Description grammars

Precedents revisited

Stouffs, Rudi

DOI
10.1177/0265813516667301

Publication date
2016

Document Version
Accepted author manuscript

Published in
Environment and Planning B: Planning & Design

Citation (APA)
Stouffs, R. (2016). Description grammars: Precedents revisited. *Environment and Planning B: Planning & Design*, *45 (2018)*(1), 124-144. https://doi.org/10.1177/0265813516667301

Description grammars: precedents revisited

Abstract

A description grammar, in conjunction with a shape grammar, serves to generate verbal descriptions of designs, next to the spatial descriptions. These verbal descriptions can also assist in guiding the generative process. This paper revisits applications of description grammars found in literature and demonstrates how they can be recast and redeveloped to make use of a general notation and implementation for description grammars. The review of this notation was the topic of a previous paper; this paper is both meant as an illustration and as a confirmation of those review analysis results.

Introduction

In his seminal "note on the descriptions of design," Stiny (1981) proposed to augment a shape grammar with a description function in order to construct verbal descriptions of designs, next to the spatial descriptions generated by the shape grammar. About thirty-five years onwards, we can find more than a few applications in literature of a description function, often denoted a description grammar, in conjunction with a shape grammar, to qualify designs both spatially and descriptionally.

In this paper we revisit a few applications of description grammars and demonstrate how these can be recast and redeveloped to make use of a general notation and implementation for description grammars. This general notation itself was derived largely from these examples and was previously (forthcoming) reviewed to explicate its strengths and limitations with respect to these and other examples. It was concluded, though, that while some differences are readily specifiable and can be attributed to

being cosmetic in nature, due to convention, a present discrepancy in the implementation, or a present omission, other differences are not, for example the adopted notation for conditionals may not support all variations in conditionals found in literature. Without further examination, it is hard to predict how well (or badly) this notation performs in light of the various description schemes.

For example, (reference withheld) has revisited Stiny's (1981) example illustrating the application of a description function with designs made up of blocks from Froebel's building gifts. One obvious alteration from this example is the explicit use of functions to retrieve the last coordinate pair of a tuple, determine the number of distinct coordinate pairs in a tuple, etc. Less obvious is the need to explicate the initial coordinate pair as a tuple of this coordinate pair and the empty entity *e*, because, otherwise, extracting the last element of the tuple would yield a single coordinate rather than the coordinate pair. Stiny (1981), instead, relegates the extraction of the last coordinate pair to the explanation provided with the rule. Finally, Stiny offers a graphical depiction of the adjacency matrix and terms the rooms "W" and "E", instead of "R0" and "R1".

In this paper, we revisit a few exemplar description schemes and demonstrate how these can be recast and redeveloped to make use of the available notation and implementation. This is both meant as an illustration and as a confirmation of the analysis results.

A study example

We start with a simple study example. Brown et al (1996, p. 156) present an example of a description function for a shape grammar over a domain of squares. The grammar

contains two rules, the first rule adds a square below and to the right at a rectilinear (i.e., taxicab) distance equal to the length of the square, and the second rule rotates a square 45° about its center (Figure 1). The description function assigns each square a description specifying a name for the square, its length, its center position and its rotation angle. The description function is specified as:

f_1: {**square**$_i$(*l*:z, *c*:(x, y), *r*:t)} → {**square**$_i$(*l*:z, *c*:(x, y), *r*:t), **square**$_j$(*l*:z, *c*:(x+z/2, y-z/2), *r*:t)}

f_2: {**square**$_i$(*l*:z, *c*:(x, y), *r*:t)} → {**square**$_i$(*l*:z, *c*:(x, y), *r*:t+45(mod 90))}

with the initial description specified as {**square**$_1$(*l*:10, *c*:(0,0), *r*:0)}

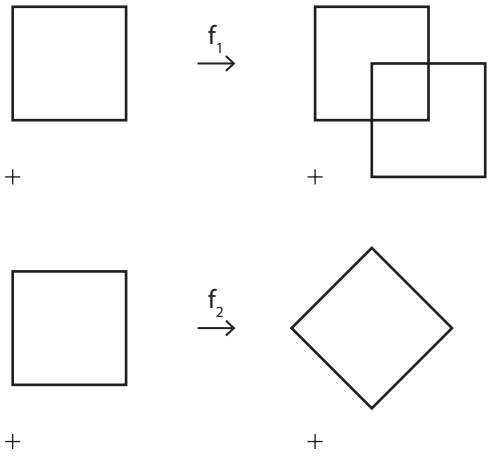


Figure 1: The two shape rules of the grammar: adding a square below and to the right, and rotating a square 45°.

Note that f_1 is explained above as adding a square below and to the right (in positive x-direction and negative y-direction). However, this does not necessarily correspond to the conjunctive shape rule. Firstly, the shape rule has 4-fold rotational symmetry and, thus, the square can be added in any of the four diagonal directions. Secondly, if the square has been rotated by 45° (using rule f_2), adding the square below and to the right

would no longer agree with the shape rule. We address these issues by removing any rotational symmetry and requiring the rotational value *r*:t in f_1 to be zero.

Different from Stiny's (1981) example, rule f_2 and, to some extent, rule f_1 can apply to any square, not just the current square or the last square resulting from rule application. As such, the difficulty lies in identifying the description corresponding to a selected square or, vice versa, the square corresponding to a selected description, under conjunctive rule application. A first solution exists in assigning each description as an attribute to the corresponding square, in a similar way as Beirão (2012) assigns descriptions to shape objects. Selecting the square then automatically selects the description as its attribute.

We present this solution in the Sortal Description Language (SDL), an interpretive language for describing *sortal* representational structures, data constructs and rules. We consider the square, with marker, as an attribute to the description. The resulting *sortal* structure, termed 'solution1', is composed of a *sort* of descriptions, a *sort* of squares (specified in terms of line segments and marker points, the latter to avoid rotational symmetry), and an additional description specifying an index:

```
sort squares : (shapes : [LineSegment]) ^ (markers : [Point]);
sort solution1 : (descriptions : [Description]) ^ squares + (index :
[Description]);
```

For the sake of readability, we ignore the specification of the squares and only define the description rules. Each rule is assigned a name, a brief explanation, and a left-hand-side (lhs) and right-hand-side (rhs) description. The descriptions are enclosed within backquotes (``...``) to distinguish the descriptions from the remaining SDL code.

```
rule f1 “add a square below and to the right at a rectilinear distance
equal to the length of the square”
<< solution1:
{ (descriptions ^ squares): { `“square”.i (“l:”, z, “c:”, c, “r:”, 0)`
},
  index: { `j` } }
>> solution1:
{ (descriptions ^ squares): {
  `“square”.i (“l:”, z, “c:”, c, “r:”, 0)`,
  `“square”.(index.value + 1) (“l:”, z, “c:”, c + (z/2,-z/2), “r:”,
0)` },
  index: { `j + 1` } };

rule f2 “rotate a square 45° about its center”
<< solution1:
{ (descriptions ^ squares): { `“square”.i (“l:”, z, “c:”, c, “r:”, t)`
},
  index: { `j` } }
>> solution1:
{ (descriptions ^ squares): { `“square”.i (“l:”, z, “c:”, c, “r:”, (t
+ 45) % 90)` },
  index: { `j` } };
```

Note that rule f_2 should only rotate the square and not the marker. In fact, we can ignore the marker in the rule specification. Next, we construct a simple derivation as a sequence of both rule applications. Each intermediate result is captured in an SDL variable.

```
form $initial = solution1:
{ (descriptions ^ squares): { `“square1” (“l:”, 10, “c:”, (0, 0),
“r:”, 0)` },
```

```
  index: { `1` } };
form $two = solution1: $initial|f1;
form $final = solution1: $two|f2;
```

Finally, we show the results of both rule applications, in an assignment to the same SDL variables.

```
form $two = solution1:
{ (descriptions ^ squares): {
    `"square1" ("l:", 10, "c:", (0, 0), "r:", 0)`,
    `"square2" ("l:", 1-, "c:", (5, -5), "r:", 0)` },
  index: { `2` } };
form $final = solution1:
{ (descriptions ^ squares): {
    `"square1" ("l:", 10, "c:", (0, 0), "r:", 45)`,
    `"square2" ("l:", 1-, "c:", (5, -5), "r:", 0)` },
  index: { `2` } };
```

An alternative, second, solution exists in assigning the index number of a square as an attribute (in the form of a numeric label) to this square, then to refer in the lhs of the description rule to this index, in order to restrict rule application to the appropriate description.

```
sort solution2 : squares ^ (indices : [Numeric]) + descriptions +
index;

rule f1_s2 "add a square below and to the right at a rectilinear
distance equal to the length of the square"
<< solution2:
{ descriptions: { `"square".i?=indices.value ("l:", z, "c:", c, "r:",
0)` },
  index: { `j` } }
```

```
>> solution2:
{ descriptions: {
  `"square".i ("l:", z, "c:", c, "r:", 0)`,
  `"square".(index.value + 1) ("l:", z, "c:", c + (z/2,-z/2), "r:",
0)` },
  index: { `j + 1` } };

rule f2 "rotate a square 45° about its center"
<< solution2:
{ descriptions: { `"square".i?=indices.value ("l:", z, "c:", c, "r:",
t)` },
  index: { `j` } }
>> solution2:
{ descriptions: { `"square".i ("l:", z, "c:", c, "r:", (t + 45) % 90)`
},
  index: { `j` } };
```

We leave the specification of the derivation and the results as an exercise to the reader.

A third solution could exist in retrieving the length value, center position and rotation angle directly from the shape. However, with the possible exception of the center position (center of gravity of the polygon), this would require particular functions to be specified as part of the grammar. We consider the second solution as preferable over this third solution and will not further explore this.

We can conclude that the differences with the original description function are both of a logical and of a cosmetic nature. The need to keep track of the number of squares is of a logical nature and can be avoided if we omit the numbering of the squares. The need for (double) quotes to delimit strings is of a cosmetic nature, and can be addressed with a more informal notation, albeit only for presentation.

Descriptions as expressions

Brown (1997, p. 30) considers description rules (denoted description functions) for volume calculation of stepped grooved shafts. These description rules require the shape rule to provide values for the diameter and length of the section when adding a new section to the shaft (Figure 2), and values for the diameter of the section and the width of the groove, when adding a circumferential groove to a section of the shaft. They are specified as:

f_1: vol → vol + $S_2.l$*π*(0.5*$S_2.d$)^2

f_2: vol → vol + $S_2.l$*π*(0.5*$S_2.d$)^2

f_3: vol → vol

f_4: vol → vol – π^2*(0.5*$g.d$)*(0.5*$g.w$)^2 + 4/3π*(0.5*$g.w$)^3

f_5: vol → vol

with the initial description specified as the value of $S.l$*π*(0.5*$S.d$)^2

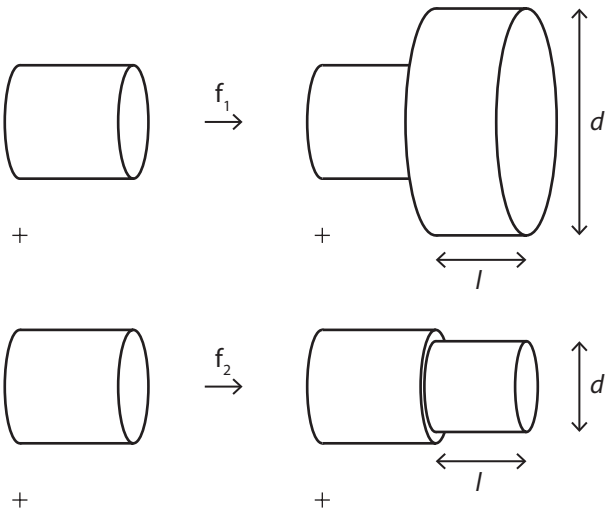


Figure 2: Two shape rules of the grammar: adding a new section with a larger diameter, and adding a new section with a smaller diameter (after Brown, 1997, p. 28).

Each rule specifies exactly one parameter in the lhs, matching the numeric value of the current volume, and either specifies a numeric expression over this parameter to update the current value or leaves this value unchanged.

We assume *sortal* structures termed ‘sections’ and ‘grooves’ to respectively represent the sections and grooves of the shaft as geometric cylinders, and define a *sortal* structure termed ‘volume’ to represent the volume descriptions (Table 1). Rather than presenting the full SDL notation, we ignore any formatting that is not absolutely necessary to capture the notation of the descriptions and description rules. We also omit rules f3 and f5 from Table 1; since these rules do not alter the description, they do not require a description rule part. Note that the label “S” is only assigned to the current section under rule application or, upon rule application, to the new section. We omit a derivation.

Table 1: Description rules for volume calculation of stepped grooved shafts.

	`(grooves : [Cylinder]) + (sections : [Cylinder]) ^ (labels : [Label]) + volume`
f1	Add a new section with a larger diameter `volume: { `vol` } → { `vol +` `rhs.sections.length:labels.label=“S” * pi *` `rhs.sections.radius:labels.label=“S”^2` }`
f2	Add a new section with a smaller diameter `volume: { `vol` } → { `vol +` `rhs.sections.length:labels.label=“S” * pi *` `rhs.sections.radius:labels.label=“S”^2` }`
f4	Add a circumferential groove

	volume: { `vol` } → { `vol − pi^2 * lhs.sections.radius * (rhs.grooves.length / 2)^2 + 4 / 3 * pi * (rhs.grooves.length / 2)^3` }

Aside from cosmetic differences, this example also shows a structural dissimilarity: the original description rules identify geometric elements by their label, while the adopted notation also requires the *sort* (or type) of the element and of the label to be identified.

Brown et al (1996, p. 162) consider a description function that generates process plans for the manufacturing of objects manufacturable by a given process. Specifically, the shape vocabulary consists of a single parametric shape (akin to a right trapezoid). The parameters are α, β, μ and ν, with $\alpha \leq \beta$ and $\mu < \nu$ (Figure 3). Additionally, the turning tool has parameters *l*, its remaining length, *h*, its height, and *m*, its minimum movement height or, alternatively, its gradient for turning.

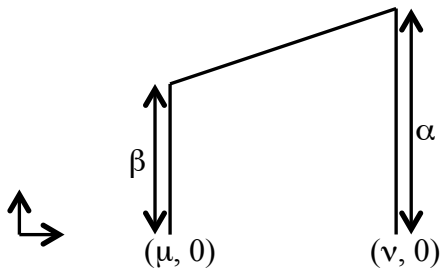


Figure 3: Selected shape vocabulary for the generation of process plans for the manufacturing of objects with a turning tool (after Brown et al, 1996).

Brown et al (1996) define ten (parametric attributed labelled) shape rules; rules 1 through 5 and rule 10 have a conjunctive description rule. A description specifies a tuple of five parts: the length, a tuple of locations, a tuple of movement heights, a tuple of letters ‘r’ or ‘f’ specifying a rough or fine surface finish, respectively, and a tuple of

removal blocks. Note that Brown et al (1996, p. 162) omit the specification of the length in rules f2, f4 and f5, suggesting that the length remains unchanged. However, an examination of the shape derivation example shows that both rules f2 and f4 may result in an update of the length value (the corresponding description derivation omits the specification of *l* altogether). For this reason, we use a value *l'* in the rhs of rules f2, f4, and f5 without further specification.

f1: $(l, L, H, S, R) \rightarrow (l, [0 \mid L], [0 \mid H], [s \mid S], [[] \mid R])$

f2: $(l, L, [h \mid H], S, R) \rightarrow (l', L, [\max(h, \alpha, \beta) \mid H], S, R)$

f3: $(l, [l_1\ l_2 \mid L], [h_1\ h_2 \mid H], [r\ r \mid S], [[]\ X \mid R]) \rightarrow (l, [l_2 \mid L], [h_2 \mid H], [r \mid S], [[((\lambda,\lambda), (\lambda,\lambda), (l-\nu,\alpha), (l-\nu,T.h)) \mid X] \mid R])$ if $X = [((l-\nu,y_1), (l-\nu,T.h), (x_3,y_3), (x_3,T.h)) \mid X1]$ and $T.m > 0$

$(l, [l_1\ l_2 \mid L], [h_1\ h_2 \mid H], [f\ f \mid S], [[]\ X \mid R]) \rightarrow (l, [l_2 \mid L], [h_2 \mid H], [f \mid S], [[((\lambda,\lambda), (\lambda,\lambda), (l-\nu,\alpha), (l-\nu,T.h)) \mid X] \mid R])$ if $X = [((l-\nu,y_1), (l-\nu,T.h), (x_3,y_3), (x_3, y_4)) \mid X1]$

$(l, [0 \mid L], [h \mid H], [s \mid S], [[] \mid R]) \rightarrow (l, [l-\nu \mid L], [h \mid H], [s \mid S], [[((\lambda,\lambda), (\lambda,\lambda), (l-\nu,\alpha), (l-\nu,T.h))] \mid R])$ otherwise

f4: $(l, L, H, S, [[((x_1,y_1), (x_2,y_2), (x_3,y_3), (x_3,y_4)) \mid X] \mid R]) \rightarrow (l', L, H, S, [[((l-\mu,\beta), (l-\mu,T_2.h), (x_3,y_3), (x_3,y_4)) \mid X] \mid R])$ if $y_1 = \alpha$ and $(\alpha-\beta)/(\mu-\nu) = (y_3-y_1)/(x_3-x_1)$ or if $x_1 = y_1 = y_2 = \lambda$

$(l, L, H, S, [X \mid R]) \rightarrow (l', L, H, S, [[((l-\mu,\beta), (l-\mu,T_2.h), (l-\nu,\alpha), (l-\nu,T_1.h)) \mid X] \mid R])$ otherwise

f5: $(l, L, H, S, [[((x_1,y_1), (x_2,y_2), (x_3,y_3), (x_3,y_4)) \mid X] \mid R]) \rightarrow (l', L, H, S, [[((l-\mu,\beta), (l-\mu,T.h+\delta*T.m), (x_3,y_3), (x_3,y_4)) \mid X] \mid R])$ if $y_1 = \alpha$ and $(\alpha-\beta)/(\mu-\nu) = (y_3-y_1)/(x_3-x_1)$ or if $x_1 = y_1 = y_2 = \lambda$

$(l, L, H, S, [X \mid R]) \rightarrow (l', L, H, S, [[((l-\mu,\beta), (l-\mu,T.h+\delta*T.m), (l-\nu,\alpha), (l-\nu,T.h)) \mid$

X] | R]) otherwise

f10: (l, L, H, S, R) → π(l, ρ(L), ρ(H), ρ(S), ρ(R))

In these rules, T refers to the tool, T_1 and T_2 distinguish the configuration of the tool with respect to the lhs, respectively, rhs of the shape rule. The function *max* returns the maximum of its arguments; the function ρ reverses a list; the function π is a grammar-specific function. The initial description is of the form (ν−μ, [], [], [], []).

For the sake of simplicity, we will assume a *sort* of *shapes* with elements having the properties alpha, beta, mu and nu, corresponding to the shape parameters α, β, μ and ν, and a sort of labels, indicating the current shape and distinguishing between the tool activities of retraction ("S"), forward movement ("N") and turning ("T"). We also assume a *sort* of *tool* element with the properties height and movement, corresponding to the parameters *h* and *m*. We omit rule f10, because of the grammar-specific function, and rule f5, because it does not appear in the derivation example presented by Brown et al (1996, p. 164) and it is very similar to rule f4. Corresponding to f1 through f4 and their variations, we define nine rules in total. We disregard any changes to the length *l* by the rules f2 and f4 (Table 2).

Table 2: Description rules for generating process plans for the manufacturing of objects with a turning tool.

	`description + (shapes : [XShape]) + (tool : [XTool])`

f1a	Start a rough cutting operation
	`description: { `(l, L, H, S, R)` } →`
	`{ `(l, [0 L], [0 H], ["r" S], [[] R])` }`

f1b	Start a fine cutting operation

	description: { `(l, L, H, S, R)` } → { `(l, [0 L], [0 H], [“f” S], [[] R])` }
f2	Move the tool along the workpiece description: { `(l, L, [h H*], S, R)` } → { `(l, L, [max([h, shapes.alpha:labels.label=“N”, shapes.beta:labels.label=“N”]) H], S, R)` }
f3a	Select option A to begin cutting description: { `(l, [l1 l2 L*], [h1 h2 H*], [“r” “r” S*], [[] [((nuprime?=(l-shapes.nu), y1), (nuprime, Th?=tool.height), (x3, y3), (x3, Th)) X*] R*])` } → { `(l, [l2 L], [h2 H], [“r” S], [[((“λ”, “λ”), (“λ”, “λ”), (nuprime, shapes.alpha), (nuprime, Th)) ((nuprime, y1), (nuprime, Th), (x3, y3), (x3, Th)) X] R])` }
f3b	Select option B to begin cutting description: { `(l, [l1 l2 L*], [h1 h2 H*], [“f” “f” S*], [[] [((nuprime?=(l-shapes.nu), y1), (nuprime, Th?=tool.height), (x3, y3), (x3, y4)) X*] R*])` } → { `(l, [l2 L], [h2 H], [“f” S], [[((“λ”, “λ”), (“λ”, “λ”), (nuprime, shapes.alpha), (nuprime, Th)) ((nuprime, y1), (nuprime, Th), (x3, y3), (x3, y4)) X] R])` }
f3c	Select option C to begin cutting description: { `(l, [0 L*], H, S, [[] R*])` } → { `(l, [l-shapes.nu L], H, S, [[((“λ”, “λ”), (“λ”, “λ”), (l-shapes.nu, shapes.alpha), (l-shapes.nu, tool.height))] R])` }
f4a	Cut a section description: { `(l, L, H, S, [[(x1,

	y1?=shapes.alpha:labels.label="T"), (x2, y2), (x3, y3?=(y1+ (shapes.alpha:labels.label="T"-shapes.beta:labels.label="T")/ (shapes.mu:labels.label="T"-shapes.nu:labels.label="T")* (x3-x1))), (x3, y4)) X*] R*])` } → { `(l, L, H, S, [[((l-shapes.mu:labels.label="T", shapes.beta:labels.label="T"), (l-shapes.mu:labels.label="T", rsh.tool.height), (x3, y3), (x3, y4)) X] R])` }
f4b	Cut a section description: { `(l, L, H, S, [[(("λ", "λ"), (x2, "λ"), (x3, y3), (x3, y4)) X*] R*])` } → { `(l, L, H, S, [[((l-shapes.mu:labels.label="T", shapes.beta:labels.label="T"), (l-shapes.mu:labels.label="T", rsh.tool.height), (x3, y3), (x3, y4)) X] R])` }
f4c	Cut a section description: { `(l, L, H, S, [X R*])` } → { `(l, L, H, S, [[((l-shapes.mu:labels.label="T", shapes.beta:labels.label="T"), (l-shapes.mu:labels.label="T", rsh.tool.height), (l-shapes.nu:labels.label="T", shapes.alpha:labels.label="T"), (l-shapes.nu:labels.label="T", lsh.tool.height)) X] R])` }

The first variant of rule f3 also specifies a condition on T.m, however as it does not appear in the lhs of the rule, this condition could not be included in rule f3a (Table 2). This is a limitation of the notation adopted.

We omit the derivation, as references to shapes and shape rules are currently not supported. However, note that when replacing the references with fixed numbers, considering additional rules where numbers may differ, the derivation of the

intermediate plan descriptions presented by Brown et al (1996, p. 164) can be demonstrated. Aside from minor cosmetic differences (quoted strings) and the structural dissimilarity of shape references, there is only one other distinction with respect to the original description rules, that is, the notation adopted from regular expressions to collect subtuples, rather than a notation borrowed from logic programming.

Verbal descriptions as reflections

Li (2001) applies a description function to the specification of a shape grammar for (teaching) the architectural style of the *Yingzao fashi* (Chinese building manual from 1103). Li considers nine descriptions and seven drawings, in parallel. The nine descriptions are u, the number of bays, v, the number of rafters, w, the number of storeys, b, the disposition of the beams, c, the number of columns in depth, x, the widths of bays, y, the length of rafters, z, the width of columns, and l, the elevations of purlins. The descriptions u, v, w and c are expressed as integers, the descriptions y and z as well; the latter describe lengths in *fen*, a unit of length. The descriptions x and l are expressed as tuples of integers; the lengths of the tuples are related to the values of u and v, respectively. Finally, the description b is expressed as a tuple of strings, these constitute the actual descriptions, as found in Liang (1983). In fact, Li (2001) considers not one but two descriptions, denoted b_c and b_e, b_c presents the Chinese version (using the Latin alphabet) and b_e the English version.

The seven drawings are the plan diagram, the section diagram, the plan (a scale drawing), the partial elevation, the roof section, the section, and the (complete) elevation. Corresponding to the seven drawings, Li's grammar is specified in seven stages, denoted as A through G. Each stage is thus concerned with a single drawing.

Also each stage uses only a subset of the nine descriptions. Stage A uses the descriptions *u* and *v* and has eight rules. Only rules A1 through A3, A5 and A6 affect descriptions *u* and *v*. The latter two rules simply assign the values to variables *u* and *v*, corresponding to the descriptions. The initial values are $i = 0$ (for *u*) and $j = 0$ (for *v*).

A1(u): $i \rightarrow i + 1$

A1(v): $j \rightarrow j + 2$

A2(u): $i \rightarrow i + 2$

A3(v): $j \rightarrow j + 2$

A5(u): $u = i$

A6(v): $v = j$

Stage B uses the descriptions *v*, *w*, *b* (b_c and b_e) and *c*, and has 48 rules (rule B10 is illustrated in Figure 4). We only present the rules that affect any of the descriptions. Description *v* is not affected, only the variable *v* is referenced. Only rule B1 affects description *w*. Description *c* is actually not explicated, its value is contained within description *b*. Rules affecting b_c and b_e are formulated as sub-rules affecting individual tuple entities. The initial values are $w = 0$, $b_c = \varnothing\ \varnothing\ \varnothing$ and $b_e = \varnothing\ \varnothing\ \varnothing$.

B1(w): $w \rightarrow w + 1$

B10: $b_{c1} \rightarrow$ *v-jia chuan wu*

$b_{c3} \rightarrow$ *yong* 2 *zhu*

$b_{e1} \rightarrow$ *v*-rafter building

$b_{e3} \rightarrow$ with 2 columns

B11: $b_{c2} \rightarrow$ *tong yan*

$b_{e2} \rightarrow$ clear span

B12: b_{c2} → *fen xin*

b_{c3} → *yong c* + 1 *zhu*

b_{e2} → centrally divided

b_{e3} → with *c* + 1 columns

B16: b_{c2} → b_{c2}, *qian zhaqian*

b_{c3} → *yong c* + 1 *zhu*

b_{e2} → b_{e2}, 1-rafter beam in front

b_{e3} → with *c* + 1 columns

B17: b_{c2} → b_{c2}, *qian rufu*

b_{c3} → *yong c* + 1 *zhu*

b_{e2} → b_{e2}, 2-rafter beam in front

b_{e3} → with *c* + 1 columns

B18: b_{c2} → b_{c2}, *qian* 3-*chuan fu*

b_{c3} → *yong c* + 1 *zhu*

b_{e2} → b_{e2}, 3-rafter beam in front

b_{e3} → with *c* + 1 columns

B19: b_{c2} → b_{c2}, *qian* 4-*chuan fu*

b_{c3} → *yong c* + 1 *zhu*

b_{e2} → b_{e2}, 4-rafter beam in front

b_{e3} → with *c* + 1 columns

B20: b_{c2} → b_{c2}, *hou zhaqian*

b_{c3} → *yong c* + 1 *zhu*

b_{e2} → b_{e2}, 1-rafter beam in back

b_{e3} → with *c* + 1 columns

B21: b_{c2} → b_{c2}, *hou rufu*

b_{c3} → *yong c* + 1 *zhu*

b_{e2} → b_{e2}, 2-rafter beam in back

b_{e3} → with *c* + 1 columns

B22: b_{c2} → b_{c2}, *hou* 3-*chuan fu*

b_{c3} → *yong c* + 1 *zhu*

b_{e2} → b_{e2}, 3-rafter beam in back

b_{e3} → with *c* + 1 columns

B23: b_{c2} → b_{c2}, *hou* 4-*chuan fu*

b_{c3} → *yong c* + 1 *zhu*

b_{e2} → b_{e2}, 4-rafter beam in back

b_{e3} → with *c* + 1 columns

B44: *qian* a_1, *hou* a_2, *qian* a_3, *hou* a_4 → *qian* a_1 a_3, *hou* a_2 a_4

a_1 in front, a_2 in back, a_3 in front, a_4 in back → a_1 a_3 in front, a_2 a_4 in back

B45: *qian* a_1, *hou* a_1 → *qian hou* a_1

a_1 in front, a_1 in back → a_1 in front and back

B46: For $a_1 + a_2 = c$

qian a_1, *hou* a_2 → a_1 *dui* a_2

b_{c3} → *yong c* – 1 *zhu*

a_1 in front, a_2 in back → a_1 abutting a_2

b_3 → with *c* – 1 columns

B47: *qian hou* a_1 a_1 → *qian hou bing* a_1

a_1 a_1 in front and back → double a_1 in front and back

B48: *qian hou* a_1 a_2 → *qian hou ge* a_1 a_2

a_1 a_2 in front and back → a_1 a_2 in both front and back

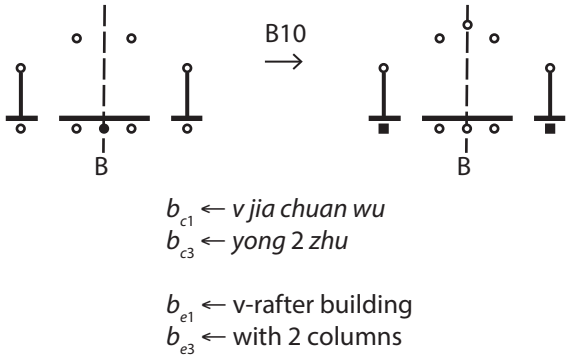


Figure 4: Rule B10 instantiates the ridge purlin and prepares the section diagram (markers) for the next sub-stage (after Li, 2001).

Stage C uses the descriptions x and y and, indirectly, references the descriptions u, v, b and c. Description x specifies the widths of bays; these are user-defined given the constraint that 300 *fen* $\geq x_1 \geq \ldots \geq x_m \geq$ 200 *fen*, where $m = (u + 1)/2$. Description y specifies the length of rafters; this is also user-defined given the constraint $y \leq$ 150 *fen*. Additionally, Li (2001, pp. 78-82) considers a counter i as part of description x to indicate the index of the current value within the tuple. Thus, practically speaking there may be five descriptions. The description ξ may be a tuple of length $(u + 1)/2$ containing the user-defined widths of bays. The description x is then a sub-tuple of ξ; rules may add a value from ξ to x in the order specified by ξ. The description i may be an integer expressing an index into the tuple x; rules may increase its value. The description y_3 may contain a single user-defined length of rafters. The description y is then initialized to 0 and subsequently assigned the value of y_3. The description rules themselves are straightforward; we omit them. We also omit the remaining stages, as they do not add anything new.

Thus, we consider rules from stage A and B that alter any of the descriptions (Table 3). We define a *sort* composed of five description *sorts*, corresponding to the descriptions u, v, w, b_c and b_e. As the compositional relationship is a disjunctive one, it suffices for each rule to specify only those component *sorts* that are affected or referenced. The latter is important, descriptions that are not explicated in the rule cannot be referenced. We omit any rules that are not used in the exemplar derivation below, mainly because they are quite similar to other rules included. We make an exception for rule B46 and its conditional specification. Note that in rule B1 (Table 3) we denote the parameter k instead of w (see above), as w refers to the description *sort*.

Table 3: Description rules for the architectural style of the *Yingzao fashi.*

	`bc + be + u + v + w`

A1	Instantiate the front centre bay
	`u: { ` i ` } → { ` i + 1 ` }`
	`v: { ` j ` } → { ` j + 2 ` }`

A2	Increase the width of the plan diagram `u: { ` i ` } → { ` i + 2 ` }`
A3	Increase the depth of the plan diagram `v: { ` j ` } → { ` j + 2 ` }`
B1	Initiate the base `w: { ` k ` } → { ` k + 1 ` }`

B10	Initiate the ridge purlins
	`bc: { ` bc1 bc2 bc3 ` } →`
	`{ ` v.value.“-jia chuan wu” bc2 “yong 2 zhu” ` }`
	`be: { ` be1 be2 be3 ` } →`

	`{ `v.value.“-rafter building” be2 “with 2 columns”` }` `v: { `j` } → { `j` }`

B12	Centrally divided
	`bc: { `bc1 bc2 “yong ”.c.“ zhu”` } →`
	`{ `bc1 “fen xin” “yong ”.(c + 1).“ zhu”` }`
	`be: { `be1 be2 “with ”.c.“ columns”` } →`
	`{ `be1 “centrally divided” “with ”.(c + 1).“ columns”` }`

B16	1-rafter beam in front
	`bc: { `bc1 bc2 “yong ”.c.“ zhu”` } →`
	`{ `bc1 bc2.“, qian zhaqian” “yong ”.(c + 1).“ zhu”` }`
	`be: { `be1 be2 “with ”.c.“ columns”` } → { `be1 be2.“, 1-rafter beam in front” “with ”.(c + 1).“ columns”` }`

B20	1-rafter beam in front
	`bc: { `bc1 bc2 “yong ”.c.“ zhu”` } →`
	`{ `bc1 bc2.“, hou zhaqian” “yong ”.(c + 1).“ zhu”` }`
	`be: { `be1 be2 “with ”.c.“ columns”` } → { `be1 be2.“, 1-rafter beam in back” “with ”.(c + 1).“ columns”` }`

B44	Combine the front beams and back beams into an intermediate description
	`bc: { `bc1 bc20.“, qian ”.bc21.“, hou ”.bc22.“, qian ”.bc23.“, hou ”.bc24 bc3` } → { `bc1 bc20.“, qian ”.bc21.“ ”.bc23.“, hou ”.bc22.“ ”.bc24 bc3` }`
	`be: { `be1 be21.“ in front, ”.be22.“ in back, ”.be23.“ in front, ”.be24.“ in back” be3` } → { `be1 be21.“ ”.be23.“ in front, ”.be22.“ ”.be24.“ in back” be3` }`

B45	Apply if number of front beams equals number of back beams `bc: { `bc1 bc20.“, qian ”.bc21.“, hou ”.bc22?=bc21 bc3` } →`

	{ `bc1 bc20.“, qian hou ”.bc21 bc3` } be: { `be1 be20.“, ”.be21.“ in front, ”.be22?=be21.“ in back” be3` } → { `be1 be20.“, ”.be21.“ in front and back” be3` }
B46	Apply if number of front and back beams equals number of columns bc: { `bc1 bc20.“, qian ”.bc21.“, hou ”.bc22 “yong ”.c.“ zhu”` } → { `bc1 bc20.“, ”.bc21.“ dui ”.bc22 “yong ”.(c - 1).“ zhu”` } be: { `be1 be20.“, ”.be21.“-rafter beam in front, ”.be22.“-rafter beam in back” “with ”.c?=(be21 + be22).“ columns”` } → { `be1 be20.“, ”.be21.“-rafter beam abutting ”.be22 “with ”.(c - 1).“ columns”` }
B47	Apply (optional) to remove repetition bc: { `bc1 bc20.“, qian hou ”.bc21?{“zhaqian”, “rufu”, “3-chuan fu”, “4-chuan fu”}.“ ".bc22?=bc21 bc3` } → { `bc1 bc20.“, qian hou bing ”.bc21 bc3` } be: { `be1 be20.“, ”.be21.“ beam ”.be22?=be21.“ beam in front and back” be3` } → { `be1 be20.“, double ”.be21.“ beam in front and back” be3` }

We present a derivation exemplified by Li (2001, pp. 60 (figure 5), 69-71 (figures 10 and 11)) (Table 4). As previously mentioned, we omit any steps that do not affect the descriptions. Additionally, we add the application of rule B47 as the last step of the derivation (erroneously omitted by Li (2001, p. 71)).

Table 4: Derivation of a 6-rafter building, centrally divided, double 1-rafter beam in front and back, with 7 columns.

	bc: { `e e e` }

	be: { `e e e` } u: { `0` } v: { `0` } w: { `0` }
A1	bc: { `e e e` } be: { `e e e` } u: { `1` } v: { `2` } w: { `0` }
A2, A2	bc: { `e e e` } be: { `e e e` } u: { `5` } v: { `2` } w: { `0` }
A3, A3	bc: { `e e e` } be: { `e e e` } u: { `5` } v: { `6` } w: { `0` }
B1, B10	bc: { `"6-jia chuan wu" e "yong 2 zhu"` } be: { `"6-rafter building" e "with 2 columns"` } u: { `5` } v: { `6` } w: { `1` }
B12	bc: { `"6-jia chuan wu" "fen xin" "yong 3 zhu"` } be: { `"6-rafter building" "centrally divided" "with 3 columns"` } u: { `5` } v: { `6` }

	w: { `1` }
B16	bc: { `"6-jia chuan wu" "fen xin, qian zhaqian" "yong 4 zhu"` } be: { `"6-rafter building" "centrally divided, 1-rafter beam in front" "with 4 columns"` } u: { `5` } v: { `6` } w: { `1` }
B20, B16, B20	bc: { `"6-jia chuan wu" "fen xin, qian zhaqian, hou zhaqian, qian zhaqian, hou zhaqian" "yong 7 zhu"` } be: { `"6-rafter building" "centrally divided, 1-rafter beam in front, 1-rafter beam in back, 1-rafter beam in front, 1-rafter beam in back" "with 7 columns"` } u: { `5` } v: { `6` } w: { `1` }
B44	bc: { `"6-jia chuan wu" "fen xin, qian zhaqian zhaqian, hou zhaqian zhaqian" "yong 7 zhu"` } be: { `"6-rafter building" "centrally divided, 1-rafter beam 1-rafter beam in front, 1-rafter beam 1-rafter beam in back" "with 7 columns"` } u: { `5` } v: { `6` } w: { `1` }
B45	bc: { `"6-jia chuan wu" "fen xin, qian hou zhaqian zhaqian" "yong 7 zhu"` } be: { `"6-rafter building" "centrally divided, 1-rafter beam 1-rafter beam in front and back" "with 7 columns"` }

	u: { `5` } v: { `6` } w: { `1` }
B47	bc: { `"6-jia chuan wu" "fen xin, qian hou bing zhaqian" "yong 7 zhu"` } be: { `"6-rafter building" "centrally divided, double 1-rafter beam in front and back" "with 7 columns"` } u: { `5` } v: { `6` } w: { `1` }

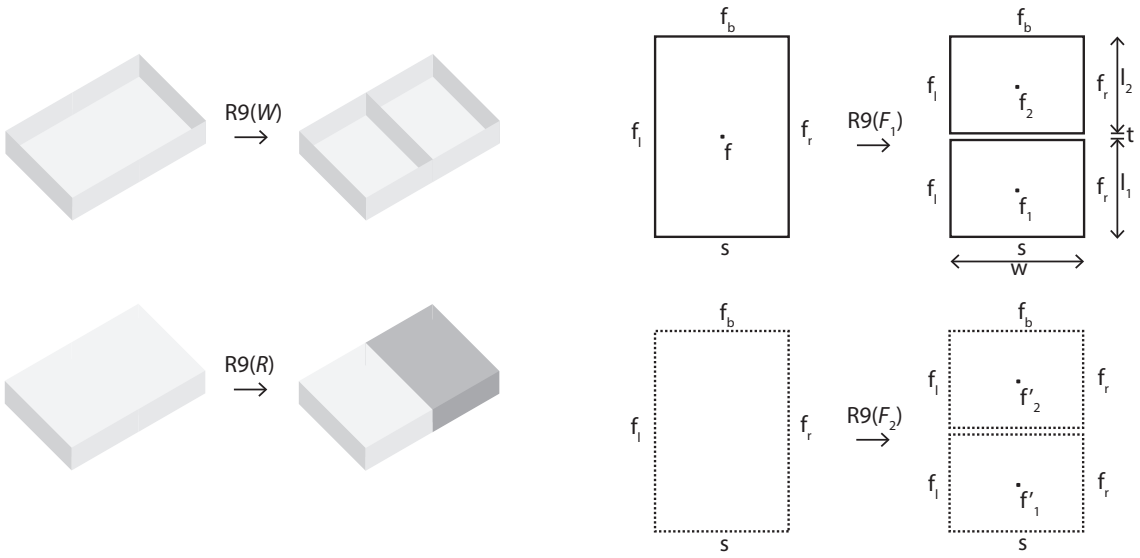
The dissimilarities between Li's more informal notation and the formal, *sortal* notation are largely cosmetic, though they are not minor. The omission of quotes, the structuring of the descriptions, alternatively, as a series of sub-descriptions and as a single description with commas where appropriate, and the informal simplification of conditionals, make Li's descriptions and description rules far superior from a human readability point of view, but prone to ambiguities and interpretations that a computer cannot handle. As an example, rule B16 adds a comma separating the existing description part and the added text "1-rafter beam in front". Often, this rule is preceded in the generation by rule B12, resulting in the existing description part "centrally divided". However, Li (2001, p. 72 (figure 12)) also presents the derivation of a "6-rafter building, 1-rafter beam in front, 2-rafter beam in back, with 4 columns". In this case, the existing description part is empty and the comma is superfluous; it is ignored in the derivation by Li. Other differences include the use of 'e' versus ∅ to denote an empty string in the *sortal* notation, the need to include rule parts for descriptions being referenced, and the inability to consider different rule parts for different elements in the tuple of strings.

Descriptions as design brief

Duarte (2001; also, 2005a) conceives a discursive grammar, which, from a technical point of view, is a combination of a shape grammar, a description grammar, and a set of heuristics. The latter is intended to constrain the rules that are applicable at each step of the design generation. From an operational viewpoint, a discursive grammar is composed of two grammars that operate in sequence. The first one, denoted a programming grammar, generates design briefs based on user and site data and omits the shape grammar part. The second one, denoted a designing grammar, uses the previously generated design brief(s) to generate designs in a particular style, and incorporates both the shape grammar part and the description grammar part. Duarte (2001) applies discursive grammars to the Portuguese housing program guidelines and evaluation system (PAHP) and the houses designed by the architect Alvaro Siza at Malagueira. Among others, Beirão (2012) extends the application of discursive grammars to urban design.

Descriptions, in these applications, take a large variety of forms, too large to address in the context of this paper. Instead, we consider the Malagueira designing grammar only, in a simplified form (Duarte, 2005b). Here, a single description represents functional zones and their adjacency relations. Duarte (2005b) explicates one example, rule 9: dissecting the outside zone into yard and sleeping zones (Figure 5). Other rules are shown, but the descriptions are only presented as conditions, not as rules.

R9: <F1; *fb*, *fr*, *ff*, li; o; *Z*> → <F1; *fb*, *fr*, *ff*, *fl*; ya, sl; *Z*–{ya, sl}>, ya, sl ∈ *Z* = {required zones}



R9(*D*): <F1; fb, fr, ff, li; o; Z> → <F1; fb, fr, ff, fl; ya, sl; Z–{ya, sl}>, ya, sl ∈ Z = {required zones}

Conditions:
$l_m < l_1$
$l_m < l_2$
$w_m > w > w_x$
$t = 0.2m$
$f_1 = o \Rightarrow f'_1 = o$
$f_2 = o \Rightarrow f'_2 = o$
$f = o \wedge f_1 \neq o \Rightarrow f'_1 = i'$
$f = o \wedge f_2 \neq o \Rightarrow f'_2 = i'$
$f = i \Rightarrow f'_1 = i \wedge f'_2 = i$

Figure 5: Rule R9 dissects the outside zone into yard and sleeping zones (after Duarte, 2005b).

Rule 9 only applies to the first floor (label 'F1'). The parameters *fb*, *fr*, *ff*, and *fl* identify the functions associated with adjacent spaces at, respectively, the back, right, front, and left side of the space currently considered for dissection. In the left-hand-side of rule 9, the parameter *fl* is replaced by the label 'li' (living room), which restricts rule application to a space adjacent to the right of the living room. The next element in the tuple identifies the function currently associated with the space being dissected, specifically, the outside zone ('o'). In the right-hand-side of rule 9, two functions are specified as the space has been dissected into two; these functions are yard zone ('ya') and sleeping zone ('sl'). The final tuple element is the set of required zones (*Z*). Rule application can only occur if both the yard zone and sleeping zone are part of the set of

required zones; this is expressed as a conditional. Additional conditionals (Figure 5) control the dimensioning and the relationships between first and second floor. If a zone on the first floor resulting from the dissection is denoted outside (e.g., yard), the same area on the second floor must be outside as well. Similarly, if a zone on the first floor resulting from the dissection is denoted inside (e.g., sleeping zone), the same area on the second floor must be inside as well.

We propose the use of two parallel descriptions, one for the specification of the zones and their adjacencies, and another for the required zones. Furthermore, we assume these descriptions only to apply to the first floor and for similar descriptions to apply to the second (and third) floor. All descriptions are considered as sets. The specification of the zones and their adjacencies considers a set of quintuples, each specifying the functions associated with adjacent spaces at the back, right, front, and left side of the space under consideration, and the function associated with the space itself. The specification of the required zones considers a set of zone labels. Rather than a (textual) description per se, this could also be considered as a set of labels; the enclosing backquotes would then be removed. Table 5 explicates description rule 9 as adapted to the general notation. As a single space is dissected into two spaces, description F1 requires a single tuple on the left-hand-side of the rule and presents two tuples on the right-hand-side of the rule. As the two resulting spaces are adjacent, their adjacency relations reflect on this. Although Duarte (2005b) does not explicate this in the description rule, in the shape rule of the second floor plan and in the control conditions specified, he identifies the effect the dissection has on the second floor plan. As such, in Table 5, we have also explicated the description rule for the second floor: one outside zone is dissected into one outside zone (above the yard zone) and one inside zone (‘i’) (above the sleeping zone). Additionally,

in the shape rules, it is assumed that the original space is adjacent at the front to the street (‘s’). This information has been added to the description rules.

Table 5: Description rule 9: dissecting the outside zone into yard and sleeping zones.

	F1 + F2 + Z1
R9	Dissect the outside zone into yard and sleeping zones F1: { `<fb, fr, "s", "li", "o">` } → { `<"sl", fr, "s", "li", "ya">`, `<fb, fr, "ya", "li", "sl">` } F2: { `<fb, fr, "s", fl, "o">` } → { `<"i", fr, "s", fl, "o">`, `<fb, fr, "o", fl, "i">` } Z1: { `"ya"`, `"sl"` } → {}

Descriptions as generative guide

Stiny revisits descriptions in “Shape: Talking about Seeing and Doing” (2006) and presents description rules for Palladian villa plans that count the number of rooms and assign plans to equivalence classes. The description rules have the form (Figure 6):

$step \rightarrow step + 1$

$(m, n) \rightarrow (m, n + 1)$

$(m, n) \rightarrow (m + 2, n)$

$N \rightarrow mn$

$N \rightarrow N - 1$

$N \rightarrow N - 2$

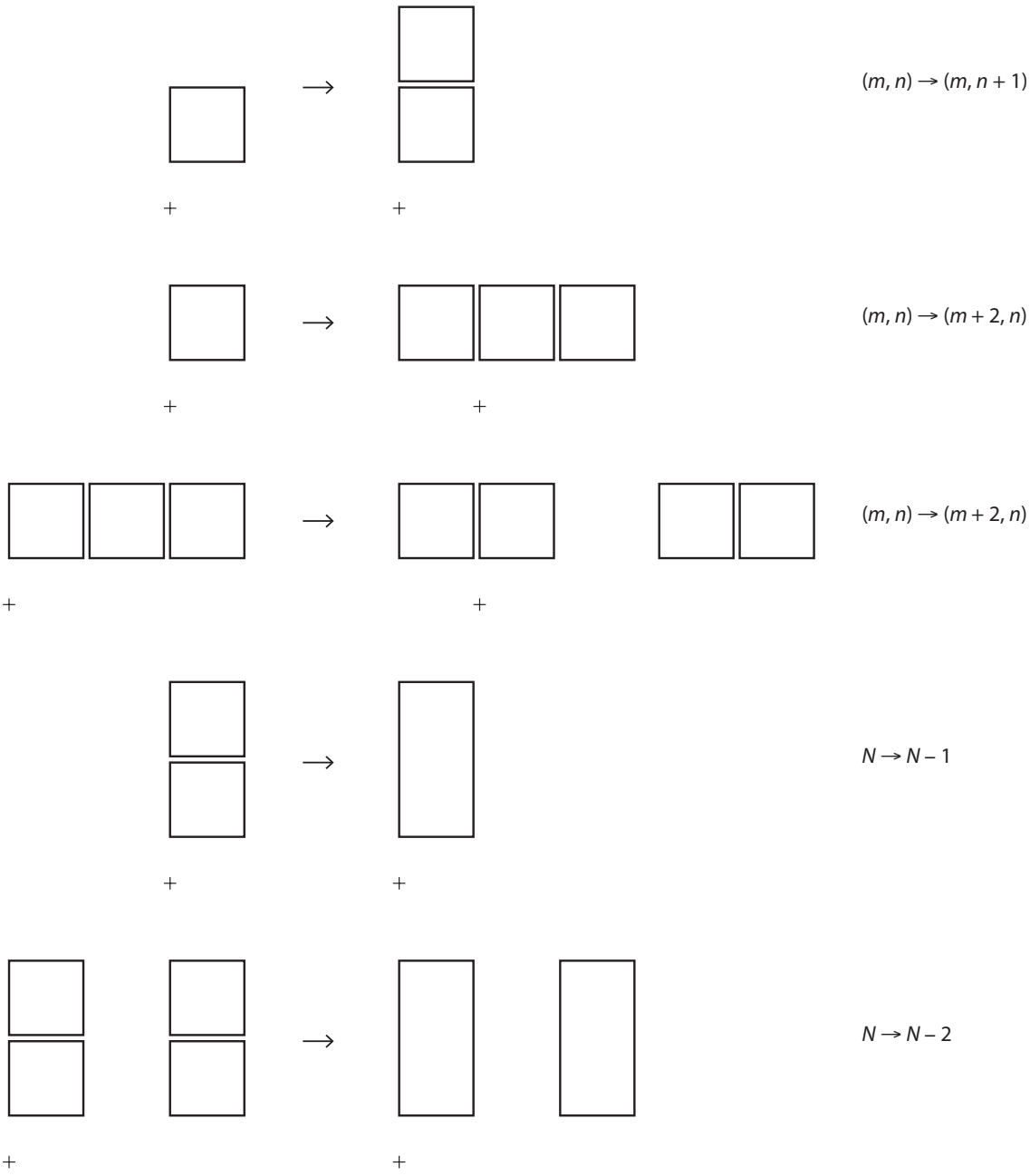


Figure 5: Exemplar rules to generate Palladian villa plans and their description components (after Stiny, 2006).

The first rule counts the number of steps in the generation. As such, it should be assigned to all the shape rules. The next two description rules count the width and the height of the grid being generated, with *m* the number of rectangles in a row and *n* the number of rectangles in a column. The former description rule accompanies shape rule 7, while the latter accompanies shape rules 3 and 4 in the Palladian grammar (Stiny and

Mitchell, 1978). The fourth description rule is invoked after the grid is completed and computes the number of rectangles, i.e., rooms. It can be assigned to shape rule 11, which generates the exterior walls and initiates the room layout rules. The last two description rules count the number of rooms as rooms are being combined into larger rooms. The second to last description rule applies to shape rules 14 and 17, whereas the last description rule applies to shape rules 12, 13, 15, 16, and 18 in the Palladian grammar. Table 6 illustrates these descriptions rules adapted to the general notation. We leave the derivation as an exercise to the reader.

Table 6: Description rules for counting the number of rules in Palladian villa plans.

	`grid + rooms + steps`

r3
r4
	Add grid cells horizontally
	`grid : { `(m, n)` } → { `(m + 2, n)` }`
	`steps : { `step` } → { `step + 1` }`

r7	Add a grid cell vertically
	`grid : { `(m, n)` } → { `(m, n + 1)` }`
	`steps : { `step` } → { `step + 1` }`

r11	Generate the exterior walls
	`grid : { `(m, n)` } → { `(m, n)` }`
	`rooms : { `N` } → { `grid.n * grid.m` }`
	`steps : { `step` } → { `step + 1` }`

r12
r13
r15
r16
	Concatenate three space into one, or four spaces into two
	`rooms : { `N` } → { `N - 2` }`
	`steps : { `step` } → { `step + 1` }`

r18	

r14
r17
	Concatenate two spaces into one
	`rooms : { `N` } → { `N - 1` }`
	`steps : { `step` } → { `step + 1` }`

Stiny (2006, pp. 362-367) explores the use of these descriptions to set goals to guide and control the design process. The number of rooms defines a partitioning within the catalogue of room layouts with a given grid. Specifying the number of rooms limits the rules that are applicable in generating room layouts. We leave the derivation as an exercise to the reader.

Ahmad (2009) proposes a description scheme to map the style characteristics of shape rules, based on the concept of semantic differential. The style characteristics are specified as numeric values quantifying opposing adjectival pairs for each shape rule. The values are collected through rule application and analyzed to characterize the style or styles of the design and of the language of designs as generated by the grammar. The mapping of the style range of the grammar may serve as a guide for grammar transformation. Ahmad presents two exemplar grammars, one for Greek temple façades and one for mobile phone designs. In the case of the Greek temple façade grammar, five rule sets are specified. Composition rule sets B and C consider style descriptor ranks reflecting on spatial relations between primitive shape elements: *Symmetric—Asymmetric*, *Monolithic—Fragmentary*, and *Stable—Directional*. Specification rule sets D, E and F consider style descriptor ranks reflecting on the primitive shape elements themselves: *Rectilinear—Curvilinear*, *Symmetric—Asymmetric*, and *Simple—Detailed*. Each rule in these sets specifies a shape transformation, a rank for each relevant style descriptor, and a weight. The style descriptor rank is specified both as a numeric, either

1 or −1, and an alphanumeric value. For example, in the case of style descriptor *Symmetric—Asymmetric*, the rank is either 1, “Symmetric”, or −1, “Asymmetric”. Derived designs collect style descriptor ranks for each rule applied, also considering the weight of the rule. Based on this collection of style descriptor ranks, the final design is ranked according to the style descriptors *Unity—Diversity*, *Balanced— Unbalanced*, *Simple—Complex*, and *Dominance*.

Ahmad (2009) presents a number of style analysis examples of student designs. The descriptions are only conceptually developed and the description rules are not formalized at all. (reference withheld) considers one of the presented designs as a case study to explore and develop an explication of the description rules. The explication is relatively straightforward; the style descriptor values are collected, for each rank, as positive and negative values separately. On the basis of this information, the style descriptor ranks for the final design are determined. (reference withheld) considers a single description for the style descriptor ranks reflecting on the primitive shape elements, and similar for the style descriptor ranks reflecting on spatial relations between primitive shape elements, and includes only the numeric values for these style descriptor ranks. Alternatively, a separate description could be maintained for each style descriptor rank, and the alphanumeric value could be included alongside the numeric value.

Al-kazzaz (2011; also, Al-kazzaz et al, 2010) considers descriptions in shape grammars for hybrid design, where the descriptions provide feedback on rule application based on comparisons between the generated design and the antecedents in the corpus. One description scheme acts as a user guide for hybrid design and is specified as a set of antecedent labels. In order to ensure that subsequent shape rule applications are taken from different antecedents, each shape rule requires one of the labels of its antecedents

to be part of the user guide and subsequently removes all the labels of its antecedents from the user guide. Al-kazzaz considers two more description schemes specifying evaluation metrics for rules and derivations, respectively. These provide feedback on the degree of innovation in hybrid design, specifically, whether the resulting design combines features from different antecedents and whether the design is sufficiently different from the antecedents in the corpus (Al-kazzaz, 2011, p. 73). The evaluation metrics provide feedback both on the rule under application, such as rule prevalence value and rule geometrical difference value, and on the design currently being derived, e.g., design diversity, design abundance, and matching degree.

Al-kazzaz (2011) demonstrates these metrics and user guide for a minaret design grammar considering a heterogeneous corpus of 12 traditional minaret designs. Each rule in the minaret design grammar identifies the antecedents this rule is derived from, as well as values for the rule metrics. Note that Al-kazzaz (2011) does not offer any explication of description rules, describing them only conceptually. (reference withheld) explores an explication of these description rules for an example presented by Al-kazzaz (2011, pp. 131–137) for a hybrid design derived using seven (original) rules.

The difficulty in explicating the metrics rules is to define the various accounts that have to be collected, which are not all straightforward. For example, design diversity relates to the number of antecedents that the generated design is derived, which implies maintaining a list of (unique) antecedents from which rules have been applied. As another example, the matching degree relates to the highest number of applied rules derived from a same antecedent, which implies maintaining the number of applied rules for each antecedent. The explication of the user guide description rules is also far from straightforward. Al-kazzaz (2011) suggests maintaining a set of antecedent labels, and removing all antecedent labels of the rule being applied from this set. However,

checking for membership of at least one of a given set of antecedent labels is a complicated matter. Instead, (reference withheld) considers a description containing a list of numbers of applied rules, one for each antecedent. Thus, a rule applies if the number of applied rules is zero for at least one antecedent, and the number of applied rules is increased by one for each antecedent of the rule. Expressing this condition requires a rule variant for each antecedent of the rule. A function returning the maximum value within this description can then be used to compute the matching degree.

Discussion and conclusion

We have revisited a number of applications of description grammars found in literature and demonstrated how they can be recast and redeveloped to make use of a general notation (and implementation) for description grammars. Though we have identified some limitations of the general notation, we have achieved an explication for each of the case studies we have addressed. Certainly, whether these explications can be considered as adequate cannot only be answered objectively, it necessarily involves some subjective evaluation as well. A final verdict would likely be made for each case study separately by the original author of the case study. Nevertheless, we can draw a few general conclusions.

An obvious conclusion would be that without a general notation (and implementation), authors embrace too much expressive freedom, resulting in description rules and grammars that are impracticable and require significant additional work in order to make them workable. Certainly, Stiny (1981) did not have the intention to explicate every aspect of his description function, for this was not necessary in order to

understand the concepts and processes presented, and the same can be said of other case studies referred to in this paper. Nevertheless, there are clear examples of where the lack of explication introduces logical inconsistencies and ambiguities that hinder a proper understanding. The study example from Brown et al (1996) is still borderline in this respect, but the case study from Al-kazzaz (2011) presents a clear example where a significant effort is required to bridge the gap from the conceptual description to a possible implementation.

The downside of this need for explication and unambiguity are the numerous cosmetic dissimilarities identified above that often reduce human readability for the sake of machine readability. Another example is structural dissimilarities, such as the choice for a notation adopted from regular expressions versus a notation borrowed from logic programming to collect subtuples. Cosmetic dissimilarities can be (partially) addressed by envisioning a presentation-only format that increases readability at the cost of unambiguousness, with a process of parsing the more explicit notation into the latter format. Structural dissimilarities may be tackled by considering variations in the notation for different uses (or authors). For example, considering that the case study of Brown et al (1996) only involves prepend operations on lists, the notation borrowed from logic programming might be perfectly preferable.

Future work may include notational variations targeting different fields of application, description patterns to address recurring issues, the investigation of a complete case study application with its original author, and weaving together formal requirements of the notation with a user's informal approach.

Acknowledgments

I would like to express my gratitude to the reviewers for their constructive comments, suggestions and insights.

References

Ahmad S, 2009, *A framework for strategic style change using goal driven grammar transformations* PhD thesis, Department of Architecture, University of Strathclyde, Glasgow, UK

Al-kazzaz D A A, 2011, *Shape grammars for hybrid component-based design* PhD thesis, Department of Architecture, University of Strathclyde, Glasgow, UK

Al-kazzaz D, Bridges A, Chase S, 2010, "Shape grammars for innovative hybrid typological design", in *Future Cities* Eds G Schmitt, L Hovestadt, L Van Gool, F Bosché, R Burkhard, S Coleman, J Halatsch, M Hansmeyer, S Konsorski-Lang, A Kunze, M Sehmi-Luck (eCAADe, Brussels) pp 187–195

Beirão J N, 2012, *CItyMaker: Designing Grammars for Urban Design* PhD thesis, Faculty of Architecture, Delft University of Technology

Brown K, 1997, "Grammatical design" *IEEE Expert* **12**(2) 27–33

Brown K N, McMahon C A, Sims Williams J H, 1996, "Describing process plans as the formal semantics of a language of shape" *Artificial Intelligence in Engineering* **10**(2) 153–169

Duarte J P, 2001, *Customizing Mass Housing: A Discursive Grammar for Siza's Malagueira Houses* PhD thesis, Department of Architecture, MIT

Duarte J P, 2005a, "A discursive grammar for customizing mass housing: the case of Siza's houses at Malagueira" *Automation in Construction* **14**(2) 265–275

Duarte J P, 2005b, "Towards the mass customization of housing: the grammar of Siza's houses at Malagueira" *Environment and Planning B: Planning and Design* **32**(3) 347–380

Li A I, 2001, *A shape grammar for teaching the architectural style of the Yingzao fashi* PhD thesis, Department of Architecture, MIT

Liang S, 1983, *Yingzaofashi zhushi*, (Zhongguo jianzhu gongye, Beijing)

Stiny G, 1981, "A note on the description of designs" *Environment and Planning B: Planning and Design* **8**(3) 257–267

Stiny G, 2006, *Shape: Talking about Seeing and Doing* (MIT, Cambridge, MA)

Stiny G, Mitchell W J, 1978, "The Palladian grammar" *Environment and Planning B: Planning and Design* **5**(1) 5–18

forthcoming, "Description grammars: a general notation" submitted to *Environment and Planning B: Planning and Design*