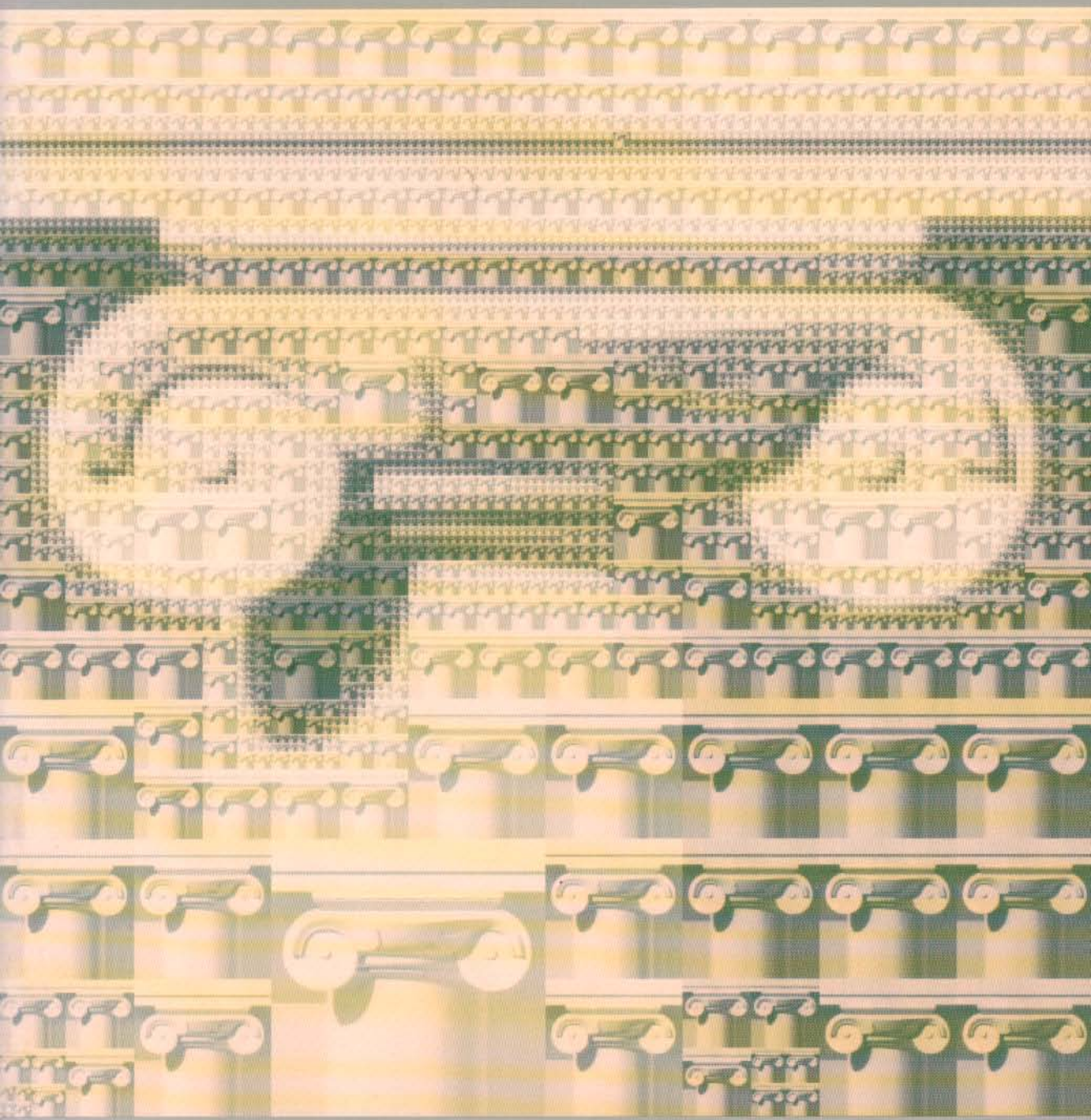
THE LOGIC OF ARCHITECTURE

DESIGN, COMPUTATION, AND COGNITION



WILLIAM J. MITCHELL

8 LANGUAGES OF ARCHITECTURAL FORM

GRAMMATICAL COMBINATION

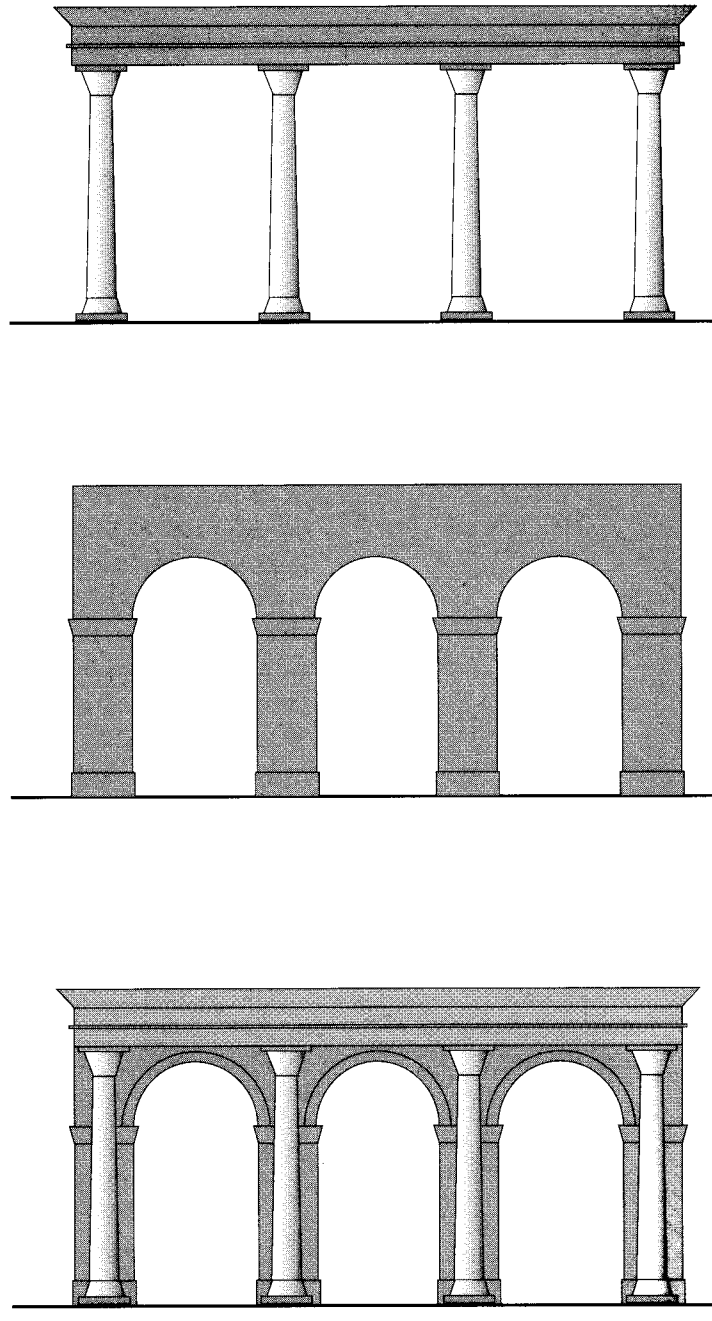
The trouble with algebras, as universes of design possibilities, is that they usually contain too much. They tend to contain vast numbers of possibilities that have no architectural meaning whatsoever, plus possibilities that are meaningful but irrelevant or uninteresting. The state-action trees that they establish contain numerous branches that are not worth exploring. We need some way to curb promiscuous combination of shapes, to tighten up the rules of the game.

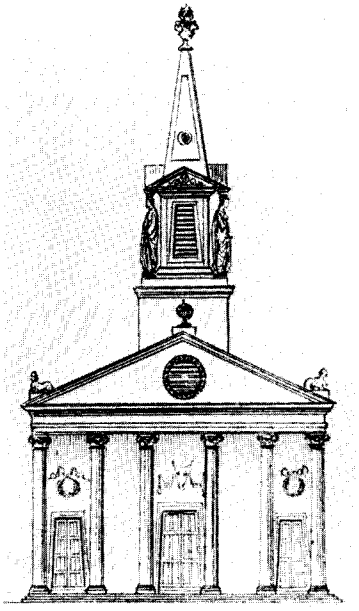
One powerful way to do this is to introduce the idea of grammatical combination of parts. We can, if we so choose, specify in the type definition of an architectural vocabulary element that it is only instantiated in certain kinds of combinations with other elements. That is, we specify certain external relations in the type definition. The analogy here with parts of speech is close; it is essential to being an English noun that it is only instantiated in English sentences in certain kinds of combinations with other words, as given by the rules of English grammar. Thus not every string of English words is an English sentence: only strings that comply with the rules of English grammar count as sentences. A clear and simple illustration of this kind of thing in architecture is provided by Alberti's handling of columns, piers, entablatures, and arches (figure 8.1), as analyzed by Rudolf Wittkower (1962):

> In his religious buildings Alberti consistently avoided the combination of arch and column. When he used columns he did, in fact, give them a straight entablature, while when he introduced arches, he made them rest on pillars with or without half-columns set against them as decoration. Alberti found the models for both forms in Roman architecture. But whereas the first motif is Greek, the Romans playing the role of mediators, the second is Roman. The first motif is based on the functional meaning of the column, the second on the cohesion and unity of the wall. To explain this latter point: in the Colosseum the arched pillars may be interpreted as residues of a pierced wall, with the half-columns, which carry the straight entablature, placed against them as ornament. In practice, therefore, Alberti's conception of the column is essentially Greek, while his conception of the arch is essentially Roman.

Critics sometimes suggest that certain usages violate rules of grammatical combination. Vitruvius warned that Ionic capitals should not be combined with Doric entablatures,

8.1
Albertian usage (following Roman precedent) of columns, entablatures, piers, and arches





8.2
"A most glaring absurdity": application of a Greek portico to a church, as depicted by Pugin in *The True Principles of Pointed or Christian Architecture*, 1841

"the usage of each class having been fixed long ago." Augustus Welby Pugin, in *The True Principles of Pointed or Christian Architecture* (1841), noted a difficulty that arose when English architects proposed to apply Greek porticos to churches (figure 8.2):

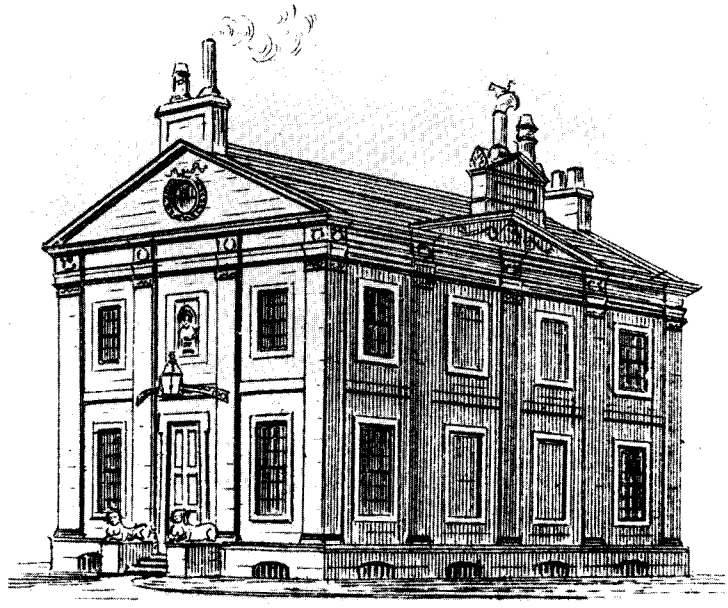
Christian churches require bells, by the sound of which the faithful may be called to their devotions. The bells, to be distinctly heard, must be suspended in a tower or belfry, and these are features utterly unknown in Greek architecture. A tower composed of a number of small porticos, set over one another, and placed in front of a mock temple, is a most glaring absurdity; nor is a tower of this description, starting out of nothing at the top of a portico, any better.

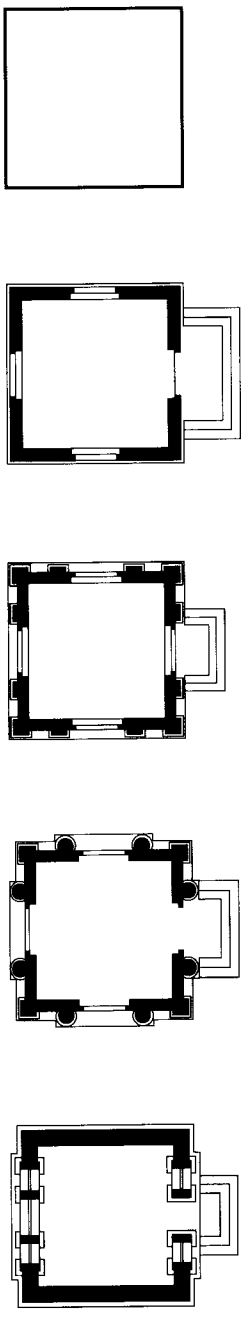
He was no better pleased by the application of Greek porticos to houses (figure 8.3).

Neither are they better adapted for domestic purposes; for it is still more absurd to see two or three tiers of windows introduced in the shell of a Greek temple, the roof of which is broken by numerous stacks of vainly disguised chimneys.

In biological thought, very similar ideas about restrictions on ranges of combinatorital possibility were formulated by Cuvier. In his *Leçons d'anatomie comparée* (1800) he noted (as Aristotle had done) that each kind of organ may take many forms, and that, if any form could coexist with any other form, a huge combinatorial universe would result. However, he argued, combinations must meet certain conditions, and only the combinations that meet these conditions actually exist in nature.

8.3
"A Grecian temple outraged in all its proportions and character": Pugin's depiction of the combination, in a house, of the Greek temple with windows and chimneys





8.4
Substitution of alternative wall and entrance treatments for a square plan (after Gibbs's *A Book of Architecture*, 1728)

SIMPLE PRESCRIPTIVE RULES

The grammatical rules of a language of architectural form, like those of a spoken language, may be specified in a variety of formats. The simplest approach, as employed, for example, by Pugin, is to display various exemplars of "correct" and "incorrect" practice. This technique has very commonly been employed by architectural theorists, from Vitruvius to the present day.

A more sophisticated approach is to state generalized prescriptive rules, as in elementary language textbooks. The Renaissance architectural theorists were particularly fond of doing this. In his Four Books of Architecture (1570) Palladio, for example, introduced rules of composition as follows:

> And altho' variety and things new may please every one, yet they ought not to be done contrary to the precepts of art, and contrary to that which reason dictates; whence one sees, that altho' the ancients did vary, yet they never departed from the universal and necessary rules of art, as shall be seen in my book of antiquities.

Typical of the prescriptive rules given by Palladio for villa designs are:

> The rooms ought to be distributed on each side of the entry and hall; and it is to be observed, that those on the right correspond with those on the left, that so the fabrick may be the same in one place as in the other. . . .
>
> The windows on the right hand ought to correspond to those on the left, and those above directly over them that are below; and the doors likewise ought to be directly over one another, that the void may be over the void, and the solid upon the solid, and all face one another, so that standing at one end of the house one may see to the other, which affords both beauty and cool air in summer, besides other conveniences.

REPLACEMENT RULES

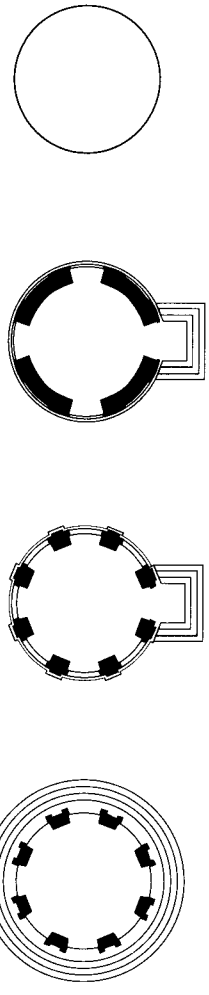
Yet another approach is to specify replacement rules. Grammarians of spoken language, for example, often set out sentence schemata like:

The_____ is green.

Then they specify the type of word substitutable for the blank, in this case a noun. Thus the schema might be expressed:

The ***Noun*** *is green.*

Noun is a variable ranging over all the English nouns: substitution of any instance of an English noun, such as *grass*, yields a grammatical English sentence.



8.5
Substitution of alternative wall and entrance treatments for a circular plan (after Gibbs, 1728)

Architectural theorists have occasionally established schemata and demonstrated substitutions in much the same way. James Gibbs (1728), for example, produced diagrams showing how various wall and entrance treatments could be substituted for each other in square and circular pavilions (figures 8.4, 8.5). In his *Encyclopaedia of Architecture* (1846), J. C. Loudon took a bare cube as the schema for a cottage, then showed how different exterior treatments might be substituted as appropriate to the owner's status. More recently, Bernard Tschumi has programmatically employed substitution of architectural elements from a chosen lexicon within the framework of a gridded ten-meter cube to generate a set of pavilions for the Park of La Villette in Paris.

RECURSIVE REPLACEMENT

Replacement rules become particularly interesting if they are applied recursively. Figure 8.6 shows how the recursive application of a simple replacement rule to an initial square yields plans for increasingly elaborate fourfold Islamic gardens, such as that of the Taj Mahal, which was originally subdivided by paths and canals into sixty-four smaller squares. Notice that the rule can be applied an unlimited number of times. Thus a very simple rule system specifies a countably infinite set of design possibilities for exploration.

Elaboration of this principle leads to a powerful technique for specifying universes of similarly-structured objects. Let us assume, for example, that we are concerned with a type of object known as a sentence, and that a sentence always consists of a noun phrase followed by a verb phrase. This can be expressed by the replacement rule[1]:

Sentence* → *NounPhrase VerbPhrase

The rule tells us that, whenever we see the string of symbols on the left-hand side, we can replace it by the string of symbols on the right-hand side. A second rule establishes the essential properties of a noun phrase:

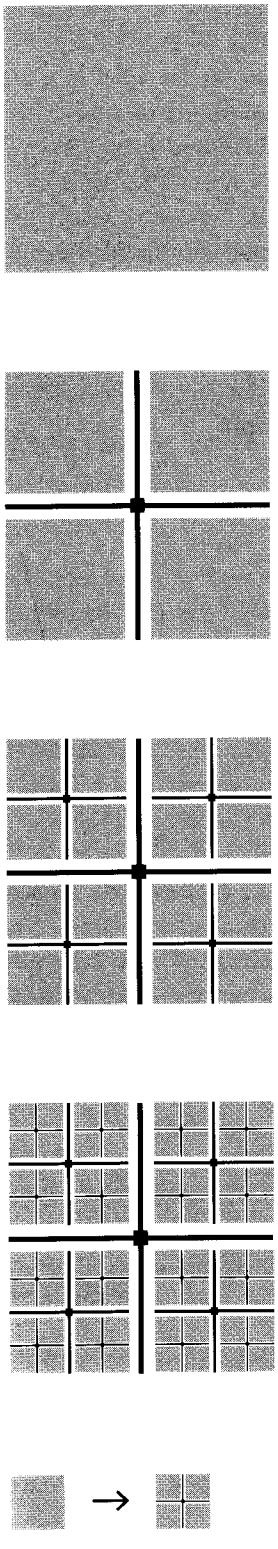
NounPhrase* → *Article Noun

And a third rule establishes the essential properties of a verb phrase:

VerbPhrase* → *Verb NounPhrase

Finally, there are rules establishing the ranges of the variables **Article**, **Noun**, and **Verb**:

Article → *a, the*
Noun → *column, beam*
Verb → *supports, loads*



8.6
The Islamic garden rule, as used in the garden of the Taj Mahal

(The comma indicates an alternative.) These rules are known as productions, and the set constitutes a production system, as follows:

Rule 1	***Sentence*** → ***NounPhrase VerbPhrase***
Rule 2	***NounPhrase*** → ***Article Noun***
Rule 3	***VerbPhrase*** → ***Verb NounPhrase***
Rule 4	***Article*** → *a, the*
Rule 5	***Noun*** → *column, beam*
Rule 6	***Verb*** → *supports, loads*

These six replacement rules, together with the starting symbol **Sentence**, constitute a phrase-structure grammar that generates a simple language.[2] Sentences in the language are generated by recursive application of the production system to the starting symbol to generate a sequence of symbol strings. A rule applies, and can be executed, whenever a substring of the current symbol string is matched by the left-hand side of a rule. We say that a new symbol string is *derived* from the current symbol string by application of that rule. A derivation is a sequence of such rule applications. Derivations terminate when no further matches can be found. Here, then, is an example of a derivation:

Starting symbol	***Sentence***
By rule 1	***NounPhrase VerbPhrase***
By rule 2	***Article Noun VerbPhrase***
By rule 3	***Article Noun Verb NounPhrase***
By rule 2	***Article Noun Verb Article Noun***
By rule 4	*a* ***Noun Verb Article Noun***
By rule 4	*a* ***Noun Verb*** *the* ***Noun***
By rule 5	*a column* ***Verb*** *the beam*
By rule 6	*a column supports the beam*
Termination	

This derivation is depicted graphically by the tree diagram in figure 8.7. Steps in derivations are often denoted as follows:

By rule 6	*a column* ***Verb*** *the beam* ⇒
	a column supports the beam

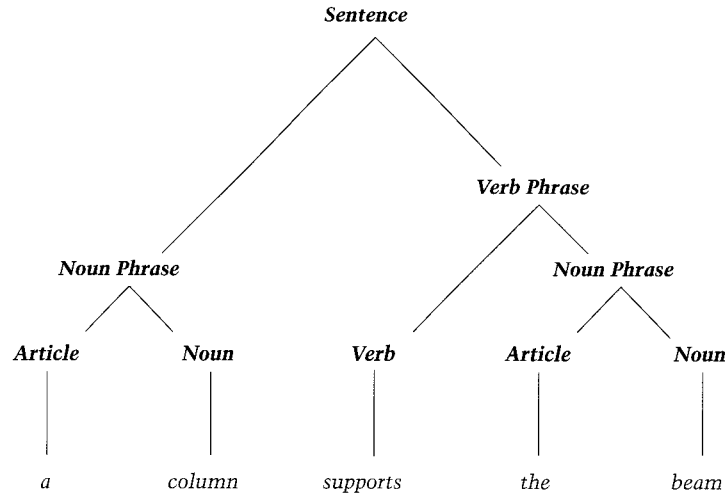
The symbol ⇒ is read "derives."

Sometimes, in a derivation process, several different substitutions are allowable. For example, the intermediate string

Article Noun Verb Article Noun

consists of five variables, each with two possible values, so $2^5 = 32$ different sentences can be produced from it. Thus a concise grammar can specify a large (in some cases countably infinite) set of sentences. Alternative derivations can be rep-

8.7
A tree diagram depicting the derivation and structure of a sentence



resented by a state-action tree in which the root is the starting symbol **Sentence**, branches represent applicable rules, internal nodes are "incomplete" sentences (strings that still contain variables), and "complete" sentences in the language (strings with no variables) are at terminal nodes.

Now let us assume that, for every production in the grammar there is a corresponding inverse, or recognition rule, as follows:

Reduction 1	***NounPhrase VerbPhrase*** → ***Sentence***
Reduction 2	***Article Noun*** → ***NounPhrase***
Reduction 3	***Verb NounPhrase*** → ***VerbPhrase***
Reduction 4	*a, the* → ***Article***
Reduction 5	*column, beam* → ***Noun***
Reduction 6	*supports, loads* → ***Verb***

These recognition rules can be applied recursively to a string of words to determine whether the string is a sentence in the language; that is, whether it complies with the grammatical rules. If the string can be reduced to the starting symbol **Sentence**, then it is a sentence: it has been shown to satisfy the predicate *sentence (**String**)*. Here is an example of a successful reduction:

Given string	*the beam loads a column*
By reduction 4	***Article*** *beam loads a column*
By reduction 4	***Article*** *beam loads* ***Article*** *column*
By reduction 5	***Article Noun*** *loads* ***Article*** *column*
By reduction 2	***NounPhrase*** *loads* ***Article*** *column*
By reduction 5	***NounPhrase*** *loads* ***Article Noun***
By reduction 2	***NounPhrase*** *loads* ***NounPhrase***
By reduction 6	***NounPhrase Verb NounPhrase***
By reduction 3	***NounPhrase VerbPhrase***
By reduction 1	***Sentence***
Termination	

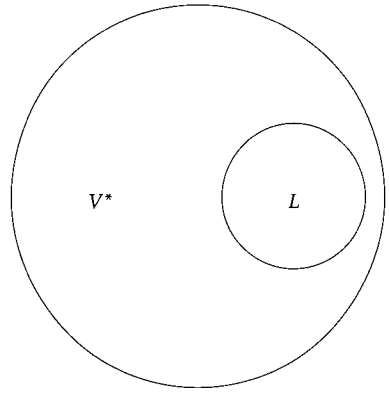
Compilers perform such reductions when they check the syntax of programs in computer languages, and language students do so when they diagram sentences.[3]

In summary, a grammar concisely encodes a type definition. If it is executed in generative mode it produces instances of the type, and if it is executed in recognition mode it determines whether or not a given object is an instance. Furthermore, the grammar assigns a syntactic structure to instances: it specifies a way (or perhaps alternative ways) of decomposing instances into parts of various recognizable types, such as noun phrases, verb phrases, and so on.

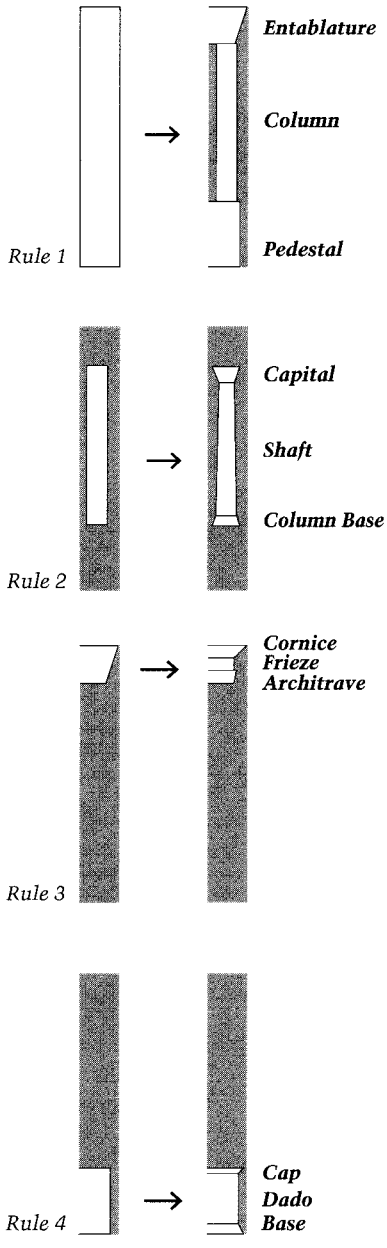
Technically, the language L specified by a grammar is a subset of the carrier set V^* of an algebra established by the vocabulary V and combination operators (figure 8.8). The vocabulary is divided into nonterminal elements (variables) and terminal elements (constants). In our example, the nonterminal vocabulary consists of the symbols **Sentence**, **NounPhrase**, **VerbPhrase**, **Article**, **Noun**, and **Verb**. The terminal vocabulary consists of six English words: *a*, *the*, *column*, *beam*, *supports*, *loads*. The concatenation operator applied to the terminal vocabulary yields an infinite number of strings, such as:

the the the
a the beam column
supports loads a

Even if we specify that a string must be exactly five words long, there will still be $6^5 = 7{,}776$ strings. But we admit as sentences only those strings of terminals that are derived from the starting symbol **Sentence** by application of the rules. Thus our grammar specifies that only thirty-two of these five-word strings are sentences in the language.



8.8
The language L specified by a grammar is a subset of the carrier set V^* of an algebra



8.9
The rules of a simple grammar that generates schematic designs for classical columns

THE PARALLEL OF THE ORDERS AS A GRAMMAR

If we want to use a grammar to restrict the possible states of a design world, we must formulate the rules in terms of the types of shapes, labels, and relations populating that world. This can be done in many different ways. Let us begin with a simple example.

Parallels of the classical orders usually specify that an order consists of three parts: pedestal, column, and entablature. This might be expressed verbally by the replacement rule:

Order* → *Pedestal Column Entablature

Then further rules of trichotomous subdivision might be added:

Pedestal* → *Base Dado Cap
Column* → *ColumnBase Shaft Capital
Entablature* → *Architrave Frieze Cornice

The same rules may be expressed graphically, as shown in figure 8.9.

So far this is not a very interesting grammar, since it merely specifies a single object subdivided into nine parts. But it begins to specify a more interesting language if we introduce rules that provide for alternative substitutions, such as:

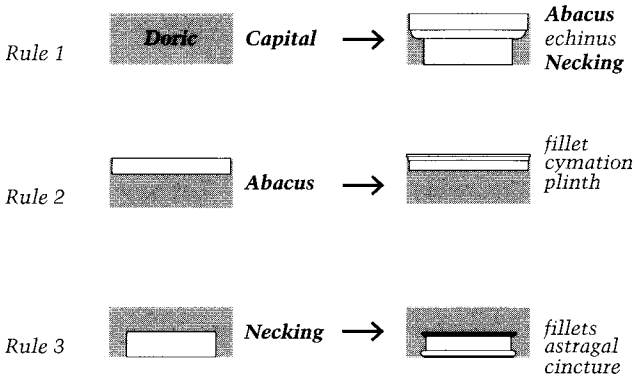
Capital* → *Doric, Ionic, Corinthian

Furthermore, we can introduce rules that tell how to detail the various parts. For example, a Doric capital is detailed as follows:

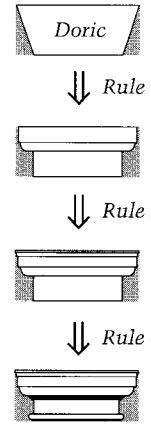
Capital* → *Necking *echinus* ***Abacus***
Abacus → *plinth cymation fillet*
Necking → *cincture astragal fillets*

Figure 8.10 shows graphic versions of these rules and figure 8.11 illustrates their application to derive a complete Doric capital. It is easy to see how a complete parallel of the orders might be encoded by such rules.

When run in generative mode these rules tell how to compose the parts of a classical column correctly: they encode knowledge of how instances of this type of artifact are put together. When run in recognition mode they tell how to parse a classical column into a hierarchy of labeled parts (figure 8.12). (These rules parse any classical column in a unique way, but in general grammars may parse objects in multiple ways.) Figure 8.13 illustrates the step-by-step reduction process by which an object is recognized and labeled as a classical column.

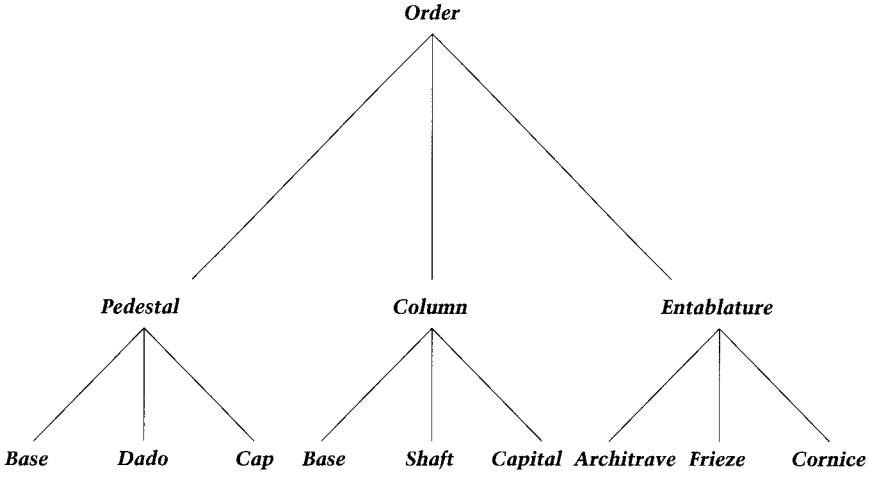


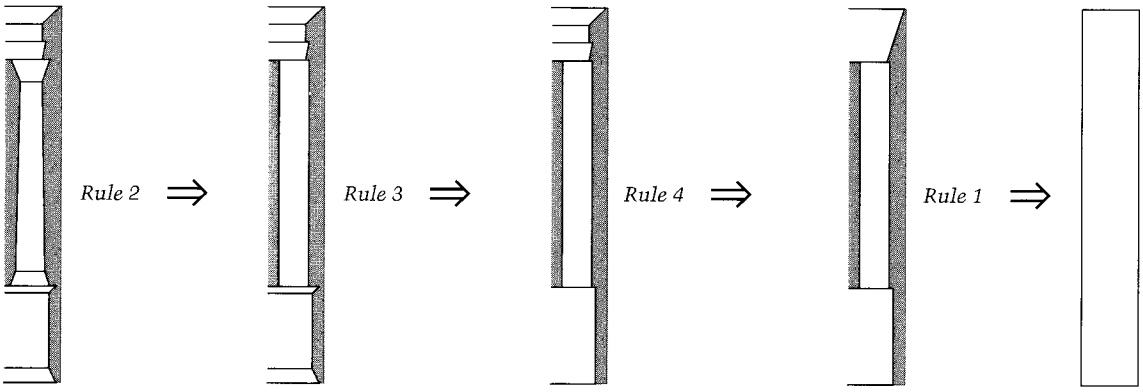
8.10
Rules for detailing a Doric capital



8.11
Top-down refinement of the design of a Doric capital

8.12
A tree diagram depicting the derivation and structure of a schematic classical order





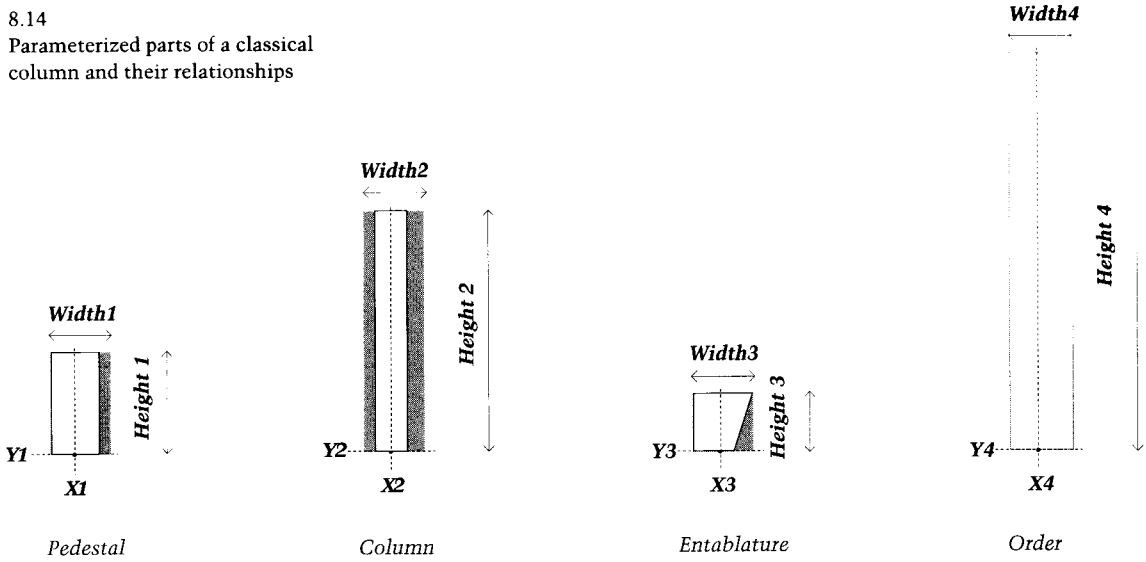
8.13
Reduction of a classical order

Notice how the classical rules, as I have expressed them here, structure a top-down design process: the designer begins with a very abstract definition and recursively refines it into a detailed drawing of an instance of the type. The design strategy is to accomplish the final goal of producing a complete, detailed drawing via solution of a sequence of subproblems. The subgoal, at each stage, is to appropriately refine the current design to the next level of detail.

Conversely, the classical rules structure a bottom-up recognition process. We first look for classical vocabulary elements such as astragals and cinctures, then check to see that these are correctly combined into higher-level components such as Doric capitals. Then we check that these higher-level components are correctly combined, and so on recursively until we can show that the whole composition not only has the correct classical parts, but is also correctly put together.

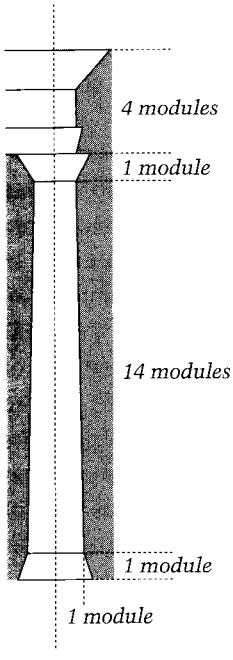
All this works because a classical order is, like a sentence, a linear sequence of elements (running bottom to top instead of left to right). Thus we can embed a grammar in an algebra in which the sole operation is concatenation of sequences of elements. But in general, if we want to specify languages of two-dimensional and three-dimensional form, we must embed grammars in algebras that have wider repertoires of operations. One way to do this is to parameterize all the vocabulary elements, as shown in figure 8.14. Assignments of values to parameters then specify geometric transformations of instances, and required spatial relationships of elements in compositions can be expressed by writing predicates of the parameters. So we can more completely express the rule governing relationships of pedestal, column, and entablature in an order by writing:

8.14
Parameterized parts of a classical column and their relationships



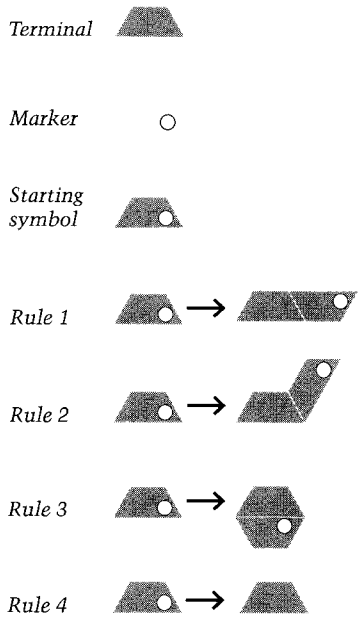
Order (X4, Y4, Height4, Width4) →
Pedestal (X1, Y1, Height1, Width1)
Column (X2, Y2, Height2, Width2)
Entablature (X3, Y3, Height3, Width3)
such that
*equal (**X4**, **X1**)*
*equal (**Y4**, **Y1**)*
*equal (**Width4**, **Width1**)*
*equal (**X1**, **X2**)*
*equal ((**Y1** + **Height1**), **Y2**)*
*equal (**Width1**, **Width2**)*
*equal ((**Y2** + **Height2**), **Y3**)*
*equal (**Width2**, **Width3**)*
*equal (**Height4**, (**Height1** + **Height2** + **Height3**))*

8.15
Correctly proportioned classical column (according to the rules given by Vignola)

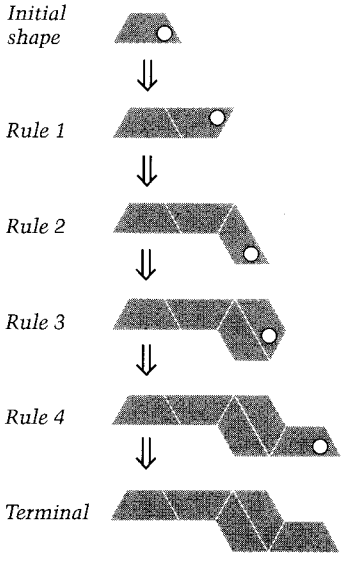


This rule could be elaborated to specify the correct classical ratio of column diameter to height, and so on (figure 8.15).

The designer's first task, in producing a design within this language, is to choose a value of **Height4** such that the order fits correctly into its context. This decomposes into subtasks of dimensioning and correctly relating pedestal, column, and entablature. The goal, in each of these subtasks, is to choose values for the variables such that the specified predicates are satisfied. Each of these subtasks then decomposes into still lower-level subtasks, and so on until values have been chosen for all the variables, predicates at every level have been satisfied, and the order has been designed down to the smallest detail.



8.16
The rules of the half-hexagon table grammar



8.17
Derivation of a design in the half-hexagon table grammar

THE HALF-HEXAGON TABLE GRAMMAR

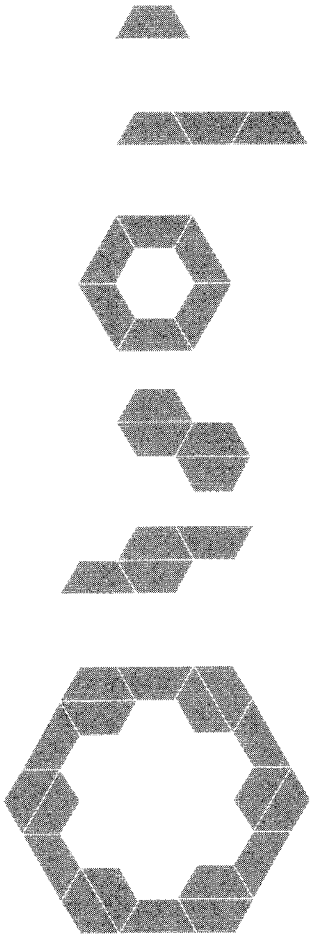
Figure 8.16 illustrates another example of a grammar defined in a two-dimensional world of closed shapes. There is one terminal vocabulary element—a half-hexagon of fixed dimension—and one marker element—a white circle. (Markers function like construction lines: they guide the development of compositions but do not appear in completed compositions.) The game defined by this grammar is conveniently played with cardboard cutouts and coins. There is no explicit parameterization of vocabulary elements. We assume, instead, that the isometric transformations (identity, translation, rotation, and reflection) may be applied to any half-hexagon or circle. Let us also assume that the half-hexagons depict tables in a furniture layout.

The positions of the markers define attachment points for new tables. The starting symbol is a single table with a single attachment point on one edge. There are three rules for attachment of new tables in different ways, and a fourth rule simply removes the attachment point to terminate the attachment process. A rule applies when its left-hand side can be brought into coincidence with a subshape by some isometric transformation. Recursive application of the rules derives sequences of arrangements such as that shown in figure 8.17. Some completed designs in the language are illustrated in figure 8.18.

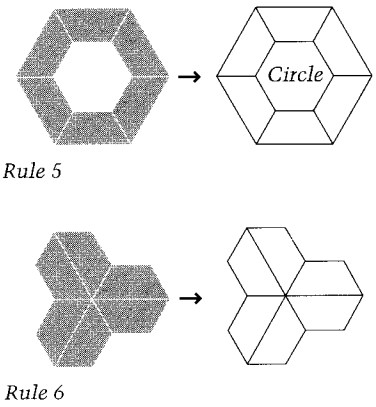
Now let us introduce two additional rules, as illustrated in figure 8.19. One recognizes and labels a type of arrangement called a circle, and the other recognizes and labels a type called a triad. We can now formulate problems such as "Construct an arrangement that includes two circles and a triad." Figure 8.20 illustrates a solution.

In this case the rules of the grammar structure a bottom-up design process. Fully-detailed components (the half-hexagon tables) are given at the outset, and the designer must assemble these (much as in a jigsaw puzzle) in such a way that specified qualities emerge. A small part of the infinite state-action tree that must be searched for a solution is shown in figure 8.21.

This tree may be searched systematically in either depth-first or breadth-first fashion. In depth-first search the designer picks an alternative from among the several available at each node, and so follows a branch deeply into the tree until a solution is found or until it becomes clear that the branch will not lead to a solution. If the branch does not lead to a solution, the designer backtracks then follows another branch, and so on. In breadth-first search, instead of diving



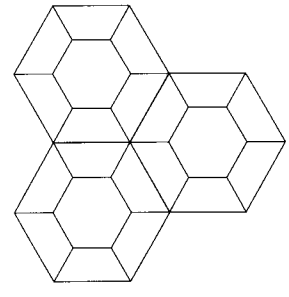
8.18
Some plan arrangements in the language specified by the half-hexagon table grammar



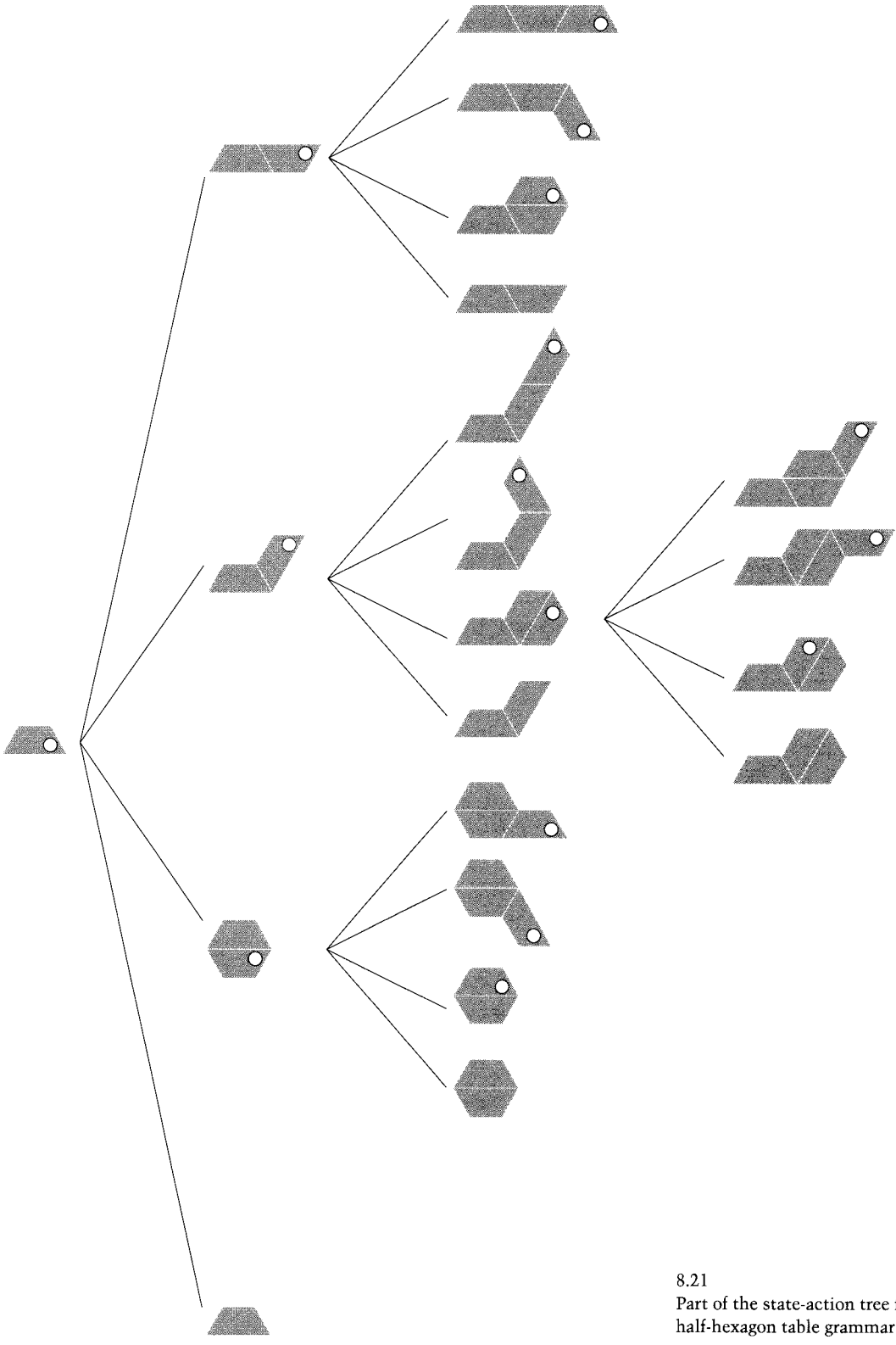
Rule 5

Rule 6

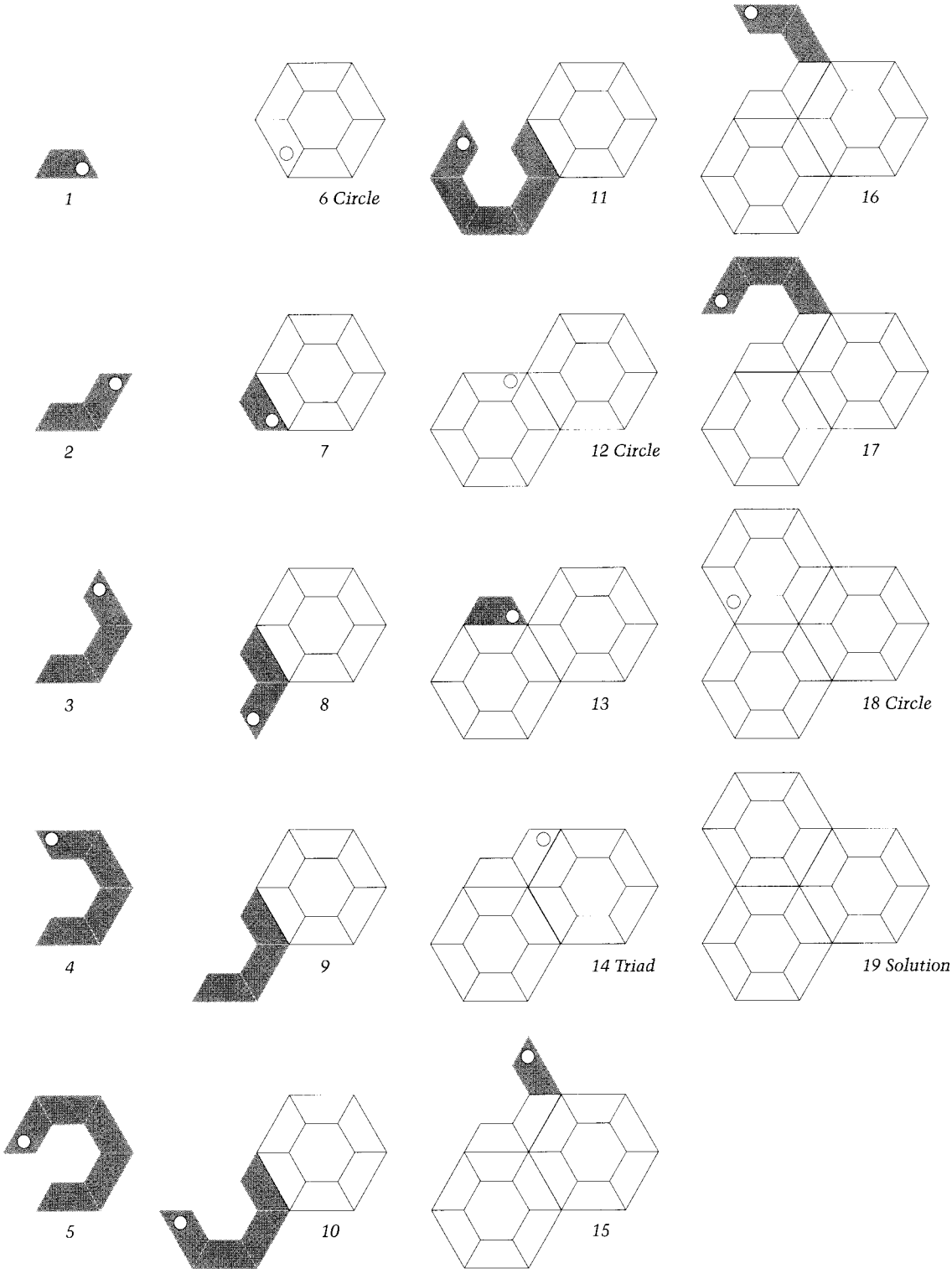
8.19
Recognition and labeling rules



8.20
A solution to the problem "Construct an arrangement that includes three circles and a triad"



8.21
Part of the state-action tree for the half-hexagon table grammar

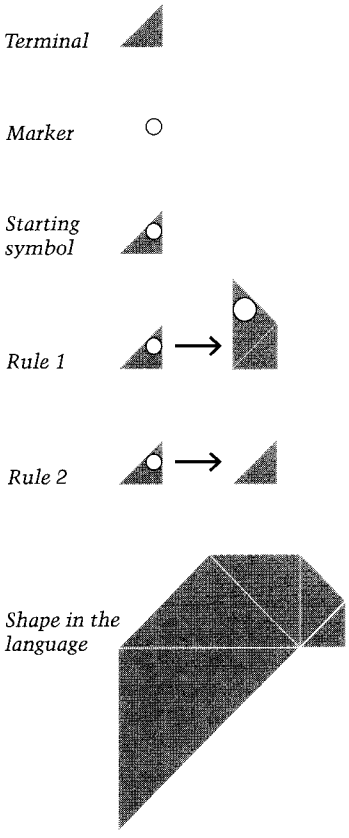


8.22
Approaching the solution via a sequence of subgoals

deeply into the tree, the designer pushes uniformly into it, level by level. All of the nodes at the first level are examined, then all of the nodes at the second level are developed and examined, and so on.

Both of these strategies become much more efficient if the designer has some good way of deciding which branches look most promising and exploring those first. In this case, the most obvious strategy is to accomplish the final goal via a sequence of subgoals of fitting individual pieces appropriately in place. The overall goal can be decomposed into subgoals of forming circles and triads. These subgoals, in turn, can be decomposed into lower-level subgoals of choosing correctly among rules one, two, and three (which fit tables against their predecessors in different ways) at each stage. If the current intermediate subgoal is to form a circle, for example, it is appropriate to choose rule two (which constructs a fragment of a circle). And if the current intermediate subgoal is to form a triad, it is appropriate to choose rule three (which constructs a fragment of a triad). By intelligently relating means to ends in this way, a designer can rapidly construct a solution (figure 8.22).

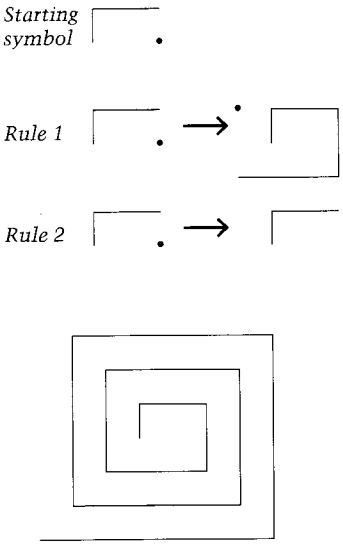
8.23
A simple grammar with scaling



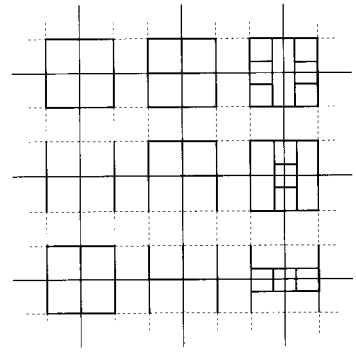
GRAMMARS WITH SCALING

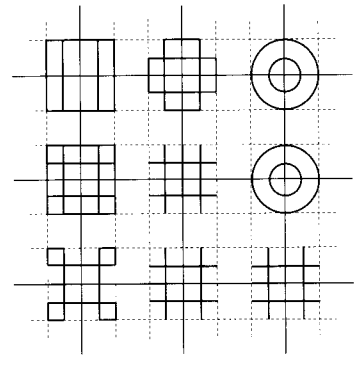
In the half-hexagon table grammar, shape equivalence is defined under isometric transformations. But we can, if we wish, define equivalence under broader classes of transformations. Figure 8.23, for example, shows a simple grammar that generates compositions of right triangles. In this case, a rule applies when its left-hand side can be brought into coincidence with a subshape by some similarity transformation.

If we think of grammars purely as abstract systems, we can define shape equivalence in any way we like. But if we want to relate a grammar to a particular construction world, it is more useful to define shape equivalence in a way that reflects the properties of that world. If real half-hexagon tables are made in just one size, for example, it makes sense to think of them as equivalent under rigid transformations, but it is *not* useful to broaden the definition of equivalence to provide for scaling. But if rooms can be made any size, we might find it useful in a floor-plan grammar to define shape equivalence under similarity or affine transformations. Figure 8.24 shows a grammar for Le Corbusier's spiral museum. Here room shapes that can be brought into coincidence by translation, right-angle rotation, and unequal scaling are taken to be equivalent.

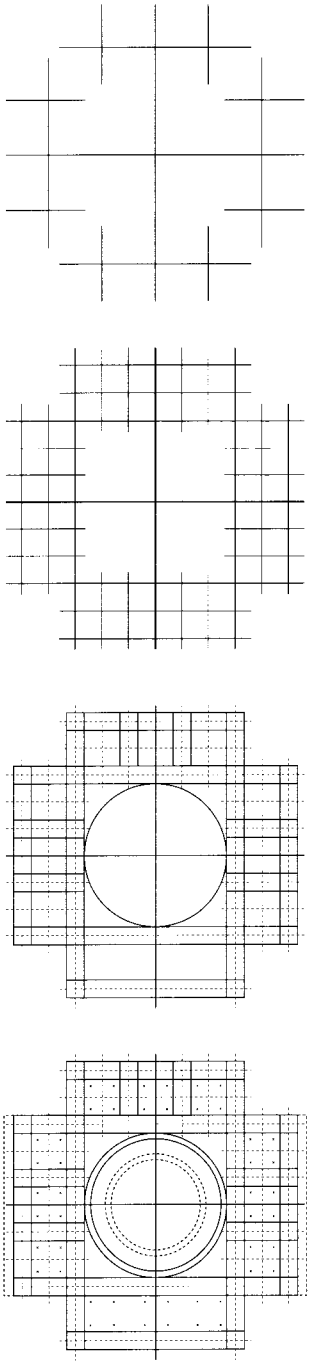


8.24
A grammar for Le Corbusier's spiral museum





8.25
Skeletons of plan construction lines (after Durand's *Partie graphique des cours d'architecture à l'Ecole Royale Polytechnique*, 1821)

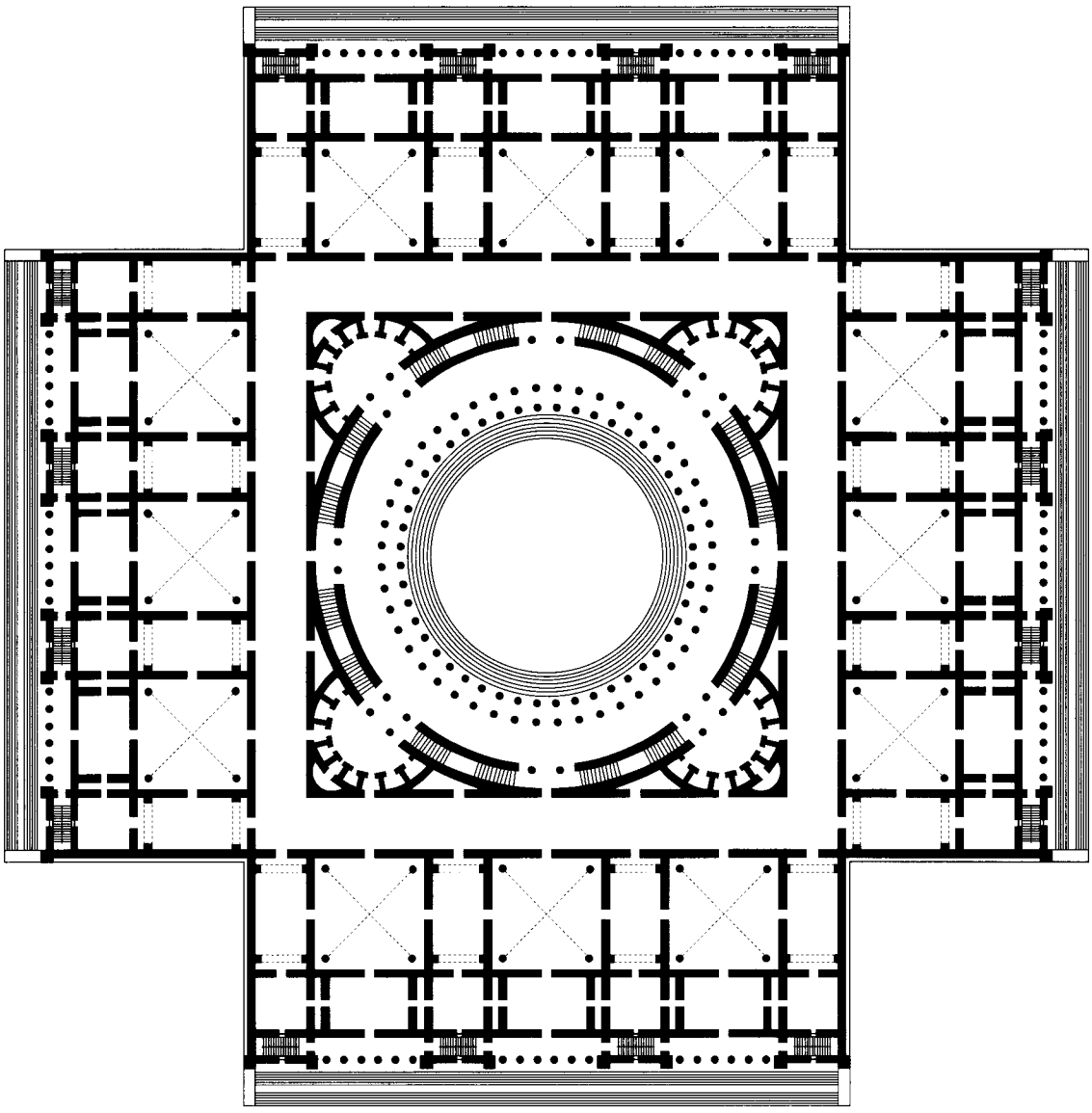


8.26
Steps in the top-down process of refining a skeleton into a fully detailed plan (after Durand's *Précis des leçons d'architecture*, 1802–5)

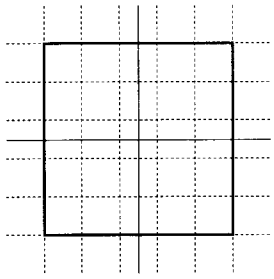
Purely top-down and purely bottom-up design processes, as illustrated by these examples, are limiting cases that we rarely encounter in practice. Instead, the rules of grammars usually establish complex domains in which exploration of possibilities may proceed in both top-down and bottom-up directions. J. N. L. Durand's *Précis des leçons d'architecture* (1802–5) and *Partie graphique des cours d'architecture* (1821) clearly illustrate this.[4]

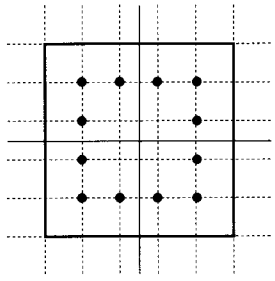
Durand's text and diagrams constitute a compendium of neoclassical design rules. Cross-axes, grids, squares and circles are taken as primitive shapes, and a diagram (figure 8.25) shows how to assemble these, in bottom-up fashion, into symmetrical skeletons of construction lines. Another key plate (figures 8.26, 8.27) illustrates the top-down process of recursively refining a skeleton into a fully detailed floor plan. Numerous additional plates illustrate a lexicon of alternative substitutions (figure 8.28) and examples of syntactically correct combinations of elements (figure 8.29). Durand's rules do not quite provide a complete, consistent specification of a classical architectural language, but it is straightforward to develop them into a grammar that does.

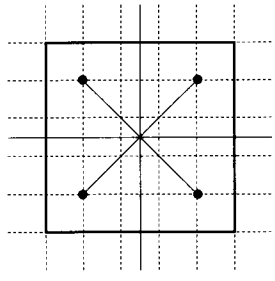
If you try to design within the framework of Durand's rules, it soon becomes evident that the strengths and weaknesses of top-down and bottom-up strategies are complementary. In top-down design the subproblems are ones of choosing and adapting elements to fit within a framework that has been established at a higher level. At a certain point it may prove impossible to substitute or to adapt any of the available elements to fit as required. This necessitates moving back to a higher level of abstraction and adjusting the overall framework so that the lower-level problems are redefined. Conversely, in bottom-up design, the subproblems are ones of assembling known pieces to achieve specified emergent properties. At a certain point it may prove impossible to find a combination of the pieces at hand that achieves the desired result. This necessitates moving back to a lower level of abstraction and redesigning the components so that they can be put together in different ways—thus redefining the higher-level problems.

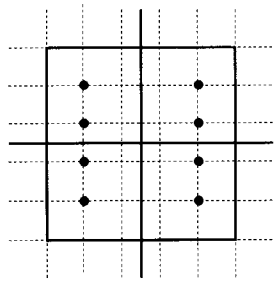


8.27
The final plan, after shapes representing architectural elements from the classical lexicon have been substituted and all construction lines have been removed

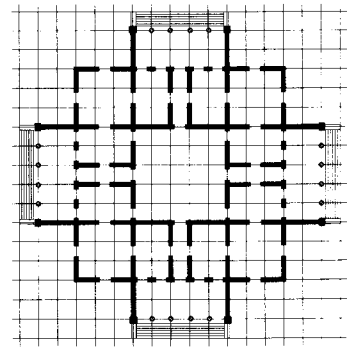


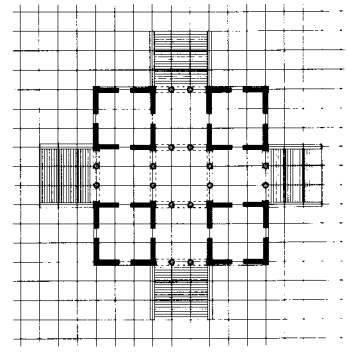


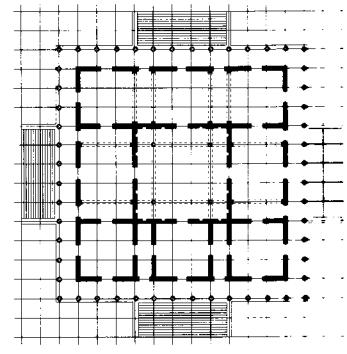




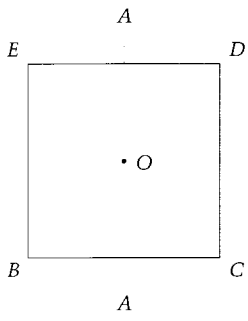
8.28
Alternative substitutions from the classical architectural lexicon (after Durand's *Partie graphique des cours d'architecture*)







8.29
Examples of syntactically correct compositions (after Durand's *Partie graphique des cours d'architecture*)

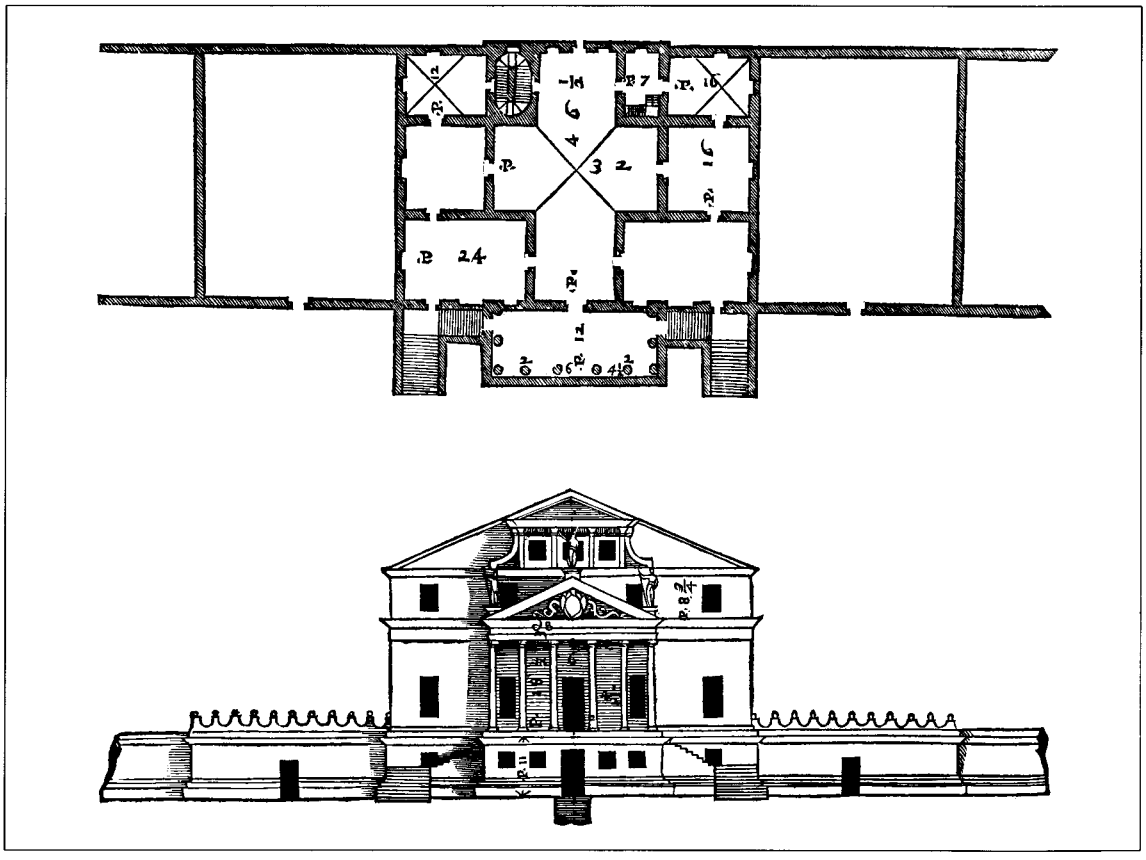


8.30
Association of labels with points and with ends of line segments

For a final example, let us consider a fairly sophisticated grammar to generate villa floor plans in the style of Palladio. Since Palladio was one of the first architects to explore plan ideas by sketching numerous variants, it is appropriate to define this grammar in a two-dimensional world of lines. It will suffice to take as the starting point the algebra of straight-line shapes under shape addition and subtraction that was introduced in chapter 4. Where L^* is the universe of two-dimensional straight-line shapes, then, the Palladian grammar specifies a subset P of L^*.

Stiny (1980) has defined, for this sort of world, a powerful type of grammar known as a parametric shape grammar.[5] Shapes, under Stiny's definition, consist of points, lines, and labels. Labels are alphabetic characters or character strings and may be associated with points and lines as illustrated in figure 8.30. Rules specify how subshapes of a composition in progress may be replaced by other shapes. A rule applies if there is a similarity transformation that will bring the shape on the left-hand side into coincidence with a subshape of the composition in progress such that there is a one-to-one match of labels. Shapes have proportion parameters, and the values of these can be left unassigned by the grammar so that dimensionless plan schemata, which can be proportioned in different ways, are produced. (When rules are applied it is assumed that assignments to the parameters of the matching shapes are the same.) Labels are used to control application of rules and to define termination conditions.

Notice that, in this formalism for defining grammars, the sharp distinction made earlier between terminal shapes (which appear in completed designs in a language) and marker shapes ("construction lines," which do not appear in completed designs) is abandoned. A derivation terminates when labels (not markers) have been eliminated. The importance of this becomes evident when we consider the roles of shapes in depictions. Where there is a sharp distinction between terminals and markers, we assume that terminals depict physical components and assemblies in the construction world, while markers are just organizing abstractions that do not depict physical objects in the construction world. Thus, in the earlier example, half-hexagons depict tables in the construction world, but the circular markers do not depict anything in the construction world. But in a language specified by a parametric shape grammar, as now defined, shapes may have more or less precise meanings in the construction world. This allows for a design to evolve smoothly from an



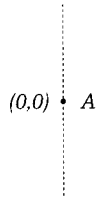
8.31
The Villa Malcontenta as shown in Palladio's *Four Books of Architecture*

initial vague sketch to final working drawings that precisely specify the sizes, shapes, materials, and locations of physical components.

The Palladian grammar given here is formulated as a parametric shape grammar.[6] It is a slight modification of a grammar published by Stiny and Mitchell in 1978. I will discuss the vocabulary and rules of the language and illustrate them through a step-by-step derivation of the plan of the Villa Malcontenta (figure 8.31).

The grammar derives plans in top-down fashion, working from the footprint and an organizing grid down to the details of walls, columns, doors, and windows. Thus it fairly closely follows Durand's method. Derivations are organized into eight main stages, as follows:

1. grid definition
2. exterior-wall definition
3. room layout
4. interior-wall realignment
5. principal entrances—porticos and exterior wall inflections
6. exterior ornamentation—columns
7. windows and doors
8. termination.



8.32
The initial shape from which all villa plans are generated: an axis through a labeled point

The initial shape, from which all plans are generated, establishes a point labeled A at the origin of the coordinate system (figure 8.32). Tartan grids are generated around this point using the rules specified in figure 8.33. Figure 8.34 shows how these rules are applied to generate a small tartan grid. The grid required for layout of the Villa Malcontenta (and derived using these rules) is shown in figure 8.35. The dimensioning lines (with arrowhead) indicate the parameters controlling grid dimensions. We will leave values unassigned here and simply assume that any assignment of values complies with Palladio's well-known rules of proportion and maintains bilateral symmetry about the central axis.

Once a grid is generated, it is circumscribed by a rectangle to define an exterior wall. This operation is performed by the rule specified in figure 8.36. The effect of applying this rule to the grid of the Villa Malcontenta is shown in figure 8.37. For clarity, the wall positions have been shaded. Meanings of shapes are now becoming less ambiguous. We can see that some of them specify locations for solid construction elements and others specify locations for habitable voids.

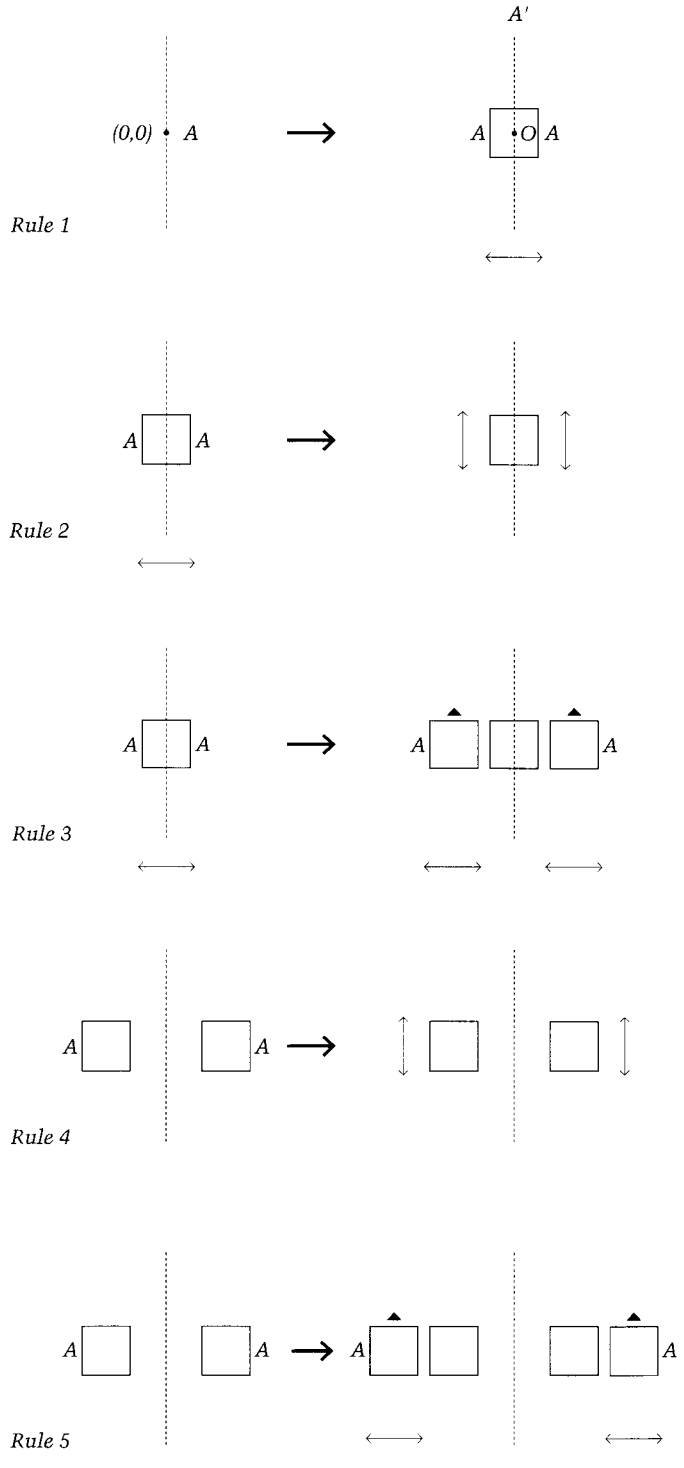
The rules specified in figure 8.38 provide for formation of rectangular, I-shaped, T-shaped and +-shaped central rooms. Application of rules 13, 12, and 18 to the Villa Malcontenta's wall pattern produces the result illustrated in figure 8.39. Notice how the meanings of lines have shifted as the design has evolved. They began as abstract gridlines, then were seen as possible boundaries between solids and voids, and are now seen as room boundaries. Similarly, the interior polygons are now seen as rooms.

Interior wall realignment is a subtlety occasionally introduced to allow achievement of appropriate dimensions and proportions. The rules of this are specified in figure 8.40. They are not applied in derivation of the Villa Malcontenta plan.

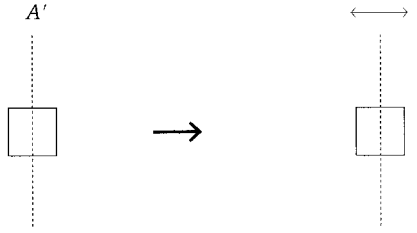
The rules for handling principal entrances are specified in figure 8.41. Palladio drew on an extensive lexicon of entrance treatments (probably in response to the varying status of his clients, and the exigencies of site and budget), so quite a few rules are needed. The Villa Malcontenta's treatment is illustrated in figure 8.42.

The addition of columns to a portico can be handled in many different ways: it becomes the occasion for an unconstrained ornamental flourish. Palladio's numerous variations

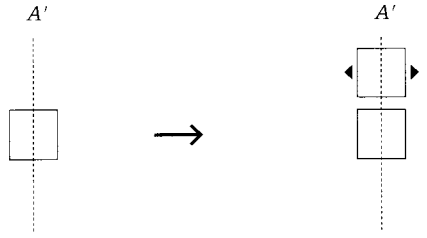
8.33
Rules for the generation of bilaterally symmetrical tartan grids



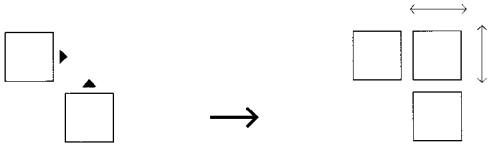
8.33 (cont.)



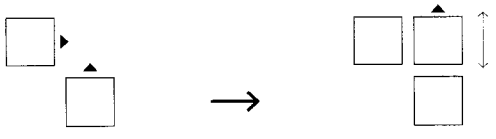
Rule 6



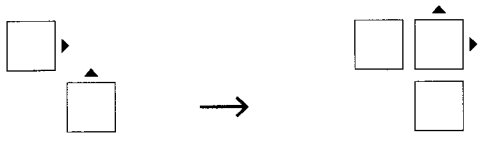
Rule 7



Rule 8

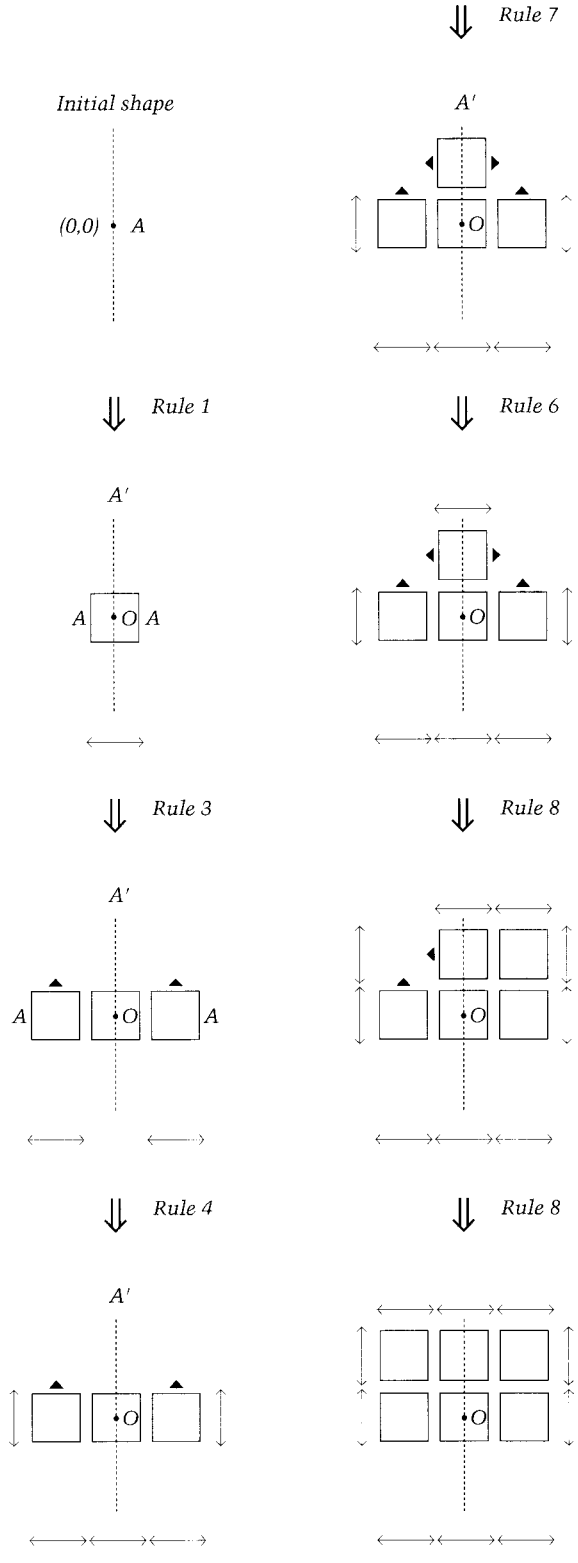


Rule 9

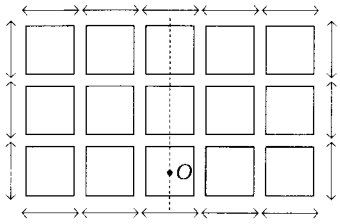


Rule 10

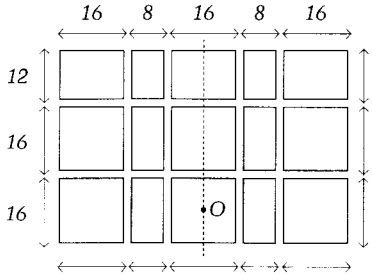
8.34
Application of the rules to generate a small tartan grid



8.35
The tartan grid generated for the Villa Malcontenta

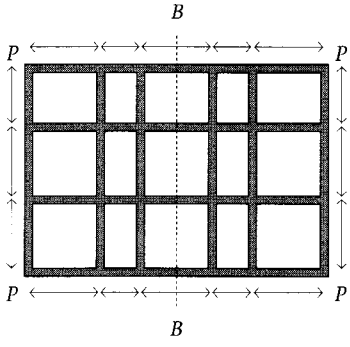


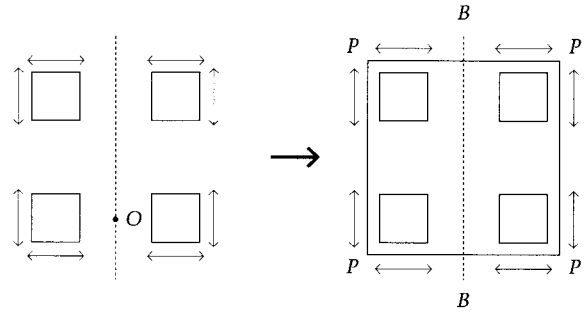
Undimensioned layout



Layout with correct Palladian values assigned to dimensioning variables

8.37
The underlying wall pattern of the Villa Malcontenta

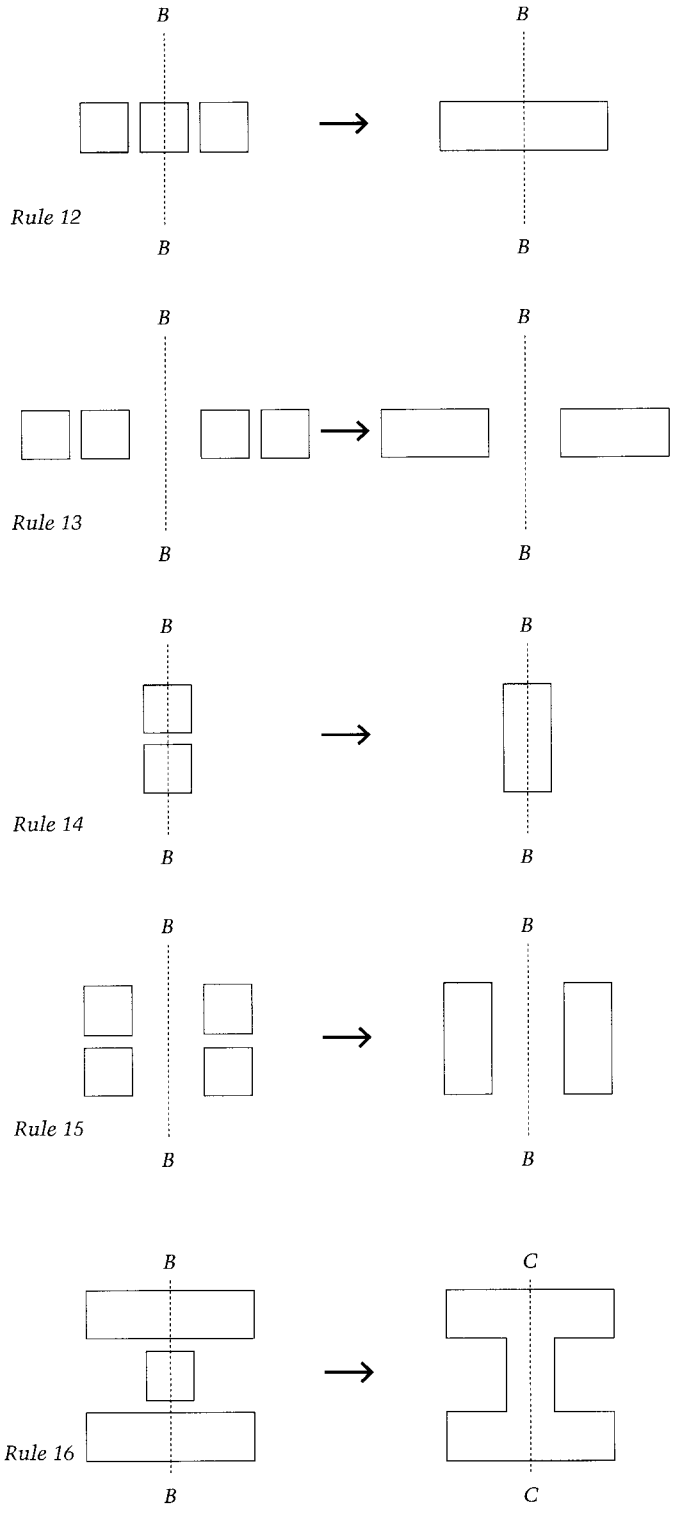




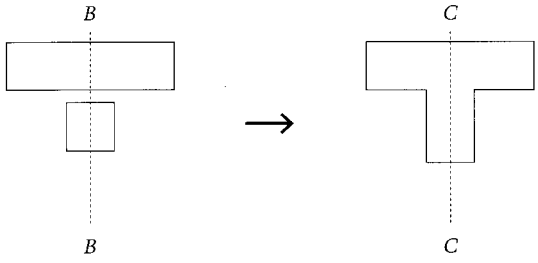
Rule 11

8.36
Rule for inscribing an outer rectangle to define exterior wall surfaces

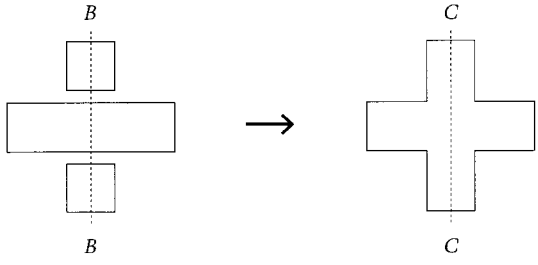
8.38
Rules for concatenation of cells to produce larger and more complex rooms



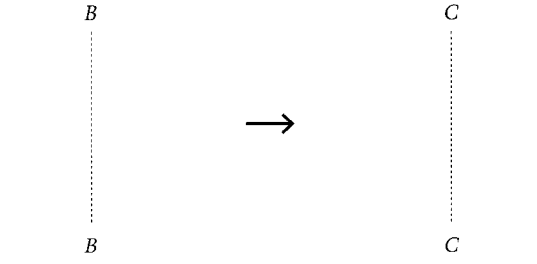
8.38 (cont.)



Rule 17

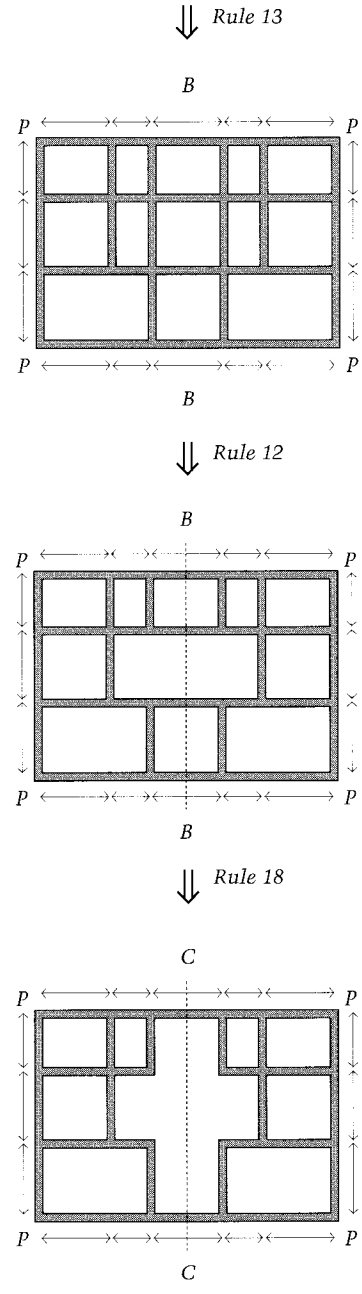


Rule 18

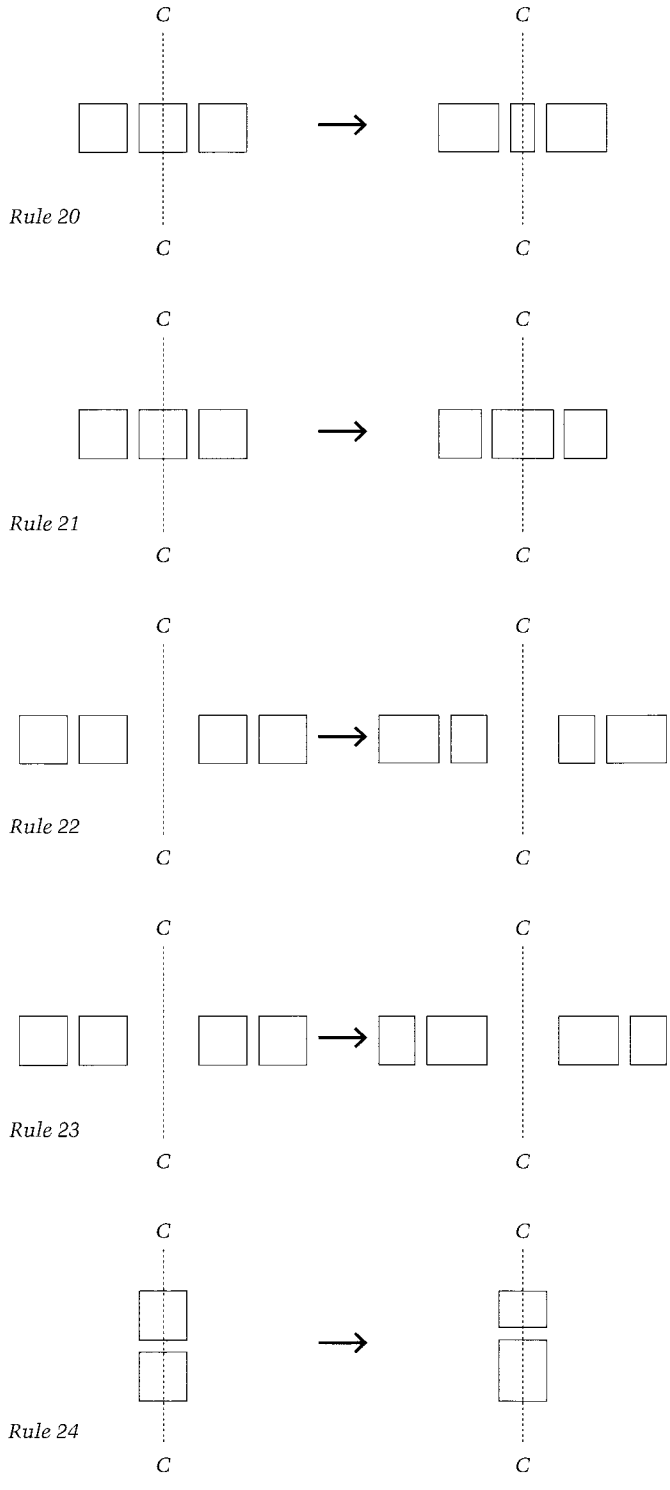


Rule 19

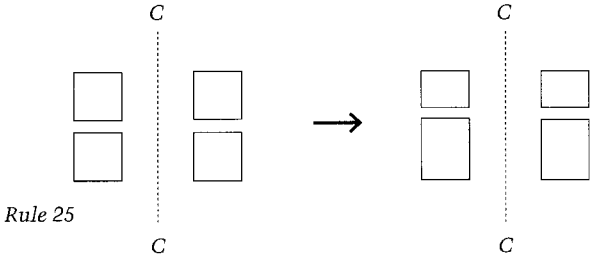
8.39
Derivation of the room layout of the Villa Malcontenta



8.40
Rules for interior wall realignment

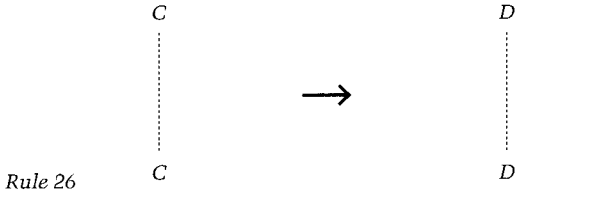


8.40 (cont.)

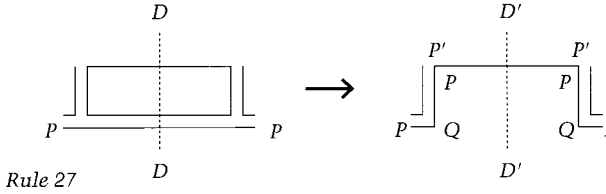


Rule 25

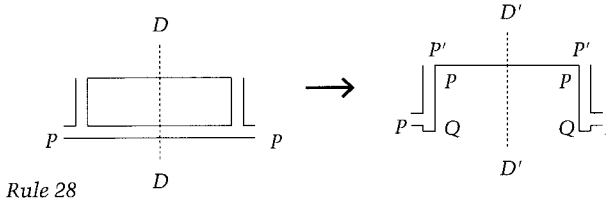
8.41
Lexical insertion rules providing for a wide variety of entrance treatments



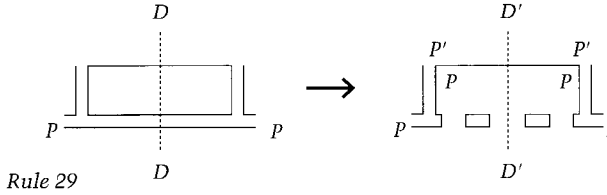
Rule 26



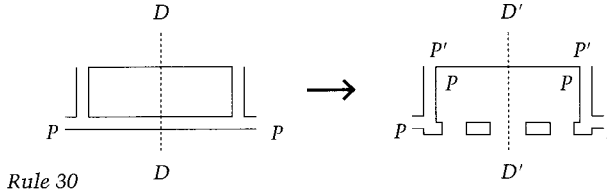
Rule 27



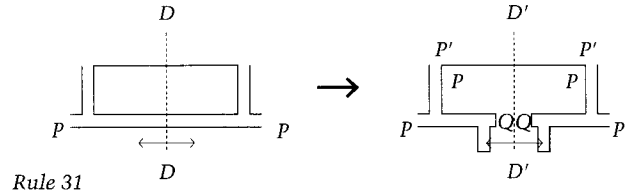
Rule 28



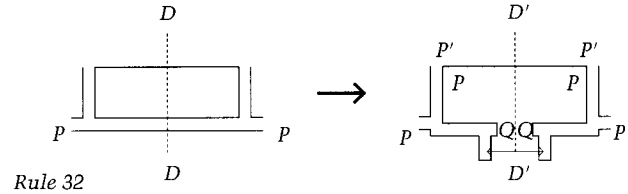
Rule 29



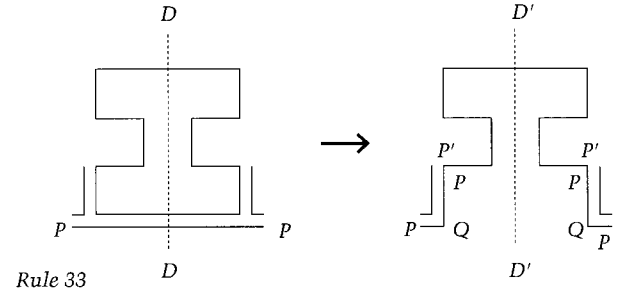
Rule 30



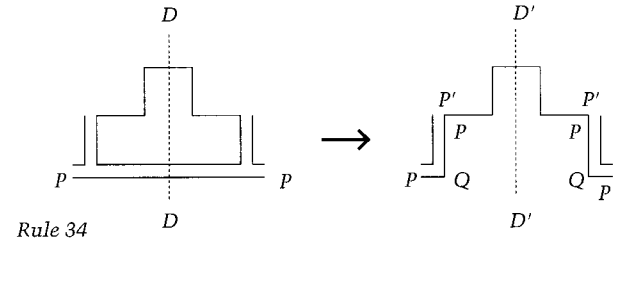
Rule 31



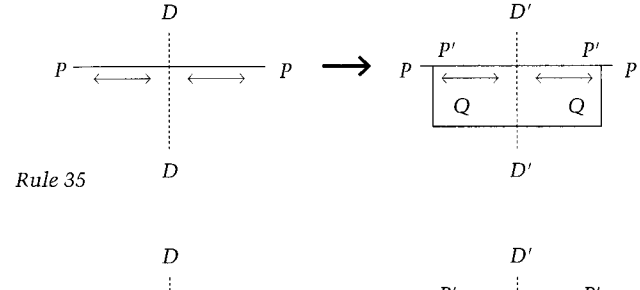
Rule 32



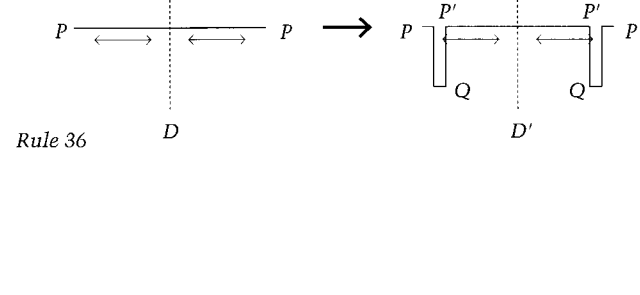
Rule 33



Rule 34

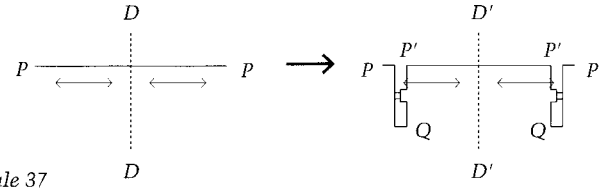


Rule 35

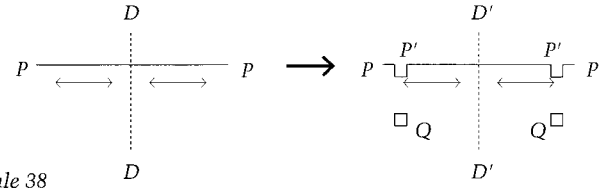


Rule 36

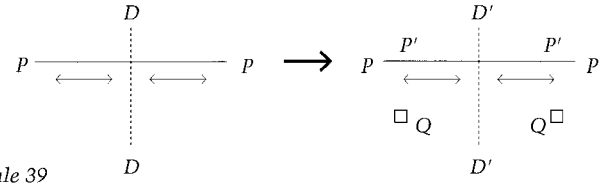
8.41 (cont.)



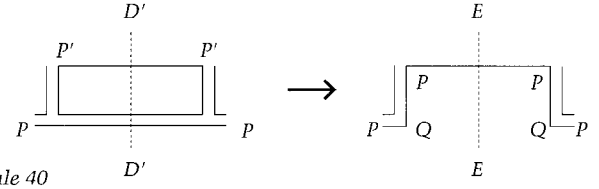
Rule 37



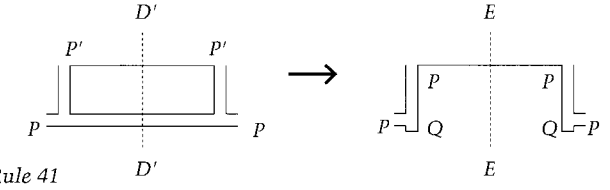
Rule 38



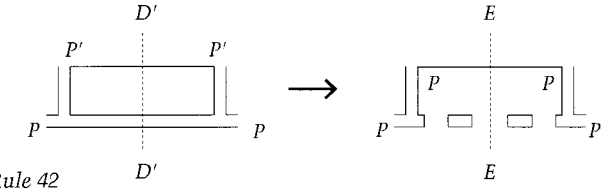
Rule 39



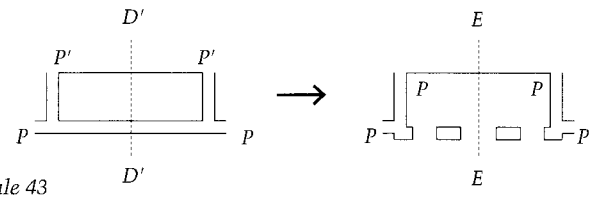
Rule 40



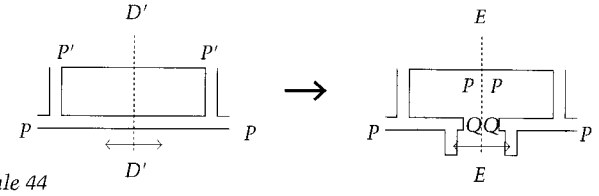
Rule 41



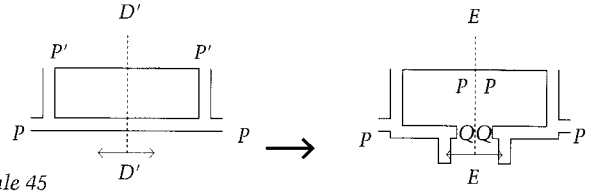
Rule 42



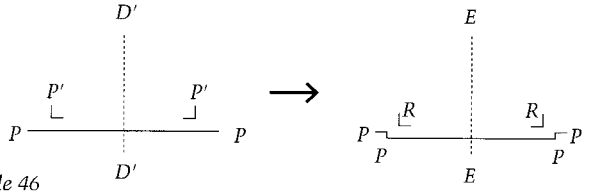
Rule 43



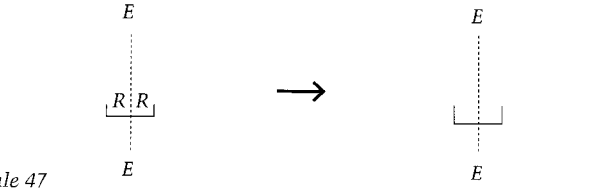
Rule 44



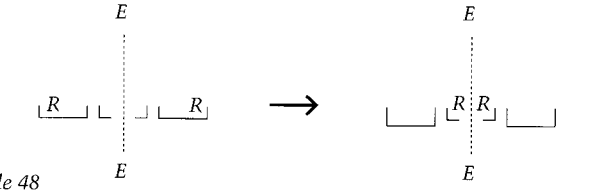
Rule 45



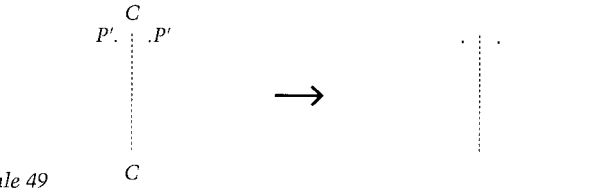
Rule 46



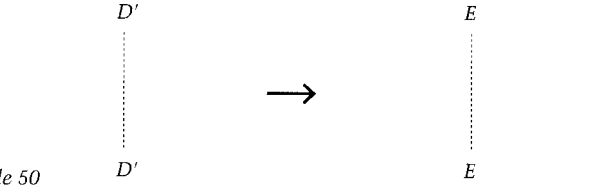
Rule 47



Rule 48

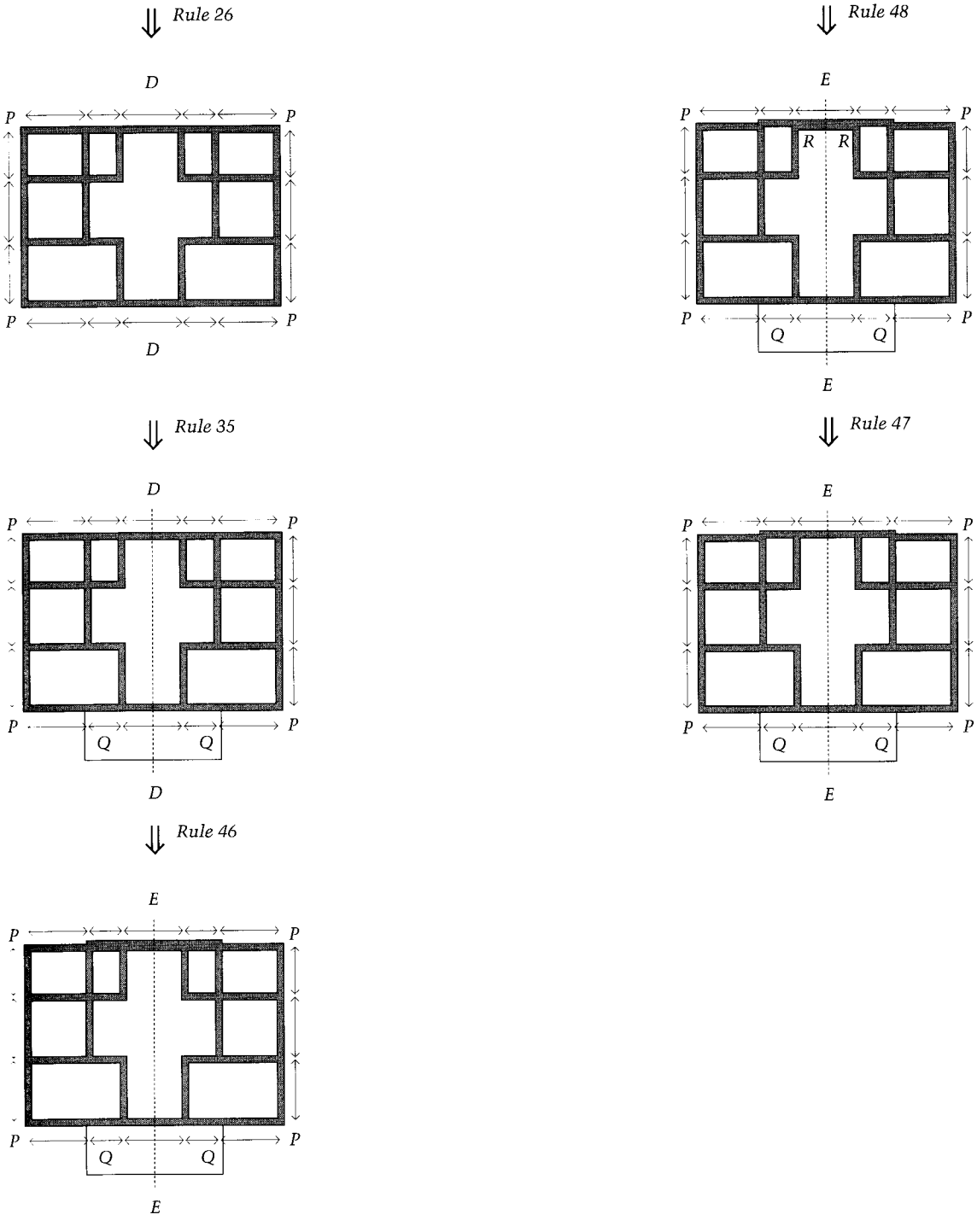


Rule 49



Rule 50

8.42
Addition of the front entrance and inflection of the back wall of the Villa Malcontenta



on the theme of portico columns are enumerated by the rules specified in figure 8.43, and figure 8.44 shows how the Villa Malcontenta columns are inserted.

The distinction that arises here between construction and enumeration of possibilities is an important one. We can encode Palladio's principles of room layout in a few simple rules, then construct layouts as required by recursive application of these rules: we do not need an exhaustive catalogue of Palladian room layouts. But it is hard to make useful generalizations about entrance treatments, so we must resort to enumerating the possibilities one by one.

By contrast, there *are* a few, strict, general principles for placement of doors and windows. These are expressed by the rules in figure 8.45, and application to the Villa Malcontenta is shown in figure 8.46.

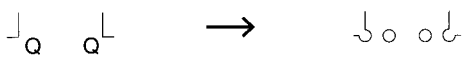
Finally, termination rules are specified in figure 8.47. For the most part these provide for erasing the construction lines and labels that were used to guide the plan generation process. When the termination rules are applied in the Villa Malcontenta derivation, the final plan shown in figure 8.48 is produced. At this stage, the mappings of shapes to construction world elements are completely unambiguous: shapes stand for rooms, walls, columns, doors, and windows. You could build from this drawing.

This grammar generates all the uniaxial[7] villa plans published in Palladio's *Four Books of Architecture* (figure 8.49), together with many plans sketched elsewhere by Palladio, and a rich catalogue of original plans in a convincing Palladian manner. Since the grid construction rules can be applied indefinitely, to generate grids of increasing size, this grammar specifies a countably infinite universe of villa designs for exploration. However, the number of room layouts possible within a grid of specified size is finite. Figure 8.50, for example, shows the twenty possibilities within a 3 by 3 grid. Notice that the Villa Angarano appears in this catalogue as number four.

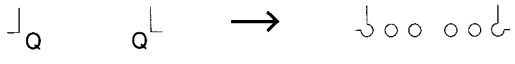
With a 5 by 3 grid the number of possible room layouts grows to 210. It is convenient to classify these according to the shapes of the central rooms. Figure 8.51 shows the complete catalogue of 119 5 by 3 layouts with rectangular central rooms. Five of the layouts from the *Four Books of Architecture* appear here: Villa Badoer, Villa Zeno, Villa Emo, Villa Ragona, and Villa Poiana.

There are only five 5 by 3 layouts with I-shaped central rooms, as shown in figure 8.52. The Villa Pisani (the only such layout to appear in the *Four Books of Architecture*) is the first of these. The other four are obvious variants on the Villa Pisani theme.

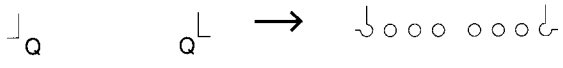
8.43
Lexical insertion rules providing for a variety of ways to add columns flanking the front entrance



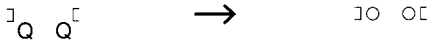
Rule 51



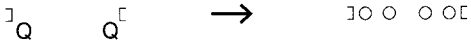
Rule 52



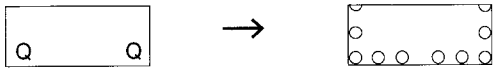
Rule 53



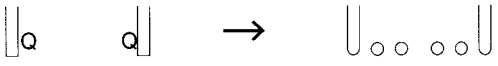
Rule 54



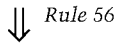
Rule 55

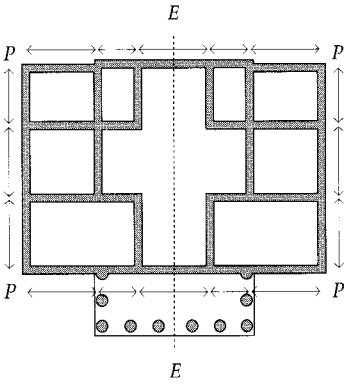


Rule 56



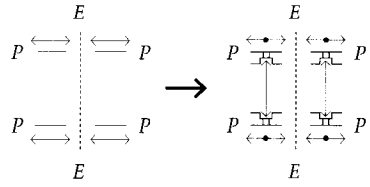
Rule 57



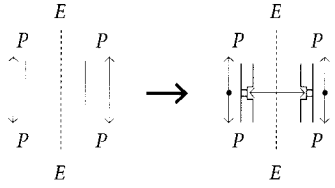


8.44
Addition of front columns to the Villa Malcontenta

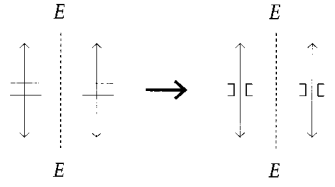
8.45
Rules for placement of doors and windows



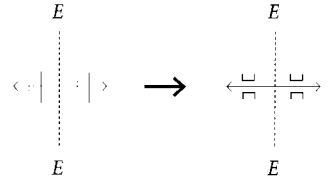
Rule 58



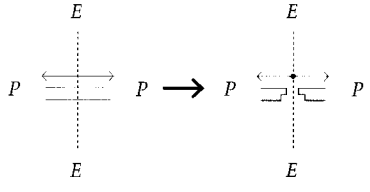
Rule 59



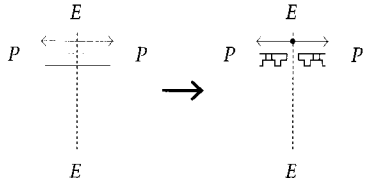
Rule 60



Rule 61

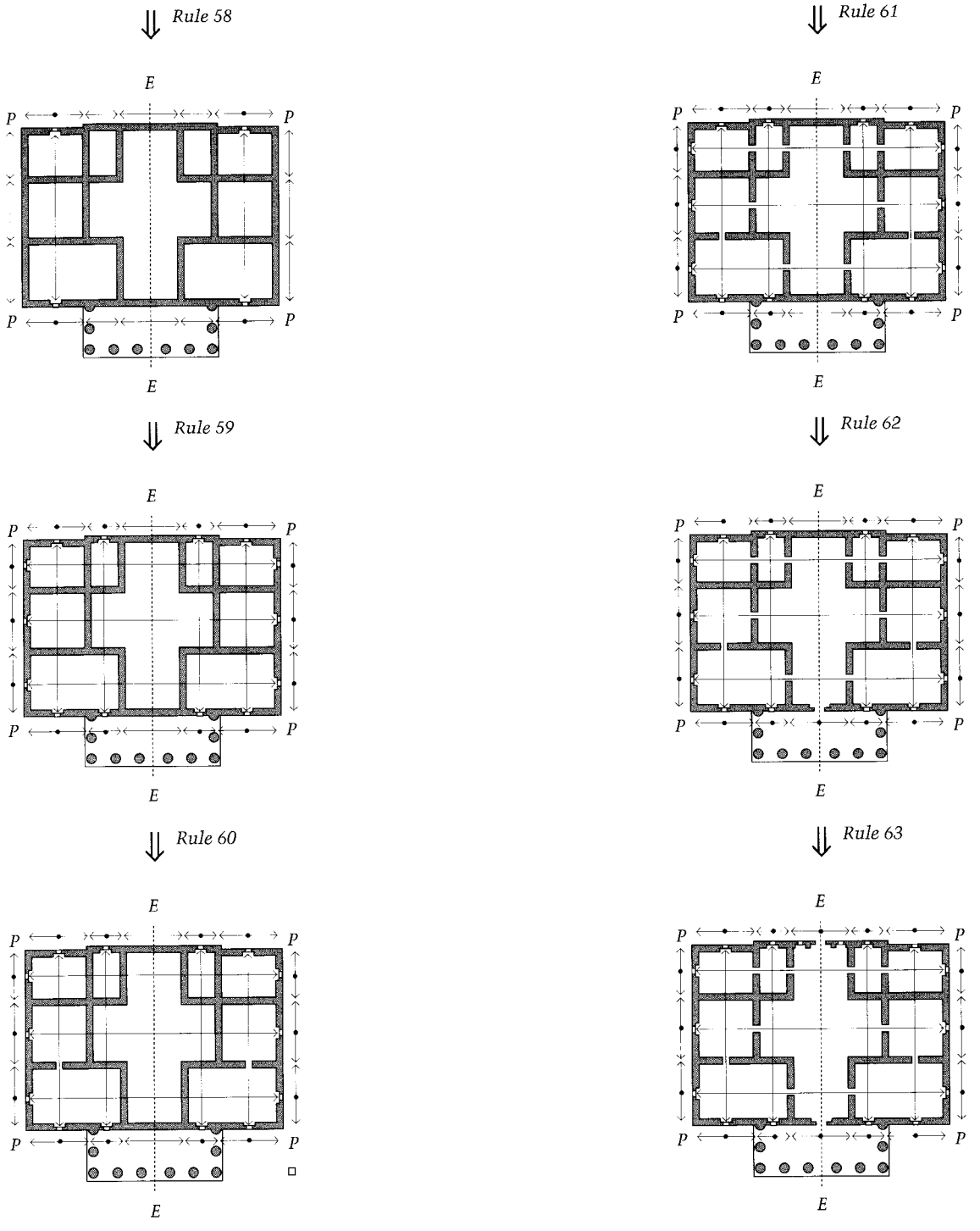


Rule 62

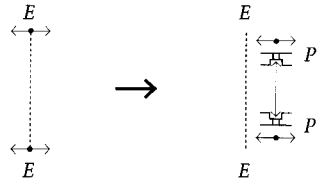


Rule 63

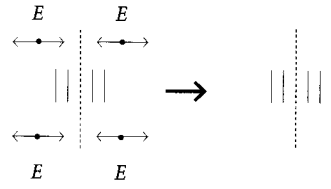
8.46
Derivation of the Villa Malcontenta's pattern of openings



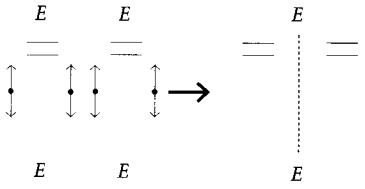
8.47
Termination rules



Rule 67



Rule 68

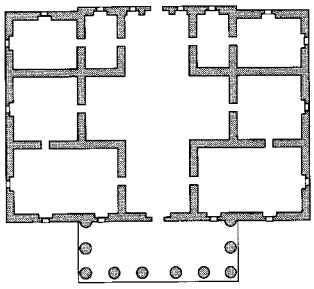


Rule 69

Rule 70

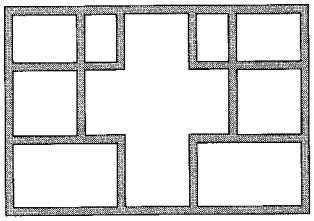
Rule 71

Rule 72

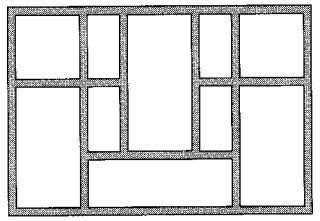


8.48
The final plan of the Villa Malcontenta, after application of termination rules to remove remaining construction lines and labels

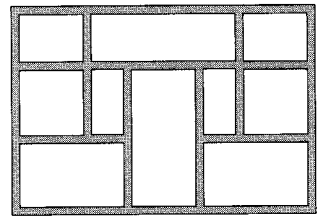
8.49
Schematic plans of the uniaxial villas published in Palladio's *Four Books of Architecture* and generated by the grammar



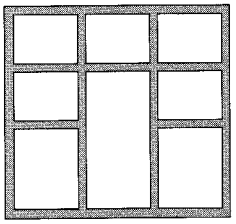
Villa Malcontenta



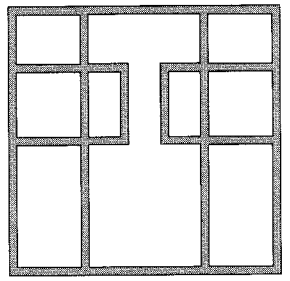
Villa Badoer



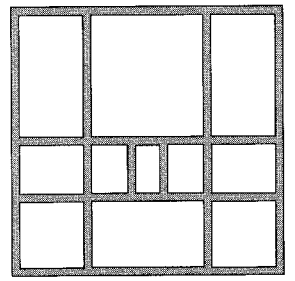
Villa Zeno



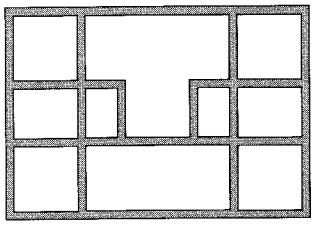
Villa Angarano



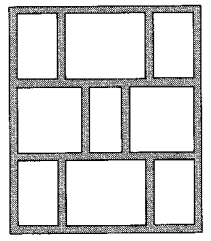
Villa Pisani



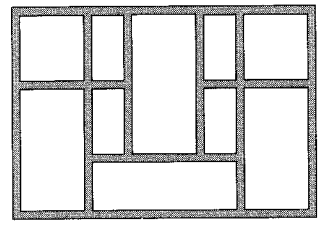
Villa Emo



Villa Sarraceno

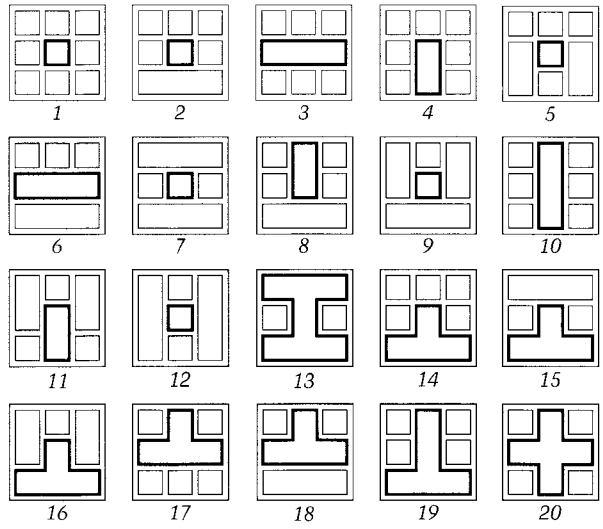


Villa Ragona

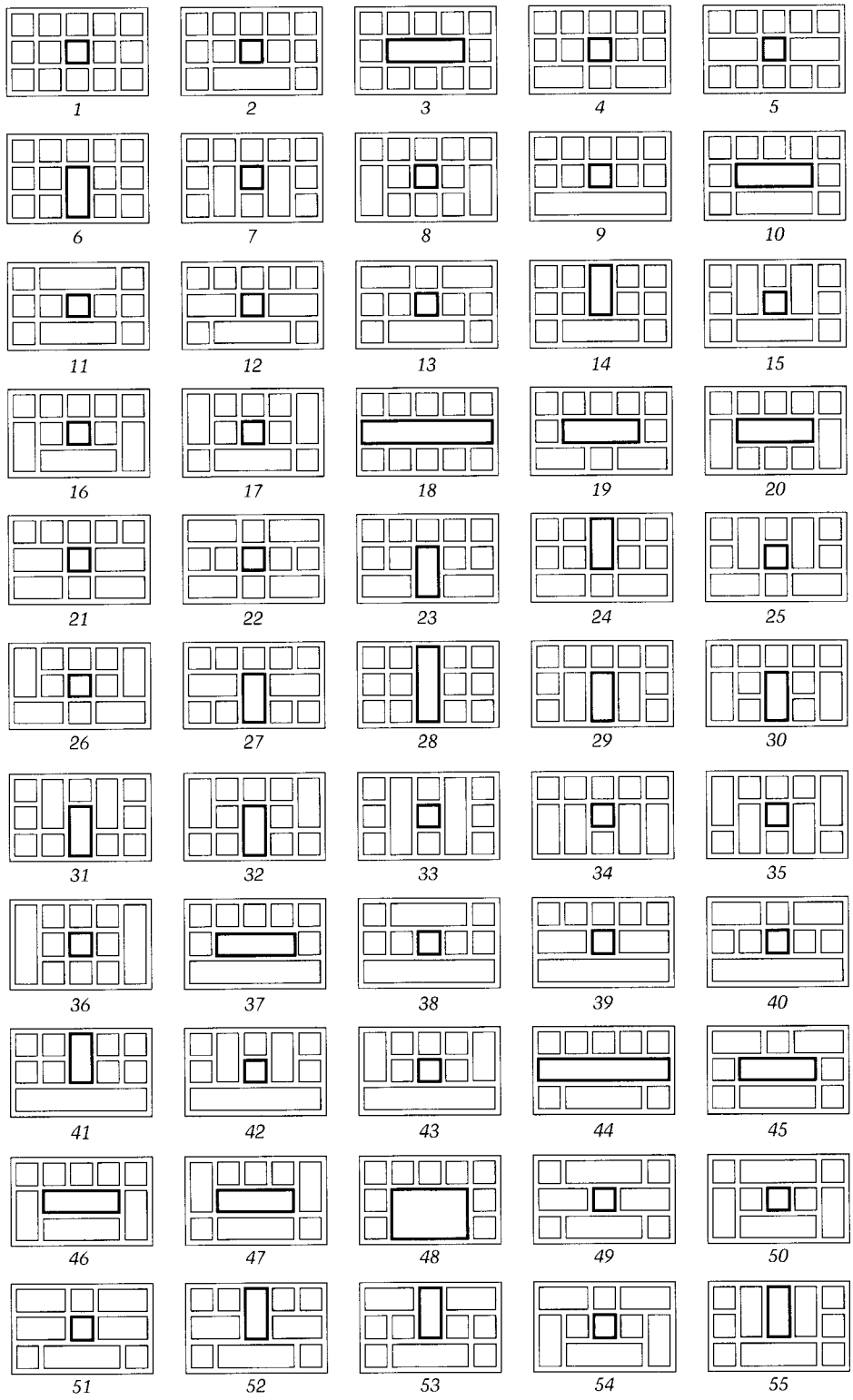


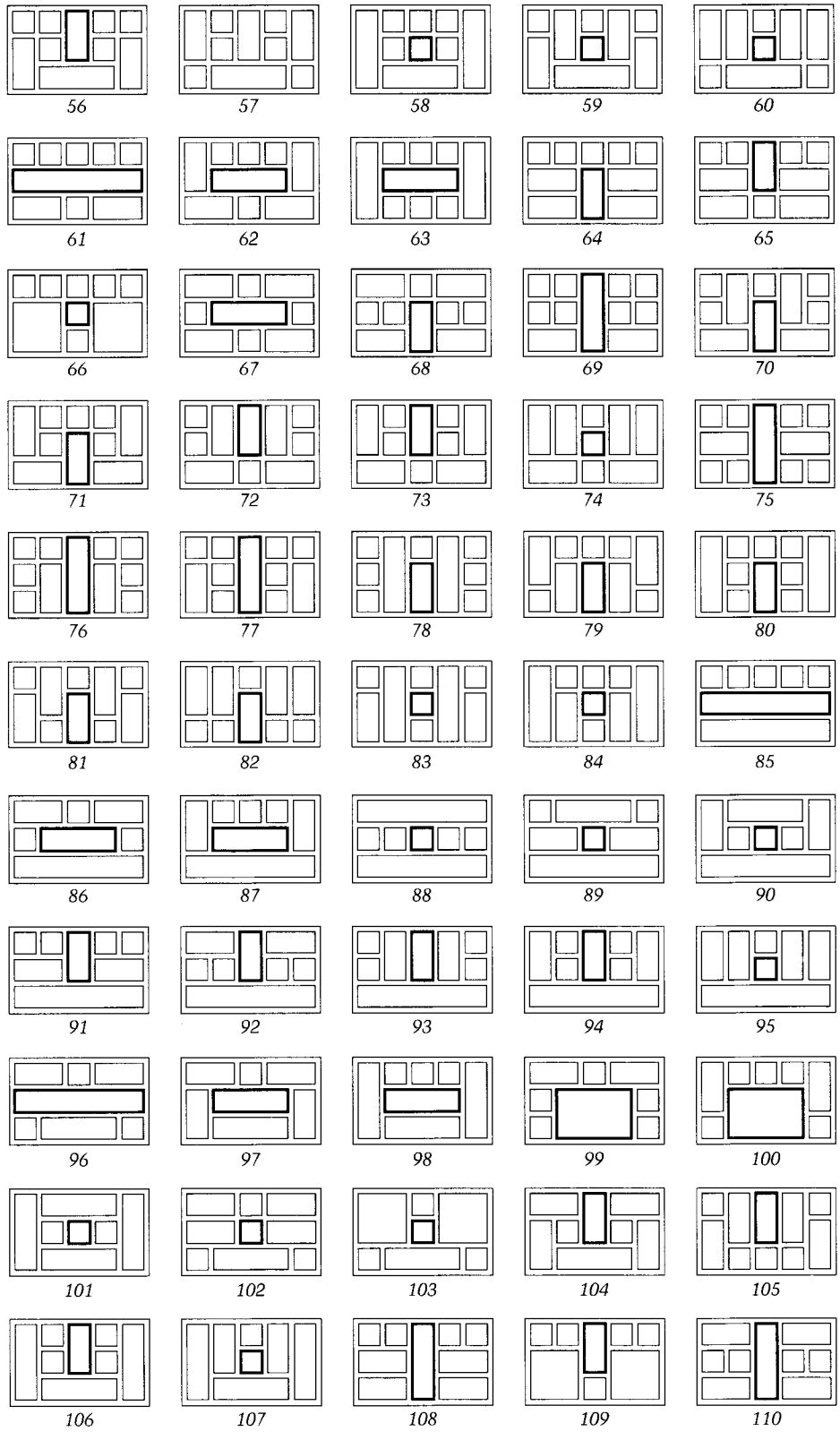
Villa Poiana

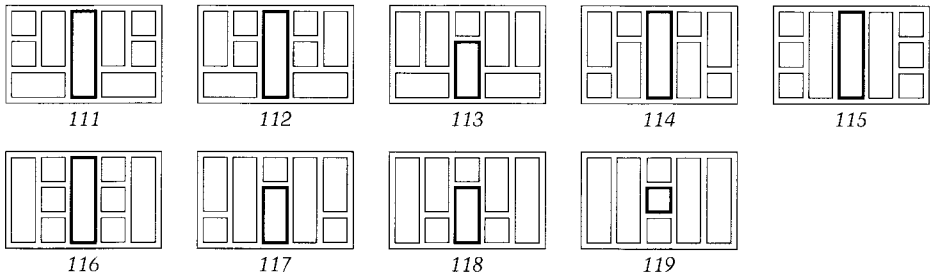
8.50
Complete enumeration of designs in the language generated by the Palladian grammar: all the 3 by 3 schematic plan layouts produced by rules 1 to 19



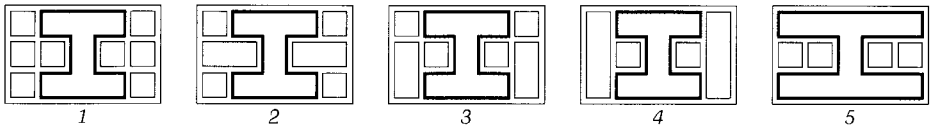
8.51
All the 5 by 3 schematic plan layouts with rectangular central rooms in the Palladian language




56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110



8.51 (cont.)



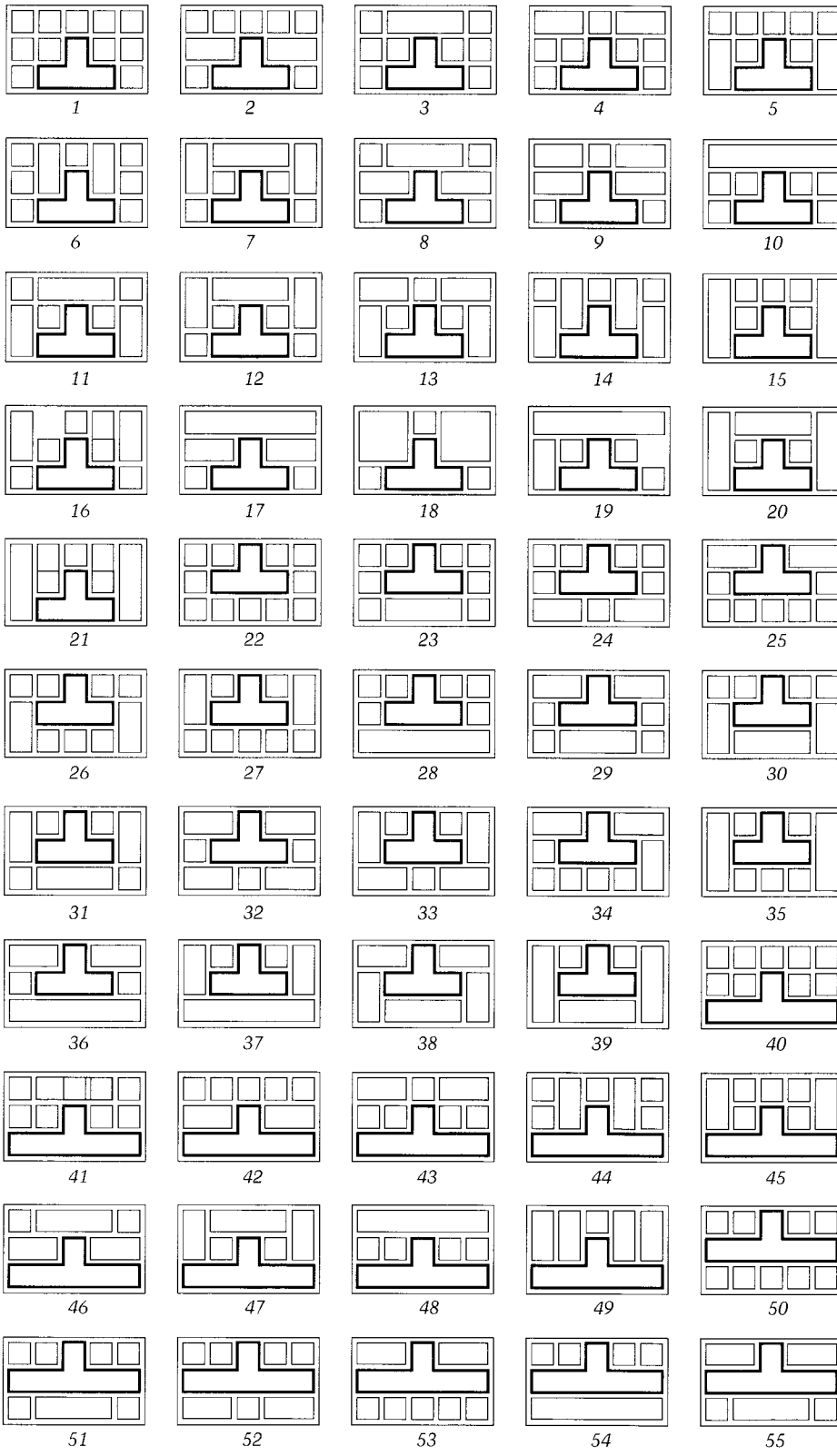
8.52
All the 5 by 3 schematic plan layouts with I-shaped central rooms in the Palladian language

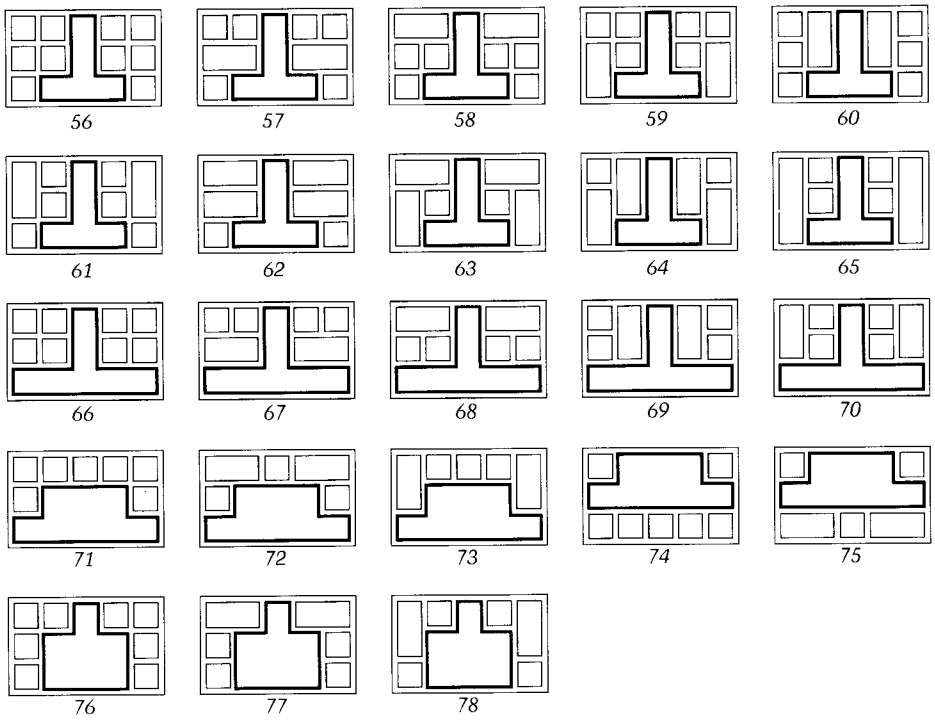
A T-shaped central room allows considerably more freedom in organization of the auxillary rooms than does an I-shaped central room. The catalogue shows 78 variants on the theme of the Villa Sarraceno (figure 8.53).

Finally, there are eight possible 5 by 3 layouts with +-shaped central rooms (figure 8.54). The Villa Malcontenta is number two, and there are five more straightforward variants on the theme of the Villa Malcontenta. The catalogue is completed by two layouts in which both arms of the cross completely span the plan.

As noted in chapter 2, Palladio thought that only a few ratios of small whole numbers were suitable for the proportions of room plans. If we assume that any room in a layout can be proportioned in any of these ratios (subject to the requirement of bilateral symmetry), and that we know at least one room dimension, the combinations of possible room proportions yield systems of simultaneous, linear, integer equations which we can attempt to solve for values of all the dimensioning parameters. In general, some of these systems will have positive solutions (the values that we seek) and some will not (meaning that the corresponding combination of room proportions cannot be realized in the given layout). Thus any layout can be developed into fully

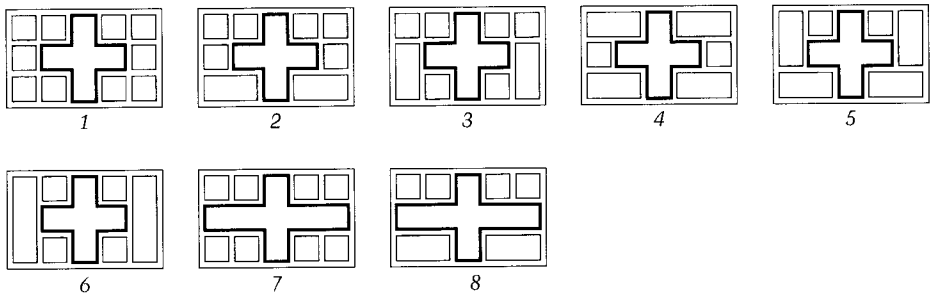
8.53
All the 5 by 3 schematic plan layouts with T-shaped central rooms in the Palladian language

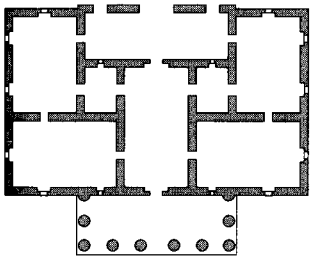




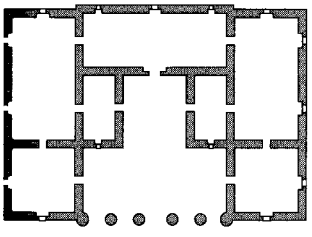
8.53 (cont.)

8.54
All the 5 by 3 schematic plan layouts with +-shaped central rooms in the Palladian language





Villa Hollywood



Villa Vine

8.55
Some original, fully developed "Palladian" villas generated by the grammar

dimensioned plans with correct Palladian proportions. Finally, any schematic plan can be detailed in numerous ways by application of rules governing selection and placement of doors, windows, and entrance features. Figure 8.55 shows some examples of original, fully developed villa plans in the language specified by the grammar.

In essence, the grammar concisely encodes knowledge of how to put together Palladian villa plans—the kind of knowledge traditionally acquired by perusal of the *Four Books* and through apprenticeship in the studio. If we consider the corresponding reductions, it also provides a way to recognize villa plans as Palladian—by successfully reducing them back to points labeled A.

DESIGN AS COMPUTATION

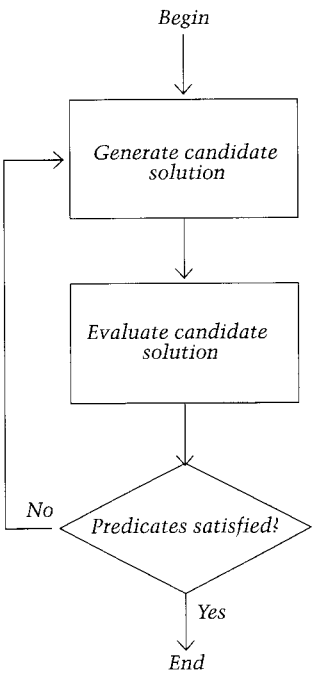
The view of design that has been developing through this discussion can now be made more precise. We have seen that a design world provides shape tokens which depict architectural elements. The rules of a grammar specify the ways in which these tokens may be manipulated. A design process is a computation of specified predicates according to these rules; that is, a sequence of operations on the shape tokens undertaken in an effort to satisfy predicates asserted about the design world.

To put this in another way, we can say that the syntactic rules governing a design world establish an architectural type (such as that known as the Palladian villa), and predicates expressed in a critical language establish the requirements of a particular moment and context.[8] The goal of the designer's computation is to instantiate the type in a way appropriate to that moment and context.

A basic structure is given to the design world by the initial vocabulary of shapes, the operators that can be applied to them, and the algebra that results. But knowledge of the type, as expressed in rules for recognition, refinement, and assembly of parts, provides much more structure.

The process of finding a solution to a design problem is a trial-and-error one of applying rules to generate candidate solutions, then computing predicates to determine whether candidate solutions are acceptable solutions.[9] The basic structure of this process is illustrated in figure 8.56. It is known, technically, as a generate-and-test process taking place in a search space.

A computational device to execute such a process needs a generation mechanism, a test mechanism, and a control strategy (rules for determining what alternative to try next).



8.56
The basic trial-and-error structure of a design process

Design intelligence may be located either in the generation mechanism or in the test mechanism. In the former case, the rules of the grammar assure with high probability (or perhaps even guarantee) that an acceptable solution will be quickly produced: the evaluation mechanism has little or nothing to do. In the latter case, the generation mechanism indiscriminately produces alternatives, and it is up to the evaluation mechanism to sort out the acceptable ones by bringing a knowledge base to bear, drawing inferences, and exploring the entailments of alternatives. In other words, you can get acceptable results by combining smart designers with dumb critics, or by teaming smart critics with dumb but energetic designers. You may prefer God (a smart designer with no need for a critic) or evolution (indiscriminate generation but deadly effective criticism).

Given a generation mechanism and a test mechanism, the efficiency with which a solution can be found will depend on the control strategy that is used. Depending on the structure of the problem, either depth-first or breadth-first search may prove to be the more efficient. And efficiency can usually be enhanced by choosing appropriate subgoals, developing more promising alternatives before less promising ones, and abandoning branches as soon as it becomes clear that they are unlikely to lead to a solution.

Different types of computational devices may be used to generate proposals, test them, and apply control strategies. For example, a computer can be used to generate alternatives by mechanically applying the rules of a shape grammar, with a human critic inspecting and testing the machine's proposals. Or a human designer might generate alternatives which are then evaluated by analysis programs or knowledge-based computer systems. In a completely manual design process all the generation and testing is performed by a human designer, or the tasks are divided among members of a design team. And in a fully automated design process the computer both generates and tests.

This formulation seems uncomfortably mechanistic and reductionist if we assume that a designer works in a closed world of fixed rules, limited knowledge, and predefined goals. This may, indeed, be the case in some specialized areas of design, but an architect usually works with a complex structure of facts, rules, and goals that can grow and change as a design process unfolds. The task is not just to solve a well-defined problem but to discover interesting possibilities and relate them, through critical discourse, to our knowledge and experience. Design elucidates *both* what we can have *and* what we want.

SUMMARY

As a foundation for designing, we need a theory of what types of shapes among the indefinitely many emergent subshapes in an evolving design should receive our attention. We have now seen that the left-hand sides of shape rules specify the shape types of interest, and that available recognition mechanisms (computer implemented, or in the human perceptual system) provide a way to find instances of these. We also need to know what to *do* with these shapes when we find them; the right-hand sides of shape rules establish potential directions for further development of the design, and the available operators (for instantiation, transformation, and combination of shapes) provide the means to carry out that development.

Ideally, the rules of an architectural grammar should provide a well-formed representation of a class of buildings in a construction world. In other words, the language specified by the grammar should include a depiction of every possible building in the class, and every design in the language should be for a possible building in that class. Of course there can be differences of opinion about the extension of a class of possible buildings, and hence about whether or not a grammar provides a well-formed representation. Some may think, for example, that the Palladian grammar specifies too many possibilities, and some may think that it specifies too few. It depends on what you understand by "Palladian villa."

I do not want to suggest that designers necessarily follow explicit grammatical rules (though they sometimes do). Nor do I want to speculate (as an orthodox, functionalist, cognitive scientist might) that a designer's mental states are states of an abstract computational device and that design is the mental derivation of shapes. The essential points are that design exploration is rarely indiscriminate trial-and-error but is more usually guided by the designer's knowledge of how to efficiently put various types of compositions together and that such knowledge can often be *made* explicit, in a concise and uniform format, by writing down shape rules.