Non-Monotonic Enumeration of Layout Alternative Using the LOOS Operators

Ulrich Flemming

EDRC 48-30-92

Non-Monotonic Enumeration of Layout Alternatives Using the LOOS Operators

Ulrich Flemming

Engineering Design Research Center
Carnegie Mellon University
Pittsburgh, PA 15213

October 1992

Abstract. This report develops a control function for application of the generation rules used in the LOOS system (Flemming et al. 88a) that does not keep the spatial relations between rectangles invariant. Such a control function is preferable to the one used in past versions of LOOS when certain forms of user interaction with the generation process are desired.

Contents

1. Introduction

LOOS is a system that supports the generation of layouts of rectangles to solve layout problems across domains and applications (Flemming et al. 88, 88a). The generation component of the system consists of generation rules and a control function that applies these rules in a particular sequence to enumerate alternative solutions to a layout problem. The function guarantees criteria of monotonicity that are important for the overall branch-and-bound strategy employed by the system.* This control function, however, is rather complicated and therefore hidden from users. The present report develops a non-monotonic control function for application of the LOOS rules that appears preferable when user interaction is essential.

Section 2 describes aspects of LOOS that are essential for an understanding of the following section. Section 3 develops a non-monotonic control function for applying the generation rules.

* Monotonicity in the LOOS sense will be introduced in Section 3.

2. The LOOS System

2.1 Representation of Layouts

To assure the desired applicability across applications and domains, the representation underlying LOOS allows for continuous variations in the location of rectangles (that is, it does not rely on an underlying grid) and for the handling of rectangles with varying dimensions, which may be fixed or variable for individual rectangles.

The chosen representation is an extension of the ***wall representation*** of rectangular dissections (Flemming 78 and 80). Figure 1(a) shows a rectangular dissection or rectangulation, that is, a rectangle subdivided into rectangular components without overlaps or 'holes'. A ***wall*** in such a configuration is a *maximal* sequence of collinear and connected line segments separating the rectangles from each other (the figure highlights one such wall). A *wall representation* of a rectangular dissection lists all of its walls and the sequence of rectangles bordering each wall from each of its sides. Such a representation is independent of the coordinates of the rectangles contained in the dissection, but *imposes constraints on them that vary with the representation.* It has been shown that wall representations of solutions to layout problems under topological and dimensional constraints can be systematically generated in a two-step process that first generates alternative wall representations and then computes for each of these a set of feasible dimensions that take the constraints imposed by the wall representation into account (Flemming 78).

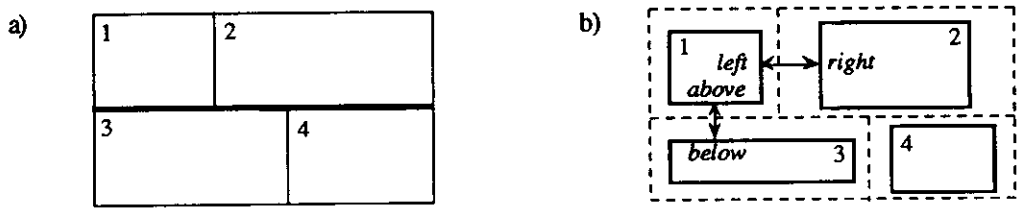

Figure 1: a) a rectangular dissection and its walls; b) a loosely-packed arrangement of rectangles that can be viewed as having the same spatial structure

Rectangular dissections constitute a very restricted class of floor plans that are practically relevant for only a limited range of problems. In order to increase the applicability of LOOS, the wall representation has been extended to include *loosely-packed arrangements of rectangles* (LPARs), which maintain non-overlap between components, but allow for holes and irregular boundaries; an example is shown in Figure 1(b). Note that a densely-packed arrangement is a LPAR; that is, rectangular dissections form a proper subset of the set of LPARs.

Observe that the wall representation of the plan in figure 1(a) implies left/right and above/below relations between the rectangles in the plan that are found also in the LPAR shown in figure 1(b); wall representations, or their equivalents, can consequently also be used to represent certain spatial relations in LPARs, *provided these LPARs can be derived from a dissection by shrinking some rectangles, or conversely by expanding the rectangles in the LPAR until they touch rectangles on all four sides and thus form a dissection.* LOOS uses the equivalent of wall representations to represent LPARs in terms of left/right and above/below relations. These relations are called *spatial* in the following. The set of spatial relations in an LPAR is called the *spatial structure* or *topology* of the LPAR. In some contexts, it is convenient to call a spatial relation also a *direction.*

Note that the spatial relations are transitive and do not have to be specified explicitly for every pair of rectangles; if, for example, a rectangle a is to the right of a rectangle b along a vertical wall, and a rectangle c is to the right of b along a second wall, c is also to the right of a. Diagrams like

the one shown in Figure 1(b) should be read in this sense. Note also that dissections are included in the LOOS representation because they represent a special case of an LPAR.

Certain sets of relations that can occur in a LPAR cannot be represented directly by a wall representation. These are the relations that occur around *non-trivial holes*, which are gaps that cannot be eliminated by extending the rectangles in an arrangement until they touch other rectangles (Flemming 89). An example is shown in Figure 2(a). LOOS circumvents this difficulty by representing non-trivial holes *explicitly as rectangles* that are marked by a special label to distinguish them from *regular* rectangles [see Figure 2(b)]. The resulting *marked structures* are able to represent *every set of spatial relations that are realizable in an LPAR* (Flemming 89) and form the basis for LOOS. In the following, the term *structure* always refers to a marked structure.

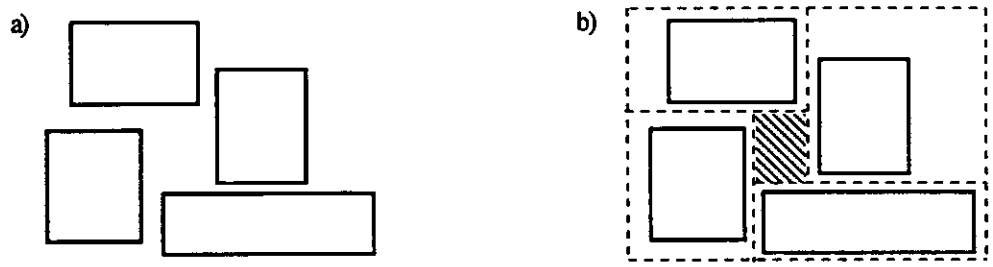

Figure 2: a) A LPAR containing a non-trivial hole; b) representation of a non-trivial hole by a special rectangle (shown hatched)

A *special rectangle* represents a non-trivial hole in a layout. Such rectangles must satisfy the following well-formedness conditions.

Well-formedness conditions for special rectangles:

(a) The walls bordering a special rectangle must form a counter-clockwise (Figure 3a) or clockwise (Figure 3b) turning pinwheel.

(b) Let the rectangles bordering the walls from the other side be denoted by a_h, r_i, b_j, and l_k, with $h = 1, ..., A$; $i = 1, ..., R$; $j = 1, ..., B$; $k = 1, ..., L$, as shown in Figure 3. In case a), neither of the rectangles a_1, r_1, b_1 or l_1 is special. In case b), neither of the rectangles a_A, r_R, b_B, or l_L is special.

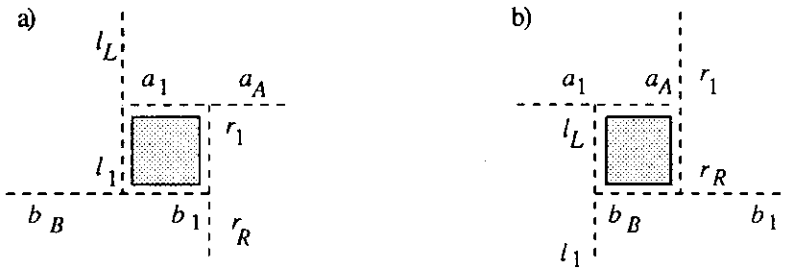

Figure 3: Rectangles surrounding a special rectangle

The LOOS representation assumes, without loss of generality, that internal walls never cross each other; that is, if two internal walls meet at a common point, they form a T-connection (see Figure 1). Crossing walls can be made to conform to this convention by the introduction of a dummy segment separating one wall at the common point into two.

In addition to the spatial relations holding between the rectangles it contains, the LOOS representation records explicitly the *upper and lower bounds for the spatial coordinates* of each rectangle implied by the relations. These *dimensional attributes* are updated after each change to the structure. An additional attribute indicates the *type of domain object* represented by a rectangle. A marked structure with these attributes is called a *configuration*.

2.2 Operators

This section describes the most important operators or rules provided by LOOS to generate and modify configurations.

a. Generation Rules

Generation rules can be used to generate structures from structures by adding one rectangle at a time, starting with a suitable initial structure, which may represent nothing more than the enclosing rectangle. Alternative structures are generated if more than one possibility for adding individual rectangles is pursued. The rules shown in Figure 4 are the two generation rules provided by LOOS. We always assume that the rectangles bordering a horizontal wall from above are ordered from left to right, and the rectangles bordering the wall from belong are ordered from right to left. Similarly, we assume that the rectangles bordering a vertical wall from the right are ordered from top to bottom, and the rectangles bordering the wall from the left are ordered from bottom to top.

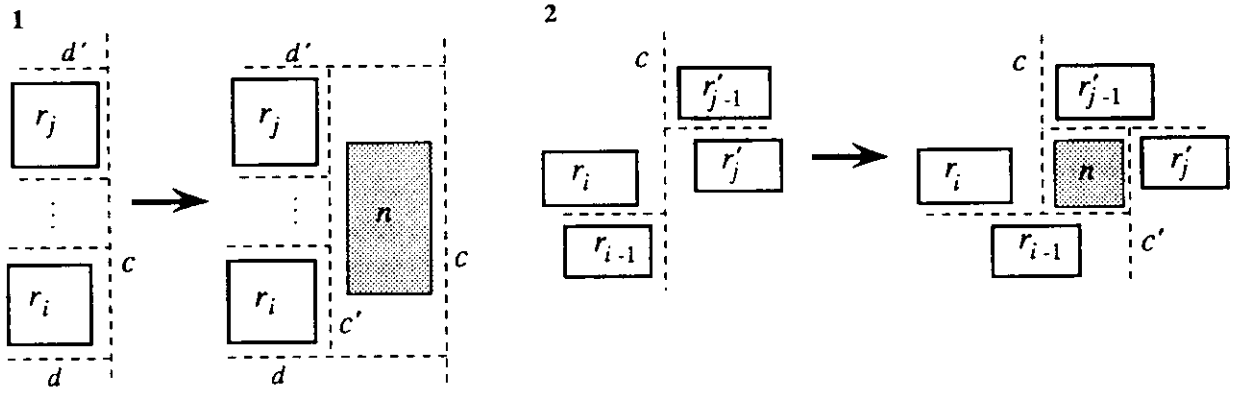

Figure 4: Generation rules

The rules are specified in the figure as *recursive rewrite rules* consisting of a *left-hand side* (LHS) and a *right-hand side* (RHS) both of which describe substructures of a structure. A rule can be *applied* to a structure, s, if s contains the LHS. The application consists of replacing the LHS in s by the RHS or of 'rewriting' s according to the rule. For example, the LHS of generation rule 1 shows a wall, c, and a sequence of rectangles r_i ..., r_j, $j \geq i \geq 1$, adjacent to c from one side; r_i may be the first rectangle and r_j the last rectangle along c on that side. The RHS shows how r_i, ..., r_j can be pushed away from c to create space for the insertion of a new rectangle, n, and a new wall, c'. In a similar way, rule 2 shows

how a wall, *c*, can be split into two to create a pinwheel configuration of walls and to insert a new rectangle in the resulting hub. This rule must be used, for example, for the creation of non-trivial holes.

Both rules can be applied in rotated or reflected versions. We will show below that if these transformations are taken into account, every structure can be generated by applying these rules in sequence, starting with a suitable initial structure.

b. Propagation Rules

The dimensional bounds for a newly inserted rectangle depend on those of the surrounding rectangles; conversely, the dimensional bounds of these rectangles are likely to become tighter through the insertion. The *propagation rules* compute the bounds for the newly inserted rectangle and propagate the resulting changes recursively through the configuration.

In (Flemming et al. 88), propagation rules for rectangles representing domain objects with fixed dimensions are described. These rules *orient* an object if its longer side fits only in one direction. Otherwise, they leave the object un-oriented. That is, orientation is not treated in LOOS as an explicit design variable, which reduces the combinatorics of search.

More recently, propagation rules that can handle rectangles with variable dimensions have been added to the system.

c. Test Rules

The configurations generated by LOOS are evaluated by a tester based on a flexible collection od test rules. Propagation and testing are not addressed in this report.

3. Non-Monotonic Enumeration of Alternatives

The basic difference between the two generation rules shown in Figure 4 is that rule 1 leaves the spatial relations between the rectangles already allocated unchanged, while rule 2 changes the relations between rectangles r_{i-1} and r'_{j-1} (and, by transitivity, the relations between further rectangles) from right/left to above/below (if applied in the orientation shown in Figure 4; similar changes occur for different orientations). As stated in the introduction, LOOS applies the rules under a rather complicated control regime, which uses rule 2 for the insertion not only of non-trivial holes, but also of 'place-holders' for regular rectangles so that the relations between these rectangles remain unchanged by the insertion of further regular rectangles (Flemming 89). This monotonicity of the spatial relations during generation is important because it enables LOOS to evaluate and consequently prune intermediate solutions with certainty as *most criteria or constraints that are not satisfied by an intermediate solution can never be satisfied by a structure generated from it.*

However, a less complex control regime may suffice in many applications, one that is still able to generate all possible structures of a given number of rectangles, including non-trivial holes, but is allowed to change spatial relations in the process. This eliminates the need for specific place-holders the significance of which may be difficult to appreciate by users unwilling to immerse themselves deeply in the technical complexities connected with the original LOOS control strategy. The present section develops such a simpler, non-monotonous control regime.

The precise specification of this regime and the proof that it is, in fact, able to produce all possible structures require a greater level of technicality. In order to be able to specify applications of generation rules independent of a specific orientation, we use the functions **I***, **P***, and **N*** to relate a spatial relation or direction * to the other relations or directions. If * is a direction, **I***, **P***, and **N*** should be read ***inverse***, ***last***, and ***next direction of*** *, respectively. The functions are defined in the following table.

*	I*	P*	N*
A (above)	B	L	R
B (below)	A	R	L
R (right)	L	A	B
L (left)	R	B	A

Table 1: Functions **I***, **P***, and **N***

For example, if a wall c borders a rectangle r in direction *, we can always say that r borders c in direction **I*** independently of *. Or for an application of rule 1 in any orientation, if wall c borders rectangles r_i, ..., r_j in direction * before the application, c will border the newly inserted rectangle, n, in direction * after the application, while rectangle r_i will be in direction **N*** of rectangle r_{i+1} before and after the application. * will be called the ***direction of application*** of rule 1. The direction of application of rule 2 is determined in the same way by the relation between wall c and rectangle r_i; figure 3 specifies both rules for * = R.

If a rule is specified in terms of a generic application direction * (that is, the spatial relations between the rectangles and walls that appear in its LHS and RHS are defined in terms of * through the functions in Table 1), applications in specific orientations can be found by associating values **A**, **B**, **R**, or **L** with *. These different orientations can also be viewed as applications of the rule under different *rotations*, and for rule 1, no further variations are needed.

It turns out that for rule 2, applications are needed also under different *reflections*. These are more difficult to reduce to a single parametric form. We therefore specify explicitly rule 2 in two

reflected versions, 2a and 2b, as shown in Figure 5. Rule 2a generates a counterclockwise and rule 2b a clockwise turning pinwheel of walls. It is understood that each version can be applied in any rotation.

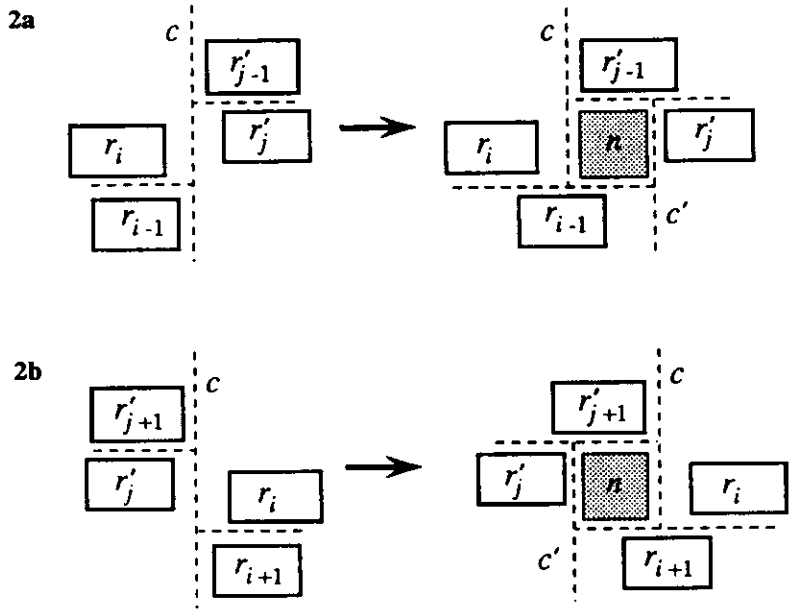

Figure 5: Rules 2a and 2b

We prove in the following that rules 1 and 2 are sufficient to generate all possible marked structures containing a given set of rectangles. This result, by itself, is not surprising. But the proof itself is important because it forms the basis for our non-monotonic control regime, in which the sequence of rule applications that create any structure is *unique, including the insertion of special rectangles.*

Theorem:

If $G_{n,h}$ is a marked structure with n regular and h special rectangles, $h \geq 0$, where the regular rectangles are labeled arbitrarily from 1 to n, there exists a sequence of applications of rules 1 and 2 that generates $G_{n,h}$ and inserts the regular rectangles in the order given by their labels.

Proof: The proof is by induction on n, where the structure $G_{1,0}$ containing one regular and no special rectangle serves as starting structure. $G_{1,0}$ contains exactly four external and no internal walls.

If $n = 2$, an application of rule 1 to each wall in $G_{1,0}$ with $r_i = r_j$ = rectangle 1 generates the four possible structures $G_{2,0}$. No other structure with 2 regular rectangles exists (that is, a structure with 2 regular rectangles cannot contain any special rectangle).

Let now $n > 2$, and let the theorem be true for all structures $G_{n-1,k}$. Suppose $G_{n,h}$ is an arbitrary structure with n regular rectangles and h special rectangles; label the regular rectangles arbitrarily from 1 to n.

Consider rectangle n. It must border a wall in each of the four directions, at least one of which must be internal because $n > 2$. The configuration of these four walls must belong to at least one of the cases shown in Figure 6 (the crossing walls at the ends of c are meant to indicate that c may form a T-joint at that point in one of two ways).

Case 1: There exists an internal wall, c', so that n is the only rectangle bordering c' on its side. Let * be the direction in which n borders c'. Denote the ordered rectangles on the other side of c' by $r_1, ..., r_m$. If neither r_1 nor r_m is a special rectangle, apply rule 1 backwards in an appropriate orientation to remove wall c' and n. This generates a well-formed structure $G_{n-1,h}$. If r_1 or r_m is a special rectangle, we have the structures shown in Figure 7. Neither of the rectangles a, b or c can be special, and a backward application of rule 2a or 2b removes the special rectangles, generating a well-formed structure $G_{n,k<h}$, from which n can be removed as before.

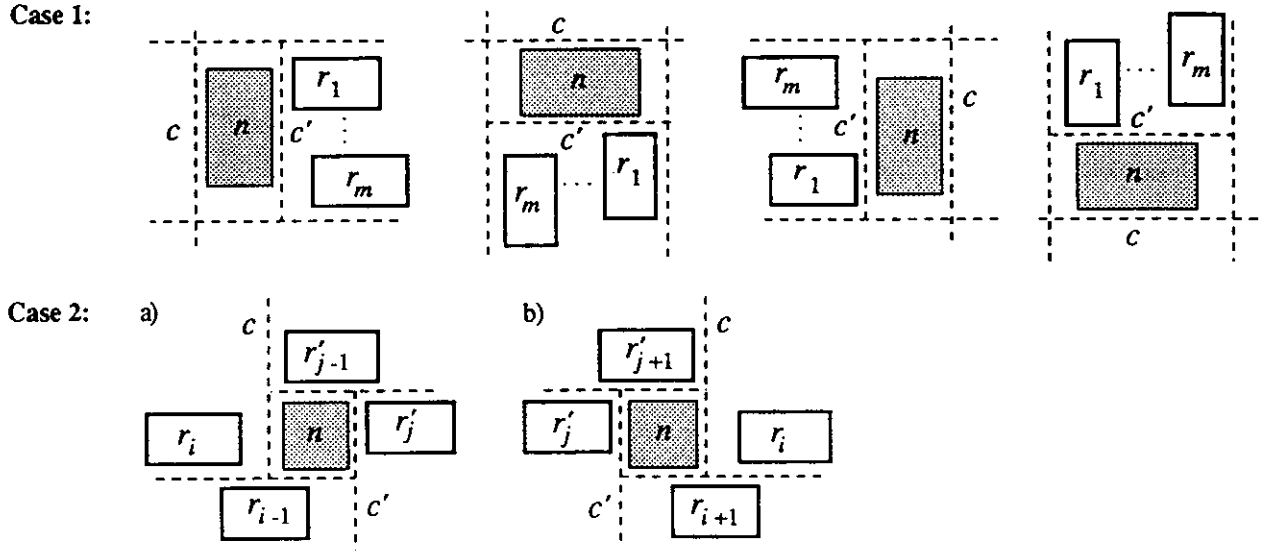

Figure 6: Cases 1 and 2

Case 2: There exists no internal wall so that n is the only rectangle bordering the wall on its side. In this case, the four walls bordering n must form a pinwheel as shown in figure 6. If neither r_i nor r'_j is a special rectangle, apply rule 2a (in case 2a) or 2b (in case 2b) backwards (in the orientation shown in figure 4) to remove wall c' and n. This generates a well-formed structure $G_{n-1,h}$. If, in case 2a, r_i or r'_j is a special rectangle, it must be a clockwise turning hub that can be removed by a backward application of rule 2b, generating a structure $G_{n,k<h}$, from which n can be removed as before. In case 2b, special rectangles can be removed in a similar way.

Thus, we can always generate a well-formed structure $G_{n-1,k\leq h}$ by backward application of the generation rules. By hypothesis, there exists a sequence of rule applications that generates $G_{n-1,k}$ by inserting rectangles 1, ..., n- 1 in that order. Clearly, $G_{n,h}$ can be generated from $G_{n1,k}$ by an application of rule 1 or 2 that inserts n, followed, possibly, by one or two applications of rules 2a or 2b that insert special rectangles. ♦

We now specify the function *non_monotonic_expand* that accepts as input a structure and inserts a new rectangle in all possible ways, including special rectangles. The generated structures are collected in a set, which is the function's output. The function is based on the above proof; that is, it reverses the reductions used in the induction to determine the required forward applications of the generation rules. The application of the propagation rules is left out in the specification.

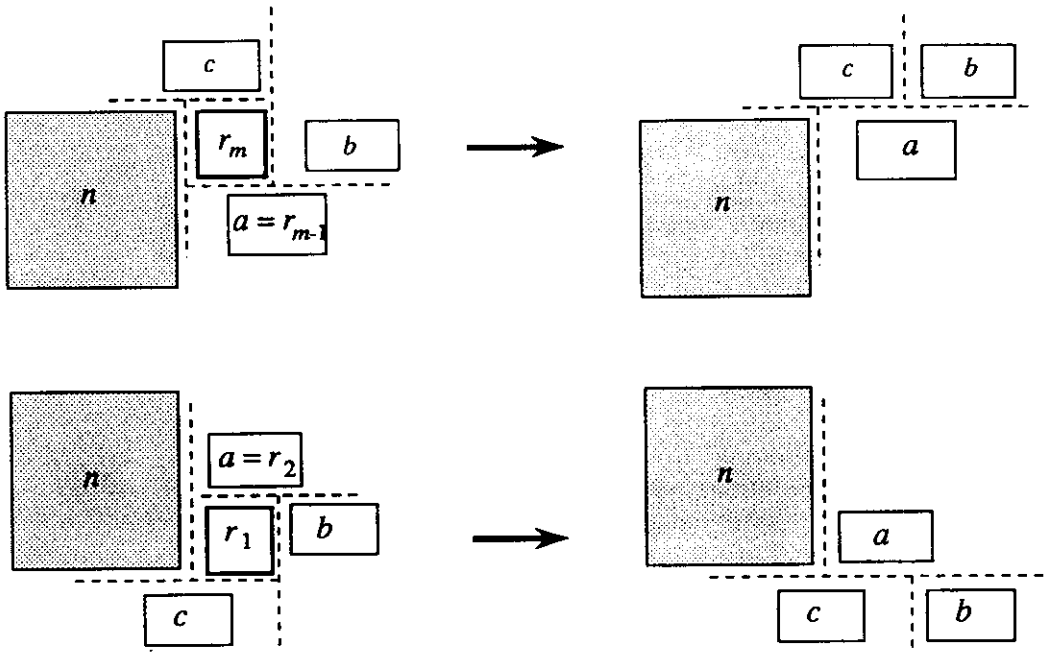

Figure 7: Special vertices at the corners of n

In addition, the procedure guarantees that each structure can be generated in only one way; that is, it avoids generating duplicates. Observe in this connection that the cases shown in Figure 6 are unique (that is, given n, there is only one way of selecting walls c and c' and rectangles r and r') except when, in case 1, n is also the only rectangle bordering c on its side (in this case, c and c' can be reversed to result in a different subcase of case 1). In case 2, uniqueness is based on the convention that for pinwheel configurations, we select only vertical walls as c and c' and label them as shown in the figure.

The specifications rely on the predicate

special (r)

which returns **T** if rectangle r is special and **F** otherwise, and on three functions:

rule_1 ($G, r, r', *, R$),

which returns a structure generated by applying rule 1 to structure G in direction * with pivots r and r' to insert rectangle R;

rule_2a ($G, r, r', *, R$),

which returns a structure generated by applying rule 2a to structure G in direction * with pivots r and r' to insert rectangle R; and

rule_2b ($G, r, r', *, R$),

which returns a structure generated by applying rule 2b to structure G in direction * with pivots r and r' to insert rectangle R.

The symbol s denotes an instance of a special rectangle, and the symbol E denotes the external rectangle that borders each of the external walls from a unique direction.

```
function non_monotonic_expand (G_{n,k}, R)
;Input: G_{n,k}   : a structure with n regular and k special rectangles, n ≥ 1, k ≥ 0
;       R         : rectangle to be inserted
begin
X ← ∅
for every wall c in G_{n,k}
        begin
        for every direction * so that c borders rectangles r_1,..., r_m in direction *
                ;; for every wall, there are two such directions!
                unless r_1 = E
                begin
                for i = 1, ..., m
                        unless special ( r_i )
                        begin
                        for j = i, ..., m
                                unless [ special ( r_j ) or ( i = 1 and j = m and (* = L or * = B) ) ]
                                begin   ;; generate all applications of rule 1 to c in direction *
                                G_{n+1,k} ← rule_1 ( G_{n,k}, r_i , r_j , *, R)
                                X ← X ∪ {G_{n+1,k} }
                                        ;; followed by insertions of special vertices
                                Y ← rule_2a_all ({G_{n+1,k} }, r_j , P*, s)
                                X ← X ∪ Y
                                X ← X ∪ rule_2b_all (Y ∪ {G_{n+1,k} }, r_i, N*, s)
                                end
                        if ( i > 1 and * = R)
                                begin   ;; generate all applications of rule 2a to c in dir. *
                                Y ← rule_2a_all ( {G_{n,k} }, r_i, *, R)
                                X ← X ∪ Y
                                        ;; followed by insertions of special vertices
                                Z ← rule_2b_all (Y , r_i, N*, s)
                                X ← X ∪ Z
                                X ← X ∪ rule-2b_all (Z ∪ Y , r'_j, P*, s) ;; r'_j as in Fig. 5
                                end
                        if ( i < m and * = L)
                                begin   ;; do the same for rule 2b
                                Y ← rule_2b_all ( {G_{n,k} }, r_i, *, R)
                                X ← X ∪ Y
                                Z ← rule_2a_all ( Y , r_i, P*, s)
                                X ← X ∪ Z
                                X ← X ∪ rule_2a_all ( Z ∪ Y , r'_j, N*, s) ;; r'_j as in Fig. 5
                                end
                        end
                end
        end
return X        ;; the set containing all structures G_{n+1,h≥k}
                ;; that can be generated from G_{n,k} , with R as the new rectangle
end
```

Functions ***rule_2a_all*** and ***rule_2b_all*** called by ***non_monotonic_expand*** are specified as follows:

```
function rule_2a_all ( X, r, *, R)

;Input: X      : a set of structures
;      r      : rectangle contained in each G ∈ X
;      *      : application direction
;      R      : rectangle to be inserted
begin
X ← ∅
for every G ∈ X
        find wall c bordering r in direction * in G
        unless [ special ( r ) or r is the first rectangle bordering c in direction I* ]
        begin
        let r'1, ..., r'm  be the rectangles bordering c  in direction *
        unless ( m = 1 )
                begin
                for j = 2, ..., m
                        unless special ( r'j)
                        X ← X ∪ { rule_2a ( G, r, r'j , *, R)}
                end
        end
return X
end
```

```
function rule_2b_all ( X, r, *, R)

;Input: X      : a set of structures
;      r      : rectangle contained in each G ∈ X
;      *      : application direction
;      R      : rectangle to be inserted
begin
X ← ∅
for every G ∈ X
        find wall c bordering r in direction * in G
        unless [special ( r ) or r is the last rectangle bordering c in direction I* ]
        begin
        let r'1, ..., r'm  be the rectangles bordering c  in direction *
        unless ( m = 1 )
                begin
                for j = 1, ..., m-1
                        unless special ( r'j)
                        X ← X ∪ { rule_2b ( G, r, r'j , *, R)}
                end
        end
return X
end
```

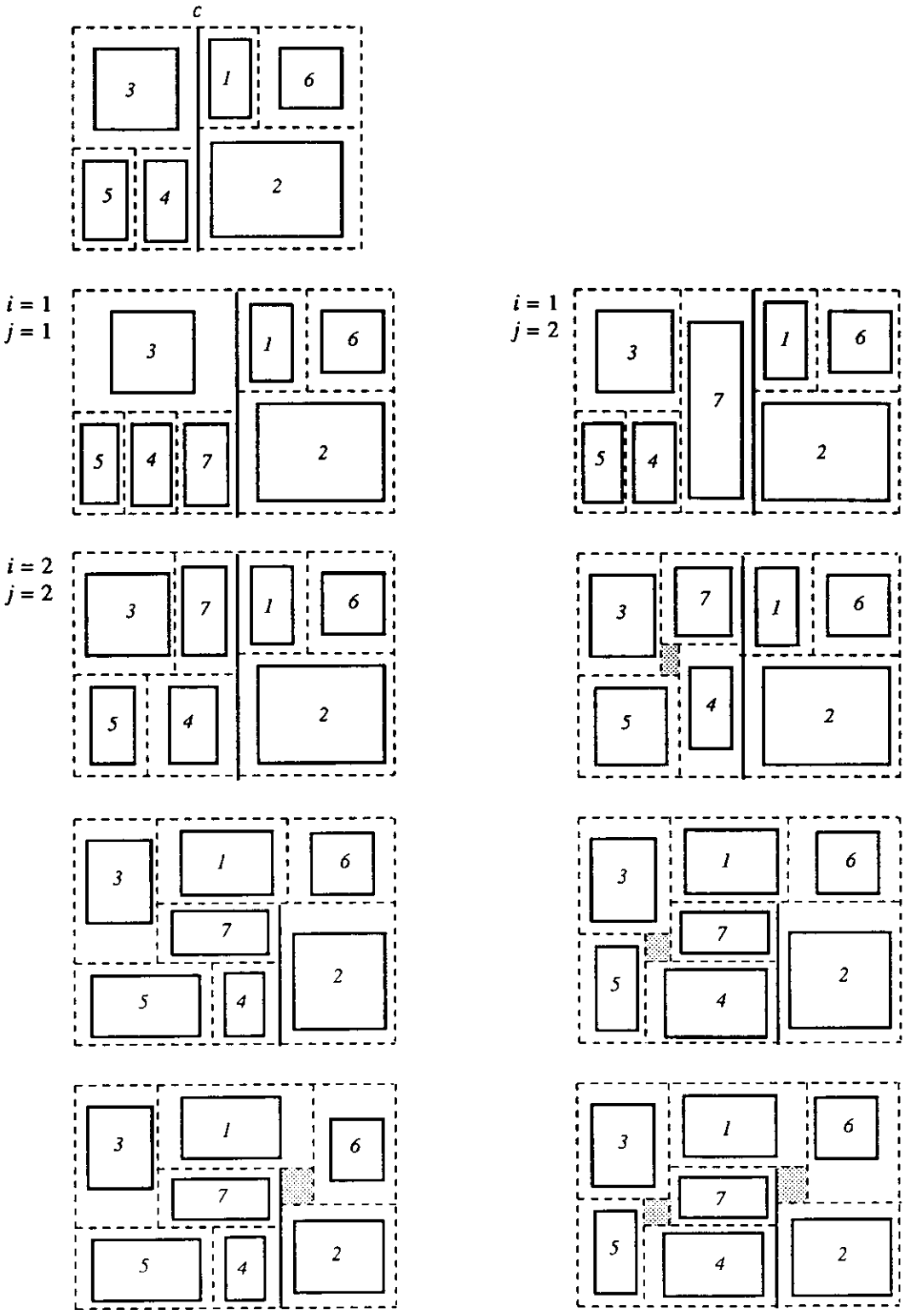

Figure 8: Illustration of non-monotonic enumeration

How these functions work is illustrated in Figure 8. It shows a structure with 6 regular rectangles and a wall, *c*. The figure shows all ways in which *non_monotonic_expand* inserts rectangle 7 along this wall in direction * = R, including all insertions of special rectangles.

References

U. Flemming (1978): "Wall representations of rectangular dissections and their use in automated space allocation" *Environment and Planning B* 5 215-232

U. Flemming (1980): "Wall representations of rectangular dissections: additional results" *Environment and Planning B* 7 247-251

U. Flemming; R. F. Coyne; T. Glavin; Hung Hsi; M. D. Rychener (1988): A generative expert system for the design of building layouts (final report). Report 48-15-89. Engineering Design Research Center. Carnegie Mellon University. Pittsburgh, PA

U. Flemming, R. Coyne, T. Glavin and M. Rychener (1988a), "A Generative Expert System for the Design of Building Layouts - Version 2," *Artificial Intelligence in Engineering; Design (Proc. Third International Conference, Palo Alto, CA)* , J. Gero, ed., New York: Elsevier, pp. 445-464.

U. Flemming (1989): "More on the representation and generation of loosely-packed arrangements of rectangles" *Environment and Planning B. Planning and Design* 16 327-359