

Estimating the interior layout of buildings using a shape grammar to capture building style

Kui Yue¹, Ramesh Krishnamurti² and Francois Grobler³

¹ Microsoft, Redmond, WA ² Carnegie Mellon University, Pittsburgh, PA 15213

³ US Army Corps of Engineers ERDC-CERL, Champaign, IL 61826;

PH (412) 268 2360 FAX: (412) 268 7819;

email: kuiyue@microsoft.com, ramesh@cmu.edu, francois.grobler@usace.army.mil

July 2010

(Revised) December 2010, March 2011

Abstract

An algorithm is described for determining the interior layout of a building given three pieces of information: i) the footprint of each story; ii) a reasonably complete set of exterior features; and iii) a shape grammar that describes the building style. Essentially, the algorithm prunes a layout tree generated by interpreting the shape grammar with constraints extracted from the footprint and exterior features. The Queen Anne House, variously located in Pittsburgh, is chosen as the exemplars of the building style. It is shown how a shape grammar interpreter for the Queen Anne House was developed and applied to the above problem. This shape grammar interpreter provides the basis for developing grammar interpreters for general parametric shapes. For the purposes of illustration and comparison, applications of the approach to two other distinct building styles are briefly described.

Keywords: Shape grammar; building style; constraint satisfaction

Introduction

Buildings are a good resource as exemplars of design, aesthetics, construction and technology. Buildings make for good ‘copy,’ inspiring designers and intriguing critics alike for their features. Buildings are grist to the mill for the historian, valuable in other ways too. During situations of urban strife, buildings become strategic entities for law enforcement. From a perspective of sustainability, buildings provide reusable material. Indeed, assessing the environmental impact of demolishing and salvaging building stock requires one to estimate the amount of renewable materials in a building. In particular, the last two examples suggest the following question: how much can one say about a building without actually stepping into it? For example, could one roughly ‘predict’ the layout of the interior of a building from observations of its exterior and surrounding features, for instance, entrances, windows, ornaments, decorations, etc? In reality, providing an answer, perhaps, while it might not be difficult for a human, particularly, for buildings in a familiar style, it is a hard computational problem.

Figure 1 shows two photographic images of a Queen Anne house, a well-established housing style in Pittsburgh, Pennsylvania. The images, clearly, portray a three-story building. On the left side photograph, it is easy to see that, on the ground floor, there is a front porch. The two columns on the left, together with the steps and red double door, ‘inform’ us that this is the main entrance. To the right, the front porch defines a patio; the large window at the back suggests that the interior must be a living room (that is, a parlor room). This is somewhat confirmed by the presence of a chimney over the roof; see the right side

photograph. The bay window and dormer shown indicate that there are two rooms (spaces) behind the living room; the room in the interior of the bay window must be another public room. Typically, rooms on the second and third floor are bedrooms. As more features are observed, say, from other views, so too will our ability to guess become better and more refined. Figure 2 shows the real floor plans for the building.

Figure 1

Figure 2

Estimating the layout of a building is a non-simple task for a machine even with present day technology. While the precise mechanisms for human recognition remain unclear, it is safe to assume that human abilities rely on reasoning based upon accumulated knowledge of the past. In the absence of such knowledge, an individual, for example, in an unfamiliar, say a different, cultural or vernacular setting, might also find it difficult to estimate the interior layout of a building, or even the nature of the building. This is typical of the kinds of problems encountered in building restoration and preservation activities. This leads us to speculate that were there ways of providing a machine with the necessary knowledge, together with algorithms to reason against input features, a machine might be able to generate possible layouts, at least, for buildings of certain special types.

Potentially, such a technology would be useful for a variety of practical applications. For example, as previously mentioned, assessing environmental impact of demolition and salvage of building stock—the city of Baltimore is a case in point (Lund and Yost 1997). Automating the process of interior layout determination would greatly assist this endeavor.

Potential application of this work includes assisting firefighters in estimating interiors of a building, for automated building restoration from remains albeit historical or by damage, etc.

Forming the Problem as a Research Question

Even for humans, it can be extremely hard to ascertain the interior layout of particular buildings from their exterior features. For example, the Walt Disney Concert Hall by Frank Gehry does not conform to any known normal architectural conventions (Meyer 2008). Clearly, it is unlikely that one can develop a general computational solution strategy for the problem of estimating interior layouts.

On the other hand, many buildings follow a pattern book (Downing 1981; Flemming et al. 1985; Hayward and Belfoure 2005); as such, this provides a handle on how this problem might be tackled. That is, there are buildings that vary according to well-defined configurational patterns, and/or certain well-established sets of regulations and dimensions. Then, collections of buildings can be recognized as belonging to particular styles. Among these, Frank Lloyd Wright's Prairie House (Koning and Eizenberg 1981), the Queen Anne House (Flemming 1987), and the Baltimore Rowhouse (Hayward 1981) are well-known exemplars, each characterized by features distinctive of their form and design. Employing knowledge about building styles might make the problem more tractable.

For a human designer, a pattern book might well suffice, whereas, from a programming perspective, a computational mechanism is needed to represent style data encapsulated in the pattern book. For this, a shape grammar (Stiny 2006) is employed although it is possible to use other rule-based or generative approaches to describing

buildings that fall within a particular style. A shape grammar provides a remarkable facility for capturing the spatial and topological aspects of building styles and for generating these designs.

For the purposes of this paper, it is assumed that the reader is familiar with the subject matter of shape grammars (Stiny 1980, 1991, 2006). Briefly, a *shape grammar* is a system of rules, each comprising, notionally, of a left and right part, both of which are parametric shapes augmented with labels (or markers). A *parametric shape* specifies a family of shapes defined by associating parameters or parametric expressions to satisfy certain conditions in the given shape. A particular member of this family is specified, by giving an *assignment* of real values to parameters that satisfies the conditions. Labels are typically associated with points, which stand in relation to the shape, although they could also be associated with elements of the shape, for example, labeled lines. A shape rule can apply to a ‘current’ layout whenever the left part can be ‘found’ in the layout under some transformation (together with and including parameter assignment), and when the rule is applied, that found part is then replaced by the right part of the rule under the same transformation to produce a new ‘current’ layout. Symbolically, $c \rightarrow t(a) + t(b)$ expresses shape rule application, where c is the current layout, $a \rightarrow b$ denotes a shape rule with left part, a , and right part, b , and t denotes a transformation. $t(a)$ is assumed to be a part of c , denoted $t(a) \leq c$. A shape grammar always has a starting shape, which serves as the initial ‘current’ layout. The collection of unlabeled nonparametric shapes generated by the grammar is its *language*, which serves to describe style. The role of a shape grammar as a descriptor of style has been richly documented in the literature for a variety of fields (Stiny and Mitchell 1978; Knight 1980; Downing and Flemming 1981; Knight 1981a,b; Koning and Eizenberg

1981; Baker and Fenves 1990; Meyer 1995; Chiou and Krishnamurti 1995; Cagdas 1996; Antonsson and Cagan 2001; Pugliese and Cagan 2002; McCormack et al. 2004; Colakoglu 2005; Duarte 2005; Stiny 2006; Duarte et al. 2007; Eilouti and Al-Jokhadar 2007).

Assuming that data on needed building features is available, the original problem can be reformulated as seeking an algorithm to determine the interior layout of a building given three pieces of information: i) the footprint of each story; ii) a reasonably complete set of exterior features, e.g., windows, chimneys and surrounding buildings; and iii) a shape grammar, which describes the building style. At this juncture, it should be noted that the focus and emphasis in this paper is solely algorithmic. That is, accuracy of determination is not based on any statistically derived metric; instead, visual comparison between generated outcome and ground truth serve as the basis for verification.

Figure 3 illustrates the general approach, which is based on the fact that, when applied exhaustively, a shape grammar generates, as a tree, the entire layout space of a building style. The approach begins with an initial layout estimation employing constraints on certain building features, specifically the ones that serve as input. Spatial and topological constraints from this estimate are then used to prune the layout tree, and ‘fix’ possible open terms in the current configuration. The layouts that remain correspond to desired layouts.

Figure 3

Essentially, the approach requires a shape grammar interpreter to apply the rules of the shape grammar for the building style, and generates a layout tree. Note that, computation-wise, both initial layout estimation and layout tree generation can involve exponential searches. However, in practice, more often than not, both procedures are

restricted in such a way that search takes polynomial time. Those shape grammars that satisfy this complexity criterion for search are tractable (Yue and Krishnamurti 2010a). In this paper, a tractable approach is examined in detail, using the Queen Anne House style as the test case. The Queen Anne House was chosen for two main reasons: the existence of a developed shape grammar (Flemming 1987) and geographic convenience afforded by the location of such houses for obtaining photographic data. To illustrate the approach presented in this paper, two other test cases, the Baltimore Rowhouse and a highrise apartment are described briefly.

Layout Tree Generation of Queen Anne houses

Flemming (et al. 1985, 1987) analyze the Queen Anne House, and present a shape grammar describing their style. In reality, the typical Queen Anne house has a non-rectangular boundary, whereas Flemming's grammar considers and specifies just neighborhood relationships of rectangular room spaces, which is sufficient to capture the Queen Anne style. Accordingly, a layout in this grammar comprises solely of rectangular spaces—non-rectangular spaces that occur in reality are approximated as such. The grammar starts with a hallway; other rooms are progressively generated by either aggregation or subdivision (that is, splitting a room into two). A subset of Flemming's shape rules is adopted to produce Queen Anne style layouts, as shown in Figure 4. A sample derivation is illustrated in Figure 5.

Figure 4

Figure 5

Labels and markers are typically employed in shape grammars. In the Queen Anne grammar, room labels serve to indicate their function—for example, H indicate a hallway; R, room; D, dining room; K, kitchen; Pt, pantry; and S, stairway. Label X is used to denote an unspecified room function. Other labels mark special parts of the building, e.g., B for back; F, front; and C, corner. Again X is used as a parametric label to denote either B or F, determined by the spatial context. Labels also indicate status, for example, R indicates the state of generating rooms, S of generating staircases, and K of generating kitchen. The labeling convention adopted in this paper follows Flemming (1987), with the exception of the R-* notation used in Figure 5. There, it specifies the shape rule applied; for example, R-1 for rule 1, R-2 for rule 2 and so on.

The entire corpus of the Queen Anne House style can be specified by a tree structure, wherein, each node represents a stage in the generation; each edge, a rule application; and each leaf node, a valid layout. The generated layout tree is used to implement—that is, to interpret—the Queen Anne grammar. In principle, and in general, interpreting a shape grammar is intractable (Yue et al. 2009). See Gips (1999) and Chau et al. (2004) on the difficulty of implementing shape grammars. However, the Queen Anne shape rules have been redesigned so that implementation is, in fact, tractable.

Implementation requires a data structure containing information, both topological on spaces (rooms) and geometrical, which, for this paper, is two-dimensional data on layouts that includes walls, doors, windows, staircases, etc. The data structure must support manipulations such as viewing a layout as a whole, viewing a layout from a particular room together with any of its neighboring spaces, or, simply, focusing on a single room. Moreover, the data structure needs to support Euclidean transformations augmented by both uniform

and anamorphic scaling. A layout rotated through multiples of 90° is still a layout in the same sense. A layout reflected is likewise a layout. To these ends, a graph-like data structure has been designed, which represents general layouts comprising rectangular spaces, and supports easy manipulation of geometric transformations for shape rule application.

Data Structure

A rectangular space is specified by a set of walls so that the space is considered rectangular by the human vision system. Figure 6 illustrates some variations: a space specified by four connected walls each adjoins pair-wise, four disjoint walls, three connected walls, or framed by four corners.

Figure 6

A graph-like data structure (also illustrated in Figure 6) consisting of nodes and edges has been designed to specify such rectangular spaces. There is a *boundary* node (colored blue representatively indicated by the label B) for each corner of the rectangular space, as well as nodes (colored green and representatively indicated by the label G) for each endpoint of a wall. These nodes are connected by either a *wall* edge (indicated by a solid line) or an *empty* edge (indicated by a dotted line). A *room* node (colored red and representatively indicated by the label R) represents the space (room), and is connected to the four corners by *diagonal* edges (indicated by dashed lines). These edges are necessary when manipulating the boundary nodes of a room unit, for instance, when dividing a wall by node insertions, creating an opening in a wall by changing its edge type, say, to *empty*, and so on. Additional information about a room is recorded in the room node, for instance, on whether there is a

staircase within the space. Windows and doors are assigned as attributes of wall edges. However, unlike conventional graph data structures, geometry information is also included in this data structure, in that the angle measure at each corner is set to 90° . A node has at most eight neighbors. A collection of these graph units can be combined to represent complex layouts comprising rectangular spaces as shown in Figure 7.

Figure 7

Transformations of the Data Structure

A shape rule applies under allowable geometric transformations. Determining exact transformations under which a shape rule applies is typically difficult, and this is at the core of the implementation of a shape grammar interpreter. Here, the target layout is assumed to comprise only rectangular spaces, and allowable transformations are specific Euclidean transformations augmented with uniform and anamorphic scaling.

When shape rule application is label-driven, translation is automatically handled. The graph-like data structure facilitates handling of uniform and anamorphic scaling, initially, by matching room names, then, by matching labels on corner nodes, and lastly, by comparing possible room ratios or dimension requirements.

Rotations and reflections still remain to be considered. Note that the data structure can always be preprocessed so that room spaces are oriented either horizontally or vertically. As rooms are rectangular, rotations are limited to multiples of 90° , and reflections about either the horizontal or vertical axes. Moreover, a vertical reflection can be viewed as a combination of a horizontal reflection and a rotation. Hence, any combination of reflections

and rotations is equivalent to some combination of horizontal reflections and rotations. Thus, the following transformations are the ones actually needed. Here, R_θ denotes a rotation through an angle θ , and H denotes a horizontal reflection.

- R_0 : default; no rotation, with possible translation and/or scale.
- R_{90} , R_{180} , R_{270} : a rotation through 90° , 180° , or 270° , respectively, possibly, augmented with translation and/or scale.
- HR_0 , HR_{90} , HR_{180} , HR_{270} : (first, a rotation through 0° , 90° , 180° , or 270° , respectively, followed by a horizontal reflection) representing a horizontal reflection, a vertical reflection, or their combination, possibly, augmented with translation and/or scale.

Transformations can be implemented on the data structure by index manipulation. See Figure 8. Each of the eight possible neighbors of a node is assigned an index from 0 to 7; indices can be transformed using modulo arithmetic. For example, $\text{index}+2 \pmod{8}$, represents a counterclockwise rotation through 90° of the neighbor vertices. Other rotations and reflections are likewise achieved.

Figure 8

By viewing the original neighbor relationship for each node with the transformed indices, the same transformation can be obtained for the whole graph. By taking advantage of this fact, in any shape rule application, only the interior layout is needed to manipulate rather than the left part of the rule. Thus, general shape rule application is simplified to the case of applying the shape rule with the default transformation, which then can be made

automatically applicable to the configuration under the different possible transformations. This gives the same results, but it is much simpler to achieve.

Implementation of the Data Structure

Using the above data structure, a layout is represented by an eight-way doubly linked list of nodes and edges. Each shape rule application manipulates this structure—in this respect, a set of common functions shared by the shape rules can be identified. These functions are implemented in an object-oriented fashion.

Classes *LNodeCorner* and *LNodeRoom* respectively represent corner and room nodes. The *LNode* class represents all other nodes. Edges are represented by the *Edge* class, with an attribute to represent edge type. To traverse a layout, it suffices to know just the handle to a node or an edge. For easy manipulation, an *InteriorLayout* class is defined to represent an interior layout configuration. There are several ways of viewing an *InteriorLayout* object: i) as a layout with a certain status; ii) as a list of rooms (room nodes); and iii) as a list of nodes and edges. Each view is useful for different contexts. For example, the last view, iii, is convenient for displaying the underlying layout, by firstly drawing all edges as well as associated components, and then drawing all nodes and their associated components. To accommodate the different views, the *InteriorLayout* maintains the following fields:

- A status marker
- Name: for display and debugging purposes
- A hash map of a room name to a list of room nodes: for fast retrieval of one or more room nodes with a given name

- A list of room nodes for the entire layout
- A list of all nodes for the entire layout
- A list of all edges for the entire layout
- A hash map of attributes to values for other status values particular to a special shape grammar

Implementation of the Queen Anne Grammar

In principle, the implementation of the Queen Anne grammar involves converting each shape rule into a piece of code that manipulates the underlying data structure. In reality, the process becomes slightly more complex, as the original Queen Anne grammar described in (Flemming 1987) is neither properly nor fully well-defined computationally. In order to convert a shape rule into a piece of code, additional constraints are required; certain shape rules need to be *de-compacted* so that different possibilities are clarified. Figure 9 illustrates one such situation involving shape rule 2 of the grammar. In the figure, the shape rule shown in (a) is applicable to the shapes shown in (b) and (c). Rule application to the shape in (b) produces a reasonable layout whereas rule application to the shape in (c) produces a room that is dimensionally too small. Although, dimensions, as such, were not important in the original Queen Anne grammar, for implementation, in order to eliminate inappropriate cases, a sense of dimension has to be incorporated.

Figure 9

Another example is the interpretation of shape rule 8 shown in Figure 10. From the shape rule on the left, it would appear that two rooms have only to partially overlap along a

wall in order for the rule to apply. However, from the sample layouts given in (Flemming 1987), it is possible for an entire wall to be shared. For a tractable shape grammar, these two cases would need to be separately specified so that implementation becomes simply a task of coding rules.

Figure 10

Figure 11 shows fifteen screenshots, labeled (a) through (o), of sample layouts generated by the implementation. Observe that the layouts in (k) and (l) are not *valid* in any realistic sense as the top rooms are, dimensionally, much too wide. However, there are no simple solutions whereby, merely adding constraints to the shape rules would result in eliminating such cases. Theoretically, during the *fixing* step, unrealistic dimensions could be obtained for such rooms so that such layouts can be removed. In the current implementation, the fixing step was not implemented; instead, it was replaced by a post-processing procedure to remove invalid layouts. There were 506 unique possible layouts, in total, generated by the implementation.

Figure 11

Layout Tree Pruning and Initial Layout Estimation

A computer implementation of a shape grammar that captures the corpus of a building style comprises, essentially, an enumeration of the possible shapes in the language of the grammar. The enumeration procedure—that is, derivations in the shape grammar—can be viewed as a tree structure, namely, a layout tree. Valid layouts correspond to certain nodes of the tree. These nodes are mostly leaf nodes, though certain internal ones are also

possible; an internal node is a layout of smaller size, whereas a leaf node represents a layout of larger size and typically with more rooms.

Accordingly layout determination reduces to ‘picking up’ nodes consistent with the feature input from the layout tree. Direct node pickup is generally difficult; it can be achieved through tree pruning—that is, by eliminating those nodes inconsistent with certain constraints and the remaining nodes as the desired results.

Shape grammars that capture building corpora are typically parametric—that is, relationships specified are generally topological, an example of which is specifying one room as being adjacent to another under certain spatial conditions, where exact dimensions are left largely unspecified. As a result, many constraints for pruning a layout tree are necessarily topological, especially for branches near the root of the tree. This makes it difficult to prune the tree as the feature input specifies objects with real dimensions. A procedure to obtain topological constraints therefore becomes necessary. For the layout estimation problem, these topological constraints are obtained through a process termed *initial layout estimation*. This process makes use of building knowledge in the form of constraints on building features.

What is more, topological constraints alone cannot prune the tree in a way that the desired final layouts can be directly determined. Prior to reaching the final layouts, variables (aka parameters) in the intermediate configurations have to be ‘fixed’ to match the feature input at a certain stage. Fixing can be progressively performed or done in a single shot. Progressive fixing relies on constraints, either directly specified by, or inferred from, the feature input, or from *a priori* knowledge, with parameters partially fixed at each step until they are all fixed. Parameter fixing of Queen Anne houses in the implementation was

progressive. It should be noted that single shot parameter fixing could be viewed as a special case of progressive parameter fixing.

Constraints on the Building Features

Building features constrain one another—for example, windows are not normally shared between rooms (or spaces). That is, a window belongs to a single room and no wall falls within this window. Another, based on practicality, requires that the minimum width of a space be at least 2'. Other constraints require specific knowledge. For Queen Anne style houses, room dimensions are approximately in the ratio of 1:2. The height of each story is determined by window and door geometries, as the height of windows to floor are typically about 3' and doors are about 7' to 8'. See Figure 12. Furthermore, once story height has been estimated, dimensions of various staircases can be estimated by using common dimensions for treads and risers ($24" \leq \text{tread} + 2 * \text{riser} \leq 25"$). The width of a staircase is a constant, typically, 3'.

Figure 12

Another example of feature interaction is that between window and staircase. As shown in Figure 13, for windows in known positions, possible positions for the staircase are greatly reduced (the impossible region for a staircase is shaded based on the principle that no staircase crosses a window in a sectional view). With further constraints that stem from the surrounding rooms, the exact position of the staircase can be narrowed down to fall within a narrow range.

Figure 13

Constraint Satisfaction

Determining the interior layout of a building involves identifying the rooms in the building and their estimated dimensions. Some of these rooms are adjacent to the building exterior. Accordingly they correspond to a set of exterior features. For instance, most rooms in Queen Anne houses have centrally-located fireplaces (Flemming 1987), which correspond to chimneys visible over the roof of the building. The abundance of such constraints among exterior building features suggests that a preliminary layout can be found by treating it as a constraint satisfaction problem.

A constraint satisfaction problem (CSP) has three components: i) a set of variables $X = \{x_1, \dots, x_n\}$; ii) a set of possible domain values, D_i , for each x_i ; and iii) a set of constraints to restrict the values that variables can simultaneously take (Russell and Norvig 2002). Various acceleration techniques, for example, forward checking and constraints propagation, have been developed to efficiently eliminate impossible values thereby speeding up solving a CSP.

In terms of estimating an initial layout, the variables, x_1, \dots, x_n , correspond to the set of rooms identified from the exterior features such as a chimney, windows, and doors. Each room is given a conservative initial dimension. Domain D_i corresponds to possible dimensions for room x_i . Constraints are either common building knowledge, for example, rules specified in building codes, or style-specific knowledge, for example, Queen Anne houses exhibit local symmetry, and symmetry axes are derivable from exterior features.

There are benefits to considering a problem as a constraint satisfaction problem. Four are identified here. Firstly, representation as a CSP is typically much closer to the original

problem: variables directly correspond to problem entities, and constraints can be expressed more explicitly without awkward translation. Secondly, CSP representations conform to standard patterns so that many algorithms have already been written in a generic fashion. Thirdly, effective generic heuristics can be developed without requiring additional, domain-specific expertise. Lastly, the structure of a constraint graph can be used to simplify the solution process, potentially leading to an exponential reduction in algorithm complexity.

The CSP algorithm for initial layout estimation starts by generating rooms with conservative dimensions in conformance to the given features. The algorithm then manipulates rooms as variables according to constraints established by common properties of buildings. Two kinds of manipulations are considered: expanding room dimensions and merging rooms. Merging two rooms eliminates a room variable—this is in contrast to the typical CSP algorithm where variables persist throughout the life of the algorithm.

CSP and Queen Anne houses

Figure 14 shows the results of applying the CSP algorithm to the first floor of a Queen Anne house in Pittsburgh, Pennsylvania—hereafter referred to as House W. The input features are locations in plan of windows, doors, and chimneys, together with possible axes of symmetry that can be inferred from the facades. Observe that, in the case of Queen Anne houses, chimneys correspond vertically to fireplaces on the first floor. (This is not always the case for North American vernacular houses.) The initial estimation can be specified in eight steps. Step 1 extends the axes of exterior walls inward (assuming a wall thickness of 1') to form wall hotspots; this enforces the tendency of interior rooms to be aligned with one another. Step 2 uses the fact that larger public rooms on the first floor have

fireplaces, which correspond to the chimneys. By projection, if the chimney falls within the interior of the footprint, then there are possible two rooms that share the chimney, with a fireplace each. If the chimney is on an exterior wall, only one room uses the chimney. Such rooms are assigned with an initial dimension of 8' \times 8'. Step 3 adjusts rooms that are close (based on a 1' threshold) so that they should align with the nearest axis. Step 4 uses the fact that rooms can contain, but do not intersect with other rooms and doors. Rooms are extended to include such features to resolve any conflict. Step 5 uses the fact that the minimum distance between two walls has to be large enough to be a useful space (usually $> 3'$). In step 6, rooms generated from the chimneys are stabilized. Step 7 specifies rooms with an initial dimension of 5' \times 5' to unassigned windows and doors. Note that a room may be left-, center-, or right aligned with a window or door. Two largely overlaying rooms are merged as one. Further, the narrow space remaining between rooms *RM1* and *RM5* is assigned to *RM5* according to the symmetry axis, *SYM-4*. Step 8 shows the final result; although incomplete, it is close to the actual condition. At this stage, there are still ambiguities that cannot be resolved without prior knowledge; these are left for the shape rules to handle.

Figure 14

In the above example, estimated rooms are restricted to rectangles. This can pose certain difficulties when evaluating buildings partially with non-rectangular rooms as illustrated in Figure 15-h for House A. The strategy adopted is to approximate the original non-rectangular rooms by rectangles, and then correspondingly project their related windows. Such simplifications retain the necessary constraints introduced by the original window features, making constraint satisfaction applicable for a wider range of layouts, and greatly

reduce the complexity in implementing geometric manipulation. Figure 15 shows the original inputs (solid lines), the simplified version (dashed line), and the manual derivation procedure based on these approximations.

Figure 15

Apart from the complexity of geometry manipulation, the implementation has to deal with issues of potential numerical roundoff and tolerance. For example, an intersection test between a rectangle (for chimney) and a line segment (for wall axis) is used to determine whether the chimney is on the boundary of a footprint or not. However, chimney data is derived from parts over the roof, whose dimensions potentially shrink. This may result in a chimney being close enough to the line representing a wall, without actually intersecting or being coincident with it. Imperfect threshold values also cause issues. For example, as shown in Figure 15-a, the assumption of a 1' wall thickness results in two hotspot wall axes (two horizontal axes in the middle) that are very close to each other; this can break the algorithm if it is not specially handled. Moreover, quite distinct from manual derivation, rooms in the upper-right corner of Figure 15-b do not merge as desired due to the difficulty in setting a ‘universal’ threshold value—as thresholds that work for a building (or a part of) may fail for another building (or another part of) of the same style.

Figure 16 shows the results of the computer implementation of the above CSP algorithm. The partial layout, determined from the initial layout estimation using constraint satisfaction, provide useful topological constraints for pruning the layout tree. For House W, shown on the left in Figure 16, there are three rooms in the front lower side; their widths have been determined. This converts to a topological constraint on the three front rooms.

Moreover, the widths of these rooms can be fixed to those respective values. The three rooms on the left side can be treated similarly. The two partial rooms on the right side convert to a slightly weaker topological constraint, namely that there are, at least, two rooms on the right side with a determined width value. For House A shown in the middle of Figure 16, the two front rooms have fixed width dimensions. For the left side, the dimension of the rear-most room is fixed; otherwise, the only sure constraint is that there are at least three rooms on that side. Likewise, there are at least two rear rooms. For House I shown on the right in Figure 16, there are, at least, three rooms and, at most, four rooms.

Figure 16

Layout Determination of Queen Anne Houses

All algorithms described or mentioned in this paper were implemented in Java using the Swing GUI toolkit on the open source Eclipse development platform. Implementing the Queen Anne grammar is equivalent to generating a layout tree whereupon layout determination becomes simply a matter of pruning the tree using constraints from the initial layout estimation. Figure 17 shows the screenshot of the computer implementation. On the left, there are four small windows: i) the *Truth* window, which shows the true layout for purposes of comparison; ii) the *Feature Input* window, which shows the features used as input for the initial layout estimation; iii) the *Derivation* window, which shows the result of the initial layout estimation by the CSP algorithm, and can be step-animated; and iv) the *Constraints* window, which shows the constraints exacted from the partial layout resulting from the initial layout estimation. On the right is shown the *Grammar* tree window. The top-left panel shows the layout tree generated by applying all the shape rules, in which those

that are crossed out correspond to layouts inconsistent with the constraints extracted. The top-right panel shows the layouts remaining after pruning the layout tree by the extracted constraints. The central panel on the top is the drawing panel; when clicking on entries of the top-right or top-left panel, the corresponding layout is displayed. The bottom is the status bar, displaying the summary of layout tree generation and pruning. Above the status bar is the rule panel, displaying all the shape rules for the Queen Anne grammar; when an entry of the layout tree is mouse selected, the currently applicable shape rules are highlighted.

Figure 17

Simple topological constraints extracted from the partial layout results of the initial layout estimation are used to prune the layout tree. For the purpose of demonstration, such constraints were extracted manually, with the proviso that they could be automatically extracted. Note that partial constraints used here come directly from the feature input instead of the initial layout estimation. Note too that each staircase is considered as a space (room) in the data structure. The three exemplar Queen Anne houses, named W, A and I, were tested on the implementation. Screenshots of the outputs are shown in Figure 16.

For House W, the extracted constraints include the following: i) the hallway is central; ii) the number of rooms to the left of the hallway is at least two; iii) the number of rooms to the right of the hallway is, at least, two and, at most, three; iv) there must be a kitchen and a staircase to the right of the hallway, v) none of the rooms directly behind the hallway is a dining room; and vi) rooms behind the hallway are aligned with the hallway.

Figure 18 shows the results by pruning the layout tree with these constraints. Layouts (a), (b) and (d) are results closest to the truth; deviations are due to omissions in the

specification of the Queen Anne grammar rules; the leftmost rear room is a sun porch room, which is not as common among Queen Anne houses and this would have to be considered a special case outwith the grammar. Layouts (c) and (e) do not appear to be correct; these layouts are easily removed when fixing layouts to real dimensions.

Figure 18

For House A, the extracted constraints include the following: i) the hallway is central; ii) the number of rooms at the right side of the hallway is two; ii) the number of rooms at the left side of the hallway is, at least, two and, at most, three; iii) there must be a kitchen and a staircase to the left side of the hallway, iv) from the front, the first two rooms contain no staircases; and v) when the left front corner room is a dining room, the right rear corner room must be a parlor room and vice versa. Figure 19 shows the results from pruning the layout tree by these constraints. Layout (b) is closer to the truth than the layout (a), which is removed when fixed to real dimensions.

Figure 19

For House I, the extracted constraints include the following: i) the hallway is central; ii) the number of rooms at the right side of the hallway is, at least, three and, at most, four; iii) there must be a kitchen and staircase to the right of the hallway, iv) from the front, the first three rooms to the right of the hallway do not have a staircase; v) the top of the layout is flat, and vi) there is no dining room behind the hallway. Figure 20 shows the results from pruning the layout tree by these constraints. Layout (d) is closest to the ground truth than the

other layouts. Again, layouts (a) and (c) are removed when fixing dimensions. Eliminating layout (b) is nontrivial, as it is almost the same as (d).

Figure 20

Other Constraints on Queen Anne Houses

As the above results show, it is advantageous to introduce other types of constraints so that the candidate layout results can be narrowed down further. In this section, ideas of how to deal with complicated boundaries are illustrated.

Footprints of Queen Anne houses typically have a non-simple boundary shape, which poses an extra difficulty. On the other hand, such boundaries place constraints on the interior layout. For example, as shown in Figure 21, the partial footprint (shown in bold lines) on the lower-left corner specifies a bulging corner room, in this case, the parlor. Moreover, this shape pattern is generally true for most Queen Anne houses. Such boundary constraints can be taken advantage of.

Figure 21

Formally, the question is to look for a (consecutive) segment sequence forming a certain pattern from which a rectangular room can be generated. For this, a representation for a sequenced pattern is developed. Figure 22 shows the target sequence pattern of the example shown in Figure 21. Walking counterclockwise along the footprint, the sequence starts from a point, turns right, then turns left three times, and finally, turns right again. To describe such a sequence mathematically, each segment of the footprint is viewed as a vector. To turn right, the cross product of two consecutive vectors is towards the negative z-axis

direction; to turn left, the cross product is toward the positive z-axis direction. Using “-“ to represent negative z-axis and “+” for positive, the sequence pattern can be represented succinctly as “-, +, +, +, -“.

Figure 22

A potential issue of the above representation is that it may be too loose to define the exact type of sequence desired. Figure 23 shows two variations of the corner-bulge-room sequence. Both have the exact same string representation as the corner-bulge-room sequence, but the actual room type is different each handled distinctly.

Figure 23

As there is no dimensional information associated with a pattern, a sequence can actually match a physically large block consisting of multiple rooms, or it may not be meaningful in any real way. To avoid such cases, further constraints are added such as the room area and ratio (see Figure). To enable iterative application of sequences, the original footprint is modified, by deleting certain points and optionally inserting new points after a sequence has been applied. See Figure 24. Figure 25 shows the results of applying the corner-bulge-room sequence on House A.

Figure 24

Figure 25

Two applications of the approach

It is natural to expect that the implementation of the CSP algorithm applied to the Queen Anne House would work for a variety of building types through simple adjustments of appropriate threshold values. However, this is not always the case even when there are relatively small morphological variations from the Queen Anne House. The Baltimore Rowhouse (Hayward 1981; Hayward and Belfoure 2005) is a case in point. On the other hand, apartment buildings do work well with the CSP algorithms although the apartments themselves are subject to a different set of grammatical rules. In this case, it is convenient to consider the layout of an apartment floor separately from the generation of the individual apartment units. Moreover, additional exterior features would also need to be considered.

The Baltimore Rowhouse

The assumptions that apply to the Queen Anne House do not readily apply to the Baltimore Rowhouse. See Figure 26. Spaces within a rowhouse tend to be narrower and deeper than in a Queen Anne house. Moreover, spaces containing fireplaces are not always symmetrical about the fireplace—this is especially true for the kitchen. Fireplaces in a rowhouse do not necessarily align with a chimney that is visible from the exterior of the building. Additionally, the simplicity of the footprint of a typical rowhouse belies the difficulty of inferring interior wall axes from the footprint alone. As a consequence of the first two reasons, initial dimensions assigned to rooms with chimneys may fall outside the building's footprint. Furthermore, chimneys do not correspond directly to fireplaces. Although it is possible to modify the CSP algorithm to work for the Baltimore Rowhouse by

revising existing constraints and adding new types of constraints, a simpler alternative for initial layout estimation can be employed.

Figure 26

Procedurally, the first floor of the rowhouse can be determined by a decision tree (Figure 27) — essentially, as a process of subdivision. The first floor is typically decomposed into two or three rectangular blocks: a block containing the parlor towards the front, a block containing the kitchen towards the rear, and an optional, smaller central block connecting the two. In a three-block rowhouse, the central block contains a pantry or a stair, while the front and rear blocks are divided into one or two rooms. The kitchen is always the rear-most space while the parlor is the front-most space. The dining room usually appears in the front block behind the parlor or in the rear block forward of the kitchen. The two cases can be distinguished by comparing the depths of the front (h_2) and rear (h_1) blocks.

Figure 27

Two-block rowhouses are more involved. Depending on the depth (h) of the front block, it can contain a single room, or be divided into a parlor and dining room possibly separated by a staircase. If the front block comprises two rooms, the staircase can occupy an enclosed space or it can be open to one or both rooms. If the front block comprises a single room, the staircase may have multiple possible arrangements. These configurations are too complicated to be handled by the decision tree, which needs further refinement by using shape rules.

Regardless of whether a layout has two or three blocks, the front door enters into the front-most room or a dedicated hallway. This is determined from the width (w) and area (s) of the front-most room.

A shape grammar was developed to capture the corpus of the Baltimore Rowhouse (Yue and Krishnamurti 2010c). Figure 28 shows sample layout results of applying the general approach by pruning the layout tree generated by the shape grammar with constraints extracted from the initial layout estimation obtained through the decision tree. Layouts are largely determined using the decision tree—the shape grammar essentially refines the layouts with added detail. Yue and Krishnamurti (2010c) provide further description of the algorithm and its implementation.

Figure 28

A Highrise Apartment

We employed the approach to a specific highrise apartment building located in Baltimore. Highrise apartment buildings pose a different kind of challenge since it is not feasible to capture all possible layouts on a floor using a single shape grammar. On the other hand, it is possible to develop a shape grammar to generate apartment units so that all possible layouts of apartment units can be derived. Assuming that there is a list of possible apartment units, a different set of exterior features are needed so that the entire floor layout can be “assembled” from the possible apartment units. Unlike a Queen Anne house, highrise apartments usually have a uniform façade, providing weak constraints by which to determine interior layouts. However, equipment pipes over the roof can be utilized, as these, more often than not directly reflect on the interior arrangement. Bathroom ventilation pipes

typically go over the roof without changing direction. Other HVAC components also provide hints for possible interior layouts.

Figure 29 shows two possible layout generated by an implementation that uses constraints from ventilation pipes. The system takes as input a list of possible apartment units as well as positions of the ventilation pipes, searches for reasonable arrangements by using common constraints such as no two units overlap, certain rooms face the exterior and so on, to eliminate unreasonable solutions. The first layout shown in (a) matches the ground truth, whereas layout shown in (b), although logically correct, does not. Additional constraints can be specified to rule out such layouts. The method here employs a shortcut in searching the solution space; if along with the step of layout tree generation, the generation of possible apartment units were included, the search space would be considerably larger and the implementation that much more complicated.

Figure 29

Discussion

In this paper, it has been shown how the interior layout of a building can be estimated provided certain feature information has been specified. The approach described in this paper depends on an underlying shape grammar. Constraints work against layouts generated by the grammar. Consequently, any change to the shape grammar should result in a change to the layouts generated. That is, in order to qualitatively improve possible layouts that can be generated would necessitate changes to the set of shape rules that specify a grammar.

Thus, the efficacy of the approach depends on the efficacy with which the style of the building has been faithfully captured.

Ambiguity can pose a challenge for layout determination as well as for shape grammar interpretation. In layout determination ambiguity is dealt with by employing a suitable set of threshold values. The difficulty lies in specifying a set of threshold values that dynamically adjusts to context and/or building type. Ambiguity in interpreting shapes for shape rule application is handled by enforcing shape grammars to be tractable (Yue and Krishnamurti 2010a). Essentially, this entails that rules can be codified.

The key idea to the approach described in this paper is that constraints are used to prune the layout tree corresponding to an underlying shape grammar with the proviso that the target building can be described by a grammar and sufficient constraints can be found to eliminate many unreasonable results. However, it is possible that there are buildings, which are difficult to be described grammatically. Likewise, it is possible that it may be difficult to find appropriate constraints to successfully prune a layout tree. On the other hand, in practice, most typical buildings follow a pattern book style, which can be described grammatically. Moreover, such buildings follow common building design and construction rules, which lend themselves to easily specified constraints. Accordingly, the approach is likely to function well for most practical buildings.

Computationally, the complexity of the approach depends on the computational costs for generating the layout tree and for pruning the tree. The former depends on shape rule application, or more accurately, shape rule interpretation. In general, this is intractable. As shown in (Yue and Krishnamurti 2010a), the complexity of rule application for a shape with

n terms, of which k are open, is a superpolynomial in k ; as k approaches $n/2$ it becomes exponential. However, in practice, it can be shown that shape grammars can be designed or enforced so that rule application is solvable in polynomial time. As shown in this paper, initial layout estimation depends on constraints that can be resolved by different approaches. Two are illustrated in this paper, the CSP algorithm employed in determining the layout for a Queen Anne house and a highrise apartment building, and the decision tree applied to a Baltimore rowhouse. Decision trees typically have polynomial time complexity. In general, the CSP algorithm requires exponential time; in practice, its computational complexity depends on the number of constraints, which determine how fast and small a solution space can be specified. Again, the number of rooms in a typical building is small; this tends to make the CSP algorithm tractable.

The approach developed in this paper requires a general parametric shape grammar interpreter, which caters to a variety of building types. For layout estimation, it is impractical to implement individual interpreters for each building style, or alternatively, for each grammar. Following this thread, the determination of building interior layouts can be used as a vehicle to investigate the practical implementation of a general shape grammar interpreter. Conversely, the former can be viewed as an application of the latter. The implementation described here has been the basis of exploring other building layout grammars and of general parametric shape grammars (Yue et al. 2009).

Conclusion

This paper describes an approach based on a process of estimation with an inadequate amount of information, namely, a footprint, annotated, as much as possible, by exterior

features, and reasoning based on the application of two kinds of spatial rules: constraints and shape rules. The former relates to common practical architectural conventions with added style considerations. Likewise, the choice of shape rules that apply, that is the choice of the shape grammar, is a matter of local style consideration. In other words, educated guesswork on building style governs the estimation.

The specific contributions of the paper are the algorithm for determining the interior layout of a building using shape grammars and the data structure for implementing shape grammars to generate buildings consisting of rectangular spaces, or can be approximated as such. Beyond these contributions, this paper reports on a novel application of shape grammars.

There are, of course, limitations to the work.

Constraints currently employed in the CSP algorithm do not extend readily to other building types; the Baltimore Rowhouse illustrates this point well. A set of constraints, which have wider applicability needs further investigation.

Ambiguity poses a challenge for both layout determination and shape grammar interpretation. In this paper, ambiguity is handled by using a set of threshold values and through preprocessing. In general, it is difficult to develop with a set of threshold values that is applicable to various contexts or building types. Further research is necessary.

Layout pruning is based on the assumption that sufficient constraints can be found to eliminate unreasonable results. It is possible that there are buildings, for which it will be difficult to find appropriate constraints. In this paper, a set of constraints tailored to the

underlying building type is employed. In practice, it is preferable to develop a set of constraints that cater to a wider range of building types.

Lastly, the approach is untested on buildings on larger scale. The highrise example was estimated by floor, essentially as a two dimensional layout problem, initially decomposed into a collection of single units. Larger buildings have higher computational complexity, and perhaps, may even be computationally intractable. To handle larger scale buildings, improvements to the algorithm are necessary.

Acknowledgement

This research was supported in part by a grant from US Army Corps of Engineers, Engineer Research and Development Center – Champaign, IL. Any opinions, findings, conclusions or recommendations presented in this paper are those of the authors and do not necessarily reflect the views of CERL. The authors would like to thank Ulrich Flemming for permission to adapt some of his figures. The authors would like to thank the reviewers for their constructive comments and concerns.

List of References

- Antonsson, E. K. & Cagan, J. (2001) *Formal Engineering Design Synthesis*. Cambridge: Cambridge University Press.
- Cagdas, G. (1996) A shape grammar model for designing row-houses. *Design Studies*, 17, 35-51.
- Chau, H. H., Chen, X., McKay, A. & Pennington, A. (2004) Evaluation of a 3D shape grammar implementation. IN GERO, J. S. (Ed.) *Design Computing and Cognition '04*. Boston.

- Chiou, S.-C. & Krishnamurti, R. (1995) The grammar of Taiwanese traditional vernacular dwellings. *Environment and Planning B: Planning and Design*, 22, 689-720.
- Colakoglu, B. (2005) Design by grammar: an interpretation and generation of vernacular hayat houses in contemporary context. *Environment and Planning B: Planning and Design*, 32, 141-149.
- Downing, A. J. (1981) *Victorian cottage residences*. New York, Dover Publications.
- Downing, F. & Flemming, U. (1981) The bungalows of Buffalo. *Environment and Planning B: Planning and Design*, 8, 269-293.
- Duarte, J. P. (2005) Towards the Mass Customization of Housing: the grammar of Siza's houses at Malagueira. *Environment and Planning B: Planning and Design*, 32, 347-380.
- Duarte, J. P., Rocha, J. M. & Soares, G. D. (2007) Unveiling the structure of the Marrakech Medina: A shape grammar and an interpreter for generating urban form. *Artif. Intell. Eng. Des. Anal. Manuf.*, 21, 317-349.
- Eilouti, B. & Al-Jokhadar, A. (2007) A Generative System for Mamluk Madrasa Form-Making. *Nexus Network Journal*, 9, 7-30.
- Flemming, U. (1987) More than the sum of parts: the grammar of Queen Anne houses. *Environment and Planning B: Planning and Design*, 14, 323-350.
- Flemming, U., Gindroz, R., Coyne, R. & Pithavadian, S. (1985) A pattern book for Shadyside. *Technical report*. Pittsburgh, PA, Department of Architecture, Carnegie Mellon University.
- Gips, J. (1999) Computer implementation of shape grammars. Cambridge, MA: NSF/MIT Workshop on Shape Computation.
- Hayward, M. E. (1981) Urban Vernacular Architecture in Nineteenth-Century Baltimore. *Winterthur Portfolio*, 16, 33-63.
- Hayward, M. E. & Belfoure, C. (2005) *The Baltimore Rowhouse*. New York: Princeton Architectural Press.
- Knight, T. W. (1980) The generation of hepplewhite-style chair back designs *Environment and Planning B: Planning and Design*, 7, 227-238.
- Knight, T. W. (1981a) The forty-one steps: the languages of Japanese tea-room designs. *Environment and Planning B: Planning and Design*, 8, 97-114.
- Knight, T. W. (1981b) Languages of design: from known to new. *Environment and Planning B: Planning and Design*, 8, 213-238.

- Koning, H. & Eizenberg, J. (1981) The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and Planning B: Planning and Design*, 8, 295-323.
- Lund, E. & Yost, P. (1997) Deconstruction - Building Disassembly and Material Salvage: The Riverdale Case Study. Upper Marlboro, MD, NAHB Research Center, Inc.
- McCormack, J. P., Cagan, J. & Vogel, C. M. (2004) Speaking the Buick language: capturing, understanding, and exploring brand identity with shape grammars. *Design Studies*, 25, 1-29.
- Meyer, E. D. C. (2008) *Frank Gehry: on line*, Princeton, N.J. :Princeton University Art Museum.
- Meyer, S. J. (1995) *A Description of the Structural Design of Tall Buildings Through the Grammar Paradigm*, PhD Thesis, Civil Engineering, Carnegie Mellon University
- Nelson, C.B. & Fenves, S. J. (1990) Manipulating shape and its function. *Journal of Computing in Civil Engineering*, 4(3), 221-238
- Pugliese, M. J. & Cagan, J. (2002) Capturing a rebel: modeling the Harley-Davidson brand through a motorcycle shape grammar. *Research in Engineering Design*, 13, 139-156.
- Russell, S. & Norvig, P. (2002) *Artificial Intelligence: A Modern Approach*, Prentice Hall.
- Stiny, G. (1980) Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design*, 7, 343-351.
- Stiny, G. (1991) The algebras of design. *Research in Engineering Design*, 2, 171-181.
- Stiny, G. (2006) *Shape: Talking about seeing and doing*, MIT Press, Cambridge.
- Stiny, G. & Mitchell, W. J. (1978) The Palladian grammar. *Environment and Planning B: Planning and Design*, 5, 5-18.
- Yue, K., Krishnamurti, R. & Grobler, F. (2009) Computation-friendly shape grammars – Details by a sub-framework over parametric 2D rectangular shapes. IN TIDAFI, T. & DORTA, T. (Eds.) *Joining languages, cultures and vision, CAAD Futures 2009, PUM, 2009*. Montreal, Canada.
- Yue, K. & Krishnamurti, R. (2010a) Tractable shape grammars. *Working paper*, School of Architecture, Carnegie Mellon University.
<http://www.andrew.cmu.edu/~ramesh/pub/distribution/working/Tsg.pdf>.
- Yue, K. & Krishnamurti, R. (2010b) A paradigm for interpreting parametric shape grammars. *Working paper*, School of Architecture, Carnegie Mellon University.
<http://www.andrew.cmu.edu/~ramesh/pub/distribution/working/PIsg.pdf>.

Yue, K. & Krishnamurti, R. (2010c) Developing a tractable shape grammar. *Working paper*, School of Architecture, Carnegie Mellon University.
<http://www.andrew.cmu.edu/~ramesh/pub/distribution/working/DTsg.pdf>.



Figure 1. House I: A Queen Anne house, Pittsburgh, Pennsylvania (Source: Photograph, Kui Yue)

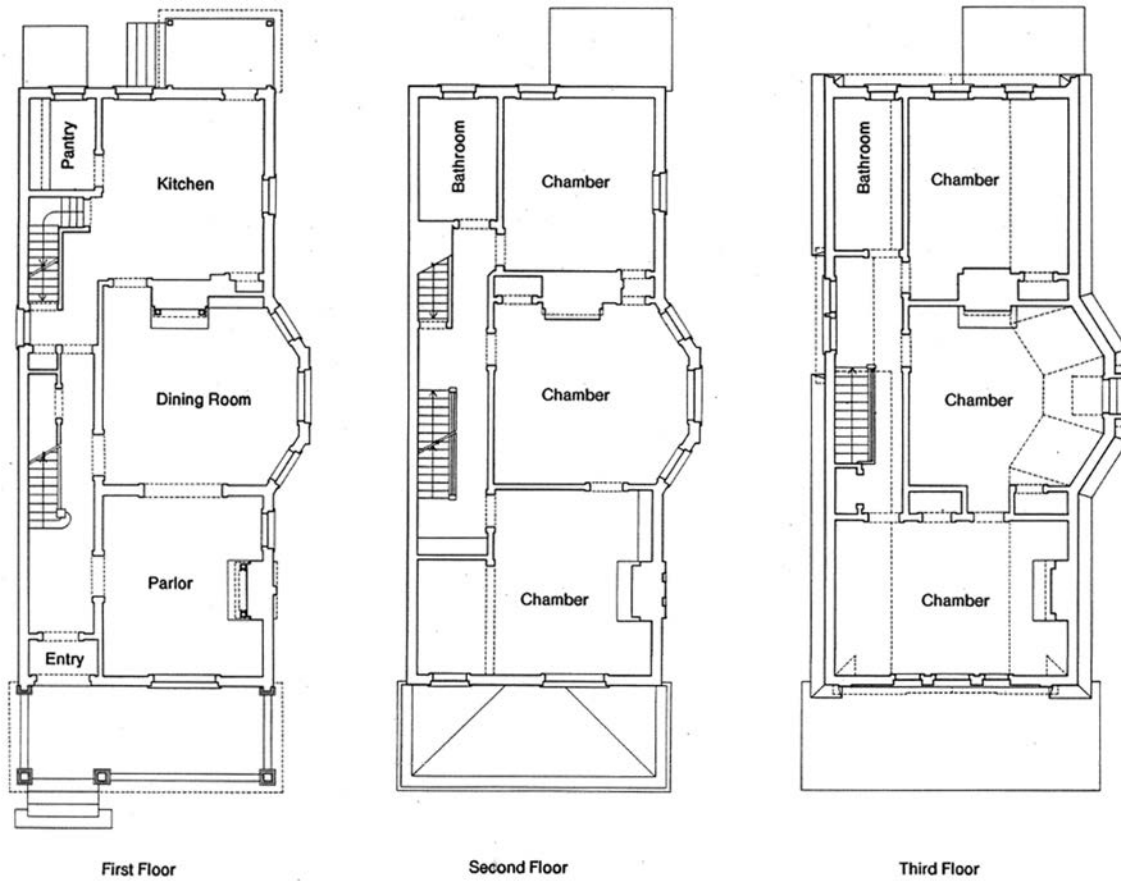


Figure 2. Floor plans for House I (Adapted and annotated by the authors from Flemming et al. (1985)

© Ulrich Flemming. Reproduced with permission.)

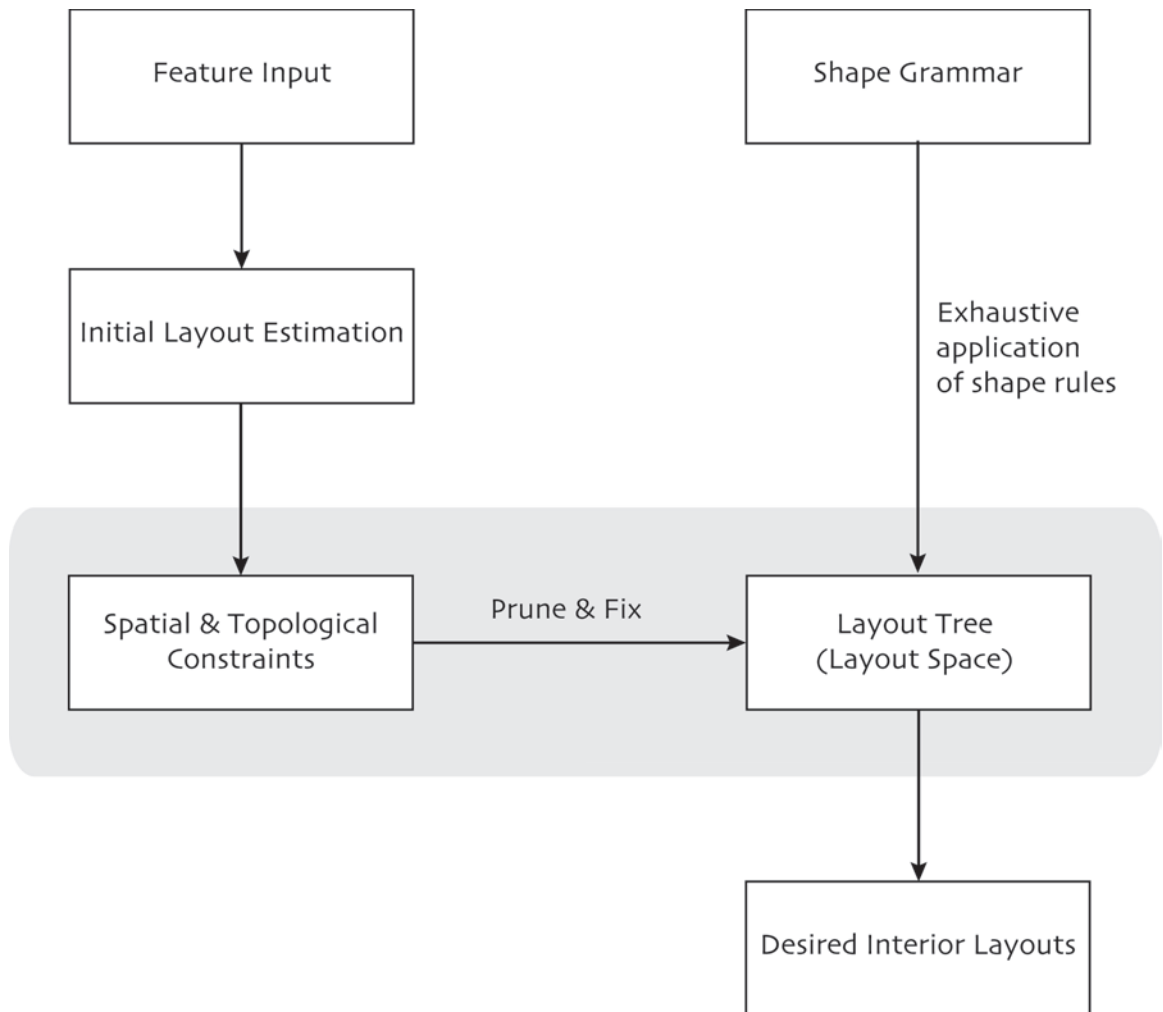


Figure 3. General approach to layout determination

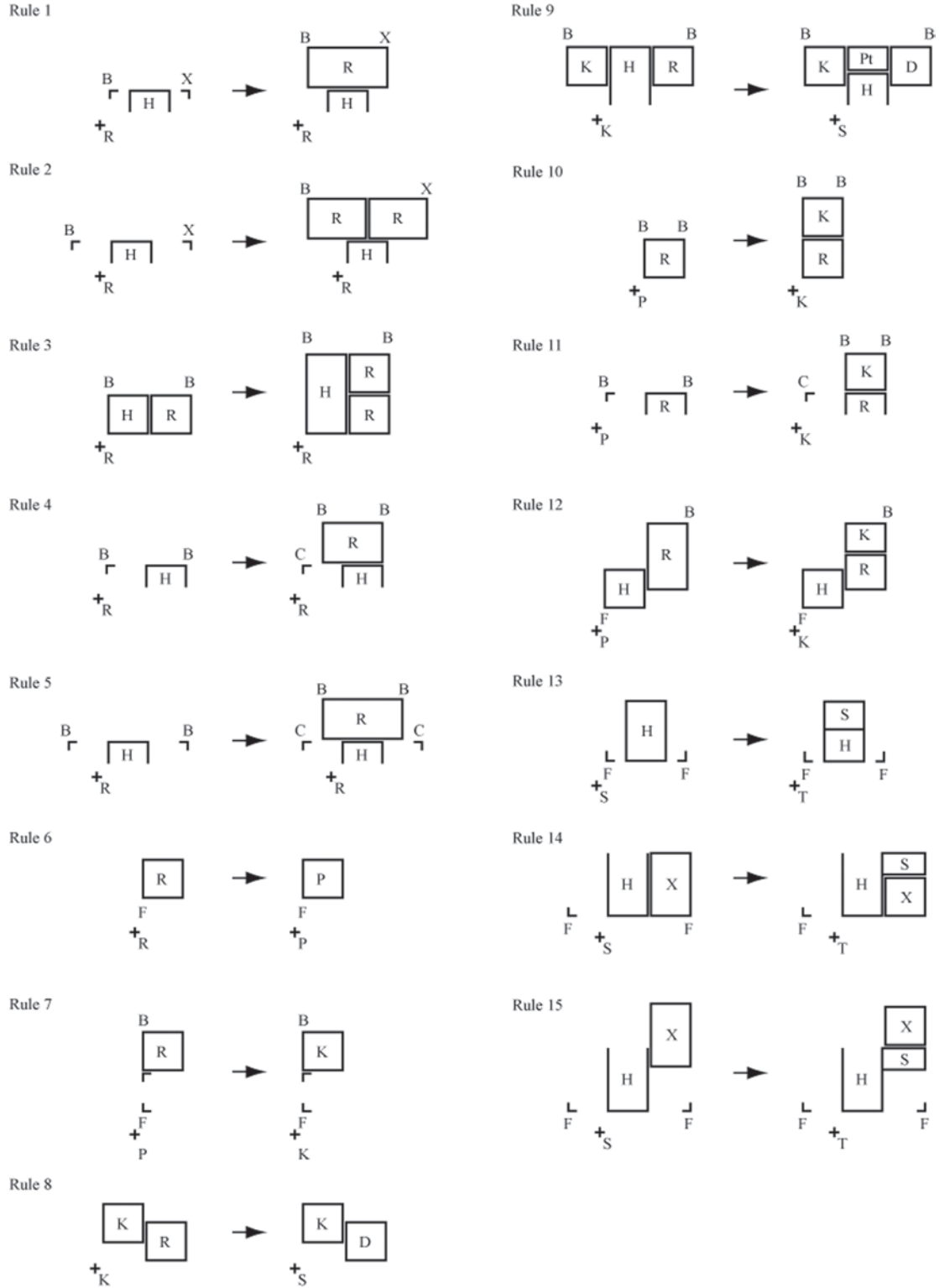


Figure 4. Shape rules to produce Queen Anne style layouts (Redrawn by the authors from Flemming (1987) . © Ulrich Flemming. Reproduced with permission)

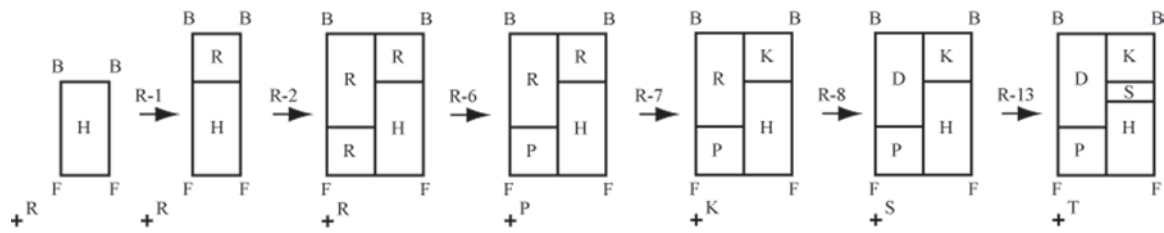


Figure 5. Derivation of a Queen Anne style layout

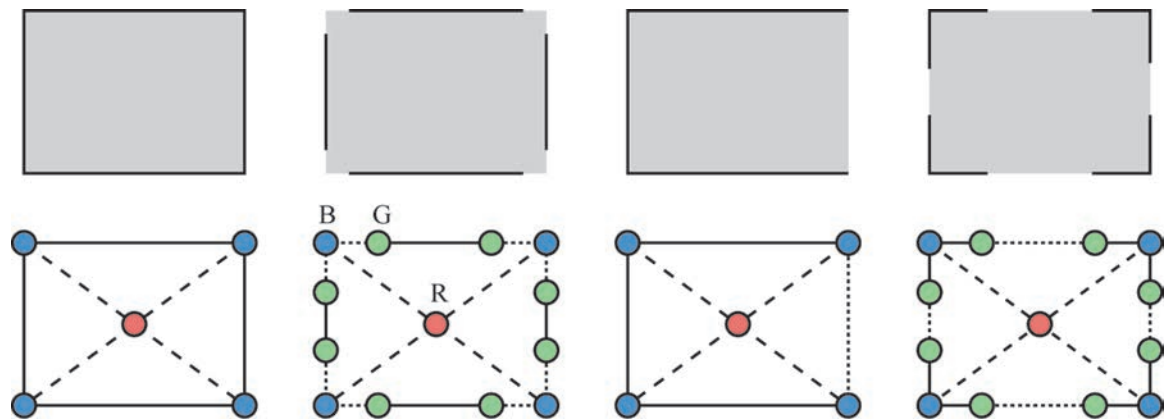


Figure 6. Example rectangular spaces and corresponding graph-like data structures

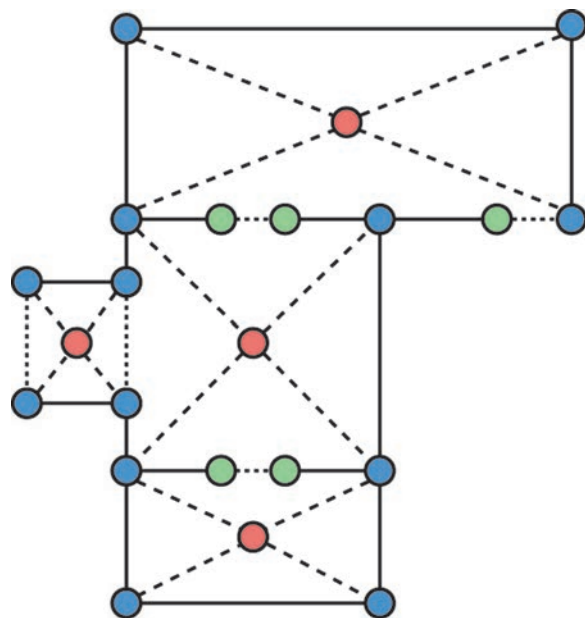


Figure 7. A layout represented by a set of graph units

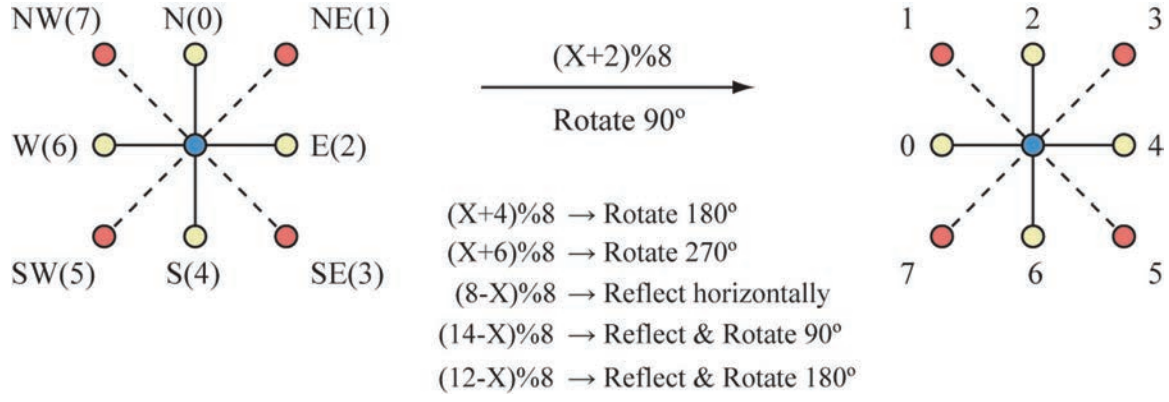


Figure 8. Transformations of the graph-like data structures

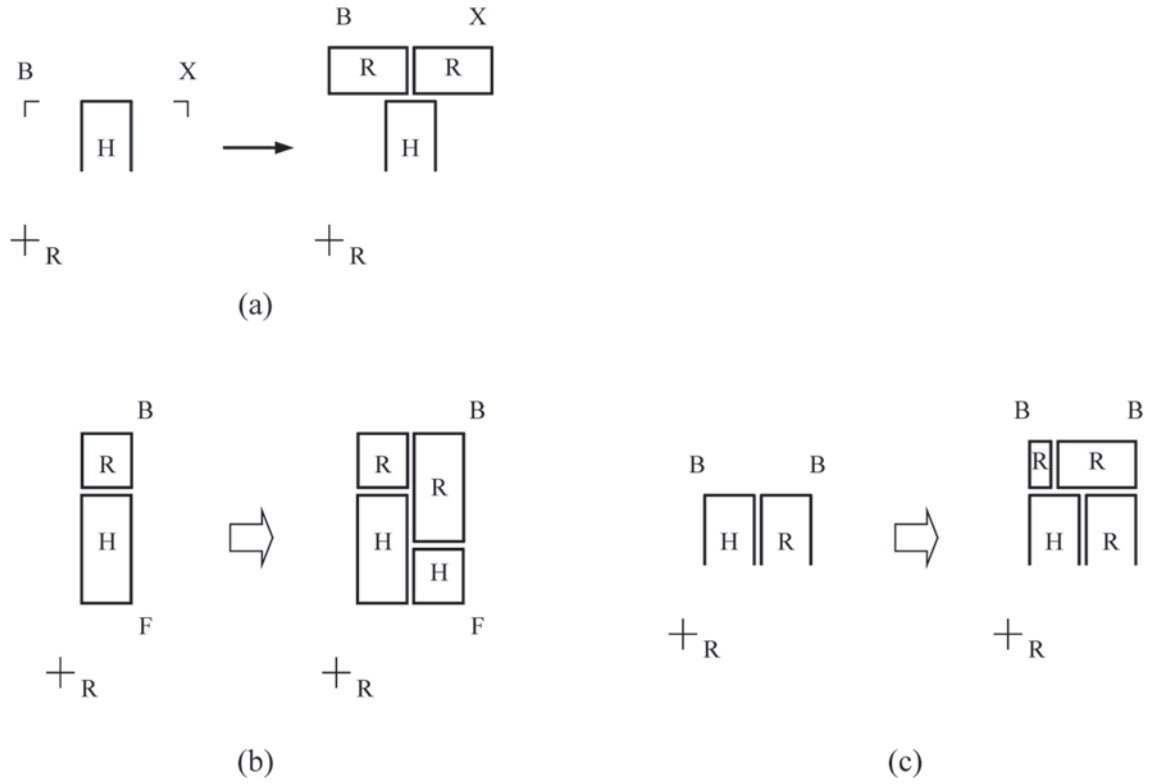


Figure 9. Two possible applications of shape rule 2

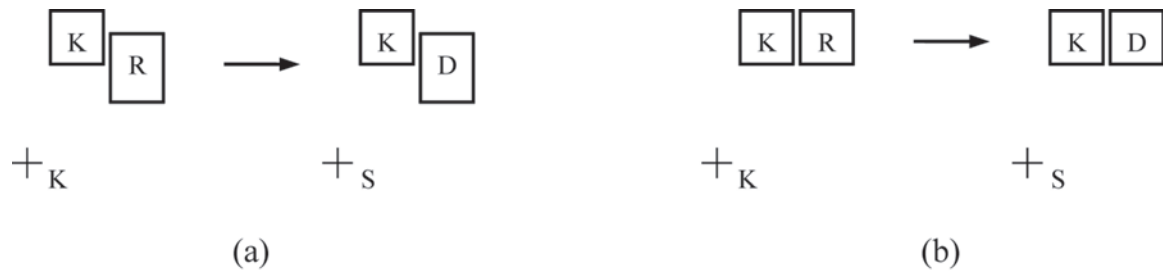


Figure 10. Two possible interpretations of shape rule 8

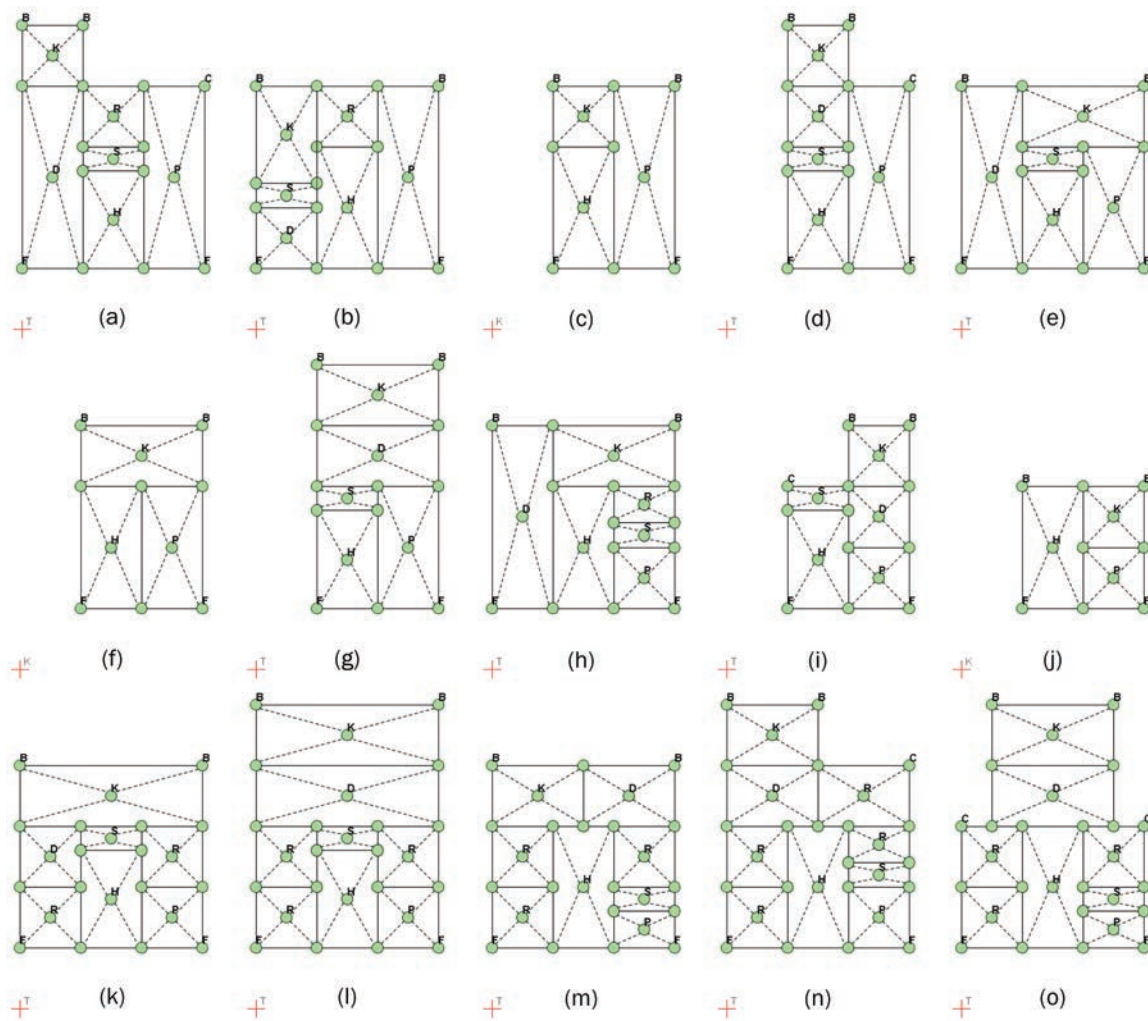


Figure 11. Sample layouts generated by the Queen Anne grammar interpreter

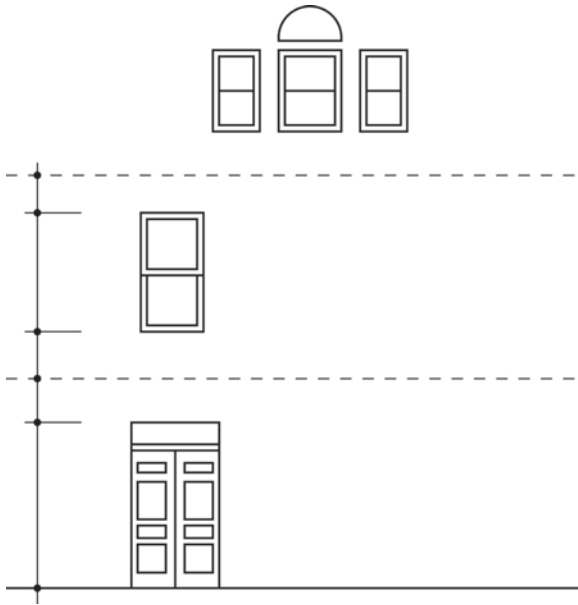


Figure 12. Windows and doors constrain the height of each story

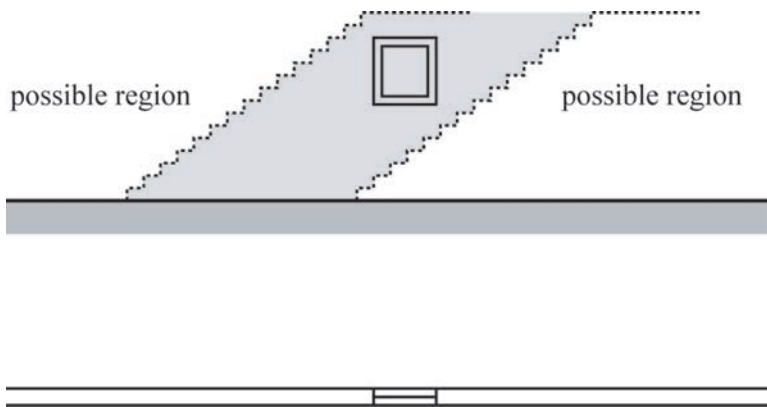


Figure 13. Window position constrains possible arrangements of a staircase



Figure 14. Initial layout estimation of Queen Anne houses by CSP

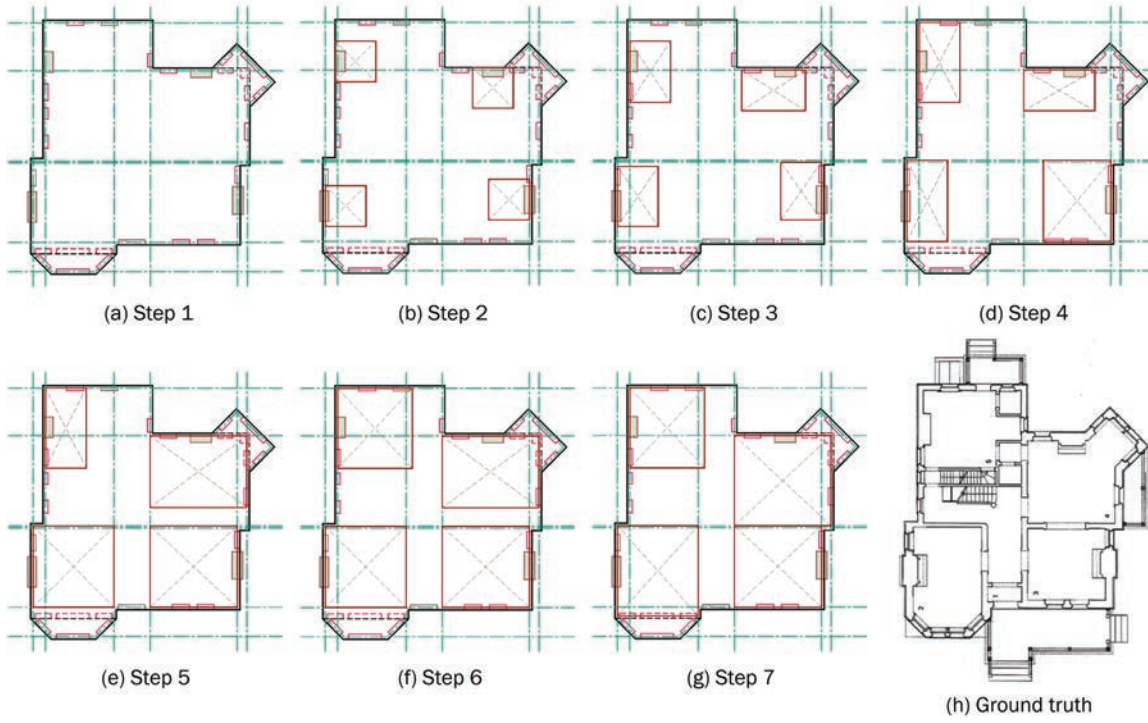


Figure 15. Manual layout derivation of House A

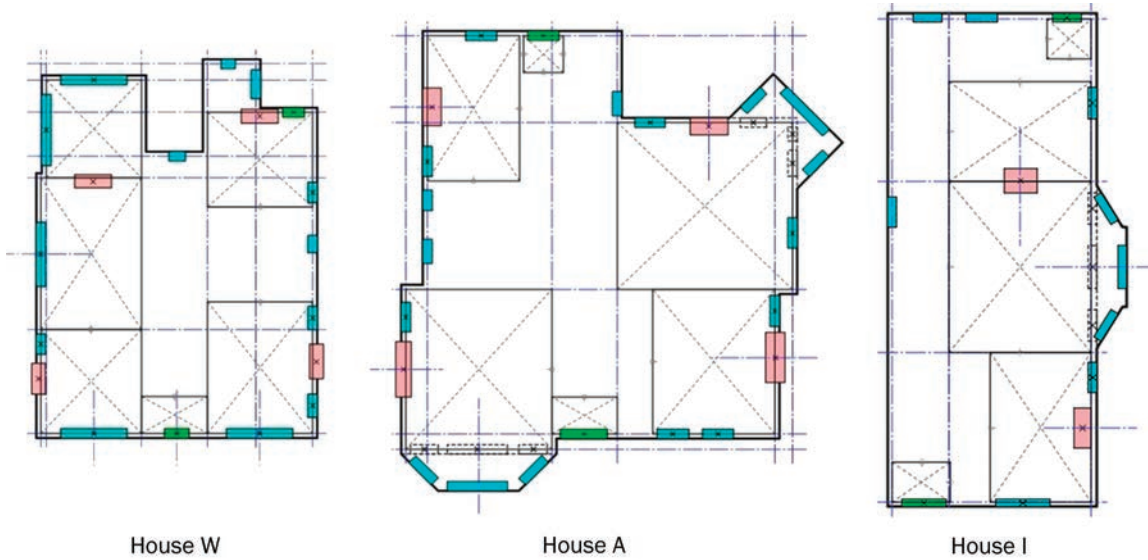


Figure 16. Results from the computer implementation of CSP

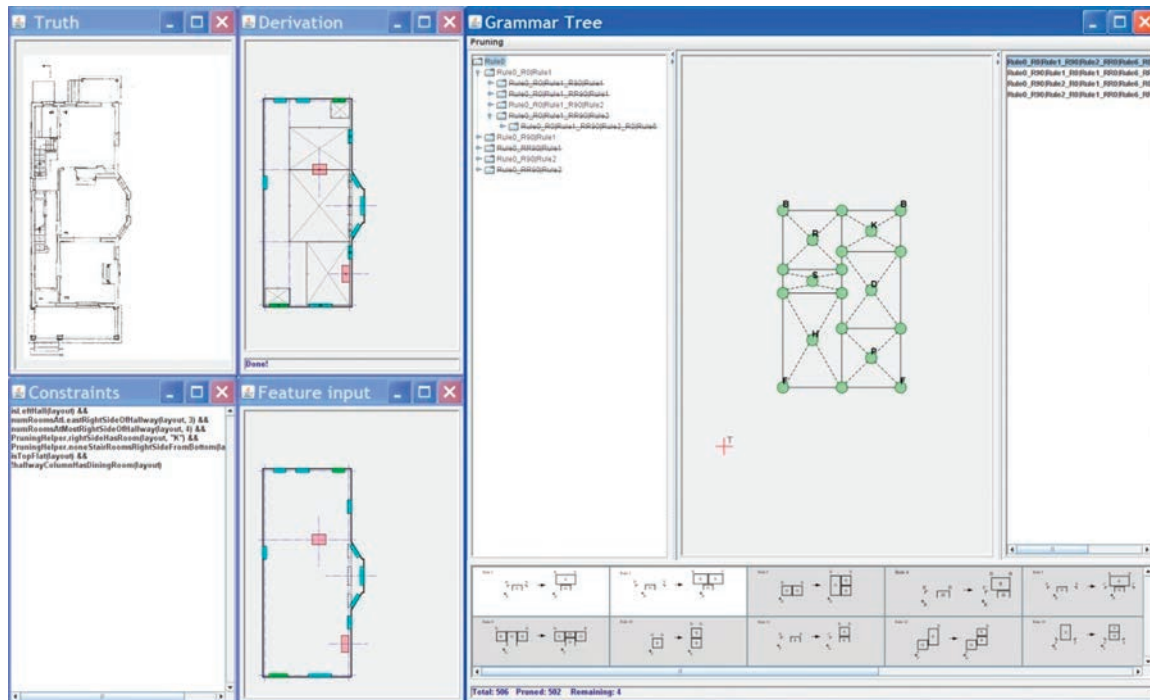


Figure 17. Screenshot of layout determination of Queen Anne houses

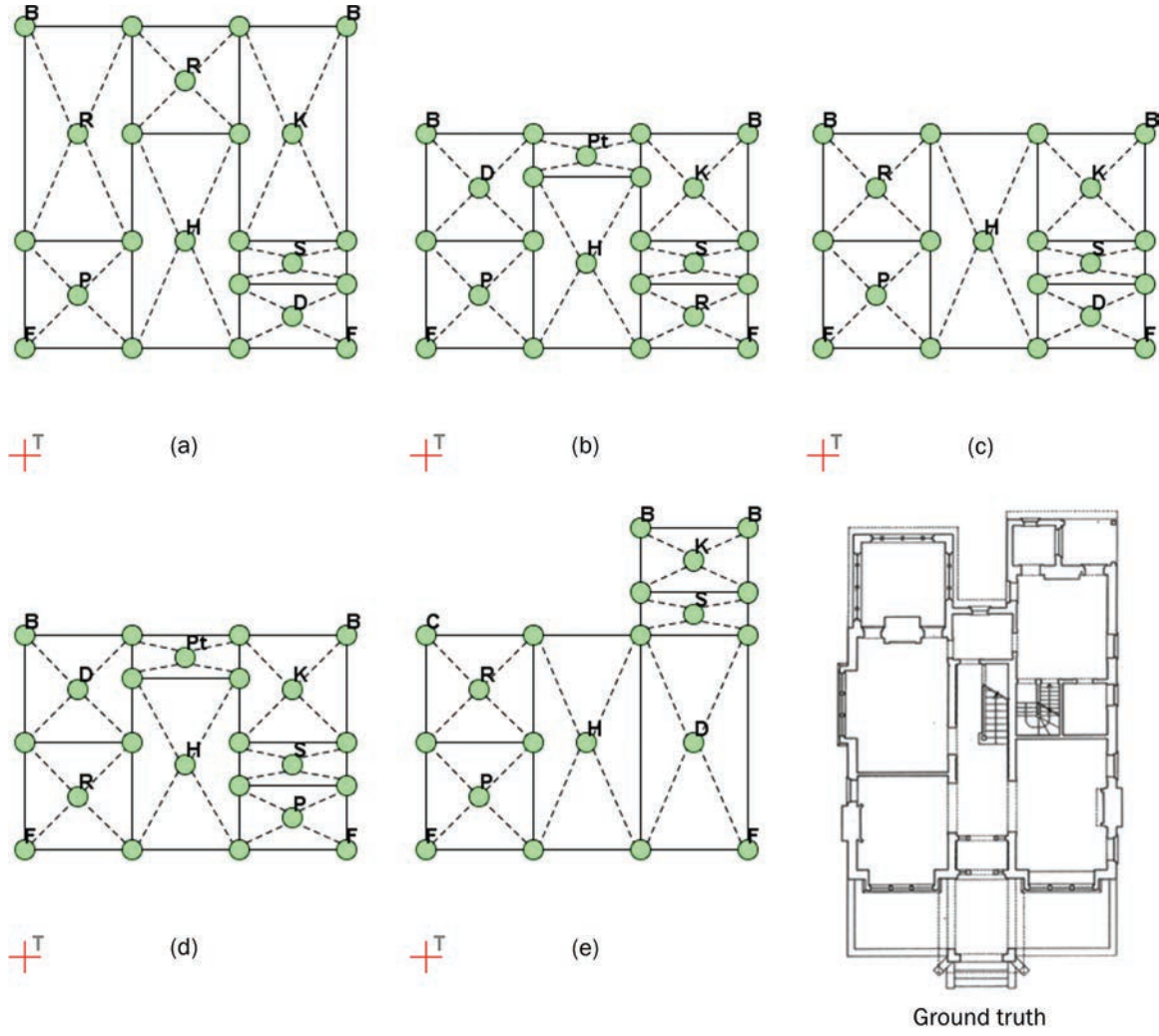


Figure 18. Layout results for House W

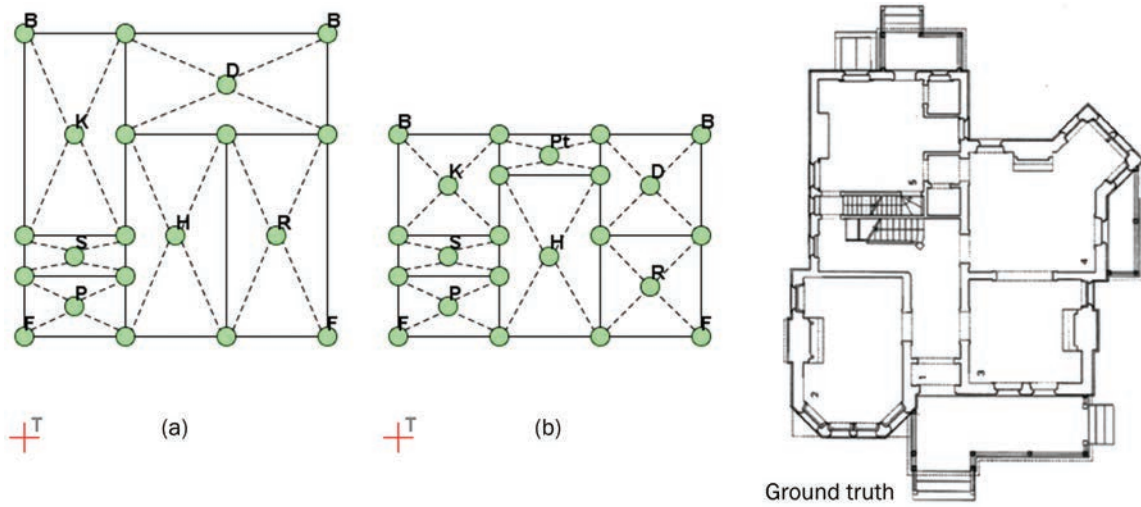


Figure 19. Layout results for House A

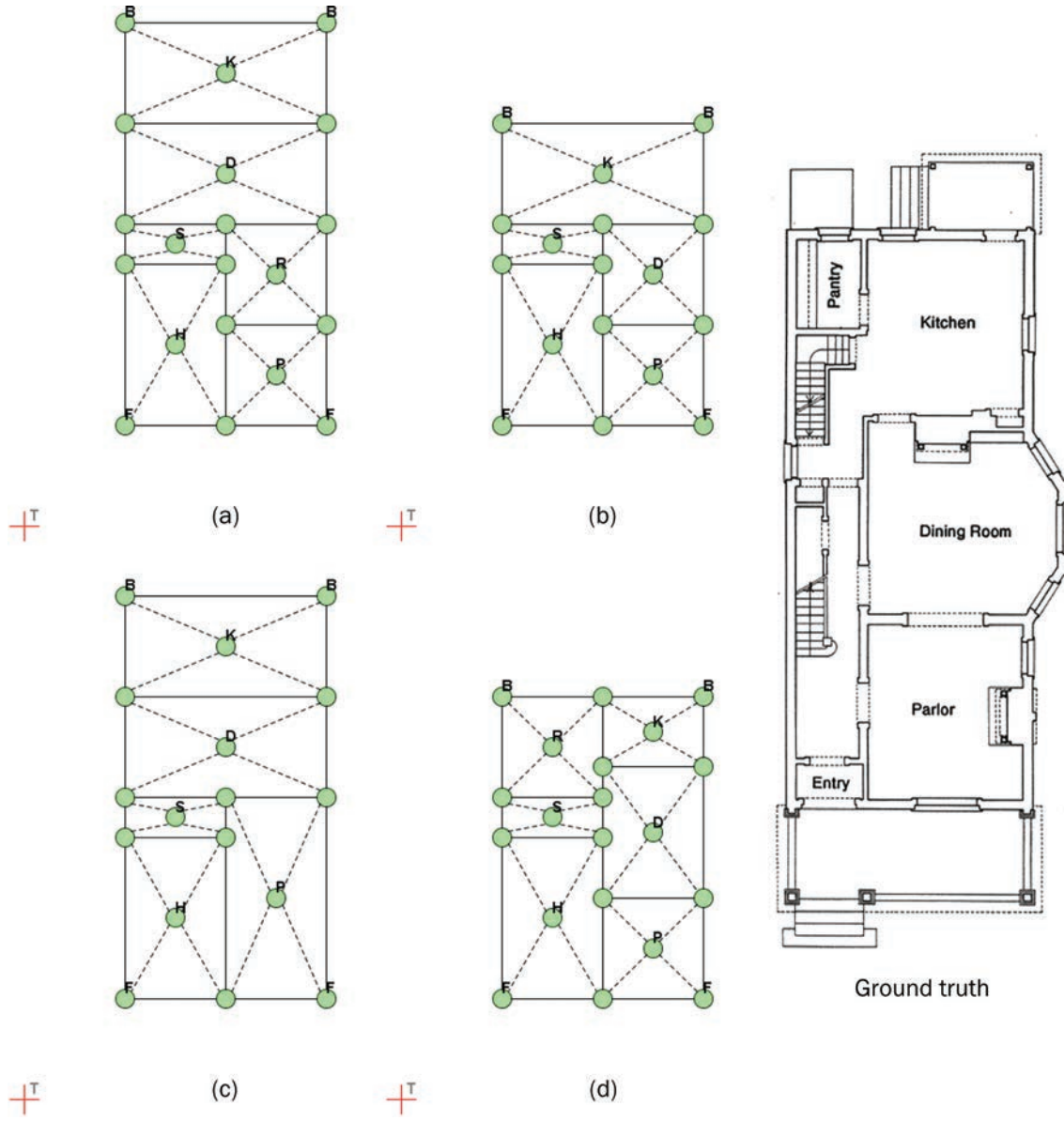


Figure 20. Layout results for House I

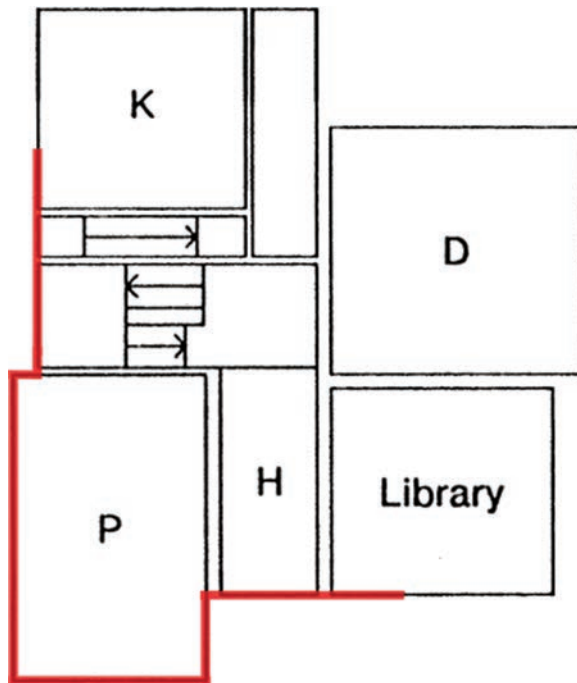
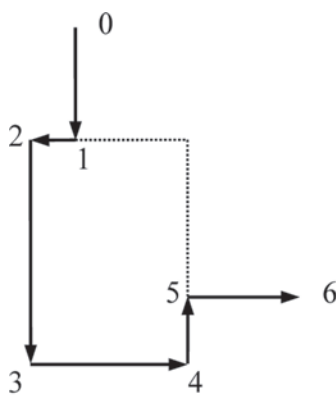


Figure 21. An example of boundary constraints



Name: *cornerBulgeRoomSeq*

Pattern: “-, +, +, +, -“

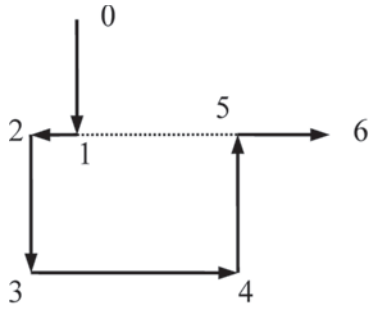
New rectangle: points {2, 3, 4}

Condition:

- Area: [min area, max area]
- Ratio: [1/3, 1]
- Other: points {1, 2, 5, 6} not collinear;
points {0, 1, 4, 5} not collinear

Point deletion: Points {2, 3, 4}

Figure 22. The corner-bulge-room sequence

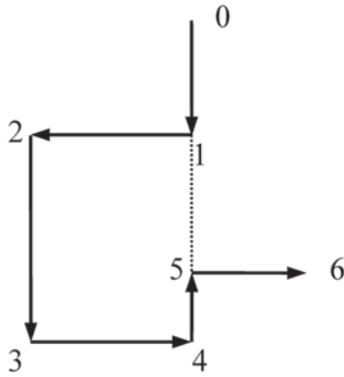


Condition:

- Other: points $\{1, 2, 5, 6\}$ collinear

Point deletion: Points $\{2, 3, 4, 5\}$

Point insertion: None



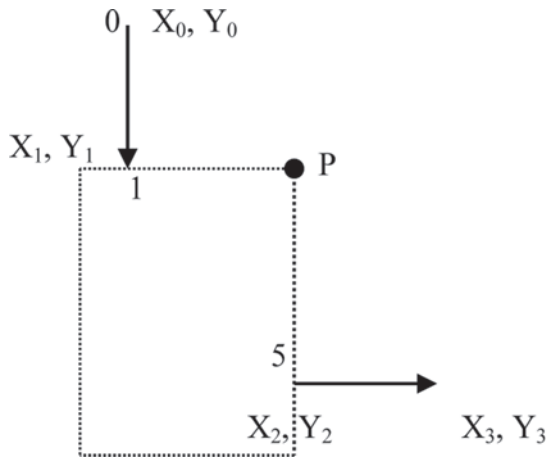
Condition:

- Other: points $\{0, 1, 4, 5\}$ collinear

Point deletion: Points $\{1, 2, 3, 4\}$

Point insertion: None

Figure 23. Variations of the corner-bulge-room sequence



if $(X_0 == X_1)$

$P = \text{new Point}(X_2, Y_1)$

else // $Y_0 == Y_1$

$P = \text{new Point}(X_1, Y_2)$

Figure 24. The new point of a corner-bulge-room sequence

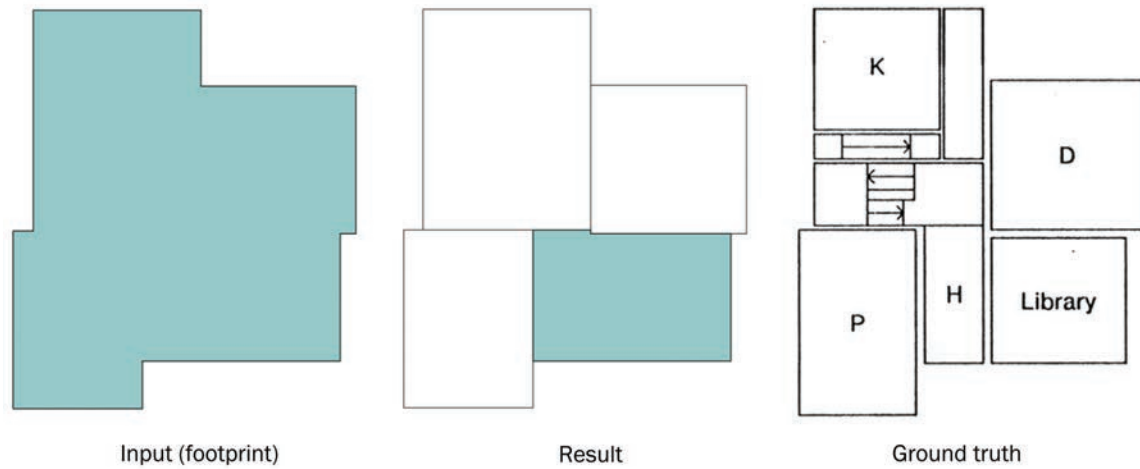


Figure 25. Application of the corner-bulge-room sequence on House A

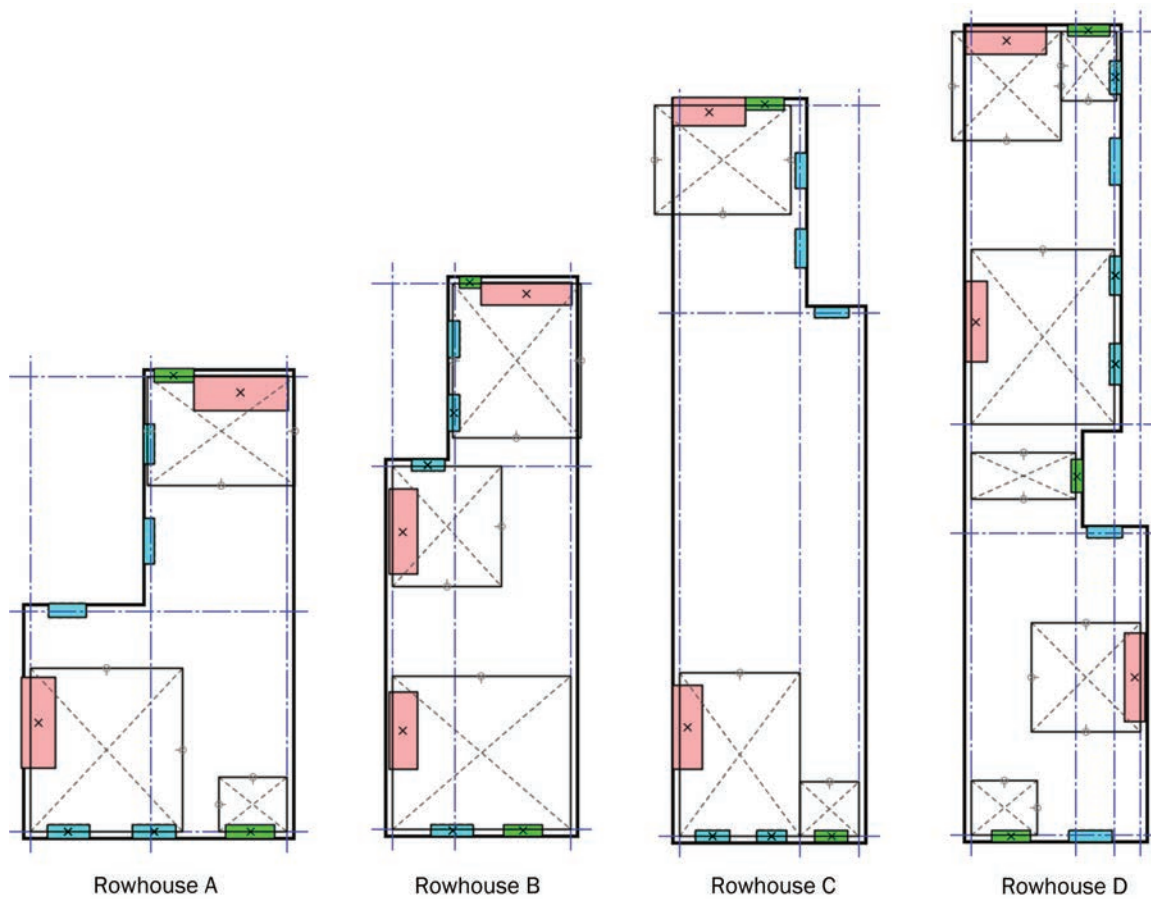


Figure 26. Results of the CSP implementation on Baltimore rowhouses

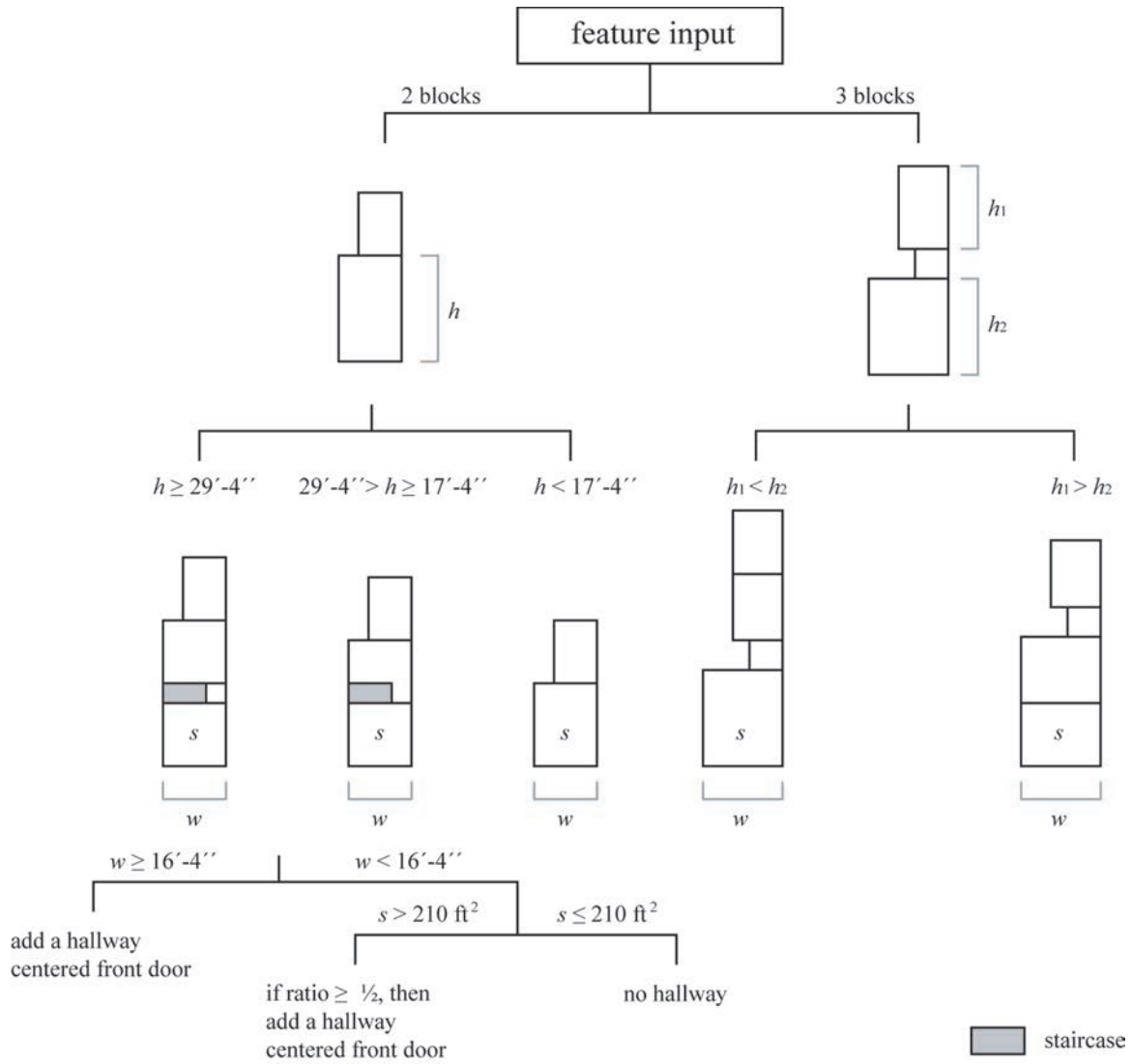


Figure 27. Space subdivision tree of Baltimore rowhouses



Figure 28. Sample layout results of Baltimore Rowhouses

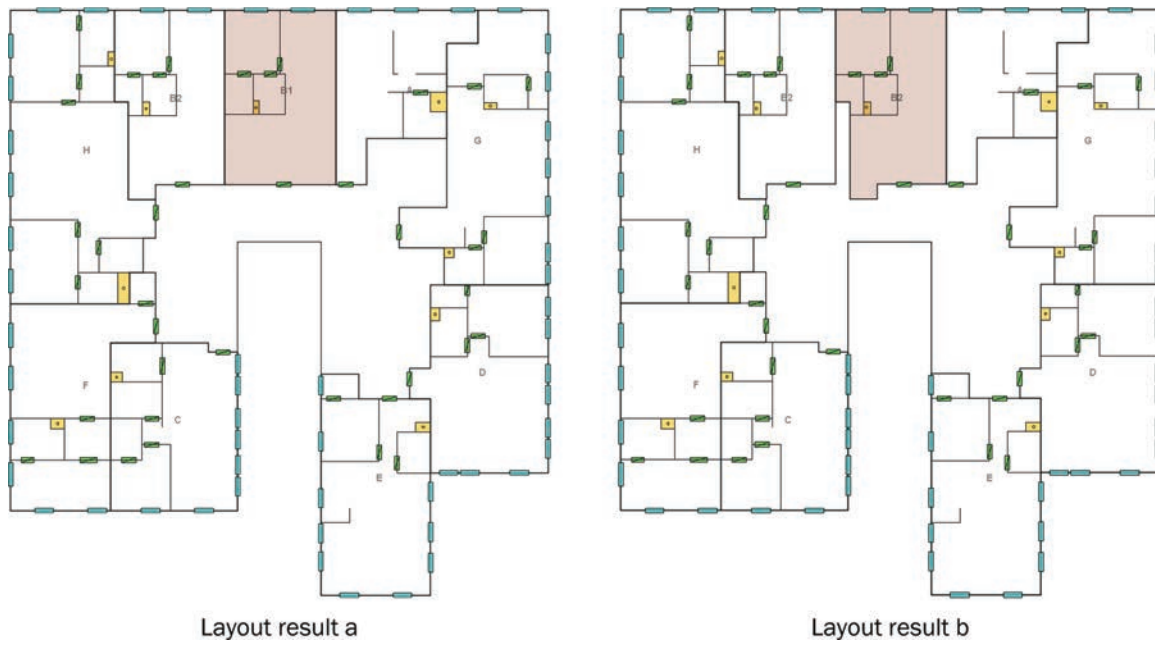


Figure 29. Two possible layout results for a highrise apartment building