Descriptive Conventions for Shape Grammars

Haldane Liew

MIT

Abstract

This paper introduces a new set of descriptive conventions for shape grammars, and illustrates how they can be used to address problems with user experience. The shape grammar formalism has been shown to be capable of generating designs such as Palladian villas, Prairie houses and Queen Anne houses. The formalism can describe the process to develop a design through the use of rules, symbols, and lines.

The user's experience in applying the rules is often neglected in the design of the grammars. This creates four problems: 1) the user is unaware of the implicit sequencing of rules, 2) the user can generate invalid design states, 3) the user is forced to apply technical rules that do not change the overall design, and 4) the user is only given a restricted set of design choices.

To address these problems, a new set of descriptive conventions has been developed that provides a layer of abstraction built on top of the formalism. These conventions are currently being implemented using the Visual LISP programming environment in AutoCAD. The program applies rules, which incorporate the use of the new conventions, to produce a design.

The conventions are based on two techniques. The first technique is an explicit control mechanism that determines the sequencing of rules based on the success or failure of a rule application. Because some design changes require more than one rule, this allows the grammar to chain a sequence of rules to create macros since. The second technique is a mechanism that demarcates an area of the drawing for query. With this technique, a rule is able to recognize void spaces in a drawing.

A comparison of the rules to construct a bi-laterally symmetrical grid in three grammars--Palladian, Yingzao Fashi, and Grid--will be used to demonstrate the advantages of the new conventions.

Introduction

Shape grammars are graphical production systems that provide a formal mechanism for generating compositions based on shapes and their spatial relationships by specifying methods to replace parts of shapes with others. A shape is composed of a finite collection of labeled or unlabeled points, lines, planes, areas, or solids (Stiny 1976). A rule in shape grammars can be written in the form A _ B where A and B are shapes. When this rule A _ B is applied, an instance of shape A is replaced with shape B. There also are parametric shape grammars, which means the shapes have parameters that can be adjusted (Stiny 1980, 1990). These kinds of shapes are called schemata, and they make it possible to define a class of shapes such as rectangles.

Researchers have been using the shape grammar formalism to define languages of design. The typical process for constructing such a grammar is to analyze a body of design work, such as Palladian villas, and then produce a shape grammar that will reproduce that body of work. If the grammar can duplicate the original designs in the analysis, then the grammar is considered to be a formal description of that language of design.

There are many examples of these types of shape grammars. Some architectural examples include Palladian villas (Stiny and Mitchell 1978), Prairie houses (Koning and Eizenberg 1981), and Queen Anne houses (Flemming 1987). The use of the formalism is not limited to architecture. Other examples include Hepplewhite chairs (Knight 1980), Mughal gardens (Stiny and Mitchell 1980), and coffee makers (Agarwal and Cagan 1998). Once the grammar is defined, it also can be used to generate new designs that are still within the language of the original body of work.

When designing a grammar, the experience of using the grammar is seldom addressed. Is it clear which rule is applicable? Can the user freely experiment with other rule sequences without worrying about getting stuck? Do the rules modify the design or just the technical mechanics of the formalism? Are the design choices presented to the user adequate? There are four general problems that occur when a grammar does not take into consideration the user's experience.

The first problem is an implied sequence of rules. To design with shape grammars, the user is presented with sets of rules. Within each set, the user decides which rule to apply. Sometimes the rules are meant to be applied in a particular order. But because the formalism can only sequence this series of rules implicitly, it is possible for the user to apply the rules in the wrong order. The result can be an incorrect design or an invalid design state, which is the second problem.

The second problem is the possibility for the user to create an invalid design state. An invalid design state is a situation in which the user is unable to proceed with the rest of the grammar because none of the rules are applicable. Typically the only means of recovering from an invalid design state is to start over from the beginning of the grammar.

The third problem is the repetition of non-design oriented rules. One method of avoiding invalid design states is to use technical mechanisms such as markers and labels to ensure the design is valid. When these markers need to be changed and modified, it is up to the user to apply these rules. This requires the user to repetitively apply rules that often do not affect the design but only the technical mechanisms. Such tasks should not be part of the user's experience in generating a design because these rules are only concerned with technical issues and not design issues.

The fourth problem is the scarcity of design options. Most grammars are written to generate designs in a concise manner. Oftentimes, the grammar does not include rules that can undo the effects of other rules. Once a rule is applied, it is permanent. Should the effects of the rule be undesirable, the user is forced to start over. It is important to allow the user to see the design in different ways to be able to carry on a conversation with the drawing (Schon, 1984). The user should be given options to permit modifying the design to learn whther the new design states are appropriate.

The remainder of the paper will compare the rules for creating a bi-laterally symmetrical grid from three grammars: Palladian villa (Stiny and Mitchell 1978), Yingzao Fashi (Li, 2001), and Grid. The first two grammars will illustrate the four problems with grammar design, and the Grid grammar will show how the new conventions address those issues.

Palladian Grammar

The Palladian Grammar is a parametric shape grammar that generates floor plans of Palladian villas. It was one of the first attempts to formalize a body of design work using shape grammars. The grammar generates the plans in 8 stages. The first stage of the grammar determines the overall size of a bi-laterally symmetrical grid that forms the basic skeleton for the plan. The second stage of the grammar combines the grids in specific ways to create the rooms. This paper is only concerned with the first stage of the grammar, which is the grid definition.

The rules for developing the bi-laterally symmetrical grid in the Palladian grammar are show in Figure 1. The rules can be organized into 4 basic groups. Rule 1 creates the initial shape. Rules 2-5 deal with the expansion and termination of the overall width of the grid. Rules 6-7 control the expansion and termination of the overall height of the grid. And rules 8-10 fill in the corners of the grid.

You will notice that some of the rules, such as rules 2-3, 4-5, 6-7 and 8-10 have the same antecedents. If more than one rule that applies, how does the user of the grammar know which rule to apply? It is up to the user of the grammar to determine which rule to use. The similarity of some of the rules may cause confusion for the user making it difficult to determine which rule is appropriate. The task at hand

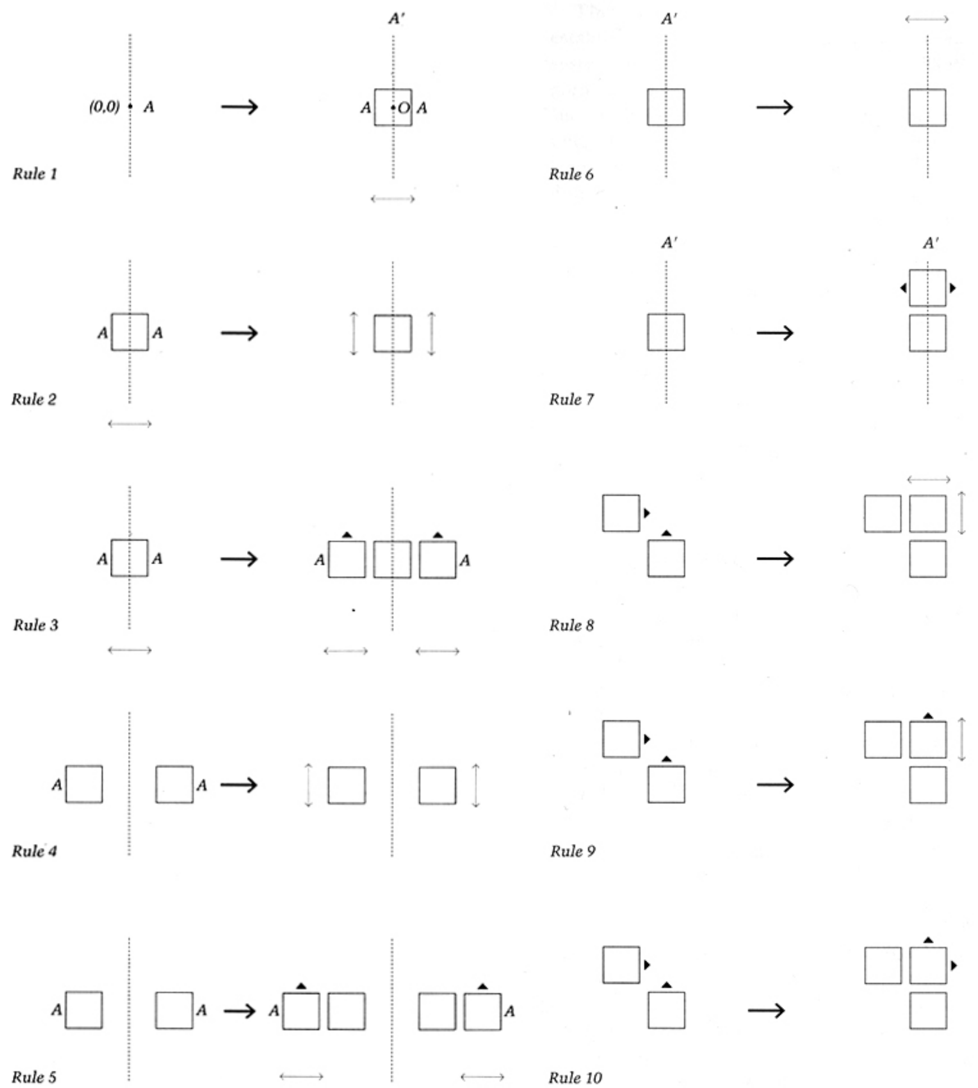

Figure 1. The rules from stage 1 of the Palladian Grammar for making a bi-laterally symmetrical grid.

here is to simply create a grid. The process to do that should be straight forward and simple. The rules should allow the user to succinctly determine the size of the grid.

Given the Palladian rules, however, it is not clear what sequence will produce a valid grid pattern. What sequence should be applied to achieve a 3x5 grid? When is it appropriate to terminate the grid if the desired grid size is 3x5? Which rules terminate the grid? The grammar assumes that the user will make the correct choice; it assumes that the user understands the implicit ordering of the rules.

To help the user make the correct choices, a sample derivation, given to complement the rules, shows one possible correct sequence to create the grid (Figure 2). Since the rules do not make explicit what rule should be applied next, the user is free to apply whichever rule is applicable. For instance, the user could apply the rules in a different sequence, such as in Figure 3. As the derivation shows, instead of applying rule 4 at the fourth step, rule 7 is applied. Instead of applying rule 8 to fill the corners, rule 10 and rule 9 are applied to fill in the corners. Although the application of the rules for making a grid in the Palladian grammar is perfectly legal, the result is an invalid design state.

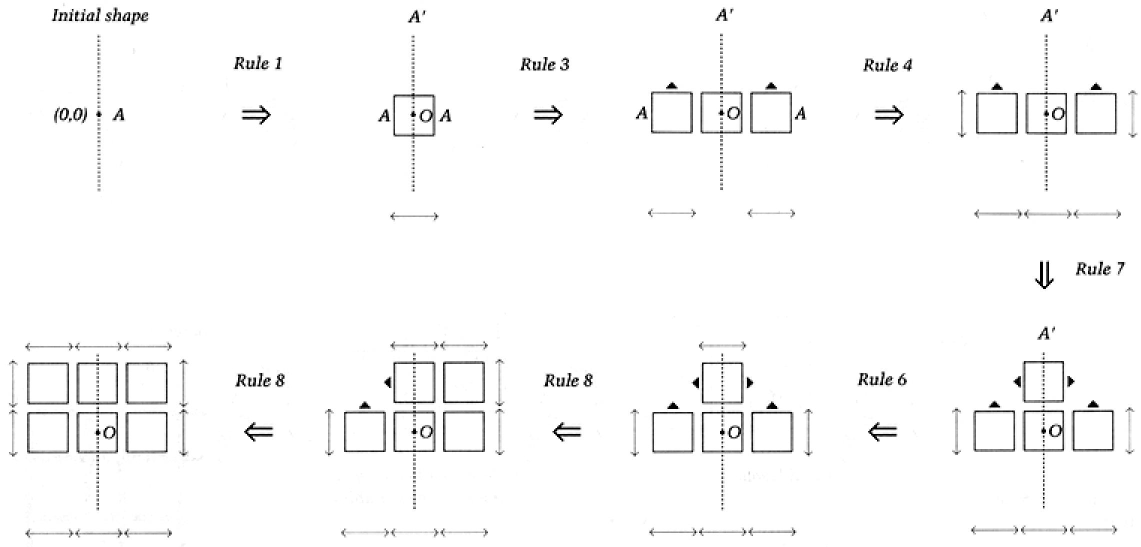

Figure 2. Correct derivation of the Palladian Grammar.

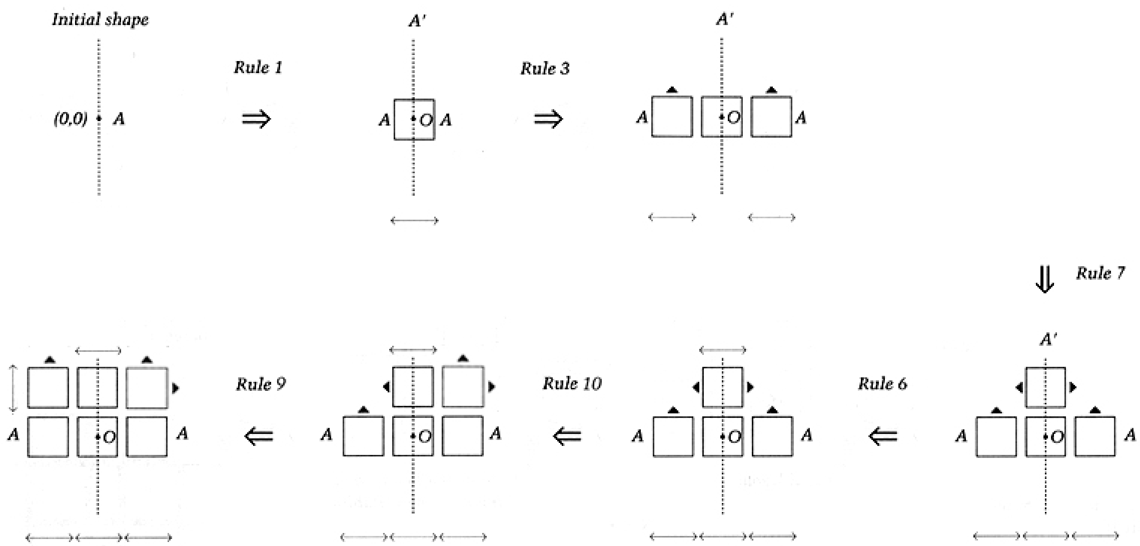

Figure 3. Legally correct derivation but with incorrect results.

Unfortunately this design state does not help us create a Palladian Villa. In fact, the user is stuck in this invalid design state unless he starts over from the beginning because the grammar has no rules to allow the user to undo the mistakes that were made at the outset. It is not helpful to give the user so many options, including options to develop invalid design states. The rules for creating a bi-laterally symmetrical grid should simply and straightforwardly show how to determine the size of the grid. It should not give users the ability to explore other design states that are not fruitful for the design. Or if it does, it should at least give them the option of undoing what they did before.

Through the analysis of the Palladian grammar, 3 problems arise: 1) The grammar has no explicit sequencing of rules. 2) It is possible to have invalid design states. 3) There is no ability to go back and undo other rule applications. Some of these problems can be resolved by re-designing the rules. One such redesign results in the Yingzao Fashi grammar.

Yingzao Fashi Grammar

The Yingzao Fashi (building standards) is a Chinese building manual written by Li Jie (d. 1110) and published in 1103. The grammar is a formalization of the architectural style described in the manual. There are 7 stages and 9 descriptions used to develop a design. This paper is only concerned with the first stage of the grammar which, is the plan diagram.

The plan diagram stage of the Yingzao Fashi grammar creates a bi-laterally symmetrical grid (Figure 4). Rule A1 creates the initial shape. Rule A2 increases the width of the grid. Rule A3 increases the height of the grid. Rule A4 fills in the corners or the grid. Rule A5 terminates the width of the grid. Rule A6 terminates the height of the grid. Rule A7 changes the termination symbol so that rule A8 can apply. Rule A8 changes the state of the drawing to go to the next stage.

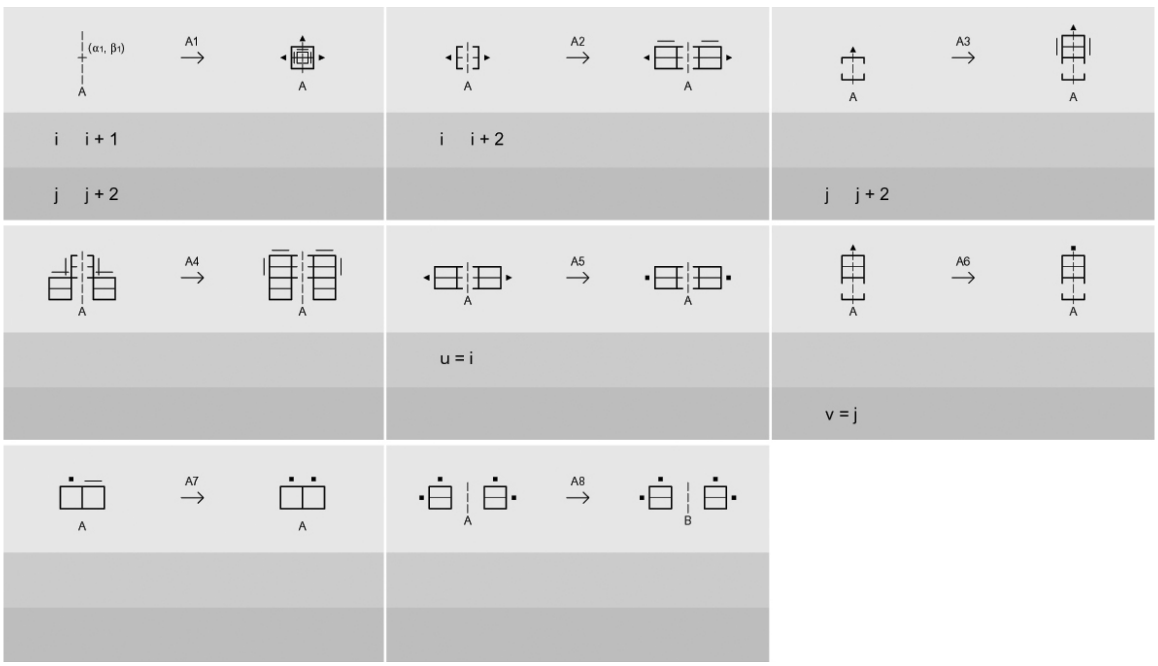

Figure 4. Yingzao Fashi rules for making a bi-laterally symmetrical grid.

There are noticeable differences between the Yingzao Fashi grammar and the Palladian grammar. The antecedents for each rule in this stage of the Yingzao Fashi grammar are unique; therefore each rule will apply in only one situation. This helps make explicit to the user which rule is applicable.

There are also fewer rules, two less to be exact, than the Palladian grid rules. Fewer rules make it easier for the user to understand the effects of the rule on the drawing. It also reduces the cognitive load on working memory (Miller 1956) so that the user can concentrate on the design.

The derivation for the grammar shows the steps required to make a 3x5 grid (Figures 5-8). (The grammar actually creates a 6x5 grid, but because I am not counting the horizontal sub-divisions inside the grid, it is effectively 3x5. Either way, it does not affect my analysis of the grammar.) Steps 1-6 (Figure 5) establish the overall size of the grid. By step 6, the derivation has made a framework that is 5 units wide and 3 units high. All that needs to be done is to fill in the remaining boxes to complete the grid.

The filling-in of the boxes begins with step 7 (Figure 6). By step 10 the entire grid is visually complete. Because the grid is complete, one would assume the next move is to go to the next stage. But that is not possible. In order to apply rule A8, which is the rule to move to the next stage, the markers at the corners of the grid have to change from two lines to two square symbols. Luckily the only rule that applies at this point is rule A7.

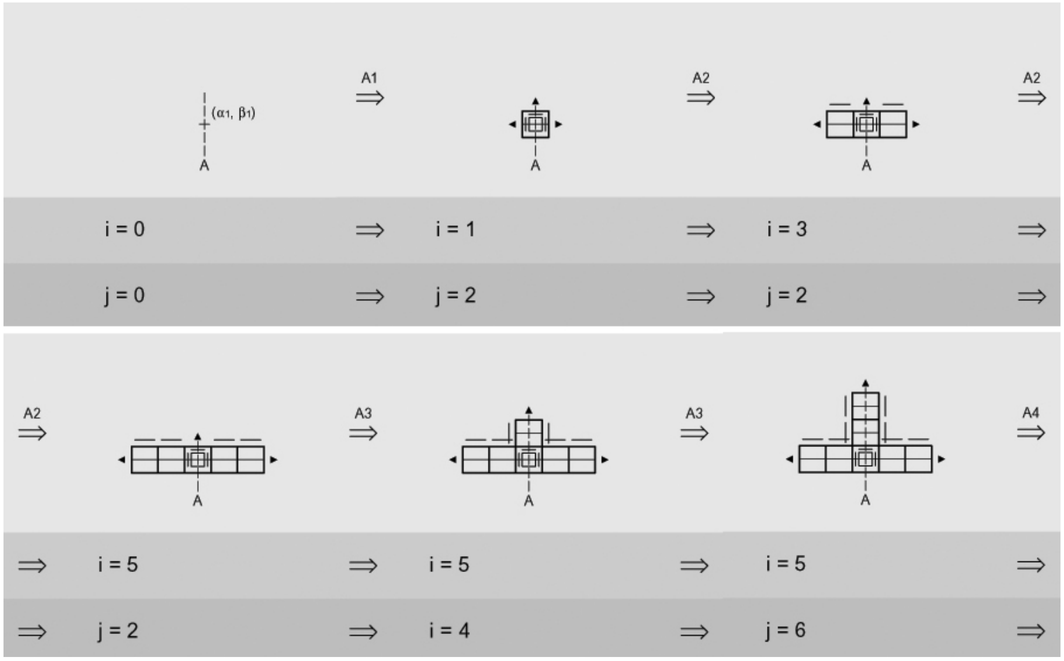

Figure 5. Derivation of the Yingzao Fashi Grammar, Steps 1-6.

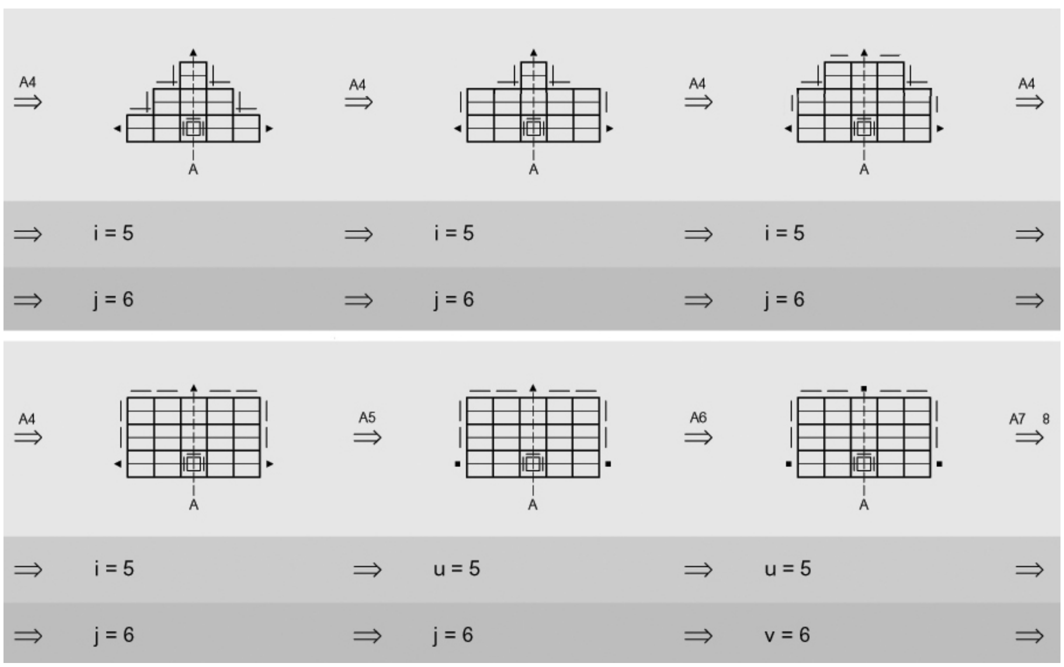

Figure 6. Derivation of the Yingzao Fashi Grammar, Steps 7-12.

The next 8 steps (Figures 7-8) all apply one rule--rule A7. The user has to manually select each rule and apply it until, finally at step 20, he can apply rule A8 and move on to the next stage of the grammar.

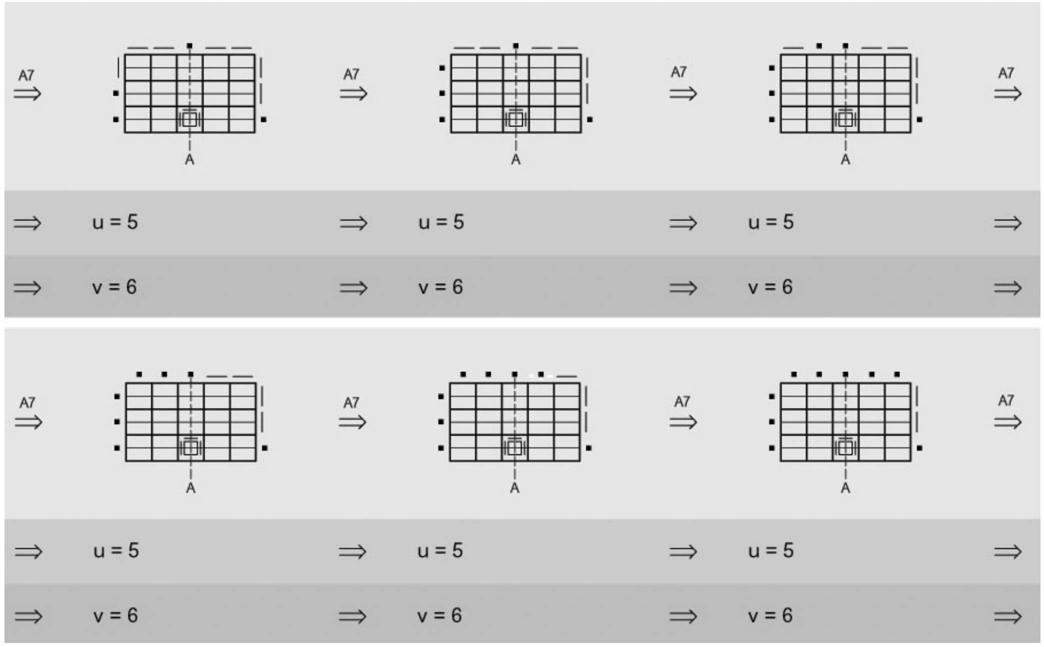

Figure 7. Derivation of the Yingzao Fashi Grammar, Steps 13-18.

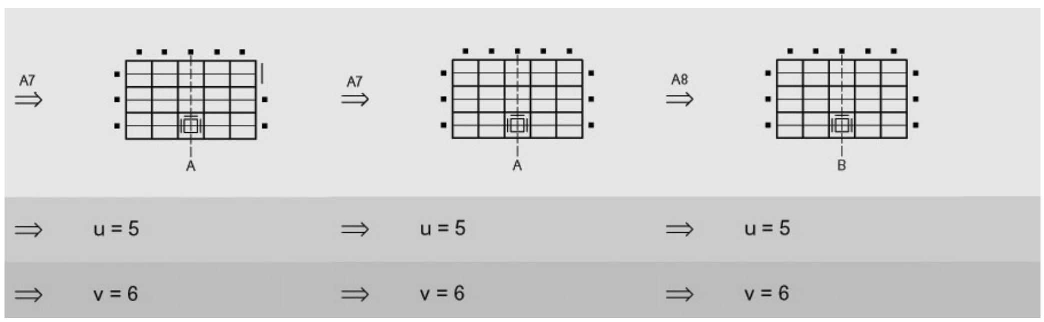

Figure 8. Derivation of the Yingzao Fashi Grammar, Steps 19-21.

The Yingzao Fashi grammar has addressed some of the deficiencies of the Palladian grammar. It has fewer and clearer rules. Each rule has a unique antecedent, which makes it clear which rule is applicable. The symbols used in the rules show the meaning of the rules: an arrow means the grid can still expand and a block means the grid is terminated. The most important improvement is that the grammar can no longer permit the user to produce an invalid design state. Each rule always leaves the drawing in a valid design state.

The other two issues, explicit rule sequencing and the ability to backtrack, have not been addressed. The sequencing of the rules is still implied. Even in situations where only one rule can be applied, the user still has to guess which rule is applicable. Also, the user is still not able to change his mind midway through the design of the grid. Should the user decide that the grid is too big, the only way to make it smaller is to start over, because the rules only allow for the expansion and not the contraction of the grid size.

The Yingzao Fashi grammar has introduced a new problem, which is the necessity for the repetitive application of a rule. In the example derivation, the user was required to apply rule A7 eight times in order to transform the drawing into a state in which rule A8 could be applied. This process was necessary to ensure that the grid was not ill-formed. The only effect rule A7 had on the drawing was to change the markers from lines to squares. This is only a technical change; the rule did not modify the overall grid.

Such mundane technical issues should not be part of the user's experience because the changes made do not affect the design. The user of a grammar is concerned with design decisions, not technical issues. The Grid grammar demonstrates how the new conventions can be used to automatically resolve technical issues so that the user can be relieved of such routine tasks.

New Descriptive Conventions

This paper addresses the four problems raised by the Palladian grammar and Yingzao Fashi grammar, by introducing two new descriptive conventions: the directive and the predicate. The directive convention is a control mechanism that makes explicit which rule to apply next. It is a simple technique that has some powerful consequences. I call this convention the *Directive* because it directs which rule, if any, should be applied next. The directive is composed of two parts: true-rule and false-rule (Figure 9). If the rule can be applied in the drawing then the subsequent rule to be applied is the true-rule. If the rule cannot be applied in the drawing then the subsequent rule will be the false-rule. The true-rule and false-rule can also be a null rule, which is a rule that does nothing.

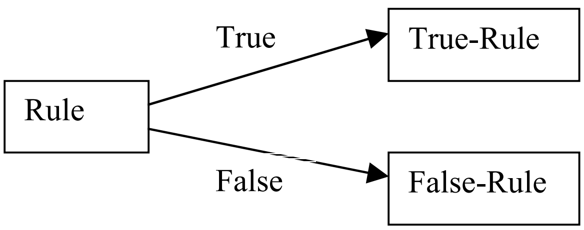

Figure 9. A diagram of the directives convention.

The most apparent use of this convention is to chain a series of rules together. A less obvious, but powerful, use of the directive is to affect changes in a drawing recursively. For instance, to change a series of squares into circles, as shown in Figure 10, without the directive, the user would have to select and apply the same rule three times. But with the use of directives, instead of repetitively applying the same rule over and over again, the user would only need to apply the rule once. And because of its recursive nature, R01 would automatically reapply itself until it can no longer find a square.

The second convention, the predicate, demarcates an area of the drawing for query. This technique allows the user to specify a portion of the drawing on which a function is applied to the shapes in that area. I call this technique the *Predicate* because it is another condition the antecedent schema of a rule has to satisfy in order to find a match. To apply the rule A_B, shape A must be part of the drawing. If shape A includes the use of the predicate convention, then in addition to finding shape A in the drawing, the predicate must also be true. If shape A is part of the drawing but the predicate is false, then the rule cannot apply.

The function used to test the predicate can be any Boolean function. For example, one possible function is to test for whether the demarcated area has any square shapes. Another function could test for whether there are any vertical lines in the specified area. But, for me, the most useful function is the void function. This function allows the rule to determine whether the demarcated area is an empty shape.

An example of the predicate using the void function is shown in Figures 11 and 12, in which two rules (R02 and R03) are compared. R02 finds any rectangle of width w and height h and adds a cross (lines joined diagonally opposed corners) in the middle of each one. R03 is the same as R02 except that the center of the rectangle is filled with a void predicate (marked by dashed lines). In other words, rule R03 will find only rectangles that have nothing inside and then add a cross within it. The predicate allows the rule to take into consideration the context of the found shapes. This is different from R02, which is only concerned with finding a rectangle regardless of context. The use of a predicate produces very different results. A parallel application of R02 to the drawing in the middle of Figure 11 produces 9 crosses, whereas a parallel application of R03 produces only 4 crosses.

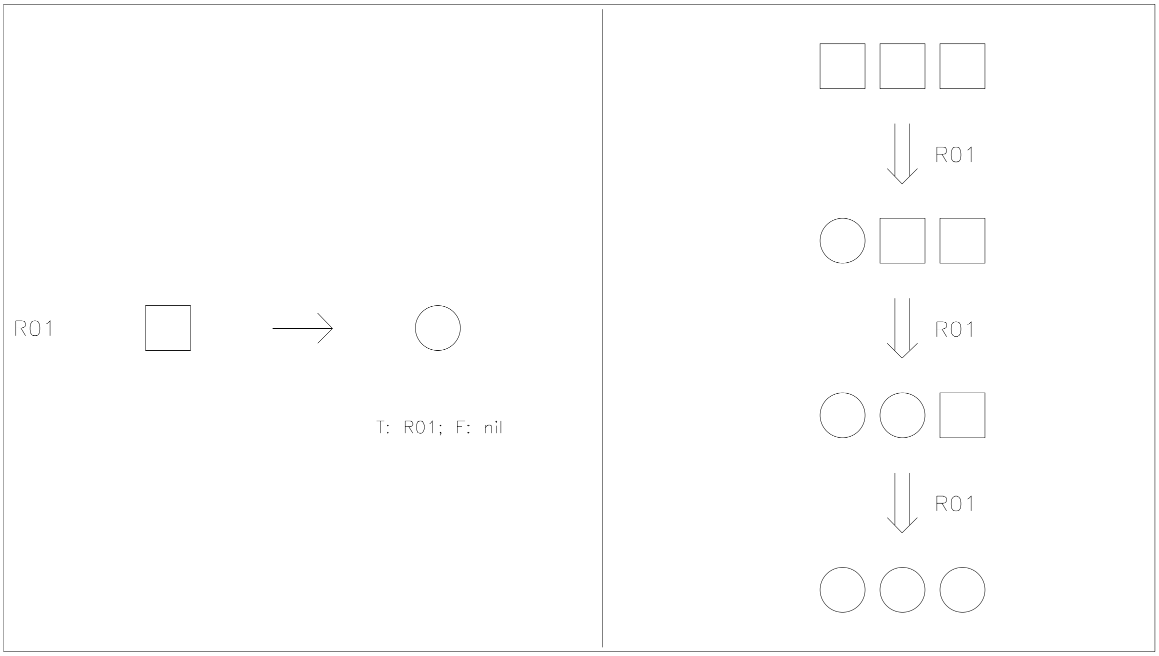

Figure 10. An example of recursive rule application using directives.

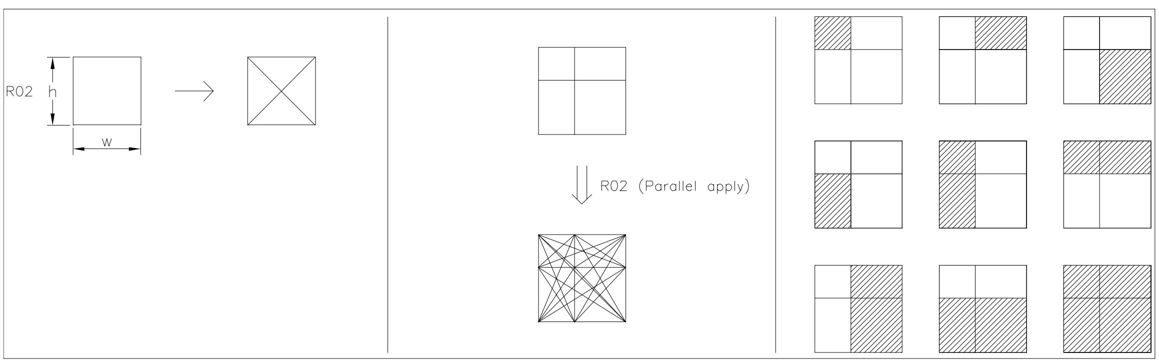

Figure 11. The parallel application of rule R02 produces 9 crosses. All the possible rectangles are enumerated on the right.

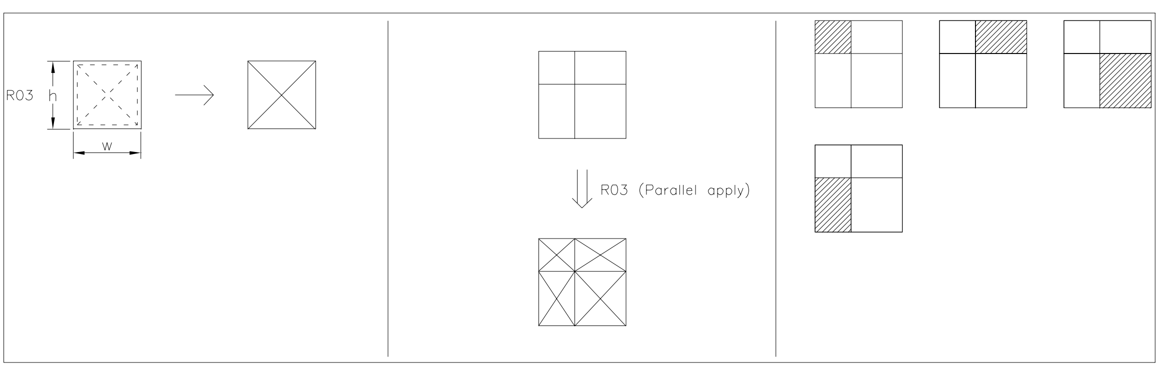

Figure 12. An example of predicates in a rule (indicated in dashed lines). The parallel application of rule R03, which uses a predicate, produces only 4 crosses. All the possible rectangles are enumerated on the right.

With these two new conventions, the designer of the grammar can specify a sequence of rules and can also detect void spaces in a drawing. The advantage of these conventions is that they can be used to address the issues raised from the analysis of the Palladian and Yingzao Fashi grammars. The Grid grammar illustrates how the conventions can be used to resolve the problems of 1) not being able to explicitly define a sequence of rules, 2) producing invalid design states, 3) applying rules that are only technically oriented, and 4) a confining set of rules which do not allow the user to backtrack.

Grid Grammar

Similar to the Palladian and Yingzao Fashi grammars, the Grid grammar generates a bi-laterally symmetrical grid. The Grid grammar, however, incorporates the use of the new conventions. The rules for the grammar are presented in Figure 13. Unlike the other grammars, the Grid grammar makes use of the directive convention to define macros or sequence of rules. Each horizontal row, such as rule R01 and R06, defines one macro in this grammar. R06 is repeated twice in the figure to clearly show the four macro rules. R05 is the only rule that does not make use of the directive convention.

The actual number of rules the user can apply is five (rules 1-5), which is fewer than both the Palladian grammar (10 rules) and the Yingzao Fashi grammar (8 rules) uses. Rules 6-8 are recursive rules that are linked with one of the first four rules through the directive.

The rules can be divided into three groups. The first group, R01 and R02, adjusts the overall height of the grid. R01 increases the height, and R02 decreases the height. The second group, R03 and R04, adjusts the overall width of the grid. R03 increases the width, and R04 decreases the width. The third group consists of only R05 which terminates this stage of the design. This set of rules gives the user the freedom to enlarge or reduce the size of the grid as he sees fit. The design of these rules also addresses the issue of giving the user the ability to go back and change the design.

Rules 6-8, on the right side of Figure 13, are linked with specific rules on the left side to define a macro. For example, R01 is linked with R06. R01 increases the height of the grid. After an increase in the grid height, the grammar ensures that all the other grid columns are increased as well by applying the recursive rule R06. R03 also uses R06 to define a macro. R02 and R07 define a macro, as does R04 and R08. Through the use of the directive convention, the correct sequencing of the rule is made explicit. For instance, after applying Rule R04, the user must apply rule R08. This sequence of rules is not left up to the user, but is embedded in the rules.

Rule 06 makes use of the predicate convention to fill in the corners of the grid. The antecedent schema of the rule looks for 3 squares that form an Lshape and uses the void predicate to ensure that the area inside the L is void of any shapes. Once that schema is found, the subsequent schema fills in the void space with another square. By repeatedly applying this rule the user can cause all expansions of the grid to produce a rectilinear shape.

Figure 14 shows one possible derivation of the grammar to create a 3x3 grid. There is an infinite number of possible derivations because the grammar allows the user to modify the grid size until the user is ready for the next stage. Regardless of which rule is applied, there will always be a valid design state.

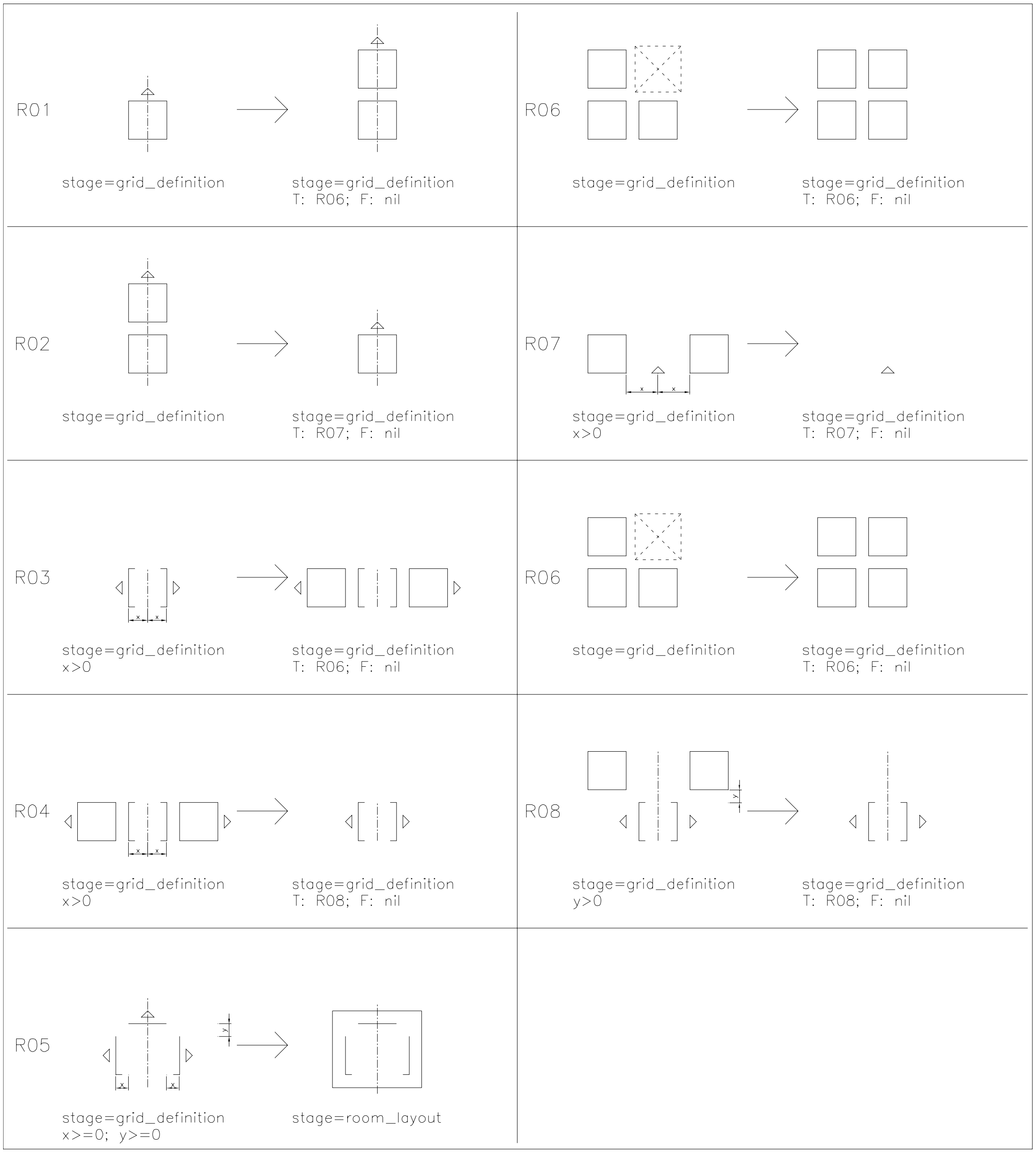

Figure 13. Grid Grammar rules.

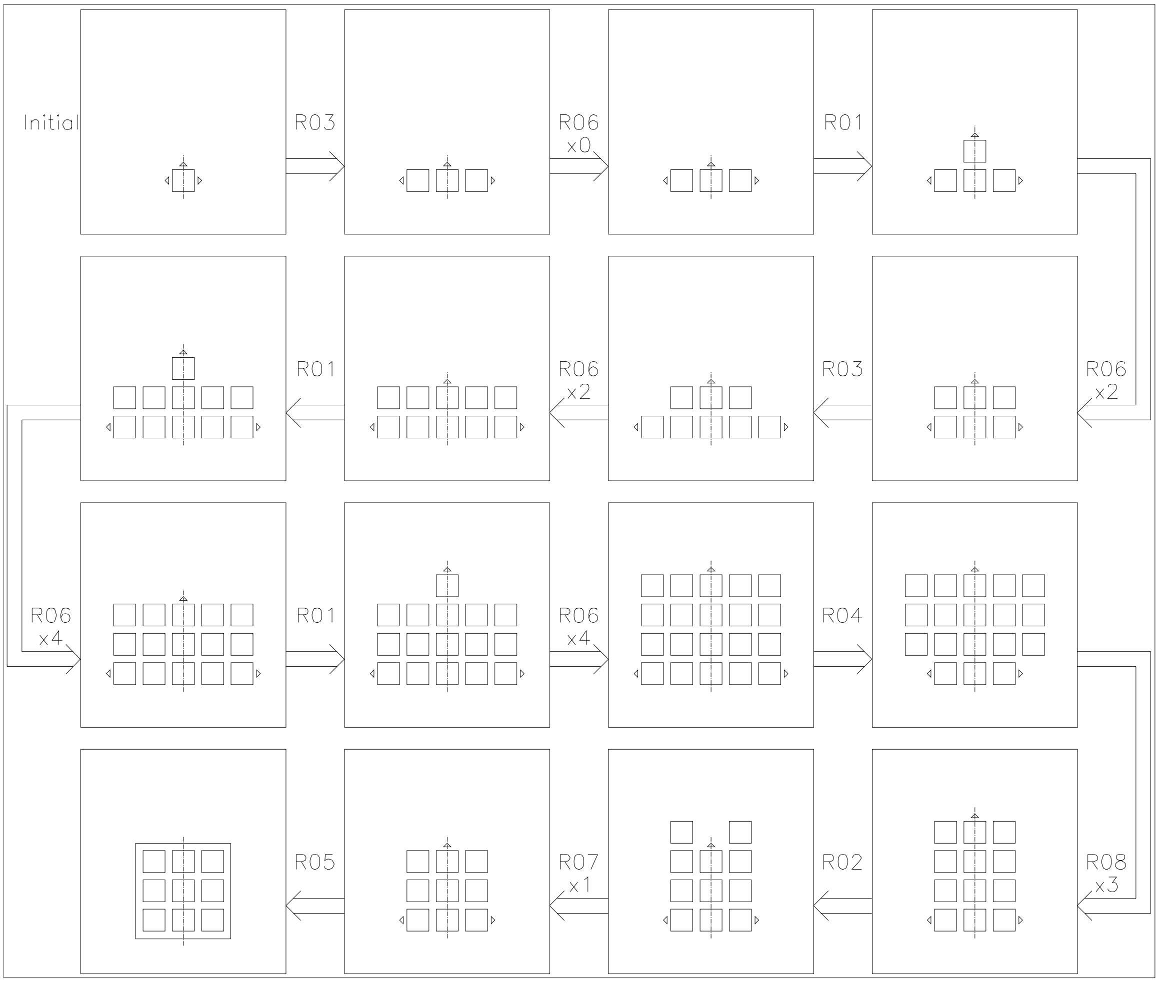

Figure 14. One derivation of the Grid grammar.

The derivation begins by expanding the width of the grid through the application of rule R03. The directive in R03 requires that rule R06 be applied next. But because the antecedent of R06 cannot be found in the drawing, the rule does not apply. The second step in the derivation is to increase the height of the grid through the use of R01. R06 is applied immediately after R01 because R06 is linked, by the directive, with R01. R06 recursively applies itself twice to finish the corners of the grid. This automated process frees the user from having to repetitively re-apply the same rule. All the user needs to be concerned about is the overall size of the grid.

Should the grid be higher or shorter? Should the grid be wider or narrower? The derivation clearly shows, unlike the other grammars, that the Grid grammar allows the user to change his mind about the size of the grid. At step 11 of the derivation, the grid size is 4x5. The user at this stage decides that the grid is too big and applies rule R04 to reduce the size. After applying R04, R08 is automatically applied 3 times to reduce the width of all the other rows in the grid. The user reduces the size of the grid yet again with R02, which automatically applies R07 to arrive at the final size of a 3x3 grid. Finally, rule R05 is applied in order to move on to the next stage.

Software Implementation

My current work is the development of a 2D shape grammar interpreter. I am currently using the Visual LISP programming environment in AutoCAD to develop a program that can apply parametric shape grammar rules that use the new conventions. The program uses a vector description format (Nagakura, 1995) to not only describe a schema but to also find a schema in a drawing.

The vector description format is used to describe points and lines. For example, a horizontal line that is 5 units long is described as (line 5 0). A shape can then be described as a series of vector displacements. To describe a parametric shape, the numbers in the vector displacement description are replaced by a variable. For example, the schema for a horizontal line of any length is described as (line x 0). A graphical user interface is being developed to facilitate the generation of the schema descriptions.

When applying a rule, the program recursively searches the drawing for all instances of the schema. The results are presented to the user through an interactive menu that highlights the found schema in the drawing. Once the user selects the desired schema, the rule application is completed by subtracting the selected schema from the drawing and adding the consequent schema from the rule.

Summary

Shape grammars demonstrate that a body of design work can be formalized through a set of rules. A common problem is the lack of consideration for the user's experience in using the grammar. As a means to address this issue, two conventions have been introduced: the directive and the predicate. Directives are explicit control sequencing mechanisms that allow the author of the grammar to define macros. Predicates make it possible to demarcate an area for query. Through the use of these conventions, grammars can be designed that provide the following advantages and benefits.

1) An explicit means to sequence a series of rules is possible. The user is made aware of the appropriate sequencing of rules.
2) Invalid design states are eliminated. The user cannot generate an invalid design state.
3) Deterministic repetitive tasks are automated. The user can be freed from applying mundane technical rules.
4) More salient design choices are possible. The user is given greater design options.

It is hoped, with the development of these new conventions coupled with their software implementation, that shape grammars will be more widely used by designers to generate designs for a set body of work as well as new designs.

References

Agarwal M and Cagan J (1998). A blend of different tastes: the language of coffee makers. *Environment and Planning B: Planning and Design 25*,205–226.

Flemming, U (1987). The Role of Shape Grammars in the Analysis and Creation of Designs. *In Computability of Design*, ed. Yehuda E. Kalay, 245 - 272. New York: John Wiley & Sons.

Knight T W (1980). The generation of Hepplewhite-style chair back designs. *Environment and Planning B: Planning and Design 7*, 227-238.

Koning, H and J Eizenberg (1981). The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and Planning B: Planning and Design 8*, 295-323.

Li, Andrew (2001). A shape grammar for teaching the architectural style of the Yingzao Fashi. Massachusetts Institute of Technology PhD Dissertation.

Miller, George (1956). The magical number seven plus or minus two: Some limits on our capacity for processing information. *Psychological Review 63*, 81-97.

Nagakura, T (1995). *Form Processing: A System for Architectural Design*. Harvard PhD Dissertation.

Schon, Donald (1984). *The Reflective Practitioner*. New York: Basic Books.

Stiny, George (1976). Two exercises in formal composition. *Environment and Planning B: Planning and Design 3* ,187-210.

Stiny, George (1980). Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design 7*, 343-351.

Stiny, George (1990). What is a design? *Environment and Planning B: Planning and Design 17*, 97-103.

Stiny, George and W. J. Mitchell (1978). The Palladian Grammar. *Environment and Planning B: Planning and Design 5*, 5-18.

Stiny, George and W. J. Mitchell (1980). The grammar of paradise: on the generation of Mughal gardens. *Environment and Planning B: Planning and Design 7*, 209-226.