

On the generation and enumeration of tessellation designs

R Krishnamurti

Centre for Configurational Studies, The Open University, Milton Keynes MK7 6AA, England

P H O'N Roe

Department of Systems Design, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

Received 11 May 1979

Abstract. Tessellation designs composed from tiles in periodic space fillings are considered. An efficient algorithmic theory for the generation and enumeration of nonequivalent designs is developed. It is shown that each design has a graphical representation as a labelled subgraph of some graph whose vertices have associated integral coordinates. Detecting isomorphisms between designs then reduces to determining permutations of the labels of the vertices of this graph and may be performed in linear time. A proof of correctness for the algorithmic theory is provided. Nine specific algorithms for various families of designs from the archimedean tessellations are presented.

Introduction

In recent years there has been striking evidence of the vital role played by certain types of combinatorial configurations in the representation of various systems. Indeed many problems in design and elsewhere are formulated in terms of spatial structures which are subject to topological and geometrical constraints. For instance, architectural plans have been represented as trivalent plane maps which have prescribed geometric realizations (March and Earl, 1977). Other examples include such diverse topics as bracing structures and adaptability patterns in buildings, both of which are related to bipartite graphs without isolated vertices (Bolker and Crapo, 1977; Harary et al, 1978); Palladian schemes are seen as line designs on an underlying grid structure (Stiny and Mitchell, 1978); biological cell morphogenesis has been modelled as a recursive aggregation of simple polygonal forms on two-dimensional lattices (Eden, 1958; 1960); the groupings called 'clusters' of sites and bonds from crystal lattices describe the high- and low-temperature constants of Ising models in statistical mechanics (Sykes et al, 1972a; 1972b; Martin, 1974; Martin and Watts, 1971); and crystal properties are intimately related to the topology of polyhedral configurations embedded in periodic two- and three-dimensional nets (Wells, 1977). This list is by no means complete. But it does serve to emphasize the following point: in each of these examples the typical combinatorial property sought is—how many *distinct* configurations are there of a given kind?

The usual method in tackling such problems is to determine a formula such as a recurrence relation, a generating polynomial, or sometimes even a constructive procedure. Unfortunately in many instances the counting problem for spatial configurations is of such difficulty that no general analytical procedure has been devised (Harary and Palmer, 1973, chapter 10). In fact for the problems considered in this paper the question of determining counting polynomials remains open. Our only recourse therefore is to resort to computer methods.

Access to computers creates its own special problems, which almost all enumeration algorithms encounter. There are two main sources of difficulty. The first is due to combinatorial explosion—namely, the number of distinct configurations increases exponentially or worse with the *order* of the configurations. Computationally there is no way of avoiding this. The second arises from the fact that spatial configurations

often correspond to one another through a set of spatial transformations. Two such spatially related configurations—referred to as *isomorphs*—cannot be regarded as distinct objects. They are instead two separate spatial manifestations of the same object. Consequently the elimination of isomorphic duplicates, termed isomorph rejection, is a matter that requires careful consideration. The importance of isomorph rejection in constructive enumeration is due to the fact that perhaps every possible permutation of the orientation in the space of the elements in the configurations may have to be examined. [The interested reader is referred to Corneil and Mathon (1978), Read (1978), Fillmore and Williamson (1974), and Williamson (1973) for an exposé of isomorph rejection for various species of combinatorial configurations.] The combined effect of these two difficulties is perhaps best captured by the following quote (Golomb and Baumert, 1965, page 524): "... most combinatorial problems grow to such an extent that there is at most one additional case beyond hand computation that can be handled by our present day high speed digital computer". Clearly in order to utilize the tremendous speed of digital computers the design of *efficient* enumeration algorithms is of paramount importance.

The efficiency of an algorithm is measured in terms of the total number of computational steps required and the amount of storage needed to house all the relevant information. These two quantities, referred to as the *time complexity* and *space complexity* respectively (Aho et al, 1974, pages 12–14), are expressed as order functionals, $O[t(p)]$ and $O[s(p)]$, where $t(p)$ and $s(p)$ are expressions in p , the problem size. An algorithm is generally deemed efficient if $t(p)$ is algebraic in p . For constructive enumeration algorithms we may adopt the definition that such an algorithm is efficient if and only if the following conditions are satisfied, where p denotes the problem size—in our case the order of the configurations.

Condition 1: Every configuration is uniquely generated in a time step bounded by a polynomial in p .

Condition 2: Isomorph rejection is performed in a time step bounded by a polynomial in p .

Condition 3: Storage required is bounded by a polynomial in p .

It is instructive to reflect upon these conditions. They are not necessarily independent or exclusive. Often it is the case that conditions 1 and 2 imply condition 3. Nevertheless it is worthwhile to include it separately. Furthermore conditions 1 and 2 are related by the fact that if condition 2 is satisfied then there must exist a 'coding mechanism' which allocates distinct codes to each isomorph of an object, which in turn influences the lexicographical order in which the objects are generated, and consequently reduces the size of the search tree (this is explained in section 3). Thus if we can satisfy condition 2 condition 1 follows more easily.

Let N_p denote the number of distinct configurations of order p . Then an efficient algorithm has time and space complexities of orders $O[t(p)N_p]$ and $O[s(p)]$ respectively, where $t(p)$ and $s(p)$ are polynomials in p . An efficient algorithm is a *good* characterization of a family of combinatorial configurations (Edmonds, 1965).

The main goal of this paper is to present a general algorithmic framework for constructing families of spatial patterns that are based on periodic tessellations. The theory developed is applied to design efficient algorithms for patterns on the archimedean tessellations. The paper consists of two basic parts. The first part, comprising sections 1 through 5, introduces the relevant terms and presents the computational theory. The second part, formed by sections 6 through 13, describes the algorithms for the illustrative examples, starting with the patterns constructed on

the regular tessellations and followed by five classes of patterns on the semiregular tessellations.

Note: All algorithms in this paper assume the standard model of computation, namely a random access machine with a sufficiently large but finite memory and a limited instruction set (Aho et al, 1974; Horowitz and Sahni, 1976; Reingold et al, 1977). The algorithmic notation is adopted from Reingold et al (1977) and complexity is based on uniform cost. The graph theory terminology corresponds to Harary (1969) and Bondy and Murty (1976). Every effort has been made to keep the presentation simple yet mathematically precise.

1 Tessellations

A *plane tessellation* is a collection of objects called *tiles* or *cells* that cover the plane without gaps or overlap. A tessellation is also known in the literature as a *tiling*, *paving*, or *mosaic*. A pair of *adjacent* tiles share a common edge, and a collection of tiles meet at a *point*. Figure 1 presents examples of plane tessellations. In a similar fashion the *solid* and *hypersolid* tessellations covering n -space, $n \geq 3$, may be defined. For the sake of visual simplicity we restrict the discussion mainly to the planar case and merely state that the relevant theory developed herein also applies to the higher-dimensional situation.

Graphically a plane tessellation may be regarded as a planar map with locally finite vertices. Consequently the combinatorial dual map represents a dual tessellation. It is important not to confuse a tessellation with its map since the former is a geometric realization of the latter. Each map may have several distinct realizations, each with distinct symmetry properties. Figure 2 shows a pair of dual maps with some representative geometric tessellations. Hence it is the geometry of the tiles superimposed on an underlying topology that essentially characterizes the tessellation.

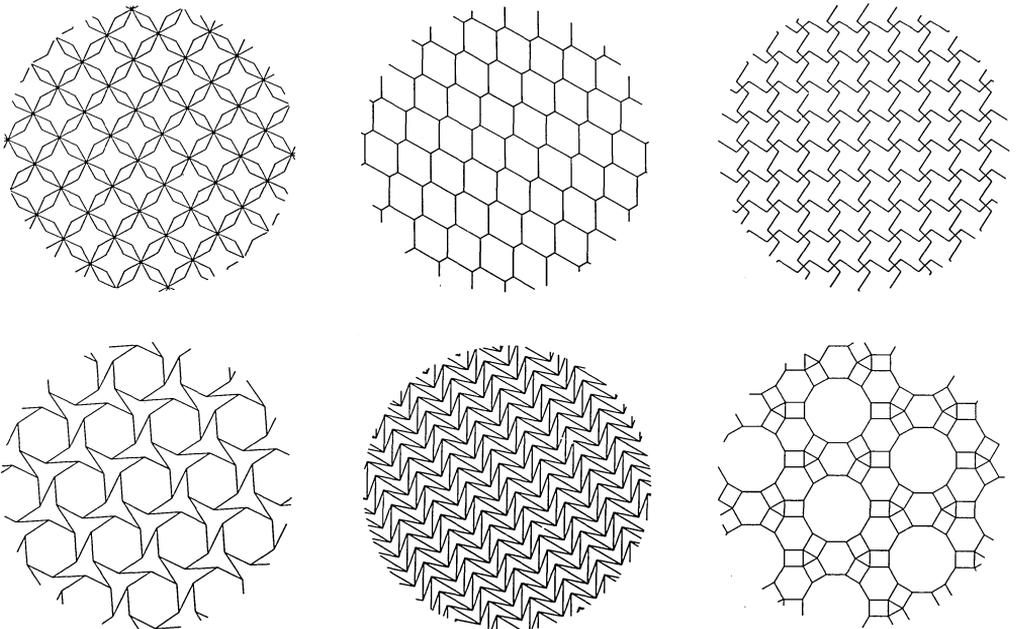
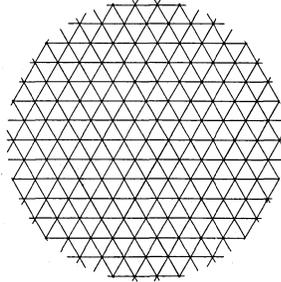
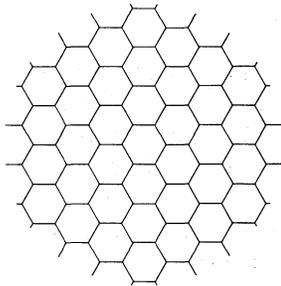


Figure 1.

A tessellation is *archimedean* if (1) every tile is a regular polygon (or polyhedron) and (2) the tiles meet at each point in the same cyclic order. There are eleven plane archimedean tessellations, and these are shown in figure 3. Observe that figures 3(a), 3(b), and 3(c) each consist of exactly one type of tile, namely the square, hexagon, and triangle respectively. These three are the *regular* tessellations, and the remaining eight are referred to as the *semiregular* tessellations. Since the polygons meet at every point in the same cyclic order, the archimedean tessellations may be conveniently designated by the ordered list of numbers $\langle n_1.n_2.\dots \rangle$, where n_i is the number of sides

Maps



Realizations

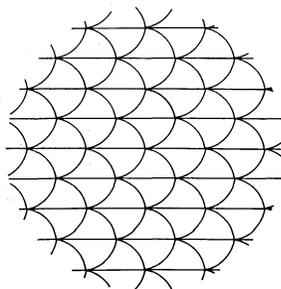
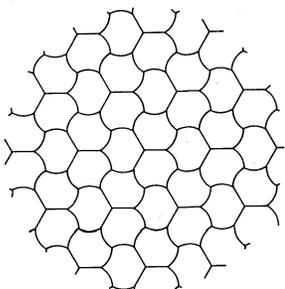
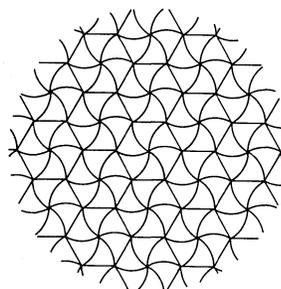
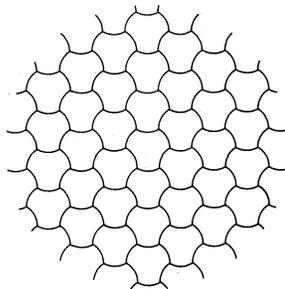
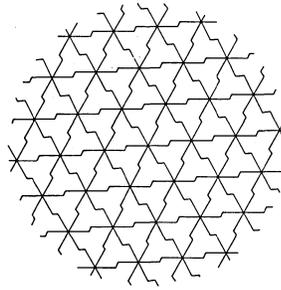
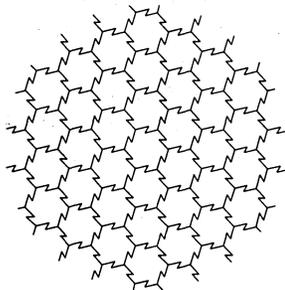


Figure 2.

of the i th polygon in the list around the cycle starting with the n_1 -gon. Often this is abbreviated to the form $\langle g_1^a . g_2^b . \dots \rangle$ in the obvious way. The geometric dual tessellations are the *Laves nets* shown in figure 4. The nets are tagged by the same symbols that

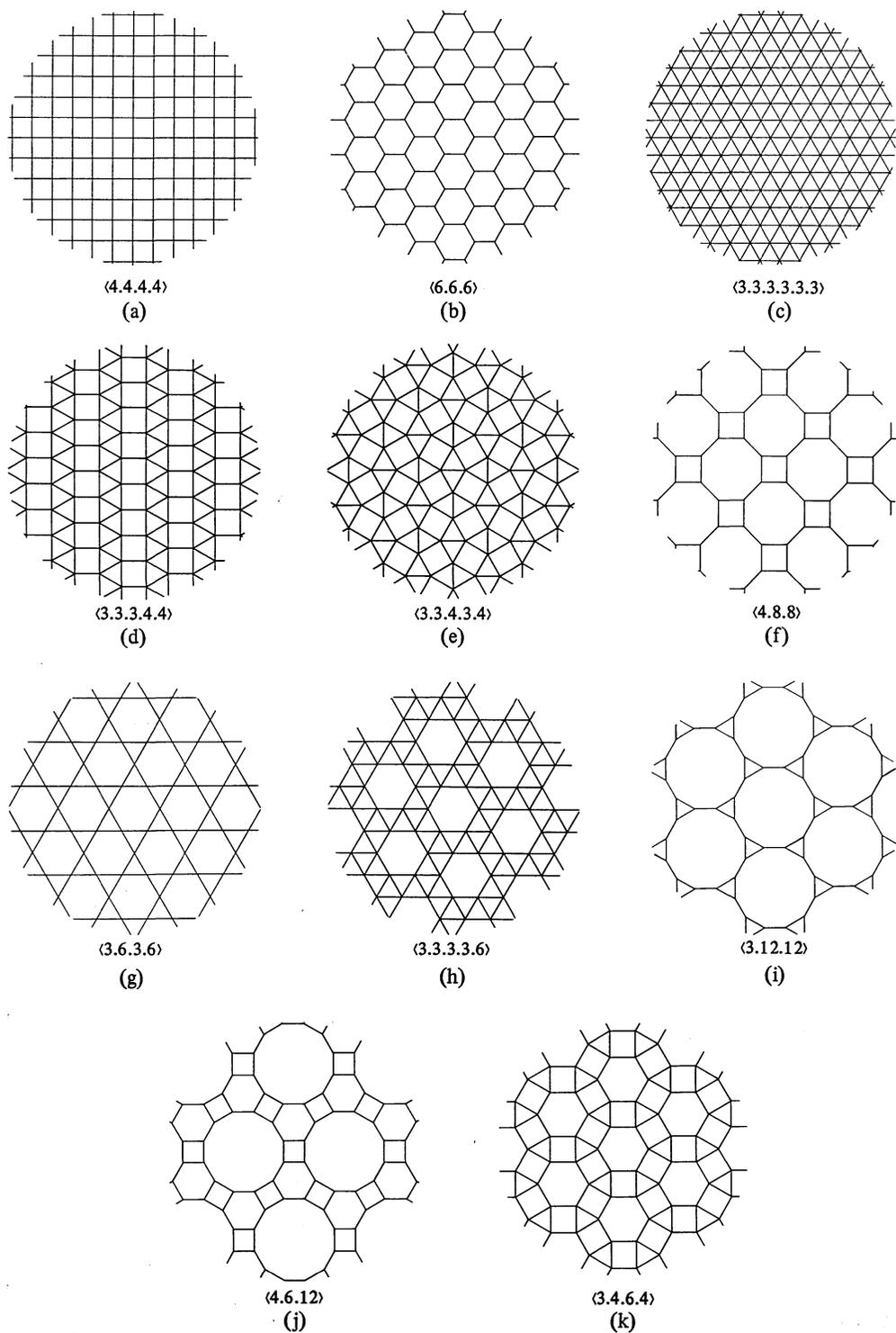


Figure 3.

identify the corresponding archimedean tessellations.

The only solid and hypersolid tessellations that are archimedean are the space coverings by the n -cube, $n \geq 3$; hence they are also regular.

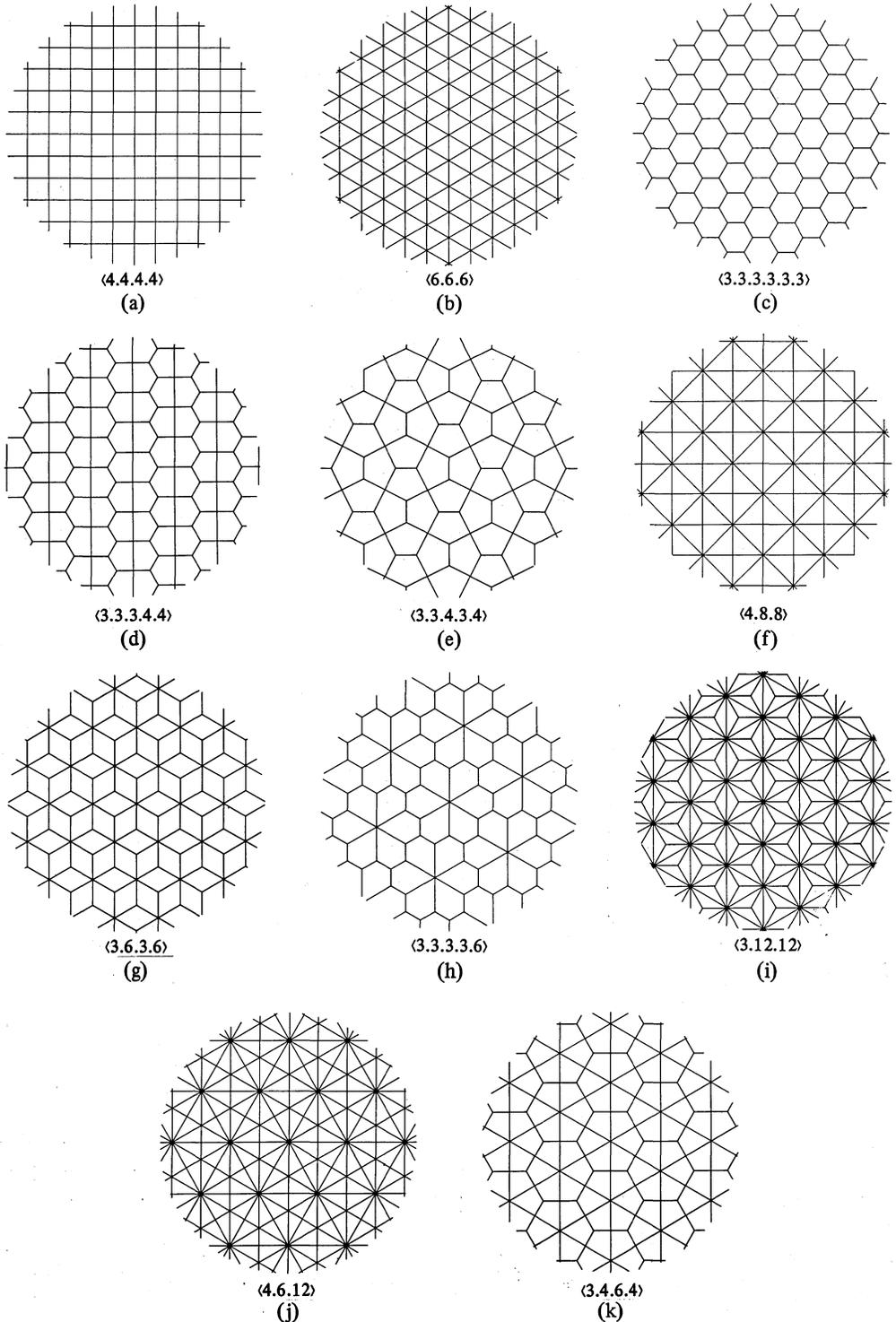


Figure 4.

2 Tessellation designs

A design on a tessellation is a finite configuration, composed of tiles of the tessellation, which can be embedded in the tessellation. Of the possible designs the most interesting class is the family of connected designs. A design is *connected* if every tile is adjacent to at least one other tile in a smaller connected design. A single tile is always connected. A design may be represented pictorially in essentially two ways: either as a two-colouring of the tessellation in which the tiles in the design are coloured differently, or as a spatial pattern composed of polygons. The two representations are illustrated in figure 5.

The designs on the regular tessellations, namely those made up of squares, hexagons, triangles, and n -cubes, $n \geq 3$, are more familiarly known as *polyominoes*, *polyhexes*, *polyiamonds*, and *poly n -cubes* respectively. The semiregular designs will be referred to as $\langle n_1, n_2, \dots \rangle$ -*patterns*, where $\langle n_1, n_2, \dots \rangle$ identifies the tessellation.

The *content* (or *order*) of a design is the number of tiles it contains. A connected design with content p is a *p-figuration*. (Henceforth the terms design and configuration will be used interchangeably.) A *p-figuration* may be recursively defined as follows: a $(p+1)$ -figuration results whenever a new tile is adjoined adjacently to a tile in a *p-figuration*. A single tile is a 1-figuration. Since configurations are embeddings in a tessellation, equivalent configurations can be defined in the following manner: two configurations are *equivalent* if they are identical embeddings after a combination of translation, rotation, and reflection. The equivalence classes under translations are *fixed*. In addition the equivalence classes of the fixed classes under a sequence of rotation and reflection are *free*. Our aim is the enumeration of the nonequivalent configurations via the counting of the representatives (called *canonical configurations*) of the free equivalence classes.

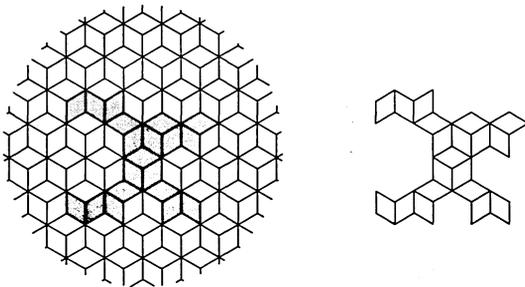


Figure 5.

2.1 Review of the literature

Previous attempts at enumerating the free equivalence classes of the tessellation designs have been directed towards the regular tessellations. Golomb (1954; 1961a; 1961b; 1961c; 1962; 1965) initiated the study by extensive analysis of polyomino properties. Kelly (1966) provided an algebraic descriptor for polyominoes and was able to demonstrate the existence of solutions for certain packing and covering problems. Eden (1960) and Klarner (1965; 1967; 1969; and with Rivest, 1973) applied highly rigorous combinatorial arguments to arrive at good upper and lower bounds for the number of p -ominoes as p approached asymptotically large values. Let S_p , H_p , and T_p respectively denote the number of p -ominoes, p -hexes, and p -iamonds. Set \tilde{S} , \tilde{H} , and \tilde{T} to the limits defined by

$$\tilde{F} = \lim_{p \rightarrow \infty} \frac{F_p}{F_{p-1}},$$

where F stands for S , H , or T as appropriate. It has been shown that \tilde{F} exists in all three cases and that

$$3.72 < \tilde{S} \leq 4.65, \quad 4 < \tilde{H} \leq 6.75, \quad \text{and} \quad 2.13 < \tilde{T} \leq 4.$$

These bounds are currently the best possible known.

Other approaches include Read's (1962) computational scheme to count polyominoes ('animals' as he calls them!) via bounding rectangles. The method is elegant in that one can obtain the counting polynomials for the p -ominoes in their boundary rectangle, but suffers from excessive time and space requirements even for small ps . Parkin et al (1967) and Lunnon (1969; 1971) reported results based on computer-implemented strategies. Their approach is essentially a restricted type of brute-force enumeration similar to the choice enumeration developed in this paper. In fact many of Lunnon's ideas have been freely borrowed.

At around about the same time Sykes and his collaborators (see, for example, Sykes et al, 1972a; 1972b; 1972c), working in statistical mechanics, employed computer methods to enumerate the number of distinct p -clusters of sites on various lattices. A cluster is an analogue of a polyomino and corresponds to a high-temperature lattice constant. Martin (1974) presents a survey of their computational methods. After literally hundreds of hours of computing, Sykes and Glen (1976) report on asymptotical results based on a Padé extrapolation technique (Gaunt and Guttman, 1974) to obtain the limits for \tilde{S} , \tilde{H} , and \tilde{T} , namely

$$\tilde{S} = 4.06 \pm 0.02, \quad \tilde{H} = 5.19 \pm 0.03, \quad \text{and} \quad \tilde{T} = 3.04 \pm 0.02.$$

These results seem to match those of Lunnon (1972) who used a logarithmic extrapolation technique.

Tilley (1970), in his MA thesis, presents efficient and attractive coding schemes which form the basis for algorithms to enumerate classes of tessellation designs called *filaments* (which are topologically equivalent to paths). His coding scheme also provides a method for defining a recurrence relation for the upper bounds on the number of filaments which gives results very close to actual counts for small ps . Lunnon (1975) extends his approach to the higher-dimensional forms and presents counts for poly n -cubes, $3 \leq n \leq 7$.

There have been attempts to find the counting polynomial for tessellation designs directly. Harary and Read (1970) obtained a counting polynomial for tree-like polyhexes with no 'periconnections'. Palmer (1972) used their approach to lay the foundations for a graph-enumerative technique by obtaining the counting polynomial for a variant of the polyomino counting problem. (The variant he considered was a relaxation of the valency restriction on the points of the square tiles—not necessarily limited to four—in the configuration.) Later Harary et al (1975), using both approaches, obtained the counting polynomials for the counting problem for arbitrary polygonal tiles under this relaxation. The problem of finding counting polynomials for the tessellation designs remains open.

3 An algorithmic technique

We now review a technique—*backtrack programming*—which is the most widely used search strategy for the constructive enumeration of combinatorial configurations. The term backtrack programming was first coined by Lehmer in the 1950s (see his survey article in 1964) and later formalized by Walker (1960) and Golomb and Baumert (1965), and more recently by Bitner and Reingold (1975). One of the earliest uses of backtracking was Tarry's (1895) maze-threading algorithm. Briefly backtracking may be described as a search for a solution (configuration) by continually extending partial solutions. At each stage of the search, if an extension of the current partial

solution is not possible the algorithm backtracks to an earlier stage (that is, to a smaller partial solution) and tries again.

Formally backtracking assumes that a configuration can be expressed as a p -tuple, (a_1, a_2, \dots, a_p) , where the a_i are members of linearly ordered lists A_i . Each p -tuple satisfies predetermined constraints. Every configuration therefore is a member of a subspace of the direct product $A^p = A_1 \times A_2 \times \dots \times A_p$ of p selection spaces. For $k \leq p$, let f_k stand for the function $f_k(a_1, a_2, \dots, a_k)$ from A^k to $\{0, 1\}$ such that

$$f_k = 1 \Rightarrow f_{l < k} = 1 .$$

The counting problem is to determine all nonequivalent (a_1, a_2, \dots, a_p) such that $f_p = 1$. The search procedure is given in ALGOL-like notation in algorithm 1.

The algorithm may be described as follows. Initially choose the null vector $()$, as the starting solution. Using the constraints determine which members of A_1 are candidates for a_1 ; let this subset be C_1 . Choose a least element of C_1 as a_1 , and now we have a partial solution, (a_1) . In general the various constraints determine which subset C_k of A_k provides possible candidates for the extension of the partial solution $(a_1, a_2, \dots, a_{k-1})$ to (a_1, a_2, \dots, a_k) . They are precisely all a s that satisfy $a_k = a$ and $f_k = 1$. If the partial solution admits no possibility for a_k then $C_k = \emptyset$, so we backtrack and make a new choice for a_{k-1}, a_{k-2} , and so on. It is helpful to picture this as a tree traversal. The subsets of $A^k, k = 0, 1, 2, \dots$, are represented as a rooted

Algorithm BACKTRACK

¶ iterative procedure for backtrack programming ¶

$C_1 \leftarrow A_1$

$k \leftarrow 1$

while $k > 0$ do

{	while $C_k \neq \emptyset$ do
	$a_k \leftarrow$ least element in C_k
	$C_k \leftarrow C_k - \{a_k\}$
	if $k = p$
	then (a_1, a_2, \dots, a_p) is a solution
else	
$k \leftarrow k + 1$	
$C_k = \{a \in A_k : f_k(a_1, a_2, \dots, a) = 1\}$	
$k \leftarrow k - 1$	

Algorithm 1.

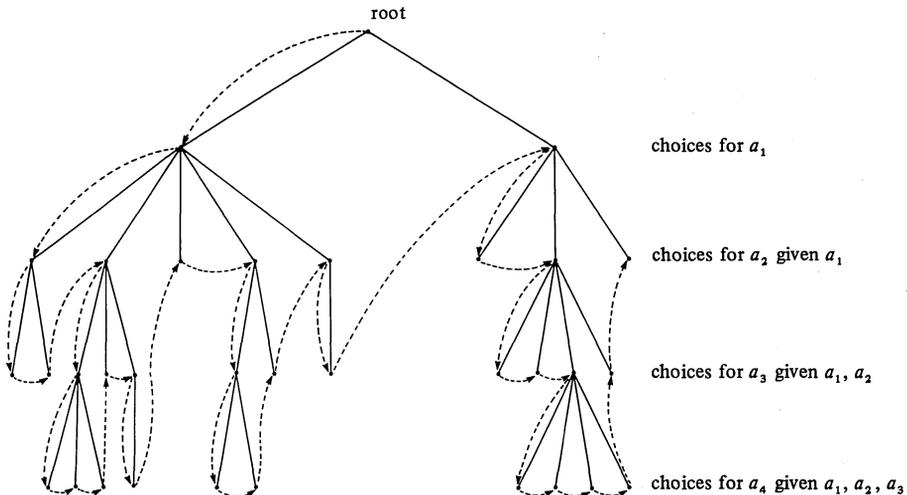


Figure 6.

tree as follows. The *root* of the tree, at the 0th level, is the null vector. Its *sons* are nodes representing the choices for a_1 . In general at the k th level the nodes represent choices for a_k given their *ancestors* a_1, a_2, \dots, a_{k-1} . The search tree assumes the form shown in figure 6. In this tree the dotted lines indicate the order in which the nodes are traversed. Since the traversal proceeds as deep as possible before backtracking it is often referred to as *depth-first search* (Tarjan, 1972).

A recursive description of backtracking is given in algorithm 2.

The simplicity of depth-first search permits an elegant proof of correctness for the algorithm. The following results are easily shown.

Lemma 1: *The search is finite.*

Proof: Since the search tree is finite and every node is traversed exactly once.

Lemma 2: *All p -tuples (a_1, a_2, \dots, a_p) are uniquely generated.*

Proof: By induction. Clearly all 1-tuples are uniquely generated. Suppose this is true for all k -tuples. All $(k+1)$ -tuples may be constructed in the following manner. For each k -tuple append an a_{k+1} from the selection space A_{k+1} . For each pair of $(k+1)$ -tuples either they differ in the last place or in the first k places.

An immediate corollary is the following lemma.

Lemma 3: *If the sequence (f_1, f_2, \dots, f_p) is well-defined then the algorithm is correct, where by well-defined is meant that*

$$f_k = 1 \Rightarrow f_{l < k} = 1 \quad \text{and} \quad f_k = 0 \Rightarrow f_{l > k} = 0 .$$

Proof: Obvious.

Remark: One must be careful when applying these lemmas directly to a particular application. It is possible to apply backtracking strategies to generate unordered p -tuples that represent configurations. In such cases two distinct ordered versions may represent identical configurations and not isomorphic variations. However, with a bit of manipulation in the way the f_k are defined, the problem of producing duplicates may be overcome. This is demonstrated for the tessellation designs.

A few comments on the efficiency of backtracking are in order. The efficiency is due to the fact that if $f_k = 0$ then $f_{l > k} = 0$, no matter what choices are made for the remaining $p - k$ components. Clearly the efficiency increases if, for many solutions (a_1, a_2, \dots, a_p) for which $f_p = 0$, $f_k = 0$ for small values of k .

Another way of increasing the efficiency is to avoid and reject partial isomorphic solutions. The rejection process employs the fact that the configurations are generated in lexicographical order. This permits us to merge several branches of the search tree

Algorithm DFS(k)

```

¶ recursive depth-first search ¶
if  $k > p$ 
then  $(a_1, a_2, \dots, a_p)$  is a solution
else
  for  $a \in A_k$  do
    if  $f_k(a_1, a_2, \dots, a) = 1$ 
      then  $\left\{ \begin{array}{l} a_k \leftarrow a \\ \text{DFS}(k+1) \end{array} \right.$ 
return
¶ invoking routine ¶
DFS(1)

```

Algorithm 2.

associated with nodes corresponding to partial isomorphic solutions. Again this is very effective for small values of k as this causes very large subtrees to be eliminated.

Furthermore when searching for solutions it is efficient to arrange the selection spaces A_k in increasing order of cardinality. It has been observed statistically that backtracking due to $C_k = \emptyset$ occurs at fixed levels and such an ordering of the A s results in fewer nodes being searched. For further pruning methods the reader is referred to Bitner and Reingold (1975).

We conclude this section by noting that backtracking is an algorithmic statement of the familiar inclusion-exclusion principle in combinatorics.

4 Representation of tessellation designs

4.1 Graphs

The graph of a tessellation is constructed in the following manner. Assign a vertex to each tile in the tessellation. Join by an edge pairs of vertices corresponding to adjacent tiles. It is easy to see that in the planar case this graph is the dual map, although in n -space, $n \geq 3$, it has sometimes been referred to as a dual tessellation. Clearly any p -figuration corresponds to a connected subgraph on p vertices of the tessellation graph. For the plane archimedean tessellations the graphs are the Laves nets (figure 4). The graph contains all the structural information necessary to specify the tessellation. For instance, in the planar case the degree of a vertex describes its corresponding tile type; if the degree is m , the tile is an m -gon. The length of the smallest cycles in the graph is the number of tiles that meet at a point. The degree sequence in these cycles provides the identification tag for the tessellation.

4.2 Coordinate representation

Since the crucial consideration in any spatial algorithm, especially of the kinds considered in this paper, is the elimination of isomorphs (equivalent configurations), it follows that an adequate mechanism for coding configurations is essential. Moreover it would improve the efficiency with respect both to space and time if these codes were unique and took on integral values (usually in the form of m -tuples, $m \leq p$). The first step in devising such a code depends upon our ability to assign integral 'coordinates' to the vertices of the tessellation graph. For the square and n -cube tessellations these coordinates are relatively simple to obtain. They are in fact simply the integral Cartesian coordinates. That is, every vertex is assigned coordinates (x_1, x_2, \dots, x_n) . The adjacent vertices have coordinates $(x_1, x_2, \dots, \eta_i, \dots, x_n)$, where $\eta_i = x_i \pm 1$ for all possible i .

For the hexagonal and triangular tessellations [figures 3(b) and 3(c)] the problem is not so straightforward. Instead we need to go through a subterfuge. Consider an arrangement of cubes in 3-space whose centers have integral coordinates (x, y, z) satisfying $x + y + z = 0$. Take a simple 45° isometric projection onto the plane as shown in figure 7(a). The result is the rhombic tessellation. Simultaneously removing all trivalent points and their associated edges gives the desired hexagonal tessellation shown in figure 7(b), in which each tile is associated with integral coordinates (x, y, z) such that $x + y + z = 0$ [see figure 7(c), where \bar{x} denotes $-x$]. Alternatively Lunnion (1972) suggests the following construction. Consider a cube with center at the origin, $(0, 0, 0)$. Pass the plane $x + y + z = 0$ through this cube, as in figure 7(d). Extending this construction to the cubic tessellation and taking the projection of this plane gives the (3.6.3.6) tessellation. Coalescing the triangle in the manner shown in figure 7(e) by the superimposed dotted hexagon gives the desired hexagon and same coordinate system as previously obtained, but in a different orientation [figure 7(f)]. The latter orientation will be used in this paper. Also, it is interesting to note that figures 7(a) and 7(e) are geometric dual tessellations.

The coordinates of the adjacent vertices of vertex (x, y, z) are given by the six-element set

$$\{(x, y \pm 1, z \mp 1), (x \mp 1, y, z \pm 1), (x \pm 1, y \mp 1, z)\}.$$

Notice that the hexagons can be coloured in a natural way by use of three colours according to whether $x - y \equiv 0, 1, 2 \pmod 3$. Now let us obliterate the tiles of a particular colour by contracting them to a point. The result is the triangular tessellation, shown in figure 8 in one of the three possible combinations of two colours. From this it is evident that a triangular tessellation corresponds to a hexagonal tessellation on two colours—that is, the set of polyiamonds is a subset of the set of two-colourable polyhexes.

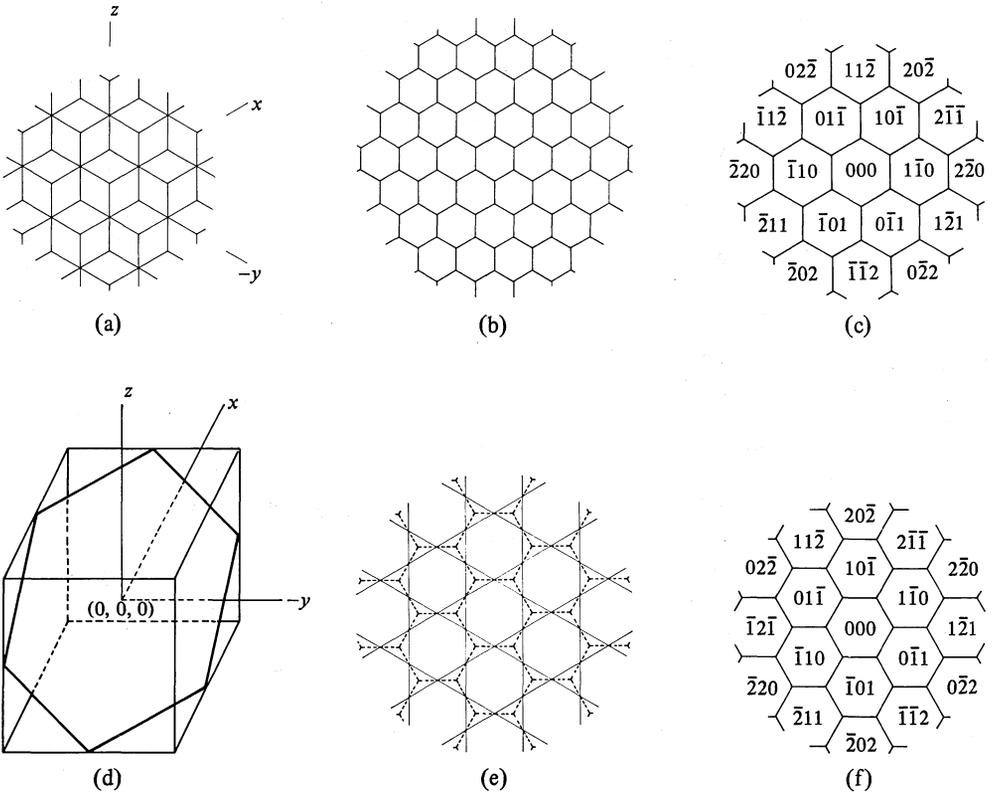


Figure 7.

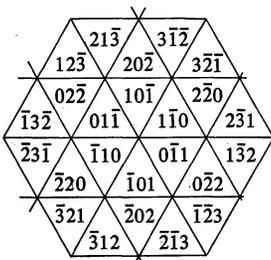


Figure 8.

Each triangular tile is either of type ‘U’ or of type ‘D’ depending on whether it points up or down respectively. The neighbours of vertex (x, y, z) are given by

$$\text{‘D’}: \{(x, y-1, z+1), (x+1, y, z-1), (x-1, y+1, z)\}$$

$$\text{‘U’}: \{(x, y+1, z-1), (x-1, y, z+1), (x+1, y-1, z)\}.$$

For convenience a triangular tessellation will be regarded as a hexagonal tessellation with forbidden ‘tiles’.

The integral coordinates for the semiregular patterns are taken up in the examples.

4.3 Bounding regions

The bounding region for a p -figuration is a smallest region that encloses it. The region is chosen so that it possesses certain symmetry properties that facilitate computation. For instance, every polyomino can be encased within a rectangle, every polyhex inside a hexagon, and every poly n -cube inside an n -rectangle (see figure 9). Clearly the outline of a tessellation design is the smallest polygon. However, from a computational viewpoint the bounding region is so chosen that it enables certain properties to be conveniently extracted, such as symmetry transformations and the coding of configurations. This is probably better illustrated when considering the specific examples.

Every p -figuration must touch all sides of its bounding region; for example, the defining lines of the rectangle or hexagon and the defining faces of an n -rectangle. In other words, a p -figuration spans its bounding region. Furthermore it can be shown that, under the given conditions, every p -figuration has a unique bounding region. An added bonus from considering bounding regions is that, for any p , the set of bounding regions is finite and can be calculated fairly easily.

A bounding region may also be regarded as an m -figuration, $m \geq p$, which contains as subconfigurations its spanning p -figurations. The graph of the bounding region, referred to as a *trellis*, is a subgraph of the tessellation graph. A p -figuration corresponds to a connected spanning subgraph on p vertices of its trellis.

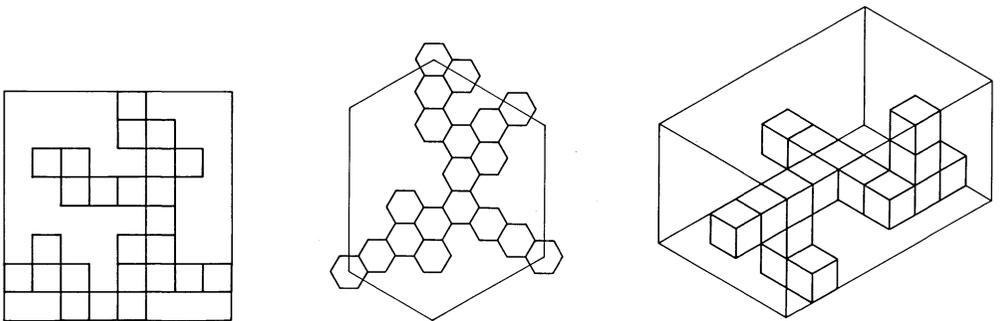


Figure 9.

4.4 Symmetry transformations

A symmetry transformation is a spatial operation that maps a configuration either onto itself or onto an isomorph. These transformations form a group which is usually the symmetry group of the dominating tile type. For the square tessellation, the symmetry group is the dihedral group D_4 ; for the hexagonal tessellation it is the group D_6 ; and for the n -cube tessellation it is the orthogonal group of rotations O_n .

Since each configuration has a unique bounding region, we need only consider the symmetry transformations that leave the bounding region invariant—usually these are small-order subgroups of the group of symmetries of the dominating tessellation tile.

Since the tiles have integral coordinates, the symmetry transformations reduce to coordinate-coordinate mappings. This can be improved upon. If we label the vertices of the trellis then a symmetry transformation is simply a permutation of the labels.

4.5 Coding of configurations

We can associate with every trellis a *word* which is a polynomial of the form

$$x_1 + x_2 + \dots + x_m ,$$

where m is the number of vertices in the trellis and x_i is a formal mark representing vertex v_i . Since a p -figuration is a subgraph of its trellis, it has associated with it a word of the form

$$\rho_1 x_1 + \rho_2 x_2 + \dots + \rho_m x_m ,$$

where $\rho_i \in Z_2 = \{0, 1\}$ for all possible i ; $\rho_i = 1$ if and only if vertex v_i is in the graph of the configuration. Clearly for a p -figuration $\sum_i \rho_i = p$. This gives a unique code for the configuration within its trellis. Let τ denote a permutation of the vertices onto themselves. Then an isomorphic configuration has the word

$$\rho_1 x_{\tau(1)} + \rho_2 x_{\tau(2)} + \dots + \rho_m x_{\tau(m)} ,$$

or equivalently

$$\rho_{\tau^{-1}(1)} x_1 + \rho_{\tau^{-1}(2)} x_2 + \dots + \rho_{\tau^{-1}(m)} x_m ,$$

where τ^{-1} is the inverse mapping. τ is an element of T_t , the group of symmetries that leave the trellis t invariant.

Since the polynomial is linear in Z_2 , we can represent it as a binary number. Hence every p -figuration corresponds to a pair (t, b) , where t identifies the trellis and b is the binary representation of the configuration word ($b = \rho_1 \rho_2 \dots \rho_m$ radix 2).

A canonical code for the representative of each free equivalence class of configurations may now be derived. Since a bounding region may be placed anywhere in the space under consideration, we need to fix the orientation of the bounding regions. We do this by defining an order relation on the defining lines of the region. This has the effect that no two bounding regions for p -figurations, with p fixed, correspond to one another through a combination of rotations and reflections. Let T_t be the group of symmetry transformations that leave the bounding region t invariant in the space. Consider a configuration b . Let $\tau(b)$ represent the transformed configuration under τ , where $\tau \in T_t$. Define the sets T'_t and T''_t as follows:

$$T'_t = \{ \tau \in T_t : \tau(b) \text{ spans } t \} ,$$

$$T''_t = \{ \tau \in T_t : \tau(b) \text{ spans a smaller bounding region} \} .$$

Let \lesssim be an order relation defined on binary numbers. Then a configuration b is the representative of a free equivalence class of configurations if and only if (1) $T''_t = \emptyset$ and (2) $b \lesssim \tau(b)$ for all $\tau \in T'_t$. Here we have (without ambiguity) allowed $\tau(b)$ to represent the binary number of the transformed configuration. Any configuration that satisfies these two conditions is called the *canonical configuration* of its free equivalence class and b is the *canonical code*. In general $T''_t = \emptyset$ and $T'_t = T_t$. In the examples considered the only configurations for which this does not hold are the (3.3.3.4.4) patterns.

Often b is unwieldy to use directly in moderately sized computers in the manner presented here. Instead we split it into smaller numbers. The code for a configuration becomes a prefixed k -tuple of the form $(t; b_1, b_2, \dots, b_k)$. Usually the prefix t is dropped from the code when the trellis under consideration is unambiguously known.

5 Computational details

5.1 A graph problem

The following problem plays an important role in generating the archimedean patterns with respect to their bounding regions. Suppose we are given a graph $G = (V, E)$, where V is the set of vertices of the graph and E the set of edges, and a finite collection of subsets of the vertices, $S^* = \{S_1, S_2, \dots\}$, where $S_i \subseteq V$ for all $S_i \in S^*$. We are required to find:

- (a) a connected labelled subgraph $P = (V_p, E_p)$, with $|V_p| = p \leq |V|$, such that $V_p \cap S \neq \emptyset$ for all $S \in S^*$; and
- (b) all possible connected labelled subgraphs that satisfy (a).

5.1.1 *Distance measures on graphs.* Let 2^V denote the power set (the set of all subsets) of V and let Z denote the integers. Then define the distance measure $d: 2^V \times 2^V \rightarrow Z$ by

$d(u, v) \equiv$ the length of the shortest path between vertices u and v ;

$d(S, v) \equiv$ the length of the shortest path between set S and vertex v
 $\equiv \min\{d(u, v): u \in S\}$.

Let $\dot{+}$ be an operation on sets such that

$d(S \dot{+} T, v) \equiv$ the length of the shortest tree linking vertex v and sets S and T .

That is, $\dot{+}$ joins S and T via a shortest tree through vertex v . Clearly we have $d(S \dot{+} T, v) \leq d(S, v) + d(T, v)$, with equality if and only if v joins sets S and T by a pair of edge-disjoint paths.

The distance measure may be extended to subgraphs. Suppose $G' = (V', E')$ is a connected subgraph of $G = (V, E)$. Let $d(G', v)$ and $d(G', S)$ denote the distances $d(V', v)$ and $d(V', S)$, where

$d(V', S) \equiv$ the length of the shortest path between a vertex in V' and a vertex in S
 $\equiv \min\{d(S, v): v \in V'\}$.

The measure $d(G', S \dot{+} T)$ may be similarly defined. Again clearly we have $d(G', S \dot{+} T) \leq d(G', S) + d(G', T)$, with equality if and only if G' is connected to sets S and T by edge-disjoint shortest paths.

If G' is the required graph P satisfying problem 5.1(a) then (1) $d(P, S) = 0$ for all $S \in S^*$ and (2) $d(P, \dot{\Sigma}S) = 0$, where $\dot{\Sigma}S = S_1 \dot{+} S_2 \dot{+} S_3 \dot{+} \dots$, summed over all $S_i \in S^*$.

Furthermore if $G' = (V', E')$ and $G'' = (V'', E'')$ are two graphs satisfying

$$V' = V'' \cup \{v\} \quad \text{and} \quad E' = E'' \cup \{(v, x) \in E: x \in V''\}$$

then the following recurrence relations must hold:

$$d(G', z) = \min\{d(G'', z), d(v, z)\}, \quad \text{for all } z \in V,$$

$$d(G', S) = \min\{d(G'', S), d(S, v)\}, \quad \text{for all } S \in S^*,$$

$$d(G', S \dot{+} T) = \min\{d(G'', S \dot{+} T), d(G'', S) + d(T, v), d(G'', T) + d(S, v), d(S \dot{+} T, v)\}.$$

These formulae give the requisite background for developing a backtracking strategy to find the required graphs.

5.2 Description of the algorithm

The main idea is recursively to construct a connected partial graph G_q on q vertices from a connected partial graph G_{q-1} on $q-1$ vertices until $q = p$.

Let v_1, v_2, \dots, v_{q-1} denote the vertices chosen in that order to generate the sequence of graphs G_1, G_2, \dots, G_{q-1} . The q th vertex, v_q , is chosen from the list of 'available' and/or 'free' vertices (explained at the end of this section), each of which must be adjacent to some vertex in G_{q-1} . This ensures the connectedness of the partial graph G_q .

To ensure that eventually G_p is the required subgraph P , it must be assured that

$$d(G_q, \dot{\Sigma}S) \leq p - q .$$

In other words, the total length of the shortest forest from G_q to the sets in S^* must not exceed the number of additional vertices required to complete the construction from G_q to $G_p = P$.

Suppose there is a graph G_q but no vertex v_{q+1} that satisfies these conditions. The search backtracks to v_{q-1} , 'forbids' the current vertex chosen as v_q , and the search restarted with a new possible candidate for v_q . When all possible choices for v_q have been exhausted, the search backtracks from search level $q-1$ to search level $q-2$, releasing all vertices forbidden for level q . The most recent candidate for v_{q-1} is forbidden. A new candidate for v_{q-1} is chosen and the search proceeds until either a graph $G_p = P$ is found or eventually the search backtracks to the initial vertex. The initial vertices are chosen from the elements of a subset, say S_1 , in S^* .

Notice any vertex chosen as v_r is forbidden whenever all possible candidates for v_{r+1} have been exhausted. Consequently any vertex u in V chosen as v_r can never be chosen as $v_q, q > r$. Furthermore any vertex that does not satisfy the distance condition at search level r cannot possibly satisfy it at search level $q, q > r$; hence it may also be forbidden from further consideration. Suppose this were not so. Then at some level r

$$d(G_r, \dot{\Sigma}S) > p - r$$

and at level $r+1$

$$d(G_{r+1}, \dot{\Sigma}S) \leq p - r - 1 .$$

Clearly the shortest forest must contain a path from v_{r+1} to the sets in S^* . Since v_{r+1} is adjacent to a vertex in G_r , $d(G_r, \dot{\Sigma}S) = d(G_{r+1}, \dot{\Sigma}S) + 1$, which implies that $d(G_r, \dot{\Sigma}S) \leq p - r$, a contradiction. Whence the following result holds.

Lemma 4: $d(G_r, \dot{\Sigma}S) > p - r \Rightarrow d(G_{q > r}, \dot{\Sigma}S) > p - q .$

The vertices of the graph by this search procedure are always forbidden at the earliest possible level. Suppose u and v are vertices that are in the same choice set—that is, are possible candidates for v_q for some q . Let the algorithm choose $v_q = u$ and $v_{r > q} = v$. By the forbidding mechanism, whenever $v_q = v$ there is no $r > q$ such that $v_r = u$. Since this is always true for every pair of vertices in every choice set at every level q , and from the fact that every connected graph can be built from a smaller connected graph, the following lemma holds.

Lemma 5: Every subgraph is uniquely generated by the algorithm.

Let f_q be a Boolean function defined as follows:

$$f_q = 1 \text{ if and only if } G_q \text{ is connected and } d(G_q, \dot{\Sigma}S) \leq p - q .$$

Immediately as a corollary of lemma 4 we have the following.

Lemma 6: The sequence of functions (f_1, f_2, \dots) is well-defined—that is,

$$f_i = 1 \Rightarrow f_{k < i} = 1 \quad \text{and} \quad f_i = 0 \Rightarrow f_{k > i} = 0 .$$

Lemmas 3, 4, 5, and 6 result in the following theorem.

Theorem 1: *The algorithm is correct and generates every configuration exactly once.*

A vertex is ‘available’ if it is not currently in G_q but is adjacent to some vertex ($\neq v_q$) in G_q . When it is adjacent to v_q and no other vertex in G_q it is ‘free’.

5.3 Data structure

In order to implement the search scheme described in section 5.2, the use of an efficient data structure is necessary.

The graph is represented by a set of *adjacency lists*, one for each vertex in the graph. The adjacency list for a vertex v is a linked list containing all vertices adjacent to it, listed in random order, and is denoted by $A(v)$. The adjacency structure for a graph comprises all its adjacency lists and is represented by two arrays, *link* and *head*, as shown in figure 10. The order of the vertices in these arrays is again random. Adjacency lists involve only memory fetches for finding a neighbour and operate in $O(1)$ time per neighbour. The adjacency structure requires $O(|E|)$ space to store the graph.

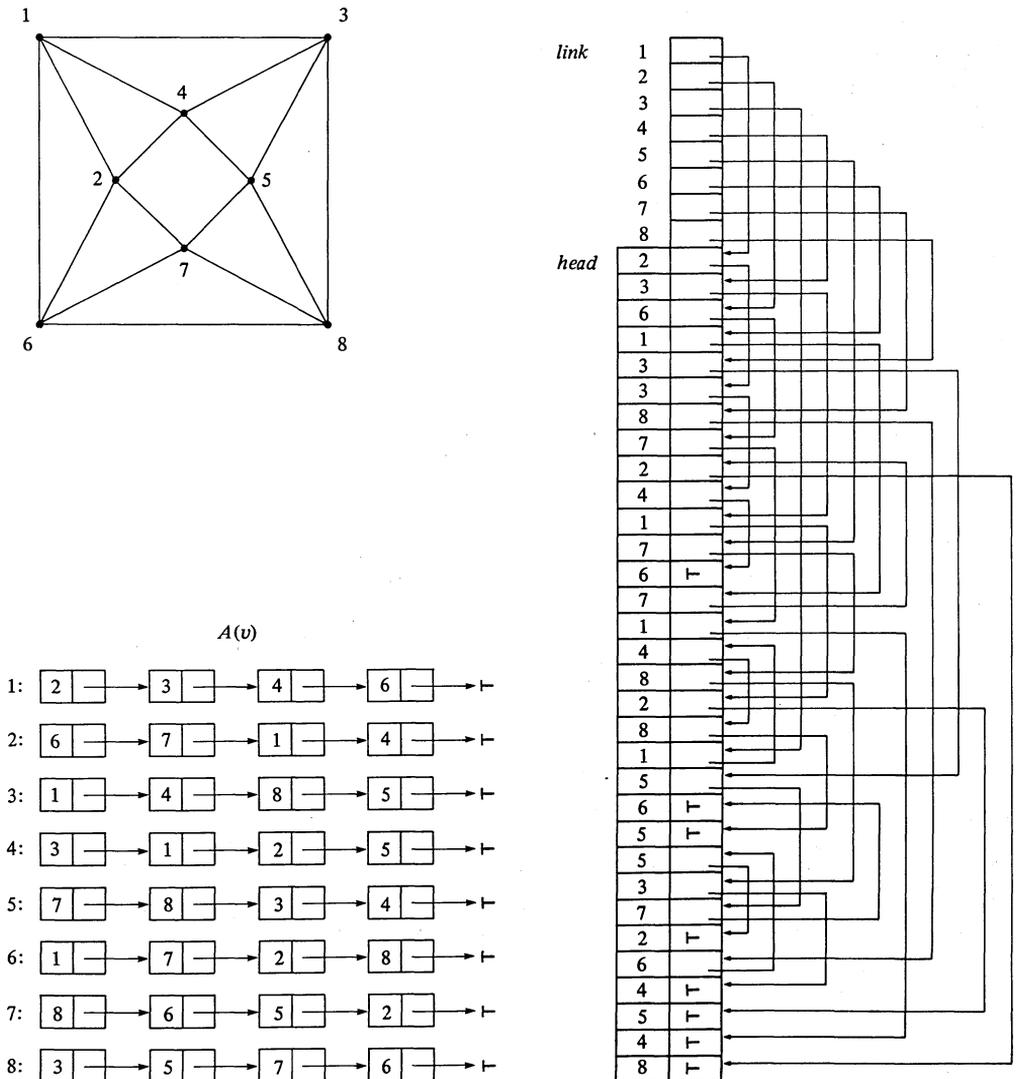


Figure 10.

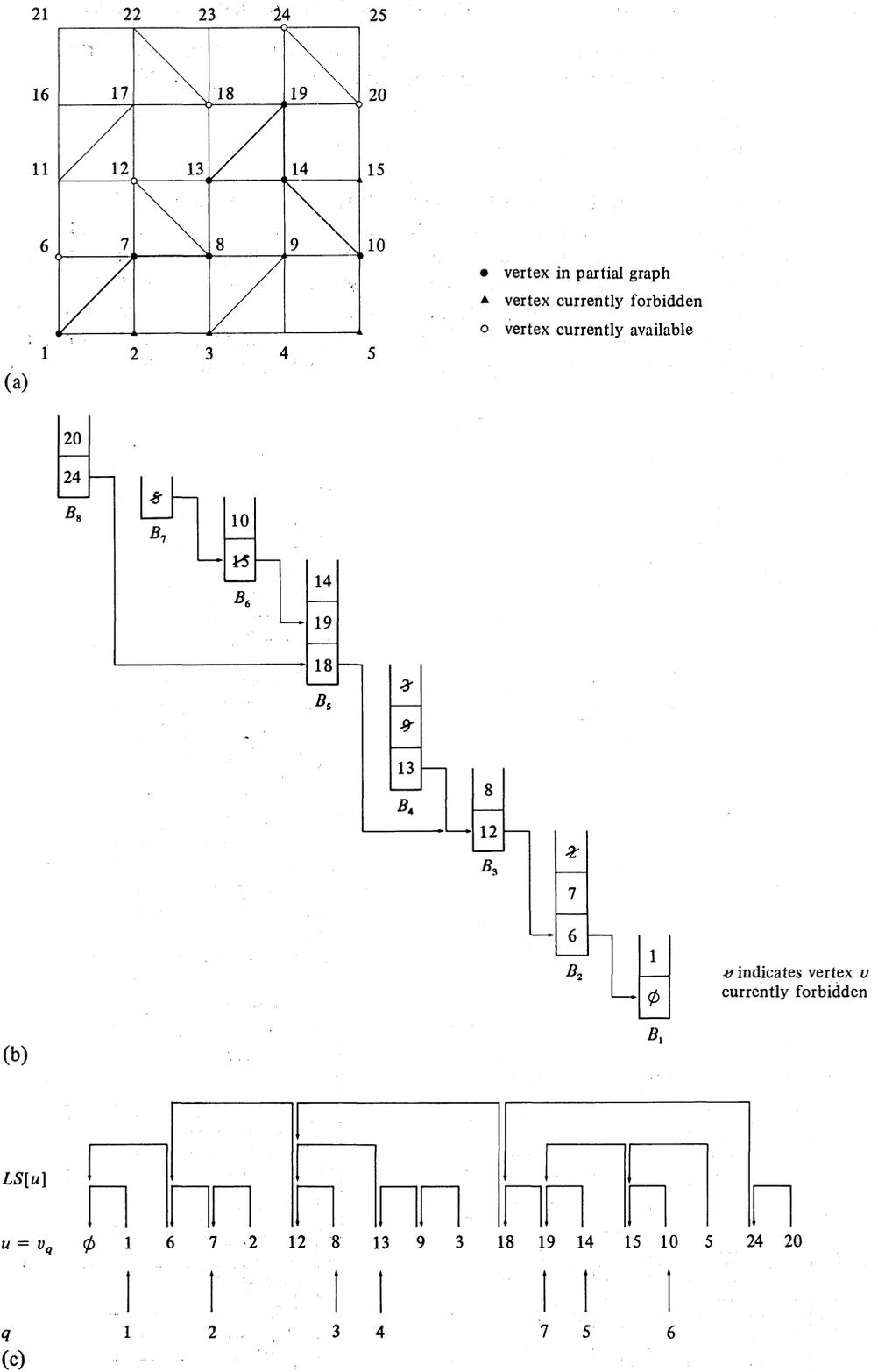


Figure 11.

A picture of the current description of the vertices is stored in an array, *picture*, which takes on values

$$picture[v] = \begin{cases} 0 & \text{if } v \text{ is free,} \\ -1 & \text{if } v \text{ is forbidden,} \\ +1 & \text{if } v \text{ is available.} \end{cases}$$

The current subgraph, G_q say, is represented by an array, *vertex*, where

$vertex[q]$ = the vertex in G_q chosen as the vertex to construct G_q from G_{q-1} .

Once the subset $vertex[1, \dots, q]$ is given, the edge set E_q can be easily constructed from the adjacency structure of G . Figure 11(a) presents one such G_q .

Since any candidate available at level r is also available as a candidate at level q , $q > r$, the list of candidates at level q contain as sublists lists of candidates for each level r , $r < q$. Let $C_1 = \{v_1 \in S_1 \in S^*\}$. Clearly the selection space for vertex v_2 is given by $C_2 = [C_1 \cup A(v_1)]$. In general $C_q = [C_{q-1} \cup A(v_{q-1})]$. Hence

$$C_p \supseteq C_{p-1} \supseteq C_{p-2} \supseteq \dots \supseteq C_1.$$

To avoid maintaining duplicate lists, a linked stack is employed. Essentially a linked stack is a linked list in which each node is a stack. The stacks may be defined as follows:

$$\begin{aligned} B_1 &= \{v_1 \in S_1 \in S^*\}, \\ B_2 &= \{v \in A(v_1): picture[v] = 0\} \\ &\vdots \\ B_q &= \{v \in A(v_{q-1}): picture[v] = 0\}. \end{aligned}$$

Clearly $C_1 = B_1$, $C_2 = B_1 \cup B_2$, $C_3 = B_1 \cup B_2 \cup B_3$, etc, where the B s are disjoint sets. Each set B_i resembles a stack and contains a pointer to a stack at a previous level. This is illustrated in figure 11(b).

This linked stack can be simulated by use of an array LS : if u is an unoccupied vertex adjacent to some vertex in the current subgraph G_q , $LS[u]$ is the 'next' candidate for vertex v_q . By 'next' is meant next in order of consideration as the q th vertex. The data structure for the linked stack LS is shown in figure 11(c).

The array LS is manipulated as follows. Suppose u is the current vertex chosen as v_q . We examine the adjacency list of u , $A(u)$, and disregard all vertices which are either forbidden ($picture = -1$) or are in LS and are available ($picture = +1$) and link the rest to the end of LS and set their $picture$ values to $+1$. The last vertex added to LS is selected as the $(q+1)$ th vertex v_{q+1} . Its $picture$ value is set to -1 and the search proceeds recursively forward. When the search backtracks, all possible G_{q+1} s obtainable from the current G_q have been explored, and we select the next choice for v_q stored in $LS[u]$ and set its $picture$ value to -1 . The process is repeated.

Eventually either a graph $G_p = P$ is found or v_q becomes *null*—that is, all possible choices for v_q have been exhausted. All vertices forbidden at level q are released to their original description and the search backtracks to the predecessor vertex, v_{q-1} .

5.4 Algorithms

The algorithms for problems 5.1(a) and 5.1(b) may be housed in the same algorithm. An ALGOL-like description is provided in algorithm 3. For problem 5.1(a) a halting condition is introduced after the first instance of $G_p = P$, indicated by the statement labelled A1. Certain observations can be made.

First, the procedure constructs all labelled graphs. To find the unlabelled graphs (those equivalent to the free equivalence class) we need to test whether the graph is canonical in the manner defined in section 4.5. This is included in statement A2,

where CANONICAL is a Boolean-valued procedure which returns a **true** or **false** value depending on whether the graph is canonical or not. A crude form of canonical testing using the word description of a configuration is shown in algorithm 4. This is improved on as we consider the individual examples.

Algorithm GRAPH(q)

```

¶ construct graph  $G_{q+1}$  from current partial graph  $G_q = (V_q, E_q)$  ¶
if  $q = p$ 
then {
  ¶  $G_p$  is a desired graph  $P$  ¶
  A1: if problem 5.1(a) then halt
  A2: if CANONICAL( $G_p$ ) then  $G_p$  is an unlabelled representative of its equivalence class
  B1: ¶  $navail$  indicates the number of unforbidden vertices in  $G$  ¶
      if  $navail > p - q$ 
          ¶ construct the stack  $B_{q+1}$ ;
          save and  $nl$  are locally declared variables ¶
          B2:  $nl \leftarrow 0$ 
           $bot \leftarrow LS[vertex[q]]$ 
          for  $w \in A(vertex[q])$  do
              if  $picture[w] = 0$ 
                  ¶ link  $w$  to bottom of  $LS$ ;
                   $bot$  points to current bottom of  $LS$  ¶
                  then {
                       $LS[w] \leftarrow bot$ 
                       $bot \leftarrow w$ 
                       $picture[w] \leftarrow 1$ 
                  }
          ¶ select the next candidate for  $v_{q+1}$  ¶
           $save \leftarrow bot$ 
          while  $bot \neq null$  do
              then {
                  B3:  $navail \leftarrow 1$ 
                   $nl \leftarrow 1$ 
                  if C:  $d(V_q \cup \{bot\}, \dot{\Sigma}S) \leq p - q - 1$ 
                      then {
                          D:
                           $vertex[q+1] \leftarrow bot$ 
                          E:
                          GRAPH( $q+1$ )
                      }
                   $bot \leftarrow LS[bot]$ 
                  ¶ restore all  $picture$  values to their original description ¶
                  while  $save \neq LS[vertex[q]]$  do
                      {  $picture[save] \leftarrow 0$ 
                         $save \leftarrow LS[save]$ 
                      }
                  B4:  $navail \leftarrow nl$ 
                   $bot \leftarrow vertex[q]$ 
              }
}
return
¶ invoking routine;
let  $G = (V, E)$  denote the graph;
vertex  $v_1$  is selected from some set, say  $S_1$ , in  $S^*$  ¶
B5:  $navail \leftarrow |V|$ 
for  $v \in S_1$  while  $navail > p$  do
    if  $picture[v] = 0$ 
        then {
             $bot \leftarrow vertex[1] \leftarrow v$ 
             $LS[v] \leftarrow null$ 
             $picture[v] \leftarrow 1$ 
            B6:  $navail \leftarrow 1$ 
            GRAPH(1)
            ¶ the following statements are included if unlabelled representatives are required;
             $T(G)$  is the group of symmetries for  $G$  ¶
            F: for  $\tau \in T(G)$  do
                if  $picture[\tau(v)] = 0$ 
                    then {  $picture[\tau(v)] \leftarrow 1$ 
                        B7:  $navail \leftarrow 1$ 
                    }
        }

```

Algorithm 3.

Second, notice we need only consider a two-valued description for the *picture* values:

$$picture[v] = \begin{cases} 1 & \text{if } v \text{ is available or forbidden,} \\ 0 & \text{otherwise.} \end{cases}$$

In other words, we need only recognize those 'free' vertices adjacent to vertex v_q at each search level q .

Third, the search tree may be further pruned. Let *navail* denote the number of unforbidden vertices—that is, the number of vertices that may be possible candidates for selection. Initially *navail* is set to the number of vertices in G , namely $|V|$. At each stage q the search can continue if and only if $navail \geq p - q$. Furthermore suppose we are interested in the free configurations. Let the vertices in $S_1 \in \mathcal{S}^*$ be labelled from 1 to $|S_1|$ inclusive. Under this condition, after every choice for v_1 has been made and all graphs G_p have been constructed from the current G_1 , we can forbid all the symmetry transforms of v_1 . The manipulation of *navail* is indicated in statements prefixed by the label B.

Statement C indicates the distance criterion that must be satisfied in order that the search may continue.

Statements D and E are dummies included for the various coding schemes developed in the individual examples.

Algorithm CANONICAL(G)

```

¶ bin is an array that stores successive powers of 2 ¶
flag ← true
¶ construct code for G ¶
code ← 0
for  $i \in \{1, \dots, p\}$  do code + ← bin[vertex[i] - 1]
¶ test isomorphic codes ¶
for  $\tau \in T(G)$  while flag do
    {
    isomorph ← 0
    for  $i \in \{1, \dots, p\}$  do isomorph + ← bin[ $\tau$ (vertex[i] - 1)]
    ¶  $\lesssim$  is an order relation (see section 4.5) ¶
    flag ← code  $\lesssim$  isomorph
    }
return (flag)

```

Algorithm 4.

In the discussion to follow we demonstrate how some of the archimedean configurations reduce to being solutions of appropriate graph problems of the type just explained.

6 Polyominoes

6.1 Bounding regions

Recall from section 4.3 that every p -omino is uniquely enclosed within a smallest rectangle. For a fixed p we may determine the set and number of bounding rectangles to house the family of p -ominoes. Let (l_1, l_2) denote the lengths of the sides of a rectangle measured along the x - and y -directions respectively. In order that l_1 and l_2 define a bounding rectangle for p -ominoes, p fixed, they must satisfy

$$l_1 + l_2 \leq p + 1, \quad (1)$$

$$l_1 l_2 \geq p, \quad (2)$$

$$l_1 \geq l_2. \quad (3)$$

Condition (1) is a restatement of the maximally stretched condition. Maximally stretched configurations are graphically equivalent to trees (see figure 12). Condition (2) states that if the bounding rectangle is pictured as a rectangular array of square cells then it must contain a sufficient number of such cells to house a p -omino. Condition (3) fixes the orientation in space of the bounding rectangle. Every (l_2, l_1) rectangle is a $\frac{1}{2}\pi$ rotation in the plane of an (l_1, l_2) rectangle. In other words every p -omino in an (l_1, l_2) rectangle corresponds to an isomorph in an (l_2, l_1) rectangle.

These conditions are plotted in figure 13. Based on this diagram the following can be shown.

6.1.1. The set of bounding regions for p -ominoes, p fixed, is given by

$$\Lambda_p^2 = \left\{ (l_1, l_2) : 1 \leq l_2 \leq \left\lceil \frac{p}{2} \right\rceil, \max \left\{ l_2, \left\lceil \frac{p}{l_2} \right\rceil \right\} \leq l_1 \leq p+1-l_2 \right\},$$

where, for any real number r , $\lceil r \rceil$ denotes the least integer greater than or equal to r .

6.1.2. The cardinality of Λ_p^2 is given by

$$n_p^2 = \lfloor \frac{1}{2}(p-1)^2 \rfloor + \frac{1}{2} \lfloor p^{1/2} \rfloor \lfloor p^{1/2} + 1 \rfloor - \sum_{k=1}^{\lfloor p^{1/2} \rfloor} \left\lceil \frac{p}{k} \right\rceil,$$

where, for any real number r , $\lfloor r \rfloor$ denotes the greatest integer less than or equal to r .

6.1.1. defines the algorithm for generating the bounding rectangles.

Suppose $(l_1, l_2) \in \Lambda_p^2$. Let $l_x = l_1 - 1$ and $l_y = l_2 - 1$. The trellis of this bounding rectangle is drawn in the plane with the vertices associated with integral coordinates (x, y) , $0 \leq x \leq l_x$, $0 \leq y \leq l_y$ (see figure 14). Label the vertices from left to right, bottom to top, in that order. That is, the label of the vertex (x, y) is given by $l_1 y + x + 1$. Define the sets of labelled points

$$\begin{aligned} S_1 &= \{1, 2, \dots, l_1\}, & S_3 &= \{l_1, 2l_1, 3l_1, \dots\}, \\ S_2 &= \{1, l_1 + 1, 2l_1 + 1, \dots\}, & S_4 &= \{l_1 l_2 - l_1 + 1, l_1 l_2 - l_1 + 2, \dots\}. \end{aligned}$$

Denote the collection of these sets by S^* .

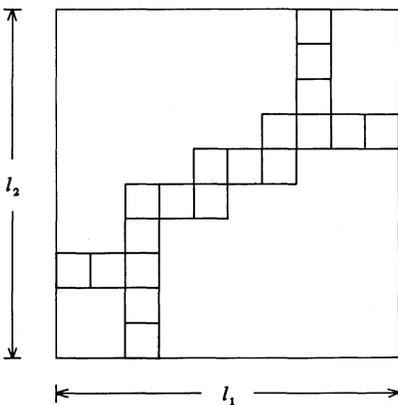


Figure 12.

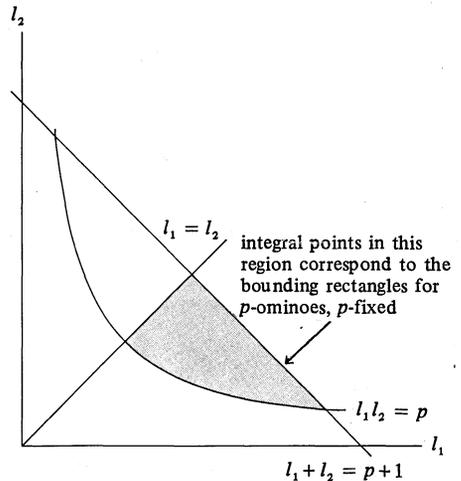


Figure 13.

It follows that the p -omino enumeration problem reduces to finding all unlabelled connected subgraphs on p vertices of the trellis such that each connected graph has a vertex in common with $S_1, S_2, S_3,$ and S_4 . Let (u_x, u_y) denote the coordinates of vertex u . Here

$$d(u, v) = \text{taxicab distance between two vertices } u \text{ and } v = |u_x - v_x| + |u_y - v_y| .$$

The other distance measures (see section 5.1.1) are given by

$$d(S_1, u) = u_y , \quad d(S_3, u) = l_x - u_x ,$$

$$d(S_2, u) = u_x , \quad d(S_4, u) = l_y - u_y ,$$

and

$$d(G_q, \dot{\Sigma}S) = \sum_i d(G_q, S_i) = \sum_i \min\{d(G_{q-1}, S_i), d(S_i, u)\} ,$$

where $V_q = V_{q-1} \cup \{u\}$.

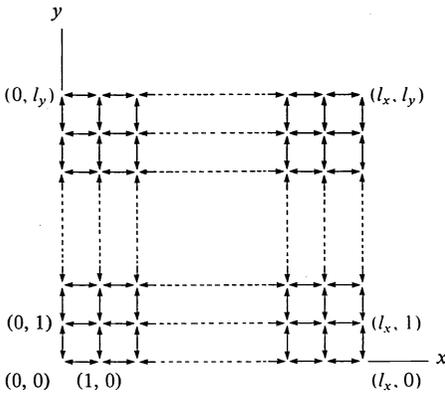


Figure 14.

6.2 Symmetries

The symmetry transformations under consideration are those that leave the trellis invariant in the plane. These transformations are isomorphic to the elements of D_4 , the dihedral group of order eight. For a trellis defined by the lines $x = 0, x = l_x, y = 0,$ and $y = l_y$, the coordinate-coordinate map representation of the transformations are presented in table 1. The last four entries may be applied only if $l_x = l_y$. Since the label of a vertex (x, y) is given by $l_1 y + x + 1$, the mappings are easily reduced to label-label maps. Computationally this allows us to define a table look-up for the symmetry transformations rather than have them computed each time a configuration is generated.

Alternatively we may use this coordinate map representation of the transformations to introduce the mechanism of synonym identification.

6.3 Synonym identification

The synonyms of a word are the words of its isomorphs. Let a word w be

$$\rho_1 x_1 + \rho_2 x_2 + \dots + \rho_{l_1 l_2} x_{l_1 l_2} .$$

Under a transformation τ the synonym w_τ is

$$\rho_1 x_{\tau(1)} + \rho_2 x_{\tau(2)} + \dots + \rho_{l_1 l_2} x_{\tau(l_1 l_2)} ,$$

or equivalently

$$\rho_{\tau^{-1}(1)}x_1 + \rho_{\tau^{-1}(2)}x_2 + \dots + \rho_{\tau^{-1}(l_1, l_2)}x_{l_1, l_2},$$

where τ^{-1} is the inverse mapping. For convenience w and w_τ are usually represented by binary strings of length $l_1 l_2$.

Imagine listing all words of length $l_1 l_2$ and content p . A word is canonical if it has no 'prior' synonym. By prior is meant lexicographically earlier. A p -omino is canonical if and only if its word is canonical.

As an example consider the 7-omino 110001100111 in a (4, 3) bounding rectangle [see figure 15(a)]. Applying the transformations $i, v, h,$ and π we have the synonyms as shown.

The rules for synonym identification are easily derived. A word is partitioned into l_2 subwords each of length l_1 :

$$(\rho_1 \rho_2 \dots \rho_{l_1})(\rho_{l_1+1} \dots \rho_{2l_1}) \dots (l_1, l_2 - l_1 + 1 \dots \rho_{l_1, l_2}).$$

When $l_x \neq l_y$, synonyms are produced by

- 6.3.1. (a) reversing the sequence of terms in each subword,
- (b) reversing the sequence of subwords,
- (c) by both (a) and (b) together.

There are at most four synonyms when $l_x \neq l_y$. Similar rules apply when $l_x = l_y$, in which case there are at most eight synonyms per word. The production of synonyms for the 7-omino of figure 15(a) is illustrated in figure 15(b).

Table 1.

Group element	Symbol	Coordinate-coordinate map: $(x, y) \rightarrow$
1 Identity	i	(x, y)
2 Reflection about $y = \frac{1}{2}l_y$	h	$(x, l_y - y)$
3 Reflection about $x = \frac{1}{2}l_x$	v	$(l_x - x, y)$
4 Rotation through π	π	$(l_x - x, l_y - y)$
5 Rotation through $\frac{1}{2}\pi$	$\frac{1}{2}\pi$	$(l_y - y, x)$
6 Rotation through $-\frac{1}{2}\pi$	$\frac{1}{2}\pi$	$(y, l_x - x)$
7 Reflection about $x = y$	R	(y, x)
8 Reflection about $x = -y$	r	$(l_y - y, l_x - x)$

} these apply only when $l_x = l_y$

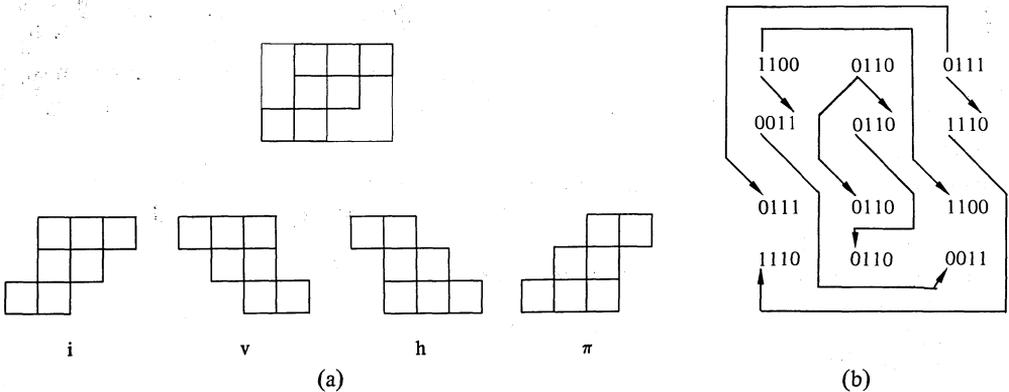


Figure 15.

6.4 Canonical word

In this section we expand the idea of synonym identification. Let the current word be partitioned as before. Each subword is the word of a row of the bounding rectangle. Each subword is an l_1 -bit expansion of a positive integer. Thus every configuration is associated with an l_2 -tuple of positive integers $(\gamma_1, \gamma_2, \dots, \gamma_{l_2})$ and uniquely by the $(l_2 + 1)$ -tuple $(l_1, \gamma_1, \gamma_2, \dots, \gamma_{l_2})$. However, when the trellis is unambiguously known the prefix l_1 may be dropped. As an example consider the 4-omino 111010 in a $(3, 2)$ rectangle. It has the 2-tuple code $(7, 2)$.

Let Γ denote the current word,

$$\Gamma = (\gamma_1, \gamma_2, \dots, \gamma_{l_2}) .$$

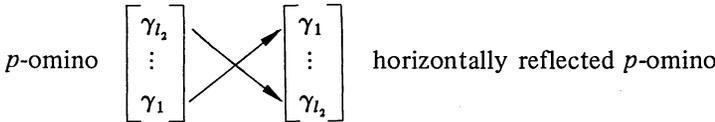
We have two cases to consider.

Case 1: $l_1 \neq l_2$

Applying rule 6.3.1(b) we get the l_2 -tuple

$$\Gamma_h = (\gamma_{l_2}, \gamma_{l_2-1}, \dots, \gamma_1) .$$

This corresponds to reflecting the p -omino about a horizontal axis ($y = \frac{1}{2}l_y$). The rule is illustrated as follows.



Applying rule 6.3.1(a) we get the l_2 -tuple

$$\Gamma_v = (\tilde{\gamma}_1, \tilde{\gamma}_2, \dots, \tilde{\gamma}_{l_2}) ,$$

where $\tilde{\gamma}_i$ is the integer representation of the binary sequence for γ_i read in reverse. Finally rule 6.3.1(c) gives

$$\Gamma_\pi = (\tilde{\gamma}_{l_2}, \tilde{\gamma}_{l_2-1}, \dots, \tilde{\gamma}_1) .$$

Γ_v and Γ_π represent a reflection about a vertical axis ($x = \frac{1}{2}l_x$) and a rotation through π respectively.

Case 2: $l_1 = l_2$

In addition to the preceding rules we do the following. Transpose the bounding region to obtain the word

$$\rho'_1 \rho'_2 \dots \rho'_{l_1 l_2} .$$

Here the word w' is formed by reading the columns of the original bounding region from bottom to top, left to right, in that order. Partition this word as before to obtain the l_2 -tuple

$$\Gamma_R = (\gamma'_1, \gamma'_2, \dots, \gamma'_{l_2}) .$$

Γ_R represents a reflection about the principal diagonal ($x = y$). Applying rules 6.3.1(a), 6.3.1(b), and 6.3.1(c) we get respectively the l_2 -tuples

$$\Gamma_{\frac{1}{2}\pi} = (\gamma'_{l_2}, \gamma'_{l_2-1}, \dots, \gamma'_1) ,$$

$$\Gamma_{\frac{3}{2}\pi} = (\tilde{\gamma}'_1, \tilde{\gamma}'_2, \dots, \tilde{\gamma}'_{l_2}) ,$$

and

$$\Gamma_I = (\tilde{\gamma}'_{l_2}, \tilde{\gamma}'_{l_2-1}, \dots, \tilde{\gamma}'_1) ,$$

where the subscripts of Γ indicate the transformations involved, namely a rotation through $\frac{1}{2}\pi$, a rotation through $-\frac{1}{2}\pi$, and a reflection about $x = -y$ respectively.

6.4.1. The relations $<$, $=$, and \leq on any two k -tuples $A = (a_1, a_2, \dots, a_k)$ and $B = (b_1, b_2, \dots, b_k)$ are defined as follows:

- (a) $A < B$ if and only if there exists a $j \leq k$ such that $a_j < b_j$ and, for all $i < j$, $a_i = b_i$;
 (b) $A = B$ if and only if, for all $i \leq k$, $a_i = b_i$;
 (c) $A \leq B$ if $A < B$ or $A = B$.

6.4.2. A word Γ is canonical if and only if it has the 'highest' code among all its synonyms:

$$\Gamma \geq \Gamma_\tau, \quad \tau \in \{h, v, \pi\}, \quad \text{if } l_1 \neq l_2;$$

$$\Gamma \geq \Gamma_\tau, \quad \tau \in \{h, v, \pi, \frac{1}{2}\pi, \overline{\frac{1}{2}\pi}, R, r\}, \quad \text{if } l_1 = l_2.$$

6.5 Algorithm

The algorithm is essentially the same as algorithm 3 with the following modifications. Statements A1, A2, D, and E are replaced by:

A: ¶ Let p -omino denote the number of free equivalence classes of polyominoes with content p ¶
 if CANONICAL then p -omino $+$ \leftarrow 1

¶ Let v_{q+1} denote $vertex[q+1]$ ¶

D: $\gamma_y[v_{q+1}] \leftarrow 2^{l_x - x[v_{q+1}]}$
 $\tilde{\gamma}_y[v_{q+1}] \leftarrow 2^{x[v_{q+1}]}$
 if $l_x = l_y$
 then $\left\{ \begin{array}{l} \gamma'_x[v_{q+1}] \leftarrow 2^{l_y - y[v_{q+1}]} \\ \tilde{\gamma}'_x[v_{q+1}] \leftarrow 2^{y[v_{q+1}]} \end{array} \right.$

E: $\gamma_y[v_{q+1}] \leftarrow 2^{l_x - x[v_{q+1}]}$
 $\tilde{\gamma}_y[v_{q+1}] \leftarrow 2^{x[v_{q+1}]}$
 if $l_x = l_y$
 then $\left\{ \begin{array}{l} \gamma'_x[v_{q+1}] \leftarrow 2^{l_y - y[v_{q+1}]} \\ \tilde{\gamma}'_x[v_{q+1}] \leftarrow 2^{y[v_{q+1}]} \end{array} \right.$

CANONICAL is a Boolean-valued procedure that tests whether or not the generated p -omino is the representative of its free equivalence class. The routine is similar to algorithm 4 except it is based on the ideas in section 6.4. D is a coding statement executed once for each addition of a vertex to the current graph. E is a decoding statement executed once after each backtrack. The arrays x and y house the coordinates for the vertices. It is clear that every configuration is generated in $O(p)$ time and that the trellis dominates storage with $O(p^2)$ space. It should be noticed that for computational convenience Γ is represented by the l_2 -tuple $(\gamma_0, \gamma_1, \gamma_y, \dots, \gamma_{l_y})$.

7 Polyhexes and polyiamonds

7.1 Bounding regions

Since polyiamonds are a two-colourable subset of polyhexes it is convenient to consider only the polyhexes. Lunnon (1972) has shown that every polyhex is contained in a bounding hexagon defined by the lines $x = a$, $x = a'$, $y = b$, $y = b'$, $z = c$, and $z = c'$, where $a' \geq a$, $b' \geq b$, and $c' \geq c$. Observe that this bounding hexagon is defined on its trellis drawn in the plane, with the vertices associated with the appropriate integral coordinate system (see section 4.2). A polyhex and its bounding hexagon are shown in figure 16.

The bounding hexagon can be described by four parameters—*diameters* l_x, l_y, l_z and *skew* χ —defined as follows. $l_x = a' - a, l_y = b' - b,$ and $l_z = c' - c.$ Let $l_a, l_{a'},$ etc denote the lengths of the sides of the hexagon lying on the lines $x = a, x = a',$ etc. From figure 16 and by use of the fact that the points on and inside the hexagon must satisfy $x + y + z = 0,$ it can be shown that

$$l_a - l_{a'} = l_b - l_{b'} = l_c - l_{c'} = a + a' + b + b' + c + c' = \text{a constant} ,$$

which is referred to as the skew, $\chi.$ It should be noticed that these parameters are integral.

Any bounding hexagon may be oriented in space in such a way that $l_x \geq l_y \geq l_z.$ Moreover, since the lengths of the sides of the hexagon are always nonnegative, $|\chi| \leq l_z.$ It is also possible to arrange the bounding hexagon in space such that χ is always greater than or equal to zero. In other words, the orientation of a bounding hexagon may be so *fixed* that

$$l_x \geq l_y \geq l_z \geq \chi \geq 0 .$$

We may determine other relationships involving the parameters $l_x, l_y, l_z,$ and $\chi.$ For instance it is easy to show that

$$l_x + l_y + l_z + \chi = 2(a' + b' + c') = 0 \pmod 2 .$$

Since $\chi \geq 0$ it follows that $l_a \geq l_{a'}, l_b \geq l_{b'},$ and $l_c \geq l_{c'}.$ The lengths of the sides may be expressed in the following terms:

$$0 \leq l_{a'} = \frac{1}{2}(-\chi - l_x + l_y + l_z) \Rightarrow l_y + l_z - \chi \geq l_x . \tag{4}$$

$$0 \leq l_{b'} = \frac{1}{2}(-\chi + l_x - l_y + l_z) \Rightarrow l_x + l_z - \chi \geq l_y . \tag{5}$$

$$0 \leq l_{c'} = \frac{1}{2}(-\chi + l_x + l_y - l_z) \Rightarrow l_x + l_y - \chi \geq l_z . \tag{6}$$

Conditions (5) and (6) are not independent of the fixing condition for the bounding hexagon.

Consider an open region in the plane defined by the lines $x = 0$ and $y = 0$ such that every point in this region has $x, y \leq 0.$ Clearly $z \geq 0$ in this region. Place a bounding hexagon inside this region. By moving this hexagon towards $(0, 0, 0)$ it is possible to align the hexagon such that $a' = b' = 0.$ These are the *spanning conditions.*

To sum up, every set of integral parameters $l_x, l_y, l_z,$ and χ such that

$$l_y + l_z - \chi \geq l_x \geq l_y \geq l_z \geq \chi \geq 0 ,$$

with

$$l_x + l_y + l_z + \chi = 0 \pmod 2 ,$$

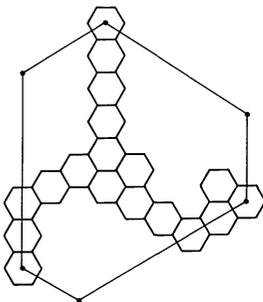


Figure 16.

uniquely determines a fixed bounding hexagon defined by the lines $x = 0, x = -l_x, y = 0, y = -l_y, z = c,$ and $z = c',$ where $c' = \frac{1}{2}(l_x + l_y + l_z + \chi)$ and $c = c' - l_z.$

We now consider the bounding hexagons for the population of polyhexes with content $p.$ Since the maximally stretched p -hex is the linear polyhex [figure 17(a)] it follows that $l_x \leq p - 1.$ From the maximal-stretching condition [see figure 17(b)] it can be shown that

$$l_x + l_y + l_z + \chi \leq 2(p - 1).$$

Since every bounding hexagon must contain at least p vertices, we have

$$p - 1 \leq -\frac{1}{4}[l_x^2 + l_y^2 + l_z^2 + \chi^2 - 2(l_x l_y + l_y l_z + l_z l_x + l_x + l_y + l_z)].$$

We may also show that

$$l_z = 0 \Rightarrow l_x = l_y = p - 1,$$

$$l_x = p - 1 \Rightarrow l_y + l_z = p - 1,$$

$$0 \leq \chi \leq (p - 1) - l_x.$$

Finally, since $l_x \leq p - 1,$ every fixed bounding hexagon may be enclosed within the triangular region which satisfies $-p < x, y \leq 0$ and $0 \leq z < p$ (figure 18). This triangular region plays an important role in developing our enumeration algorithms.

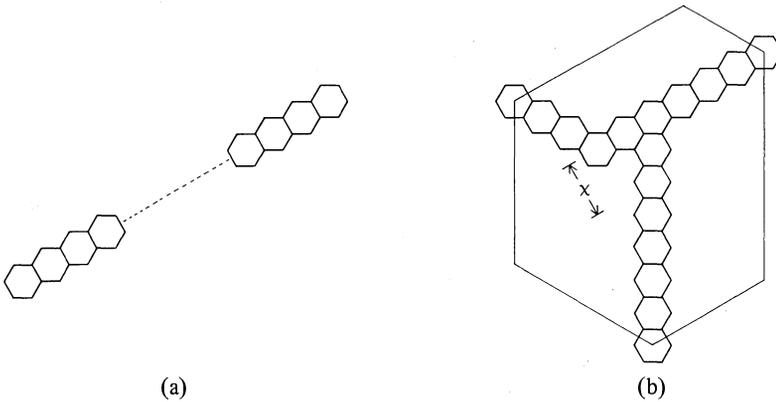


Figure 17.

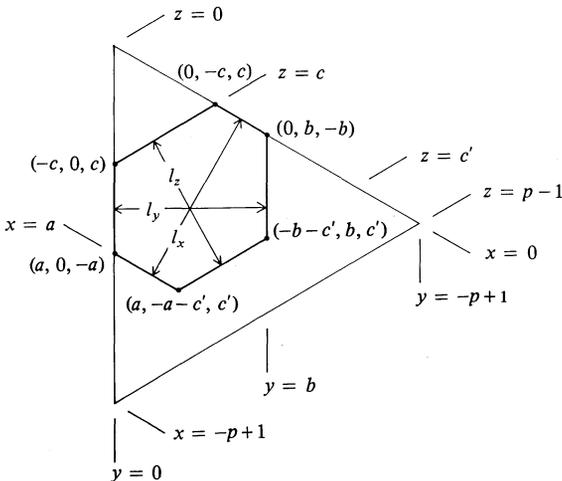


Figure 18.

7.2 Coding scheme

Consider the triangular region with vertices labelled from 1 to $\binom{p+1}{2}$ as shown in figure 19. Here every vertex (x, y, z) satisfies $x+y+z = 0$ together with $-p < x, y \leq 0$ and $0 \leq z < p$. This region may be described by the word

$$x_1 + x_2 + \dots + x_{\binom{p+1}{2}}.$$

Every configuration in this region has a word of the form

$$\rho_1 x_1 + \rho_2 x_2 + \dots + \rho_{\binom{p+1}{2}} x_{\binom{p+1}{2}}.$$

where $\rho_i \in \{0, 1\}$ for all i . This is more conveniently expressed as the binary string

$$\rho_1 \rho_2 \dots \rho_{\binom{p+1}{2}}.$$

Partition this into subwords of the form

$$(\rho_1)(\rho_2 \rho_3)(\rho_4 \rho_5 \rho_6) \dots \left(\rho_{\binom{p+1}{2}-p+1} \dots \rho_{\binom{p+1}{2}} \right).$$

Each subword may be expressed as a positive integer. Thus every configuration is associated with a p -tuple

$$\Delta = (\delta_{-p+1}, \delta_{-p+2}, \dots, \delta_0) \leq (1, 3, 7, \dots, 2^p - 1),$$

where δ_i corresponds to the contributions of the vertices on the line $x = i$. Clearly $\delta_i \leq 2^{p-i} - 1$.

Every bounding hexagon is a subword of the word of the triangular region of the form

$$\Delta = (0, 0, \dots, 0, \delta_{-l_x}, \dots, \delta_0),$$

which can be pared down to a $(l_x + 1)$ -tuple in the obvious way.

A p -hex is a linear polynomial over \mathbb{Z}_2 with content p of the word of its bounding hexagon. Each p -hex has a word of the form

$$\Gamma = (\gamma_{-l_x}, \gamma_{-l_x+1}, \dots, \gamma_0).$$

Each vertex (x, y, z) in the polyhex contributes

$$2^{-y} \text{ to } \gamma_x.$$

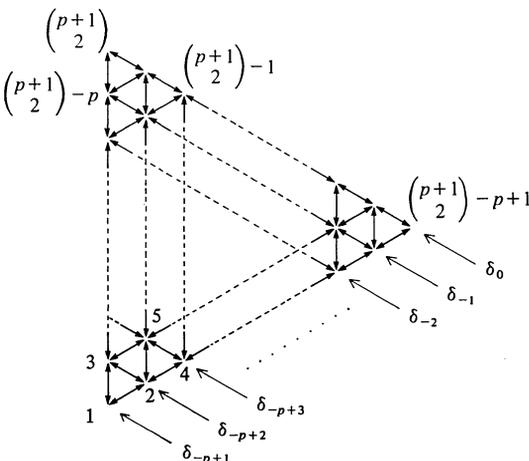


Figure 19.

A p -hex is a representative of its free equivalence class if and only if

$$\Gamma \geq \Gamma_\tau, \quad \tau \in T_h,$$

where T_h is the group of symmetries that leave the bounding hexagon h invariant in the plane and Γ_τ is the code of the isomorph under τ .

This coding scheme applies equally to the polyiamonds.

7.3 Symmetries

The symmetry transformations that leave a given bounding hexagon invariant in the plane are isomorphic to the elements of D_6 , the dihedral group of order twelve. The symmetry motions for a regular hexagon are as follows:

- d_1 identity
- d_2 rotation through π ,
- d_3 vertical reflection about x -axis,
- d_4 vertical reflection about y -axis,
- d_5 vertical reflection about z -axis,
- d_6 horizontal reflection about x -axis,
- d_7 horizontal reflection about y -axis,
- d_8 horizontal reflection about z -axis,
- d_9 rotation through $\frac{2}{3}\pi$,
- d_{10} rotation through $\frac{4}{3}\pi$,
- d_{11} rotation through $\frac{1}{3}\pi$,
- d_{12} rotation through $\frac{5}{3}\pi$.

All axes of rotation and reflection pass through the centre and are illustrated in figure 20. For a bounding hexagon defined by the lines $x = 0$, $x = a$, $y = 0$, $y = b$, $z = c$, and $z = c'$ we need only consider certain subgroups of D_6 . The following possibilities arise.

7.3.1.(a) $l_x \neq l_y \neq l_z$, $\chi \neq 0$.

In this case the bounding hexagon possesses only the identity symmetry. That is, all polyhexes are canonically generated within this type of bounding hexagon.

7.3.1.(b) $l_x \neq l_y \neq l_z$, $\chi = 0$.

Here $l_a = l'_a$, $l_b = l'_b$, and $l_c = l'_c$. In this case a rotation through π (d_2) also leaves the hexagon invariant in the plane.

7.3.2.(a) $l_x = l_y$, $\chi \neq 0$.

Here $l_a = l_b$, $l'_a = l'_b$, and $a = b$. The hexagon remains invariant under a vertical reflection about the z -axis (d_5). Of course the hexagon is invariant under identity (d_1).

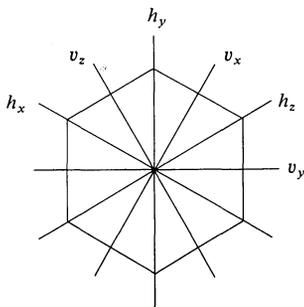


Figure 20.

7.3.2.(b) $l_x = l_y, \chi = 0$.

Here $l_a = l_b = l'_a = l'_b, a = b$, and $l_c = l'_c$. Reflection about the z -axis such that sides l_c and l'_c are interchanged gives the transformation d_8 . It is also invariant under d_1, d_2 , and d_5 .

In a similar fashion the remaining cases can be completed. We will be content to state the properties of the bounding hexagon and the elements that leave the bounding hexagon invariant.

7.3.3.(a) $l_y = l_z, \chi \neq 0$.

$l_b = l_c, l'_b = l'_c$, and $b = c - c'$; d_1 and d_3 .

7.3.3.(b) $l_y = l_z, \chi = 0$.

$l_b = l'_b = l_c = l'_c, b = c - c'$, and $l_a = l'_a$; d_1, d_2, d_3 , and d_6 .

7.3.4.(a) $l_x = l_z, \chi \neq 0$.

$l_a = l_c, l'_a = l'_c$, and $a = c - c'$; d_1 and d_4 .

7.3.4.(b) $l_x = l_z, \chi = 0$.

$l_a = l'_a = l_c = l'_c, a = c - c'$, and $l_b = l'_b$; d_1, d_2, d_4 , and d_7 .

7.3.5.(a) $l_x = l_y = l_z, \chi \neq 0$.

$l_a = l_b = l_c, l'_a = l'_b = l'_c, a = b = c - c'$; d_1, d_3, d_4, d_5, d_9 , and d_{10} .

7.3.5.(b) $l_x = l_y = l_z, \chi = 0$.

$l_a = l_b = l_c = l'_a = l'_b = l'_c, a = b = c - c'$; all elements of D_6 .

The symmetry elements may be described as coordinate-coordinate mappings. The preceding list of transformations is summarized in tables 2 and 3. Table 2 gives the coordinate map representation for the elements of D_6 . Table 3 gives the subgroups of D_6 that apply depending upon the conditions that are imposed on a *fixed* bounding hexagon.

In the case of polyiamonds the transformations apply only if 'colouring' is preserved; that is, the transformations must imply an isomorphism of the three colours (given by $x - y \pmod 3$) onto themselves. Since vertex adjacency is preserved by any symmetry

Table 2.

Group element	Symbol	Coordinate-coordinate map: $(x, y, z) \rightarrow$
1 Identity	i	(x, y, z)
2 Rotation through π	π	$(a-x, b-y, c+c'-z)$
Vertical reflection about the axis that bisects:		
3 the x lines	v_x	$(x, -c'+z, c'+y)$
4 the y lines	v_y	$(-c'+z, y, c'+x)$
5 the z lines	v_z	(y, x, z)
Horizontal reflection about:		
6 $x = -c$	h_x	$(a-x, c-z, c-y)$
7 $y = -c$	h_y	$(c'+a-z, b-y, c'+a-x)$
8 $z = -a = -b$	h_z	$(b-y, a-x, c+c'-z)$
9 Rotation through $\frac{2}{3}\pi$	$\frac{2}{3}\pi$	$(-c'+z, x, c'+y)$
10 Rotation through $-\frac{2}{3}\pi$	$\frac{2}{3}\pi$	$(y, -c'+z, c'+x)$
11 Rotation through $\frac{1}{3}\pi$	$\frac{1}{3}\pi$	$(b-y, c-z, c'+a-x)$
12 Rotation through $-\frac{1}{3}\pi$	$\frac{1}{3}\pi$	$(c'+a-z, b-x, c-y)$

motion that leaves a bounding hexagon invariant, this will always hold. Consequently this enables any two-colourable polyhex to be transformed into an equivalent two-colourable polyhex—or, a polyiamond into an equivalent polyiamond.

It is interesting to note that we need only consider the x and y coordinates of the transformations for coding purposes. For instance, if (x, y, z) is a vertex in a polyhex and if $\Gamma_{\frac{1}{3}\pi}$, the isomorphic polyhex under the transformation $\frac{1}{3}\pi$ (see table 2), is represented by the $(l_x + 1)$ -tuple $(\gamma'_{-l_x}, \gamma'_{-l_x+1}, \dots, \gamma'_0)$, then the vertex contributes

$$2^{z-c} = 2^{-x-y-c} \text{ to } \gamma'_{b-y} .$$

Table 3.

Conditions on l_x, l_y, l_z	$\chi > 0$	$\chi = 0$
$l_x > l_y > l_z$	i	i, π
$l_x > l_y = l_z$	i, v_x	i, π, v_x, h_x
$l_x = l_y > l_z$	i, v_z	i, π, v_z, h_z
$l_x = l_y = l_z$	$i, v_x, v_y, v_z, \frac{2}{3}\pi, \frac{2}{3}\pi$	all

7.4 Algorithms

There are essentially two possible approaches to enumerating polyhexes. Both employ the same method, namely determining the connected subgraphs of order p of the graph (trellis) of the bounding hexagon. The first approach applies algorithm 3, and it is left as an exercise to the reader to determine the appropriate distance condition to be applied at statement C. It must be noticed that moving along an edge in any direction changes two of the three coordinates; consequently, if $u = (u_x, u_y, u_z)$ and $v = (v_x, v_y, v_z)$ are two vertices,

$$d(u, v) = \frac{1}{2}(|u_x - v_x| + |u_y - v_y| + |u_z - v_z|) .$$

The second approach views this enumeration problem in another light. Let us regard the triangular region as a ‘super trellis’, and the problem reduces to finding a connected graph on p vertices and the corresponding trellis this graph spans. The same algorithm, namely algorithm 3, is applicable with the following modifications. The search is always started by selecting as the initial vertex a vertex whose x -coordinate is 0. From section 7.1 there is only one p -hex which contains the vertex $(0, -p+1, p-1)$. (Since $y = 0$ must be a line of the fixed bounding hexagon, $l_y = p-1$ in this case.) Consequently we need only start the search with vertices whose z -coordinate is less than $p-1$ and whose x -coordinate equals 0. We can forbid $(0, -p+1, p-1)$ from further consideration.

The distance conditions must be so chosen that eventually the bounding hexagon satisfies all the orientation conditions described earlier. It can be shown that the following three conditions are required. Let $l_x^q, l_y^q, l_z^q, \chi^q$, etc denote the diameters and skew etc at the q th recursion level. Let $\check{x}, \hat{x}, \check{y}, \hat{y}, \check{z}, \hat{z}$ denote the minimums and maximums of the corresponding coordinates of the current configuration depending on whether the coordinate is capped by a $\check{}$ or a $\hat{}$ respectively. Then

$$l_x^q = -\min\{\check{x}(G_{q-1}), x(v_q)\} = -\check{x}(G_q) ,$$

$$l_y^q = [\hat{y}(G_q) = \max\{\hat{y}(G_{q-1}), y(v_q)\}] - [\check{y}(G_q) = \min\{\check{y}(G_{q-1}), y(v_q)\}] ,$$

$$l_z^q = [\hat{z}(G_q) = \max\{\hat{z}(G_{q-1}), z(v_q)\}] - [\check{z}(G_q) = \min\{\check{z}(G_{q-1}), z(v_q)\}] ,$$

and

$$\chi^q = \check{x}(G_q) + \check{y}(G_q) + \hat{y}(G_q) + \check{z}(G_q) + \hat{z}(G_q) .$$

Statement C in algorithm 3 is replaced by

$$\max\{0, l_y^q - l_x^q\} - \hat{y}(G_q) \leq p - q, \tag{7}$$

which will eventually ensure that $l_x \geq l_y$ and $y = 0$ is a line of the bounding hexagon,

$$\max\{0, l_y^q - l_x^q\} + \max\{0, l_z^q - l_y^q + \hat{y}(G_q)\} \leq p - q, \tag{8}$$

which together with condition (1) will eventually ensure that $l_y \geq l_z$, and

$$\max\{0, \max\{l_y^q, l_z^q\} - l_x^q\} + \max\{0, -\chi^q\} \leq p - q, \tag{9}$$

which together with conditions (7) and (8) will eventually ensure that $l_y + l_z - \chi \geq l_x$ and $\chi \geq 0$. If conditions (7), (8), and (9) are satisfied, v_q is in G_q and the search may proceed recursively forward.

For polyiamonds the colours of the first two vertices selected are noted and the remaining vertices are chosen from these two colours. If the colour of v_1 is c_1 and that of v_2 is c_2 then the third or forbidden colour is $-(c_1 + c_2) \bmod 3$. Colour c_2 must be different from c_1 since v_2 is adjacent to v_1 .

8 Poly n -cubes

Poly n -cubes, $n \geq 3$, are the last examples of configurations on the regular tessellations considered in this paper. The enumeration of poly n -cubes follows in exactly the same fashion as the enumeration of polyominoes. We outline briefly the steps involved.

8.1 Bounding regions

For poly n -cubes the bounding region is a n -rectangle of dimensions l_1, l_2, \dots, l_n . The necessary and sufficient conditions for any n -rectangle defined by the n -tuple (l_1, l_2, \dots, l_n) to be a bounding region for the population of p - n -cubes are:

$$\sum_i l_i \leq p + n - 1 \quad \text{maximal-stretching condition;} \tag{10}$$

$$\prod_i l_i \geq p \quad \text{sufficiency of unit } n\text{-cubes;} \tag{11}$$

$$l_1 \geq l_2 \geq \dots \geq l_n \geq 1 \quad \text{fixing the orientation of the bounding region.} \tag{12}$$

These can be solved in the same manner as was done for polyominoes. For example, it can be shown that

$$1 \leq l_n \leq \left\lfloor \frac{p+n-1}{n} \right\rfloor.$$

Consider the case when $n = 3$; then

$$1 \leq l_3 \leq \left\lfloor \frac{p+2}{3} \right\rfloor.$$

A diagram similar to figure 13 is drawn in figure 21. It describes the conditions when l_3 is equal to a fixed value, say k . From this diagram the following can be demonstrated.

8.1.1. The set of triples (l_1, l_2, l_3) which correspond to the bounding regions for p -cubes, p fixed, is given by

$$\Lambda_p^3 = \left\{ (l_1, l_2, l_3): 1 \leq l_3 \leq \left\lfloor \frac{p+2}{3} \right\rfloor, l_3 \leq l_2 \leq \left\lfloor \frac{p+2-l_3}{2} \right\rfloor, \max\left\{l_2, \left\lfloor \frac{p}{l_2 l_3} \right\rfloor\right\} \leq l_1 \leq p+2-l_2-l_3 \right\}.$$

8.1.2. The cardinality of Λ_p^3 is

$$n_p^3 = \sum_{j=1}^{\lfloor p^{1/2} \rfloor} \left\{ \left\lfloor \frac{(p+4-3j)^2}{4} \right\rfloor + \frac{1}{2} \left\lfloor \left(\frac{p}{j} \right)^{1/2} \right\rfloor \left[\left\lfloor \left(\frac{p}{j} \right)^{1/2} \right\rfloor + 1 \right] - \sum_{i=j}^{\lfloor (p/j)^{1/2} \rfloor} \left\lfloor \frac{p}{ij} \right\rfloor \right\} + \sum_{j=\lfloor p^{1/2} \rfloor + 1}^{\lfloor (p+2)^{3/2} \rfloor} \left\lfloor \frac{(p+4-3j)^2}{4} \right\rfloor.$$

The extension to arbitrary n -space is obvious. To see why just fix l_n at some value, say k . Substituting this in conditions (10) through (12) reduces it to finding an $(n-1)$ -tuple of integers (l_1, \dots, l_{n-1}) with $l_{n-1} \geq k$. This process may be repeated inductively to obtain the set Λ_p^n given as follows.

8.1.3. The set of n -tuples (l_1, l_2, \dots, l_n) which correspond to the bounding regions for p - n -cubes, p fixed, is given by

$$\Lambda_p^n = \left\{ (l_1, l_2, \dots, l_n): 1 \leq l_n \leq \left\lfloor \frac{p+n-1}{n} \right\rfloor, l_n \leq l_{n-1} \leq \left\lfloor \frac{p+n-1-l_n}{n-1} \right\rfloor, \right. \\ \left. l_{n-1} \leq l_{n-2} \leq \left\lfloor \frac{p+n-1-l_n-l_{n-1}}{n-2} \right\rfloor, \dots, l_3 \leq l_2 \leq \left\lfloor \frac{p+n-1-\sum_{j \geq 3} l_j}{2} \right\rfloor, \right. \\ \left. \max \left\{ l_2, \left\lfloor \frac{p}{\prod_{i \geq 2} l_i} \right\rfloor \right\} \leq l_1 \leq p+n-1 - \sum_{i \geq 2} l_i \right\}.$$

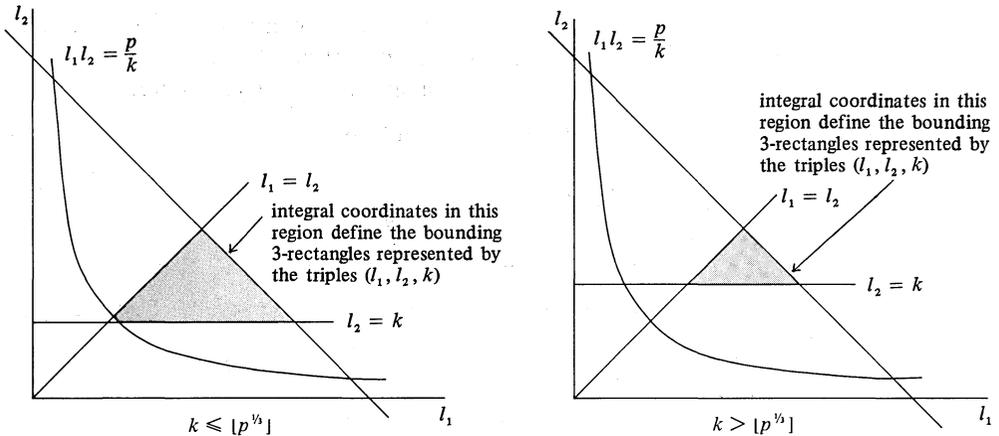


Figure 21.

8.2 Coding scheme

The bounding region for p -cubes is drawn in figure 22 as a trellis with vertices associated with integral coordinates. Recall that each vertex corresponds to a unit cube. Also $l_x = l_1 - 1$, $l_y = l_2 - 1$, and $l_z = l_3 - 1$.

The word representing this region can be partitioned into a $l_2 l_3$ -tuple of integers as follows:

$$\Gamma = \begin{matrix} k = 0 & k = 1 & \dots & k = l_z \\ (j = 0, \dots, j = l_y) & (j = 0, \dots, j = l_y) & \dots & (j = 0, \dots, j = l_y) \end{matrix} \\ x = 0 \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right] \dots \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right] \dots \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right] \dots \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right] \\ x = 1 \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right] \dots \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right] \dots \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right] \dots \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right] \\ \vdots \\ x = l_x \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right] \dots \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right] \dots \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right] \dots \left[\begin{matrix} \dots \\ \dots \\ \dots \end{matrix} \right].$$

This may be written as

$$\Gamma = (\gamma_{00}, \gamma_{01}, \dots, \gamma_{0l_y}, \dots, \gamma_{zy}, \dots, \gamma_{l_z0}, \dots, \gamma_{l_zl_y}) ,$$

where each γ_{zy} represents a column of the previous expression and

$$0 \leq \gamma_{zy} \leq 2^l - 1, \quad 0 \leq z \leq l_z, \quad 0 \leq y \leq l_y .$$

The coding operator is defined as follows: each vertex (x, y, z) in the connected subgraph of the trellis representing a p -cube contributes

$$2^{l_x - x} \text{ to } \gamma_{zy} .$$

Clearly the $l_2 l_3$ -tuple representation of a p -cube is unique, and from it the p -cube is uniquely decipherable or reconstructible.

The γ s may be represented by the array $code[1, \dots, l_2 l_3]$. Define the array $loc[0, \dots, l_z]$ as follows:

$$loc[1] = 1 ,$$

$$loc[z] = loc[z - 1] + l_2, \quad z \geq 2 .$$

Then the computational step required to code a vertex (x, y, z) is given by

$$code[loc[z] + y] + \leftarrow 2^{l_x - x} .$$

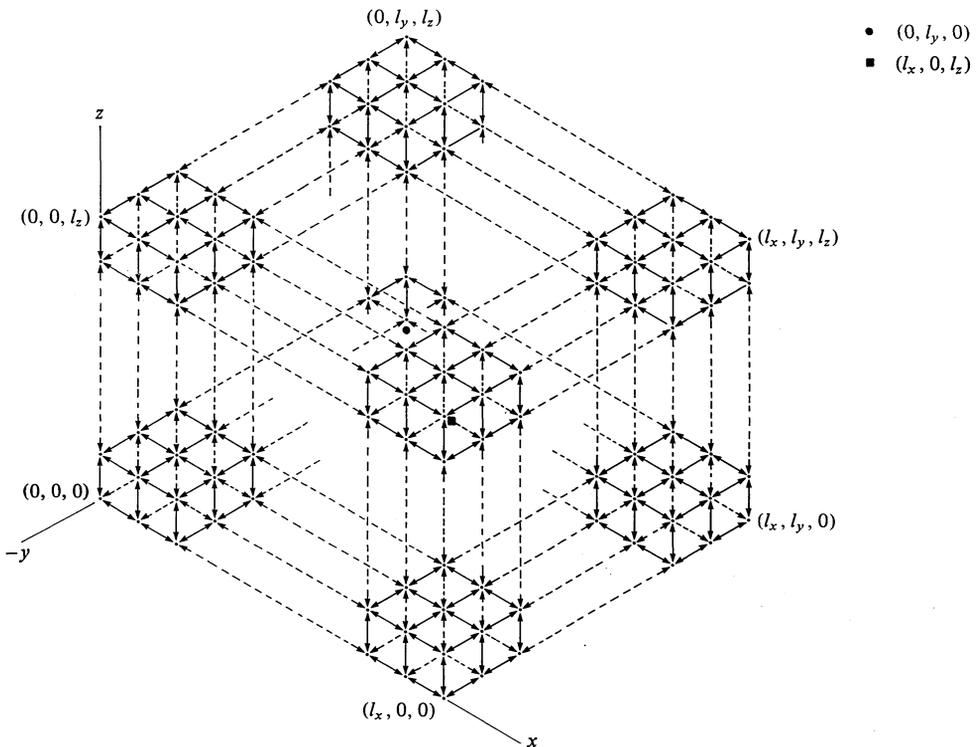


Figure 22.

8.3 Canonical word and symmetries

A word is canonical if and only if $\Gamma \geq \Gamma_\tau, \tau \in T_t$, where T_t is a group of symmetries that leave the bounding region invariant. T_t is usually a small-order subgroup of the even subgroup of O_n , the orthogonal group of rotations of order $2^n(n!)$. For $n = 3$ consider the cube in figure 23, wherein the various axes of rotations are indicated. The even elements of O_3 are given in table 4. The minimum conditions which must be satisfied in order that an element may be applied are also presented.

For $n > 3$ the group elements no longer correspond to simple axial rotations. Instead they are rotations about planes. As shown by Littlewood (1931), the group of rotations, O_n , for the n -dimensional hypercuboid can be obtained from the following generators.

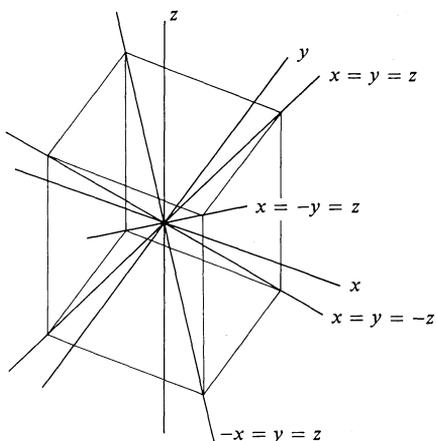


Figure 23.

Table 4.

Group element	Symbol	Coordinate-coordinate map: $(x, y, z) \rightarrow$
1 Identity	i	(x, y, z)
Rotation through π about:		
2 the x -axis	π_x	$(x, l_y - y, l_z - z)$
3 the y -axis	π_y	$(l_x - x, y, l_z - z)$
4 the z -axis	π_z	$(l_x - x, l_y - y, z)$
Rotation through $\pm \frac{1}{2}\pi$ about:		
5 } the x -axis	$\left\{ \begin{matrix} \frac{1}{2}\pi_x \\ \frac{1}{2}\pi_x \end{matrix} \right\}$	$\left. \begin{matrix} (x, z, l_y - y) \\ (x, l_z - z, y) \end{matrix} \right\} l_y = l_z$
6 } the y -axis	$\left\{ \begin{matrix} \frac{1}{2}\pi_y \\ \frac{1}{2}\pi_y \end{matrix} \right\}$	$\left. \begin{matrix} (z, y, l_x - x) \\ (l_z - z, y, x) \end{matrix} \right\} l_z = l_x$
7 } the z -axis	$\left\{ \begin{matrix} \frac{1}{2}\pi_z \\ \frac{1}{2}\pi_z \end{matrix} \right\}$	$\left. \begin{matrix} (y, l_x - x, z) \\ (l_y - y, x, z) \end{matrix} \right\} l_x = l_y$
8 } the z -axis	$\left\{ \begin{matrix} \frac{1}{2}\pi_z \\ \frac{1}{2}\pi_z \end{matrix} \right\}$	$\left. \begin{matrix} (y, l_x - x, z) \\ (l_y - y, x, z) \end{matrix} \right\} l_x = l_y$
9 } the z -axis	$\left\{ \begin{matrix} \frac{1}{2}\pi_z \\ \frac{1}{2}\pi_z \end{matrix} \right\}$	$\left. \begin{matrix} (y, l_x - x, z) \\ (l_y - y, x, z) \end{matrix} \right\} l_x = l_y$
10 } the z -axis	$\left\{ \begin{matrix} \frac{1}{2}\pi_z \\ \frac{1}{2}\pi_z \end{matrix} \right\}$	$\left. \begin{matrix} (y, l_x - x, z) \\ (l_y - y, x, z) \end{matrix} \right\} l_x = l_y$
Compositions of the above:		
11 $(\frac{1}{2}\pi_x)\pi_z = (\frac{1}{2}\pi_x)\pi_y$	c_{xy}	$(l_x - x, l_z - z, l_y - y)$
12 $(\frac{1}{2}\pi_x)\pi_y = (\frac{1}{2}\pi_x)\pi_z$	c_{xz}	$(l_x - x, z, y)$
13 $(\frac{1}{2}\pi_y)\pi_x = (\frac{1}{2}\pi_y)\pi_z$	c_{yz}	$(z, l_y - y, x)$
14 $(\frac{1}{2}\pi_y)\pi_z = (\frac{1}{2}\pi_y)\pi_x$	c_{yx}	$(l_z - z, l_y - y, l_x - x)$
15 $(\frac{1}{2}\pi_z)\pi_y = (\frac{1}{2}\pi_z)\pi_x$	c_{zx}	$(l_y - y, l_x - x, l_z - z)$
16 $(\frac{1}{2}\pi_z)\pi_x = (\frac{1}{2}\pi_z)\pi_y$	c_{zy}	$(y, x, l_z - z)$
Rotations through $\pm \frac{1}{3}\pi$ about:		
17 } the $x = y = z$ diagonal	$\left\{ \begin{matrix} r_{xyz} \\ \bar{r}_{xyz} \end{matrix} \right\}$	$\left. \begin{matrix} (z, x, y) \\ (y, z, x) \end{matrix} \right\}$
18 } the $-x = y = z$ diagonal	$\left\{ \begin{matrix} r_{\bar{x}yz} \\ \bar{r}_{\bar{x}yz} \end{matrix} \right\}$	$\left. \begin{matrix} (l_y - y, z, l_x - x) \\ (l_z - z, l_x - x, y) \end{matrix} \right\}$
19 } the $x = -y = z$ diagonal	$\left\{ \begin{matrix} r_{x\bar{y}z} \\ \bar{r}_{x\bar{y}z} \end{matrix} \right\}$	$\left. \begin{matrix} (l_y - y, l_z - z, x) \\ (z, l_x - x, l_y - y) \end{matrix} \right\}$
20 } the $x = -y = z$ diagonal	$\left\{ \begin{matrix} r_{x\bar{y}z} \\ \bar{r}_{x\bar{y}z} \end{matrix} \right\}$	$\left. \begin{matrix} (l_y - y, l_z - z, x) \\ (z, l_x - x, l_y - y) \end{matrix} \right\}$
21 } the $x = -y = z$ diagonal	$\left\{ \begin{matrix} r_{x\bar{y}z} \\ \bar{r}_{x\bar{y}z} \end{matrix} \right\}$	$\left. \begin{matrix} (l_y - y, l_z - z, x) \\ (z, l_x - x, l_y - y) \end{matrix} \right\}$
22 } the $x = -y = z$ diagonal	$\left\{ \begin{matrix} r_{x\bar{y}z} \\ \bar{r}_{x\bar{y}z} \end{matrix} \right\}$	$\left. \begin{matrix} (l_y - y, l_z - z, x) \\ (z, l_x - x, l_y - y) \end{matrix} \right\}$
23 } the $x = y = -z$ diagonal	$\left\{ \begin{matrix} r_{xy\bar{z}} \\ \bar{r}_{xy\bar{z}} \end{matrix} \right\}$	$\left. \begin{matrix} (y, l_z - z, l_x - x) \\ (l_z - z, x, l_y - y) \end{matrix} \right\}$
24 } the $x = y = -z$ diagonal	$\left\{ \begin{matrix} r_{xy\bar{z}} \\ \bar{r}_{xy\bar{z}} \end{matrix} \right\}$	$\left. \begin{matrix} (y, l_z - z, l_x - x) \\ (l_z - z, x, l_y - y) \end{matrix} \right\}$

8.3.1. Axial inversion:

$$R_i: \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ l_{x_i} - x_i \\ \vdots \\ x_n \end{bmatrix}.$$

8.3.2. Axial interchange:

$$Q_{ij}: \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_j \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix}, \quad l_{x_i} = l_{x_j}.$$

An element is *even* if and only if the number of axial inversions and interchanges is even. The R_i form a group, R_n^* , of order 2^n . The Q_{ij} form a group, Q_n^* , which is isomorphic to the symmetric group of order $n!$, S_n , the group of the permutations of

Table 5.

Group element	Symbol	Coordinate-coordinate map: (x, y, z, w) \rightarrow	Parity (0 = even, 1 = odd)
1 Identity	i	(x, y, z, w)	0
Axial inversion along:			
2 the x -axis	R_x	($l_x - x, y, z, w$)	1
3 the y -axis	R_y	($x, l_y - y, z, w$)	
4 the z -axis	R_z	($x, y, l_z - z, w$)	
5 the w -axis	R_w	($x, y, z, l_w - w$)	
Two axial inversions in:			
6 the xy -plane	R_{xy}	($l_x - x, l_y - y, z, w$)	0
7 the xz -plane	R_{xz}	($l_x - x, y, l_z - z, w$)	
8 the xw -plane	R_{xw}	($l_x - x, y, z, l_w - w$)	
9 the yz -plane	R_{yz}	($x, l_y - y, l_z - z, w$)	
10 the yw -plane	R_{yw}	($x, l_y - y, z, l_w - w$)	
11 the zw -plane	R_{zw}	($x, y, l_z - z, l_w - w$)	
Three axial inversions in:			
12 the constant- w plane	R_{xyz}	($l_x - x, l_y - y, l_z - z, w$)	1
13 the constant- z plane	R_{xyw}	($l_x - x, l_y - y, z, l_w - w$)	
14 the constant- y plane	R_{xzw}	($l_x - x, y, l_z - z, l_w - w$)	
15 the constant- x plane	R_{yzw}	($x, l_y - y, l_z - z, l_w - w$)	
16 Complete axial inversion of the 4-rectangle	R_{xyzw}	($l_x - x, l_y - y, l_z - z, l_w - w$)	0

n elements. To find the elements of O_n , take the product of R_n^* and Q_n^* under group composition. The even elements correspond to products of elements of the same parity (that is either both even or both odd). It can be shown that the following are satisfied:

$$\begin{aligned}
 R_i^2 &= Q_{ij}^2 = \text{identity}, \\
 R_i R_j &= R_j R_i, \\
 Q_{ij} &= Q_{ji}, \\
 Q_{ij} Q_{jk} &= Q_{ik} Q_{ij} = Q_{jk} Q_{ik}, \\
 R_i Q_{ij} &= Q_{ij} R_j, \\
 R_j Q_{ij} &= Q_{ij} R_i, \\
 R_i Q_{jk} &= Q_{jk} R_i, \quad i \neq j, i \neq k.
 \end{aligned}$$

For $n = 4$ the groups R_4^* and Q_4^* are presented in tables 5 and 6 respectively. The elements of R_4^* are designated by the combination of axial inversions involved. For instance R_{ijk} is equivalent to $R_i R_j R_k$. The elements of Q_4^* are designated by the cyclic notation for the corresponding permutations of the numbers 1, 2, 3, and 4.

Table 6.

Group element	Symbol	Coordinate-coordinate map: $(x, y, z, w) \rightarrow$	Conditions under which they apply	Parity (0 = even, 1 = odd)		
1 Identity	i	(x, y, z, w)		0		
Axial interchange						
2	Q_{xy}	(y, x, z, w)	$ \left. \begin{aligned} l_x &= l_y \\ l_x &= l_z \\ l_x &= l_w \\ l_y &= l_z \\ l_y &= l_w \\ l_z &= l_w \end{aligned} \right\} $	1		
3	Q_{xz}	(z, y, x, w)				
4	Q_{xw}	(w, y, z, x)				
5	Q_{yz}	(x, z, y, w)				
6	Q_{yw}	(x, w, z, y)				
7	Q_{zw}	(x, y, w, z)				
Pairs of axial interchanges						
8	Q_{xyz}	(z, x, y, w)	$ \left. \begin{aligned} l_x &= l_y = l_z \\ l_x &= l_y = l_w \\ l_x &= l_z = l_w \\ l_y &= l_z = l_w \\ (l_x = l_y) \wedge (l_z = l_w) \\ (l_x = l_z) \wedge (l_y = l_w) \\ (l_x = l_w) \wedge (l_y = l_z) \end{aligned} \right\} $	0		
9	Q_{zyx}	(y, z, x, w)				
10	Q_{xyw}	(w, x, z, y)				
11	Q_{xwy}	(y, w, z, x)				
12	Q_{xzw}	(w, y, x, z)				
13	Q_xwz	(z, y, w, x)				
14	Q_{yzw}	(x, w, y, z)				
15	$Q_{y wz}$	(x, z, w, y)				
16	$Q_{(xy)(zw)}$	(y, x, w, z)				
17	$Q_{(xz)(yw)}$	(z, w, x, y)				
18	$Q_{(xw)(yz)}$	(w, z, y, x)				
Three axial interchanges						
19	Q_{xyzw}	(w, x, y, z)			$l_x = l_y = l_z = l_w$	1
20	Q_{xzyw}	(w, z, x, y)				
21	Q_{xwyz}	(z, w, y, x)				
22	Q_{xywz}	(z, x, w, y)				
23	$Q_{xzw y}$	(y, w, x, z)				
24	Q_{xwzy}	(y, z, w, x)				

8.4 Algorithm

The graph problem may be formulated as was done for the polyominoes. For $n = 3$ the spanning sets are given by

$$S_1 = \{(x, y, z): z = 0\}, \quad S_3 = \{(x, y, z): y = 0\}, \quad S_5 = \{(x, y, z): x = 0\},$$

$$S_2 = \{(x, y, z): z = l_z\}, \quad S_4 = \{(x, y, z): y = l_y\}, \quad S_6 = \{(x, y, z): x = l_x\}.$$

Let $u = (u_x, u_y, u_z)$ and $v = (v_x, v_y, v_z)$ be two vertices. The distance measures are given by

$$d(u, v) = |u_x - v_x| + |u_y - v_y| + |u_z - v_z|$$

and

$$d(S_1, u) = u_z, \quad d(S_3, u) = u_y, \quad d(S_5, u) = u_x,$$

$$d(S_2, u) = l_z - u_z, \quad d(S_4, u) = l_y - u_y, \quad d(S_6, u) = l_x - u_x,$$

with

$$d(G_q, \dot{\Sigma}S) = \sum d(G_q, S), \quad \text{for any } G_q.$$

The algorithm is then the same as algorithm 3, but with the appropriate modifications. For $n > 3$ the distance measures are similarly defined.

We conclude by observing that every poly n -cube in an $(l_1, l_2, \dots, l_k, 1, 1, \dots, 1)$ bounding n -rectangle corresponds to a poly k -cube in the bounding k -rectangle (l_1, l_2, \dots, l_k) . Thus, for instance, if we have the list of polyominoes, we need only augment this list by determining those polycubes in $(l_1, l_2, l_3 \geq 2)$ bounding regions to obtain the total set of polycubes.

9 <4.8.8>-patterns

Consider the square tessellation. Apply the following exchange operation in the manner indicated in figure 24:



The result is the <4.8.8>-tessellation with associated integral coordinates—namely the integral Cartesian coordinates in the plane. Observe that every diagonal consists of exactly one kind of tile: either all octagons or all squares. This can be further strengthened. Designate any tile in the tessellation as the origin. Then the octagons have coordinates (x, y) such that either all have $x+y$ even or all have $x+y$ odd. For a given parity of the octagons, the squares have the opposite parity.

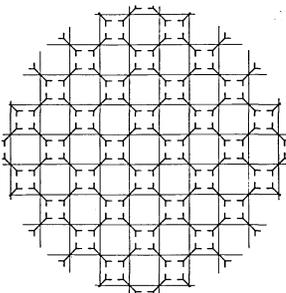


Figure 24.

9.1 Bounding regions

Every <4.8.8>-pattern can be encased in a rectangle. Let (l_1, l_2) denote the sides of the rectangle. Set the tile at the bottom left-hand corner as the origin. The vertices of the corresponding trellis have coordinates that satisfy $0 \leq x \leq l_x = l_1 - 1$ and $0 \leq y \leq l_y = l_2 - 1$. There are two types of bounding regions depending on which type of tile is associated with the origin (see figure 25). Fixing $l_1 \geq l_2$ defines an initial orientation for the bounding rectangles. Moreover, in order to obtain distinct bounding rectangles—that is, so that no type 1 rectangle transforms into a corresponding type 2 rectangle or vice versa—it must be ensured that, if the bounding rectangles of one type are permitted all combinations for the parity of l_1 and l_2 , then the rectangles of the other type must have l_1 and l_2 odd. Let (l_1, l_2, t) denote a bounding rectangle of type t . Then the following can be shown.

9.1.1. The set of bounding rectangles for the <4.8.8>-patterns with content p is given by

$$\psi_p = \left\{ (l_1, l_2): 1 \leq l_2 \leq p, \max\left\{\left\lceil \frac{p}{l_2} \right\rceil, l_2\right\} \leq l_1 \leq p \right\},$$

$${}_1\psi_p = \{(l_1, l_2, 1): (l_1, l_2) \in \psi_p\},$$

$${}_2\psi_p = \{(l_1, l_2, 2): (l_1, l_2) \in \psi_p - \{(p, p)\}; l_1, l_2 \text{ odd}\}.$$

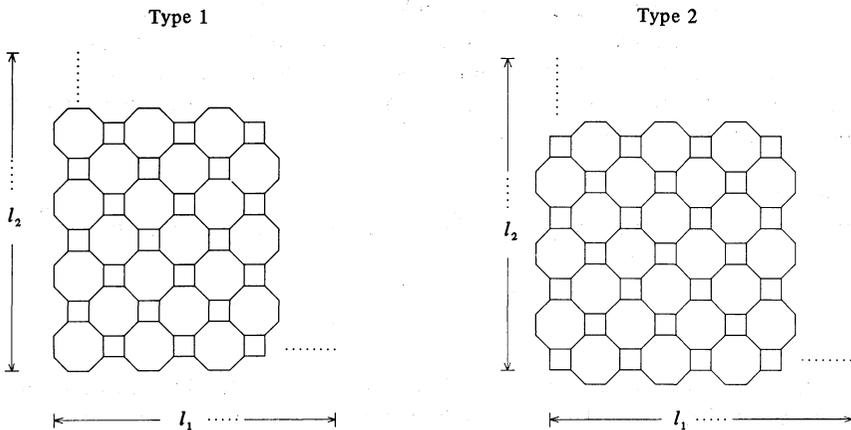


Figure 25.

9.2 Trellis

The trellises for the bounding rectangles of both types are shown in figure 26, with the vertices associated with the integral coordinates. Let (x, y) denote a vertex.

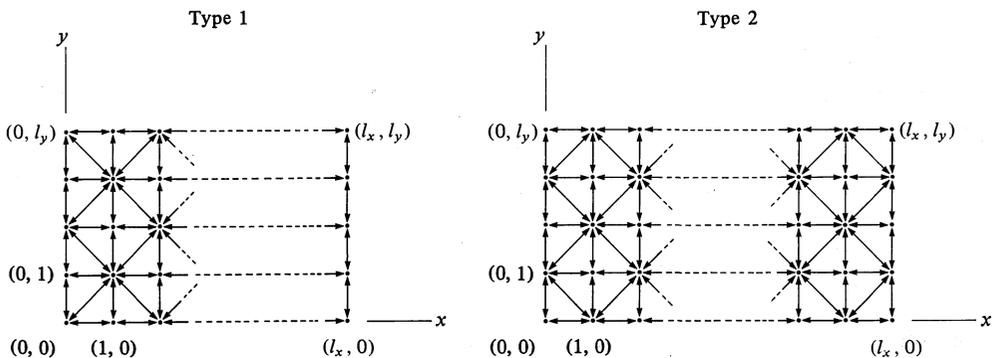


Figure 26.

Define sets N_1 and N_2 as follows:

$$N_1(x, y) = \{(x+1, y), (x-1, y), (x, y-1), (x, y+1)\},$$

$$N_2(x, y) = \{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\}.$$

If (x, y) represents an octagon, the adjacent vertices have coordinates given by $N_1(x, y) \cup N_2(x, y)$, where the vertices in N_1 represent squares and those in N_2 represent octagons. If (x, y) represents a square its neighbouring vertices are given by N_1 , which represents only octagons.

9.3 Symmetries

The symmetry transformations that apply correspond to the elements of D_4 , the dihedral group of order eight, described as coordinate-coordinate maps in table 1. Table 7 describes the elements that apply for the different conditions on the bounding rectangles.

Table 7.

Parity (0 = even, 1 = odd)		Elements of D_4 which apply
l_1	l_2	
0	0	i, π
0	1	i, h
1	0	i, v
1	1	$\begin{cases} i, h, v, \pi & \text{if } l_1 > l_2 \\ \text{all} & \text{if } l_1 = l_2 \end{cases}$

9.4 Algorithm

It should be apparent how the (4.8.8)-patterns of content p reduce to a graph problem of type 5.1(b). The set S^* is defined in the same manner as was done for the polyominoes. The chief difficulty is in calculating $d(G_q, \dot{\Sigma} S)$ at each stage q .

If the initial vertex is always chosen from set S_1 , then $d(G_q, \dot{\Sigma}_{i \geq 2} S) \leq \sum_{i \geq 2} d(G_q, S_i)$.

We have

$$d(G_q, S_2 \dot{+} S_4) = \min \left\{ d(G_{q-1}, S_2 \dot{+} S_4), \max \{ u_x, l_y - u_y \}, \right. \\ \left. d(G_{q-1}, S_2) + l_y - u_y, d(G_{q-1}, S_4) + u_x \right\},$$

$$d(G_q, S_3 \dot{+} S_4) = \min \left\{ d(G_{q-1}, S_3 \dot{+} S_4), \max \{ l_y - u_y, l_x - u_x \}, \right. \\ \left. d(G_{q-1}, S_3) + l_y - u_y, d(G_{q-1}, S_4) + l_x - u_x \right\},$$

and

$$d(G_q, \dot{\Sigma} S) = \min \left\{ d(G_{q-1}, \dot{\Sigma} S), d(G_q, S_3 \dot{+} S_4) + d(G_q, S_2), \right. \\ \left. d(G_q, S_2 \dot{+} S_4) + d(G_q, S_3), \sum_{i \geq 2} d(G_q, S_i) \right\}.$$

For all $i \geq 2$, $d(S_i, v_q)$ and $d(G_q, S_i)$ are defined in the same way as was done for the polyominoes. For $q = 1$

$$d(G_q, \dot{\Sigma} S) = \min \left\{ \max \{ l_y, l_x - u_x \} + u_x, \max \{ l_y, u_x \} + l_x - u_x \right\},$$

where $v_1 = u = (u_x, u_y)$.

The distance criterion in the statement labelled C in algorithm 3 is replaced by the formulae given here. The search can be reduced by noting that whenever $l_1 l_2 = p$ there is only one, obvious, pattern, and when $l_1 = l_2 = p$ then again there is only one, obvious, pattern for type 1 rectangles. Otherwise the algorithm is identical to that for polyomino enumerations.

10 (3.3.4.4)-patterns

We may regard the (3.3.3.4.4)-tessellation as alternating strips of squares and triangles as shown in figure 27. Suppose the strips containing the triangles are uniformly distorted by the simultaneous application of one of the following exchange operations:



or



The result is a tessellation consisting of alternating strips of squares and divided squares. In other words the exchange operations map the (3.3.3.4.4)-tessellation onto the square grid. Clearly there are several possible ways to define this mapping, each depending upon the orientation of the tessellation and the square grid. For convenience we will assume the mapping and the orientation shown in figure 28. For the remainder of the discussion the triangles will assume the orientation indicated.

Since every square, simple or divided, occupies a square in the grid, we may associate integer Cartesian coordinates with each. To distinguish between the tile types—that is, between the squares and oriented triangles—we introduce a third coordinate referred to as a *tile designator*. The tile designator σ for the three types of tiles are as follows:

tile			
designator σ	0	1	-1

Thus every tile in the (3.3.3.4.4)-tessellation may be associated with integral coordinates (x, y, σ) , where $(x, y) \in \mathbb{Z}^2$ and $\sigma \in \mathbb{Z}_3$.

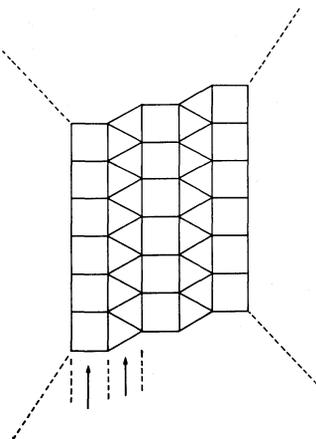


Figure 27.

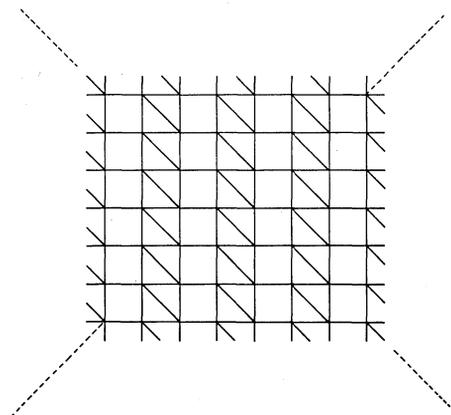


Figure 28.

The neighbours N of a given tile (x, y, σ) are given by

$$N(x, y, 0) = \{(x+1, y, 1), (x-1, y, -1), (x, y-1, 0), (x, y+1, 0)\},$$

or

$$N(x, y, \sigma \neq 0) = \{(x-\sigma, y, 0), (x, y, -\sigma), (x, y-\sigma, -\sigma)\}.$$

Furthermore if we select any square, simple or divided, as the origin $(0, 0, \sigma)$, the triangular tiles have x -coordinates that are either all even or all odd.

10.1 Bounding regions

The bounding regions for (3.3.3.4.4)-patterns with the given integral coordinate system may be chosen as a rectangles. Let l_1 and l_2 respectively denote the sides of the rectangle along the x - and y -directions. The graph (trellis) of the bounding region may be drawn in the plane with the vertices associated with their respective coordinates as shown in figure 29.

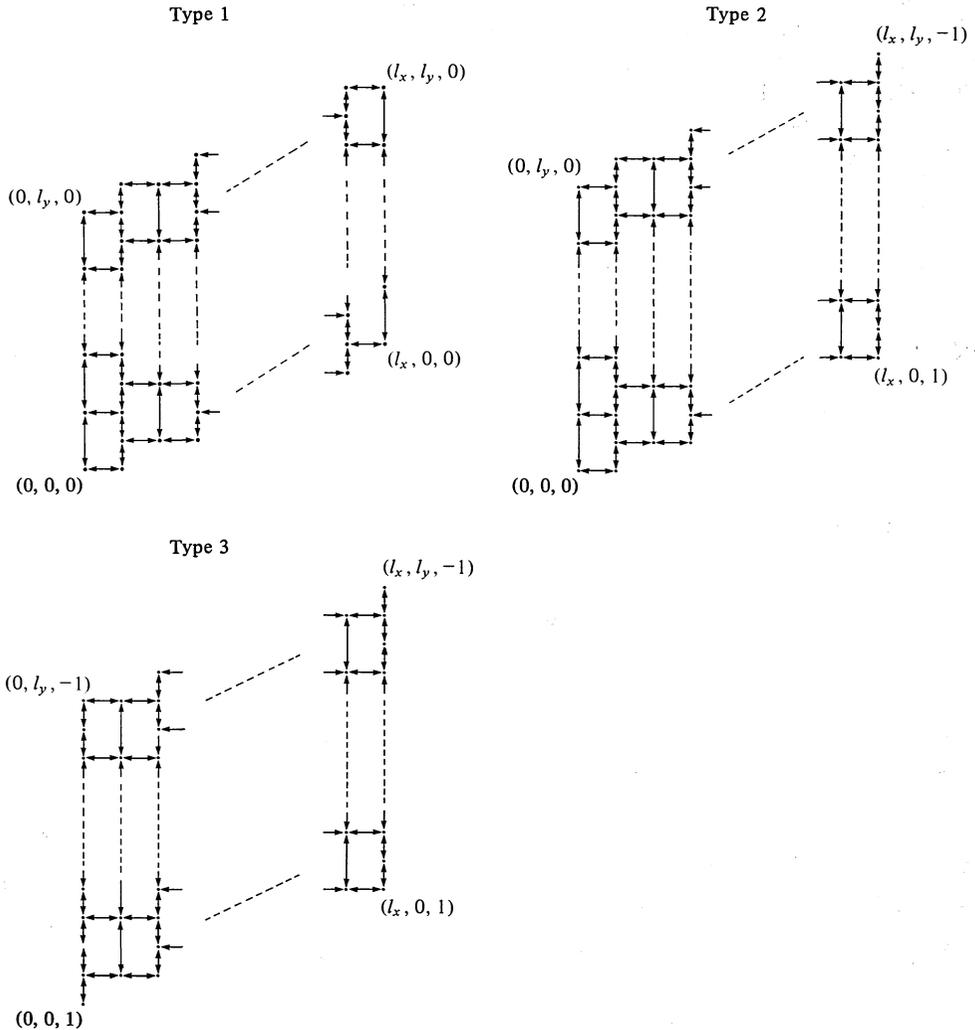


Figure 29.

Let $l_x = l_1 - 1$ and $l_y = l_2 - 1$. Let the vertices of the trellis take on coordinates in the positive quadrant, with a vertex chosen as the origin. Then every configuration is a subgraph of its trellis which it *spans* if and only if there is a vertex in the lines $x = 0, x - l_x, y = 0, \text{ and } y = l_y$.

There are two trivial configurations for any content p . These are shown in figure 30. Every other pattern must contain at least one square tile and one triangular tile. These patterns must span one of three types of bounding regions.

Type 1: l_1 is odd and a simple square is chosen as the origin.

Type 2: l_1 is even and a simple square is chosen as the origin.

Type 3: l_1 is odd and a divided square is chosen as the origin.

The three types of bounding rectangles are shown in figure 31, and their corresponding trellises in figure 29.

Let (l_1, l_2, t) denote a bounding rectangle of sides l_1 and l_2 and type t . When $l_2 = 1$ there are two trivial bounding rectangles, each of which contributes towards a single pattern. There is a trivial type 1 bounding rectangle of size $(\frac{2}{3}[p-1]+1, 1)$ if $p \equiv 1 \pmod 3$, a trivial type 2 bounding rectangle of size $(2[\frac{1}{3}p], 1)$ if $p \equiv 0, 2 \pmod 3$, and a trivial type 3 bounding rectangle of size $(2[\frac{1}{3}p]+1, 1)$ for all p . Consequently we may assume that $l_2 \geq 2$. For a fixed p we may determine the set of bounding rectangles that house all the nontrivial (3.3.3.4.4)-patterns of content p .

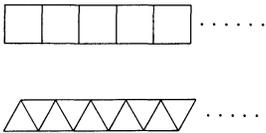


Figure 30.

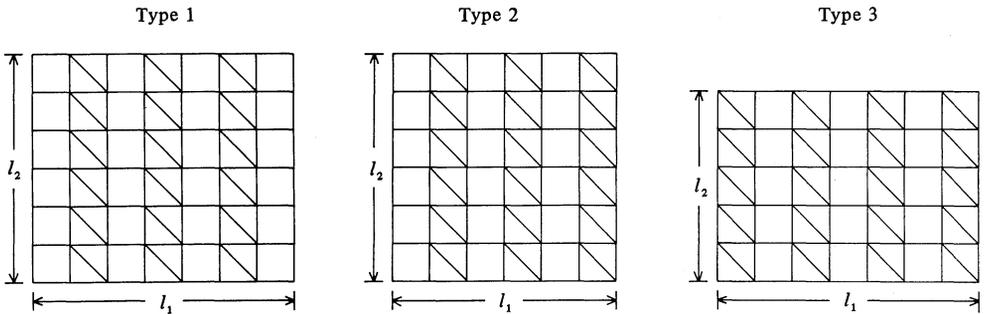


Figure 31.

10.1.1. Type 1.

Let $l_1 = 2l'_1 + 1 \geq 3$. Then l_1 and l_2 must satisfy:

$l_2 + 2l'_1 + 1 \leq p + 1$ maximal-stretching condition;

$l_2(3l'_1 + 1) \geq p$ sufficiency of tiles;

$\left. \begin{matrix} 1 \leq l'_1 \leq \frac{p-2}{2} \\ 2 \leq l_2 \leq p-2 \end{matrix} \right\}$ conditions on l_1 and l_2 .

Let $\mu = 3l'_1 + 1$. Rewriting the conditions we have

$$\begin{aligned} l_2 + \frac{2}{3}(\mu - 1) &\leq p, \\ l_2 \mu &\geq p, \\ 2 &\leq l_2 \leq p - 2, \\ 4 &\leq \mu \leq p - 1, \\ \mu - 1 &= 0 \pmod{3}. \end{aligned}$$

Solving these inequalities gives the sets

$${}_1\psi'_p = \left\{ (\mu, l_2): 2 \leq l_2 \leq p - 2, \max\left\{\left\lceil \frac{p}{l_2} \right\rceil, 4\right\} \leq \mu \leq \lfloor \frac{3}{2}(p - l_2) \rfloor + 1, \mu = 1 \pmod{3} \right\}$$

and

$${}_1\psi_p = \left\{ (l_1, l_2, 1): (\mu, l_2) \in {}_1\psi'_p, l_1 = \frac{2\mu + 1}{3} \right\}.$$

The set ${}_1\psi_p$ defines the algorithm for generating type 1 rectangles.

10.2.1. Type 2.

Let $l_1 = 2l'_1 \geq 2$. Again we have

$$\begin{aligned} l_2 + 2l'_1 &\leq p + 1, \\ l_2(3l'_1) &\geq p, \\ 2 &\leq l_2 \leq p - 1, \\ 1 &\leq l'_1 \leq \frac{p + 1}{3}. \end{aligned}$$

Solving these inequalities we get the sets

$${}_2\psi'_p = \left\{ (\mu, l_2): 2 \leq l_2 \leq p - 1, \max\left\{\left\lceil \frac{p}{l_2} \right\rceil, 3\right\} \leq \mu \leq \lfloor \frac{3}{2}(p + 1 - l_2) \rfloor, \mu = 0 \pmod{3} \right\}$$

and

$${}_2\psi_p = \left\{ (l_1, l_2, 2): (\mu, l_2) \in {}_2\psi'_p, l_1 = \frac{2\mu}{3} \right\}.$$

10.1.3. Type 3.

Let $l_1 = 2l'_1 + 1 \geq 3$. The necessary and sufficient conditions on l_1 and l_2 are

$$\begin{aligned} l_2 + 2l'_1 + 1 &\leq p + 1, \\ l_2(3l'_1 + 2) &\geq p, \\ 2 &\leq l_2 \leq p - 2, \\ 1 &\leq l'_1 \leq \frac{p - 2}{2}. \end{aligned}$$

From which we get the sets

$${}_3\psi'_p = \left\{ (\mu, l_2): 2 \leq l_2 \leq p - 2, \max\left\{\left\lceil \frac{p}{l_2} \right\rceil, 5\right\} \leq \mu \leq \lfloor \frac{3}{2}(p - l_2) \rfloor + 2, \mu = 2 \pmod{3} \right\}$$

and

$${}_3\psi_p = \left\{ (l_1, l_2, 3): (\mu, l_2) \in {}_3\psi'_p, l_1 = \frac{2\mu - 1}{3} \right\}.$$

As an example let $p = 5$. Then the nontrivial bounding rectangles are given by

$${}_1\psi_p = \{(3, 2, 1), (3, 3, 1)\},$$

$${}_2\psi_p = \{(2, 2, 2), (4, 2, 2), (2, 3, 2), (2, 4, 2)\},$$

$${}_3\psi_p = \{(3, 2, 3), (3, 3, 3)\}.$$

It is possible to combine the definition of the three sets by a single algorithm. This is left as an exercise to the reader.

10.2 Coding

We may represent any pattern within its bounding rectangle (l_1, l_2, t) by the word

$$\rho_1 x_1 + \rho_2 x_2 + \dots + \rho_m x_m,$$

where $m = l_2[\frac{3}{2}l_1 + \frac{1}{2}(t-2)]$ and each ρ_i takes on a value in $\{0, 1\}$. The subscripts refer to the tile number within its bounding region, labelled with increasing values from left to right, bottom to top, in that order. Any tile (x, y, σ) has the label

$$y[\frac{3}{2}l_1 + \frac{1}{2}(t-2)] + 1 + [\frac{3}{2}(x-\sigma)] + \sigma.$$

As in the case of polyominoes the word may be partitioned into l_2 subwords, each subword representing a row of the bounding rectangle. That is, every pattern corresponds to an l_2 -tuple $\Gamma = (\gamma_0, \gamma_1, \dots, \gamma_{l_2})$, where each tile (x, y, σ) in the pattern contributes

$$2^{\frac{3}{2}l_1 + \frac{1}{2}(t+1) - [1 + [\frac{3}{2}(x-\sigma)] + \sigma]} \text{ to } \gamma_y.$$

10.3 Symmetries

We may employ the preceding vector description to define a canonical pattern—that is, the pattern that represents its free equivalence class. However, the choice of a rectangle as the bounding region, together with the asymmetry due to distortion of the triangles, poses certain difficulties. To appreciate this point it is perhaps best to see what the bounding rectangle looks like in the original (3.3.3.4.4)-tessellation. The actual arrangements of the tiles which correspond to the various types of bounding rectangles are shown in figure 32.

A symmetry transformation is one that leaves the bounding region invariant in the space. One may observe from figure 32 that a rotation through π leaves the arrangement of tiles corresponding to rectangles of types 1 and 3 invariant in the plane. Consequently a π rotation of the bounding region may describe an isomorph of the current spanning pattern.

Other symmetry transformations do apply. Consider, as an example, the subpatterns of a type 1 rectangle indicated by the unshaded sections of figures 33(a) and 33(b).

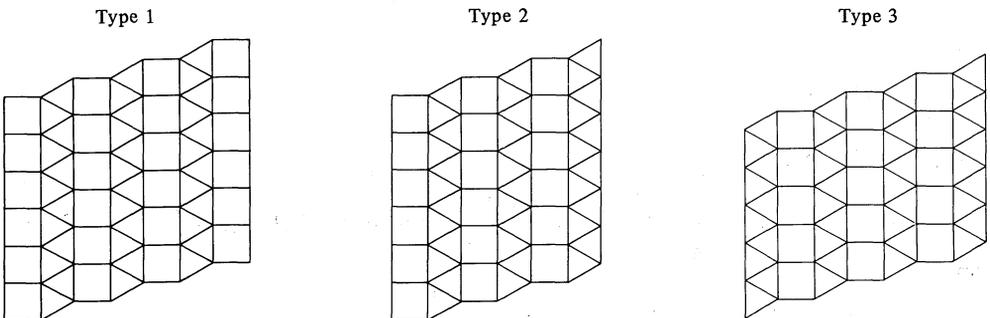


Figure 32.

Suppose there is a pattern which spans the bounding rectangle as well as the unshaded region, and also does not contain any tile belonging to the shaded section. Clearly this region is invariant under a horizontal reflection about the line $y = \frac{1}{2}l_y$. Moreover there are other types of horizontal reflections that may apply depending upon the subregion under consideration. Each type of horizontal reflection leaves one of the columns of simple squares invariant in the plane, with the exception of two special cases in the cases of bounding rectangles of types 2 and 3 (see figure 34). For example, let h_0 and $h_{\bar{0}}$ respectively denote the horizontal reflections that leave the regions in figures 33(a) and 33(b) invariant in the plane. Notice that h_0 leaves the *first* column of simple squares invariant in the plane, and $h_{\bar{0}}$ leaves the *last* column of simple squares invariant in the plane. For particular bounding rectangles of types 1, 2, and 3, the various regions that remain invariant under such horizontal reflections are shown in figure 34.

Further, in the cases of rectangles of types 1 and 3, we may compose these horizontal reflections with the symmetry transformation π . Let $f g$ denote the composition $f(g(\))$. Then, for example, for the reflections h_0 and $h_{\bar{0}}$ of figure 33, we have $v_0 = h_0\pi = \pi h_{\bar{0}}$ and $v_{\bar{0}} = h_{\bar{0}}\pi = \pi h_0$. They are labelled by the letter v since they may have the effect of a reflection about the line $x = \frac{1}{2}l_x$, a vertical reflection.

In general let c be the x -coordinate of a column of simple squares. The coordinate-map transformation of the horizontal reflection, h_c , that leaves this column invariant in the plane is given by:

$$h_c: (x, y, \sigma) \rightarrow (x, l_y - y - \lfloor \frac{1}{2}(x - c - \sigma) \rfloor, \sigma) .$$

For rectangles of types 1 and 3 the corresponding reflection about the line $x = \frac{1}{2}l_x$ is given by:

$$v_c \equiv h_c\pi: (x, y, \sigma) \rightarrow (l_x - x, y - \lfloor \frac{1}{2}(l_x - x - c + \sigma) \rfloor, -\sigma) .$$

Furthermore, for rectangles of types 1 and 3, for any h_c there exists an $h_{\bar{c}}$ such that $h_c = \pi h_{\bar{c}}\pi$ and $h_{\bar{c}} = \pi h_c\pi$. If c is the x -coordinate of the j th column from the left then \bar{c} is the x -coordinate of the j th column from the right.

In the cases of rectangles of types 2 and 3 there are subregions which remain invariant in the plane under horizontal reflection yet do not preserve any column of simple squares. They correspond to the diagrams marked with asterisks in figure 34. Notice that these regions either preserve the first column of triangular tiles (with the exception of the topmost $\sigma = -1$ tile) or the last column of triangular tiles (with the exception of the bottommost $\sigma = 1$ tile). We will denote the horizontal reflection of these two kinds of regions respectively by h_0^* and $h_{\bar{0}}^*$. The coordinate-map transformations for h_0^* and $h_{\bar{0}}^*$ may be determined from the coordinate-map

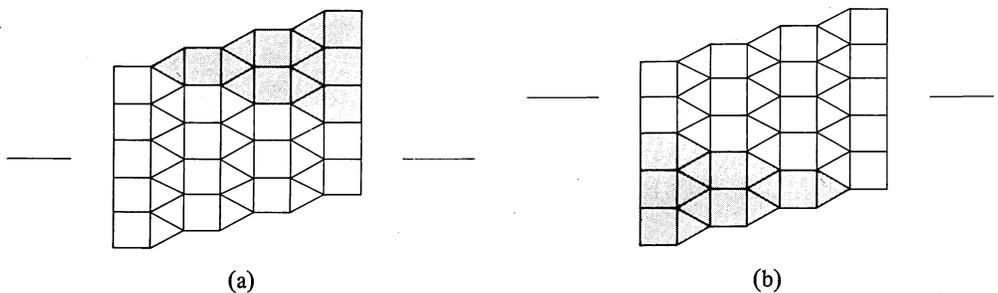


Figure 33.

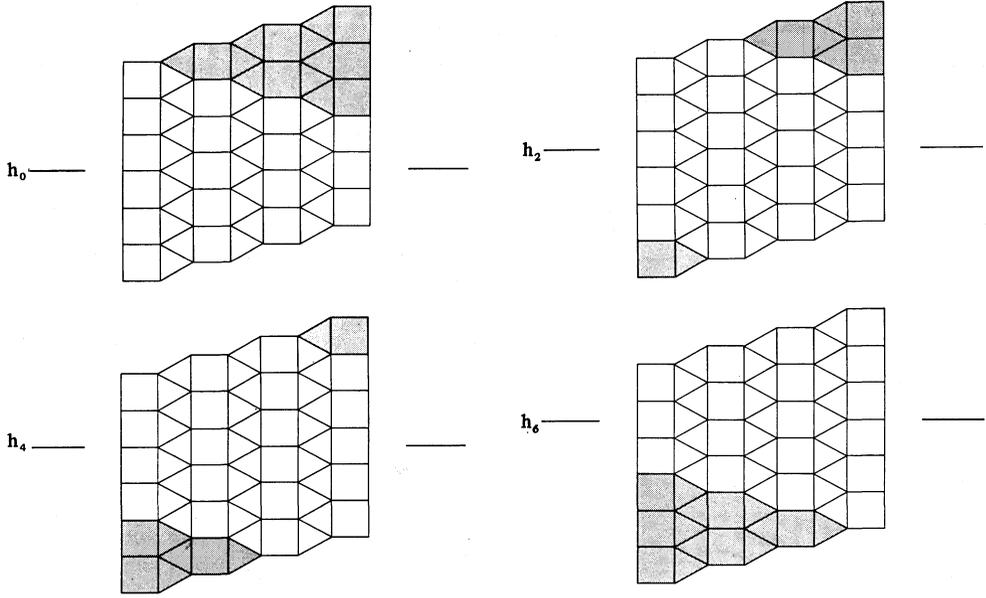
transformation for h_c . They are given by:

$$h_0^*: (x, y, \sigma) \rightarrow (x, l_y - y - 1 - \lfloor \frac{1}{2}(x - 1 - \sigma) \rfloor, \sigma)$$

and

$$h_0^{*2}: (x, y, \sigma) \rightarrow (x, l_y - y + 1 - \lfloor \frac{1}{2}(x - l_x + 1 - \sigma) \rfloor, \sigma)$$

For type 2 rectangles only h_0^{*2} applies. In the case of type 3 rectangles we have the
Type 1



$$(h_6 \equiv h_6, h_2 \equiv h_4, h_4 \equiv h_2, h_6 \equiv h_0)$$

Type 2

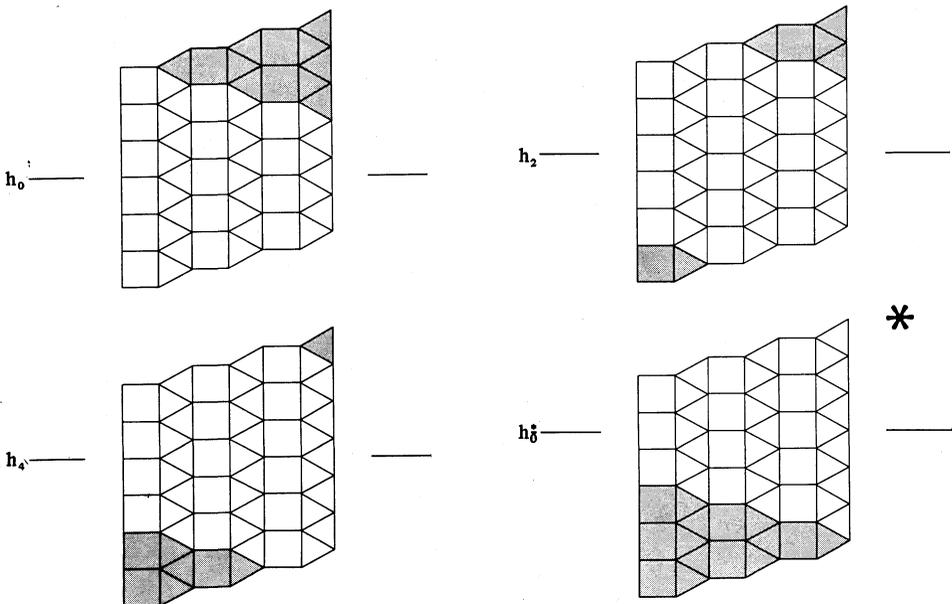


Figure 34.

additional transformations v_0^* and v_0^* given by:

$$v_0^* \equiv h_0^* \pi: (x, y, \sigma) \rightarrow (l_x - x, y - 1 - \lfloor \frac{1}{2}(l_x - x - 1 + \sigma) \rfloor, -\sigma),$$

and

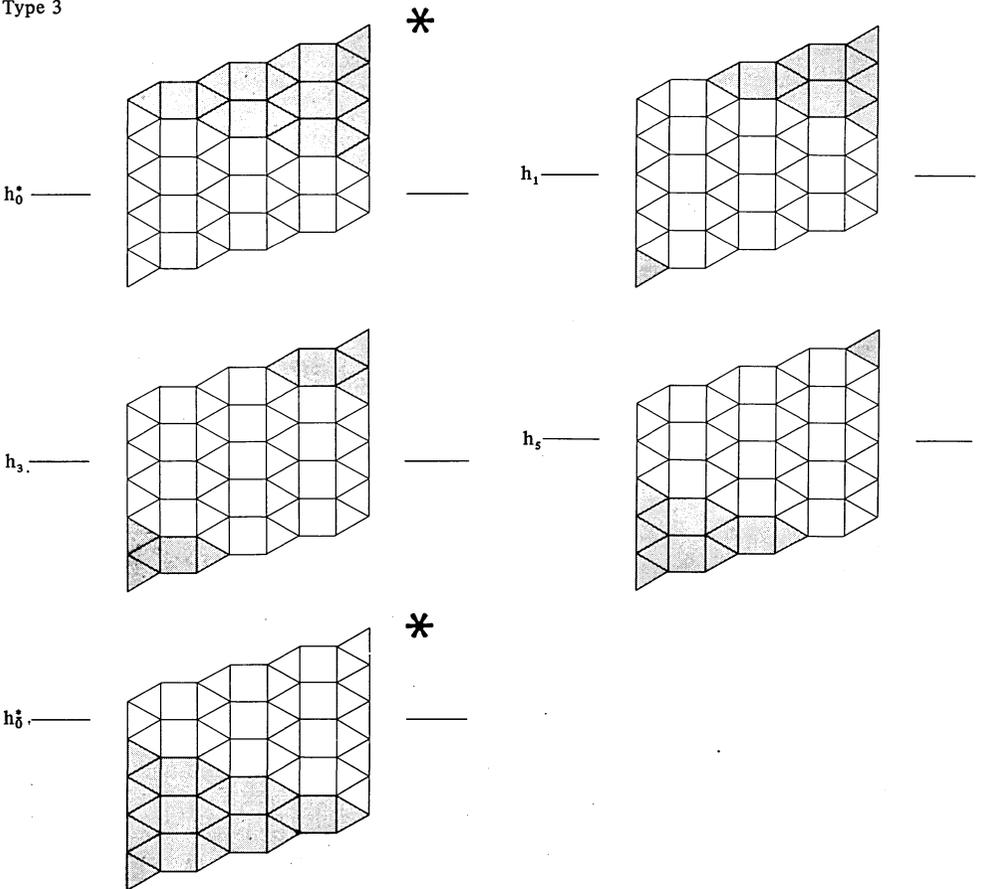
$$v_0^* \equiv h_0^* \pi: (x, y, \sigma) \rightarrow (l_x - x, y + 1 - \lfloor \frac{1}{2}(-x + 1 + \sigma) \rfloor, -\sigma).$$

All the symmetry transformations, represented as coordinate-coordinate maps, are shown in table 8.

Rather than ask the question whether or not the pattern may have a horizontally reflected isomorph by examining the subregion it spans, it is convenient to apply the transformation directly to the bounding rectangle. Since the transformation leaves the subregion invariant in the plane, it has the effect of mapping the shaded tiles to positions in the plane whose y -coordinates are either less than 0 or greater than l_y . Thus, if a pattern has a tile which is mapped to a position outside the bounding rectangle, it cannot possibly possess a horizontally reflected isomorph.

On the other hand we may have a pattern that spans the subregion yet under a horizontal reflection spans a smaller bounding rectangle. Examples of such patterns are shown in figure 35. In figure 35(a) the isomorphic pattern does not contain a tile which has a y -coordinate equal to l_y , and in figure 35(b) the isomorphic pattern does not contain a tile which has a y -coordinate equal to 0.

Type 3



$$(h_1 \equiv h_5, h_3 \equiv h_3, h_5 \equiv h_1)$$

Figure 34 (continued).

Table 8.

Group element	Symbol	Coordinate-coordinate map: $(x, y, \sigma) \rightarrow$	Types of rectangle
1 Rotation through π	π	$(l_x - x, l_y - y, -\sigma)$	1, 3
Reflection about the horizontal:			
2	h_c	$(x, l_y - y - [\frac{1}{2}(x - c - \sigma)], \sigma)$	1, 2, 3
3	h_0^*	$(x, l_y - y - 1 - [\frac{1}{2}(x - 1 - \sigma)], \sigma)$	3
4	h_0^*	$(x, l_y - y + 1 - [\frac{1}{2}(x - l_x + 1 - \sigma)], \sigma)$	2, 3
Compositions of the above:			
5	v_c	$(l_x - x, y - [\frac{1}{2}(l_x - x - c + \sigma)], -\sigma)$	1, 3
6	v_0^*	$(l_x - x, y - 1 - [\frac{1}{2}(l_x - x - 1 + \sigma)], -\sigma)$	3
7	v_0^*	$(l_x - x, y + 1 - [\frac{1}{2}(-x + 1 + \sigma)], -\sigma)$	3

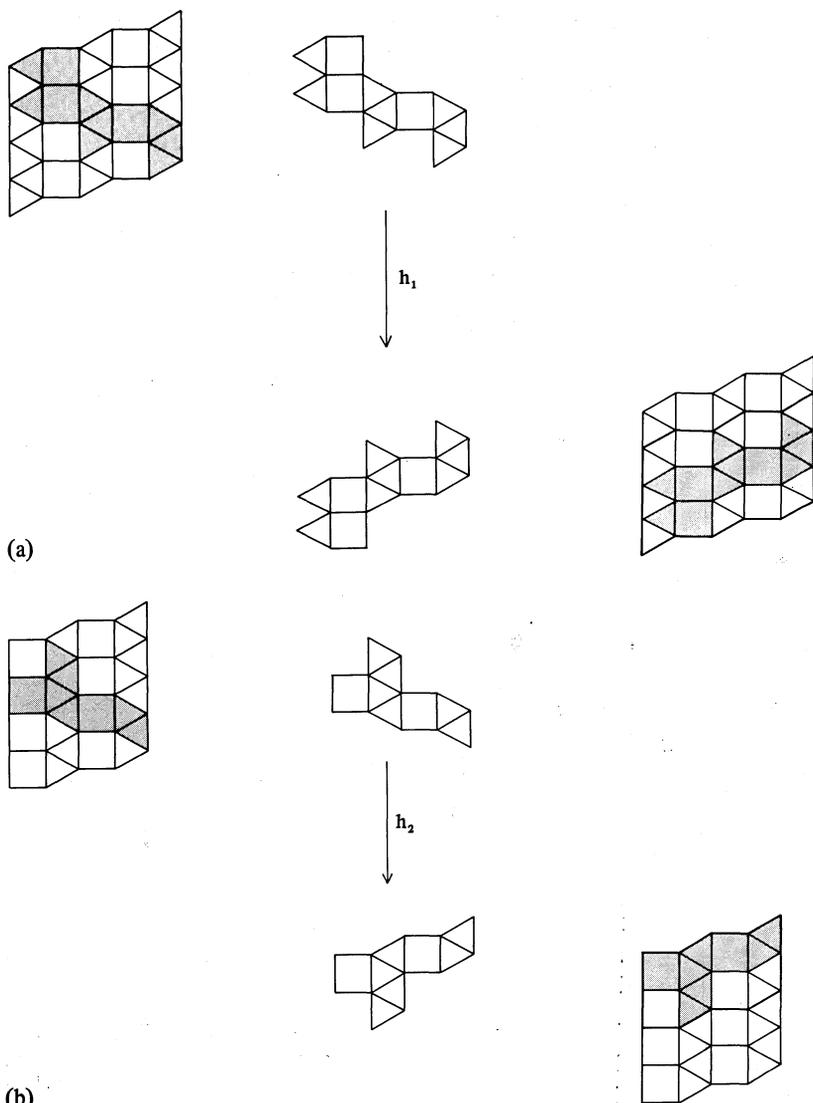


Figure 35.

Let $\Gamma = (\gamma_0, \gamma_1, \dots, \gamma_{l_y})$ and $\Gamma_\tau = (\gamma'_0, \gamma'_1, \dots, \gamma'_{l_y})$, where τ is an element of the group of symmetry transformations T_t for a rectangle t . Suppose

$$T'_t = \{\tau \in T_t : \text{the pattern under } \tau \text{ has a tile lying outside the bounding rectangle}\}$$

and

$$T''_t = \{\tau \in T_t : \text{the pattern under } \tau \text{ has } \gamma'_0 = 0 \text{ or } \gamma'_{l_y} = 0\}.$$

Then any word Γ is canonical if and only if

$$T''_t = \emptyset \quad \text{and} \quad \Gamma \geq \Gamma_\tau, \quad \tau \in T_t - T'_t.$$

10.4 Algorithm

It is easily seen that the nontrivial (3.3.3.4.4)-pattern enumeration problem reduces to the graph problem 5.1(b). Let $S^* = \{S_1, S_2, S_3, S_4\}$, where

$$S_1 = \{(x, 0, \sigma)\}, \quad S_2 = \{(x, l_y, \sigma)\}, \quad S_3 = \{(0, y, \sigma)\}, \quad S_4 = \{(l_x, y, \sigma)\}.$$

Let t denote the type of the bounding rectangle and let $u = (u_x, u_y, u_\sigma)$ be a vertex in the corresponding trellis. Then the following distance measures may be defined.

10.4.1. $u_\sigma = 0$.

$$d(S_1, u) = u_y,$$

$$d(S_2, u) = l_y - u_y,$$

$$d(S_3, u) = \begin{cases} \frac{3}{2}u_x & \text{if } t = 1, 2, \\ \frac{3}{2}(u_x - 1) + 1 & \text{if } t = 3, \end{cases}$$

$$d(S_4, u) = \begin{cases} \frac{3}{2}(l_x - u_x) & \text{if } t = 1, \\ \frac{3}{2}(l_x - u_x - 1) + 1 & \text{if } t = 2, 3. \end{cases}$$

10.4.2. $u_\sigma = \pm 1$.

$$d(S_1, u) = \min\left\{\max\{2u_y - \frac{1}{2}(1 + u_\sigma), u_y\}, u_y + 1 + k_1\right\},$$

$$d(S_2, u) = \min\left\{\max\{2(l_y - u_y) + \frac{1}{2}(u_\sigma - 1), l_y - u_y\}, l_y - u_y + 1 + k_2\right\},$$

$$d(S_3, u) = \begin{cases} \frac{3}{2}(u_x - u_\sigma) + u_\sigma & \text{if } t = 1, 2, \\ \max\{\frac{3}{2}u_x - \frac{1}{2}(1 + u_\sigma), u_x\} & \text{if } t = 3, \end{cases}$$

$$d(S_4, u) = \begin{cases} \frac{3}{2}(l_x - u_x + u_\sigma) - u_\sigma & \text{if } t = 1, \\ \frac{3}{2}(l_x - u_x) + \frac{1}{2}(u_\sigma - 1) & \text{if } t = 2, 3, \end{cases}$$

where

$$k_1 = \begin{cases} 1 & \text{if } u_x = l_x \text{ and } u_\sigma = -1, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$k_2 = \begin{cases} 1 & \text{if } u_x = 0 \text{ and } u_\sigma = 1, \\ 0 & \text{otherwise.} \end{cases}$$

We also need the following distance measures:

$$d(S_i \dot{+} S_j, u) = \min\{d(S_i, u) + d(S_j, u), d(S_i \cap S_j, u)\}$$

for $(i, j) \in \{(1, 3), (1, 4), (2, 3), (2, 4)\}$.

Let G_1, G_2, \dots, G_{q-1} be the sequence of graphs constructed. Let v_q be the current candidate. Then v_q is in G_q if and only if

$$d(G_q, \dot{\Sigma}S) = \min\{d(G_{q-1}, \dot{\Sigma}S), d(G_q, S_1 \dot{+} S_4) + d(G_q, S_2 \dot{+} S_3), \\ d(G_q, S_1 \dot{+} S_3) + d(G_q, S_2 \dot{+} S_4)\} \leq p - q,$$

where the $d(G_q, S_i \dot{+} S_j)$ are calculated according to the recurrence formulae in section 5.1.1. This is the distance criterion that must be satisfied to generate all (3.3.3.4.4)-patterns with content p .

11 (3.3.4.3.4)-patterns

The (3.3.4.3.4)-tessellation may be seen as a bidirectional arrangement of strips in which each strip consists of alternating squares and pairs of triangles as shown in figure 36. We may regard this tessellation as being composed of elastic bands which when perturbed may take on the appearance of the tessellation consisting of squares and right-angled isosceles triangles depicted in figure 37. In other words the (3.3.4.3.4)-tessellation may be mapped onto the square grid, with the triangles oriented in the manner shown in figure 37. Notice that the triangles manifest themselves in one of four distinct orientations which occur as two separate pairs, each of which forms a divided square. This allows us to define an integral coordinate system for the (3.3.4.3.4)-tessellation.

Each square, simple or divided, in the grid is allocated integer Cartesian coordinates (x, y) of the plane. To this is added a third coordinate, σ_1 , which distinguishes between the orientations associated with the squares, and a further fourth coordinate, σ_2 , which distinguishes between the triangles that form a divided square. The pair (σ_1, σ_2) may be regarded as the *tile designator*. That is, every tile is associated with four integral quantities, $(x, y, \sigma_1, \sigma_2)$, where $(x, y) \in \mathbb{Z}^2$ and $(\sigma_1, \sigma_2) \in (\mathbb{Z}_3)^2$. The values σ_1, σ_2 taken on for each type of tile are as follows:

tile						
tile designator	$\left\{ \begin{array}{l} \sigma_1 \\ \sigma_2 \end{array} \right.$	0	1	1	-1	-1
		0	1	-1	1	-1

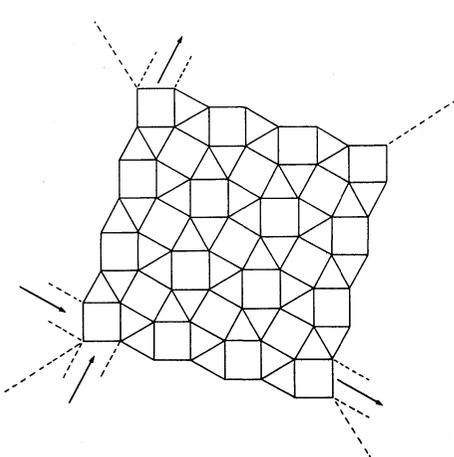


Figure 36.

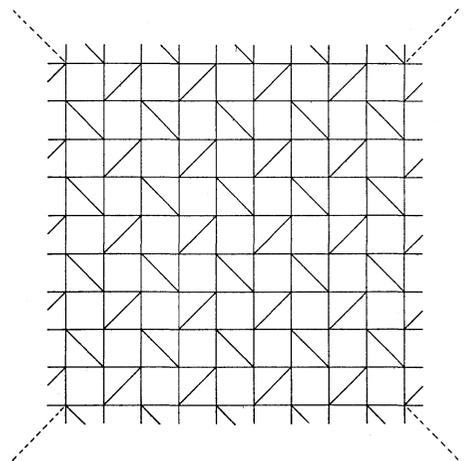


Figure 37.

The neighbours N of any tile $(x, y, \sigma_1, \sigma_2)$ are given by

$$N(x, y, 0, 0) = \begin{cases} \{(x, y+1, 1, 1), (x, y-1, 1, -1), (x+1, y, -1, 1), (x-1, y, -1, -1)\} \\ \text{or} \\ \{(x, y+1, -1, -1), (x, y-1, -1, 1), (x+1, y, 1, 1), (x-1, y, 1, -1)\} \end{cases}$$

and

$$N(x, y, \sigma_1 \neq 0, \sigma_2 \neq 0) = \{(x - \sigma_2, y, 0, 0), (x, y - \sigma_1 \sigma_2, 0, 0), (x, y, \sigma_1, -\sigma_2)\}.$$

By assigning the origin to any tile, we may observe that the triangular tiles have (x, y) coordinates such that either all of them have $x+y$ odd or all of them have $x+y$ even. Moreover any diagonal consisting of divided squares consists of oppositely oriented squares in an alternating fashion.

11.1 Bounding regions

Every (3.3.4.3.4)-pattern may be encased within a rectangle. There are two types of bounding rectangles: one which has a simple square associated with the origin and whose immediate x -neighbour is a $\sigma_1 = -1$ divided square, and the other has a $\sigma_1 = 1$ divided square as the origin. Let l_1 and l_2 denote the sides of the rectangles along the x - and y -directions respectively. For both types of rectangles $l_2 \leq l_1$, and in addition for the type 2 rectangle l_1 and l_2 must be odd. It may be easily verified that, for any other choice both for origin and orientation, the resulting rectangle can be mapped into one of these two types of rectangles through a symmetry motion in the plane. In short these two types characterize the *fixing* of the bounding regions in the plane. The two types of bounding rectangles and the arrangement of tiles in the (3.3.4.3.4)-tessellation to which they correspond are shown in figure 38.

We may derive the explicit algorithms for the set of bounding regions that house the population of (3.3.4.3.4)-patterns with content p . As usual let a bounding rectangle be denoted by the triple (l_1, l_2, t) , where t refers to the type of the rectangle.

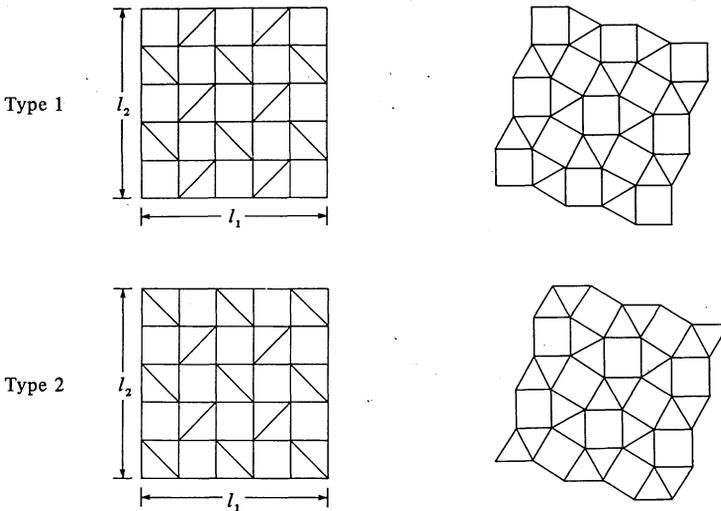


Figure 38.

11.1.1. Type 1.

It can be easily demonstrated that the following conditions on l_1 and l_2 must be satisfied:

- $l_1 + l_2 \leq p + 1$ maximal-stretching condition;
- $l_1 l_2 \geq \frac{2}{3} p$ sufficiency of tiles;
- $1 \leq l_2 \leq l_1$ orientation of rectangle.

Solving these inequalities gives the set

$${}_1\psi_p = \left\{ \left(\left\lceil \frac{2p}{3} \right\rceil, 1, 1 \right) \right\} \cup \left\{ (l_1, l_2, 1) : 2 \leq l_2 \leq \left\lfloor \frac{p+1}{2} \right\rfloor, \max \left\{ l_2, \left\lceil \frac{2p}{3l_2} \right\rceil \right\} \leq l_1 \leq p+1-l_2 \right\}.$$

11.1.2. Type 2.

Let od be the odd ceiling function

$$od(x) = x + (1 - x \bmod 2).$$

In this case l_1 and l_2 must satisfy

- $l_1 + l_2 \leq p + 1,$
- $l_1 l_2 \geq \frac{2p-1}{3},$
- $1 \leq l_2 \leq l_1,$
- l_1, l_2 odd.

From which we get the set

$${}_2\psi_p = \left\{ \left(2 \left\lceil \frac{p}{3} \right\rceil + 1, 1, 2 \right) \right\} \cup \left\{ (l_1, l_2, 2) : l_2 \in \{3, 5, \dots, od\left(\left\lfloor \frac{p-1}{2} \right\rfloor\right)\}, \max \left\{ l_2, od\left(\left\lceil \frac{2p-1}{3l_2} \right\rceil\right) \right\} \leq l_1 \leq od(p-l_2) \right\}.$$

At this stage we may remark that, for a fixed p , each of the two trivial bounding rectangles—that is, with $l_2 = 1$ —contributes towards a single pattern. We may therefore disregard these rectangles and from now on assume that $l_2 \geq 2$.

11.2 Coding

The trellis of a bounding rectangle (l_1, l_2, t) is constructed in the usual manner. Let $l_x = l_1 - 1$ and $l_y = l_2 - 1$. We may then associate with the vertices of the trellis coordinates $(x, y, \sigma_1, \sigma_2)$, where $0 \leq x \leq l_x$ and $0 \leq y \leq l_y$. We may also assign labels to the vertices in the following manner. Let loc be the function defined as follows:

- $loc(0) = 0,$
- $loc(1) = l_x + \lfloor \frac{1}{2} l_x \rfloor + t,$
- $loc(y) = loc(y-2) + 3l_x + 3, \quad y \geq 2.$

Then a vertex $(x, y, \sigma_1, \sigma_2)$ has the label

$$loc(y) + \lfloor \frac{3}{2} (x - \sigma_2) \rfloor + (1 + \sigma_2).$$

A (3.3.4.3.4)-pattern may be associated with a word defined in the usual manner. As in the case of polyominoes and (3.3.3.4.4)-patterns this word may be partitioned into l_2 subwords, where each subword represents an integral quantity. In other words every (3.3.4.3.4)-pattern in an (l_1, l_2, t) rectangle is associated with an l_2 -tuple $\Gamma = (\gamma_0, \gamma_1, \dots, \gamma_{l_2})$. Each vertex $(x, y, \sigma_1, \sigma_2)$ in a pattern contributes to γ_y

$$2^{l_x + \lfloor \frac{1}{2} l_x \rfloor + t - \lfloor \frac{3}{2} (x - \sigma_2) \rfloor - (1 + \sigma_2)} \quad \text{if } y = 0 \pmod 2$$

or

$$2^{l_x + \lfloor \frac{1}{2} l_x \rfloor + 3 - t - \lfloor \frac{3}{2} (x - \sigma_2) \rfloor - (1 + \sigma_2)} \quad \text{if } y = 1 \pmod 2 .$$

11.3 Symmetries

If we examine the arrangements of tiles in the (3.3.4.3.4)-tessellation that correspond to the boundary rectangles defined on the square grid, we observe that the bidirectional distortion of the triangular tiles essentially hides the asymmetric nature of the arrangements. Figure 39 illustrates examples of arrangements that correspond to the different conditions imposed on l_1 and l_2 . From figure 39 it may be seen that only when both l_1 and l_2 are odd do the arrangements of the tiles possess any kind of symmetry. In short when either l_1 or l_2 is even the (3.3.4.3.4)-patterns are uniquely generated within their bounding regions. That is, every pattern in these bounding rectangles is the representative of its free equivalence class.

Let us consider the situation when both l_1 and l_2 are odd. Two cases arise: (1) $l_2 < l_1$ and (2) $l_1 = l_2$. When $l_2 < l_1$ there is only one symmetry motion in the plane which leaves the bounding rectangle invariant. This is a rotation through π about the centre. Notice that this movement, although preserving the simple squares (which in any event it must), does interchange the orientation of the triangular tiles within a divided square. That is a $\sigma_2 = 1$ tile transforms into a $\sigma_2 = -1$ tile and vice versa. Consequently we may describe a π rotation by the following coordinate mapping:

$$\pi: (x, y, \sigma_1, \sigma_2) \rightarrow (l_x - x, l_y - y, \sigma_1, -\sigma_2) .$$

When $l_1 = l_2$ other symmetry motions also leave the bounding rectangle invariant. For instance a $\frac{1}{2}\pi$ or a $-\frac{1}{2}\pi$ rotation about the centre leaves a type 1 rectangle invariant in the plane. However, these motions interchange a $\sigma_1 = 1$ square into a $\sigma_1 = -1$ square and vice versa. The coordinate-map forms of the two symmetry transformations are:

$$\frac{1}{2}\pi: (x, y, \sigma_1, \sigma_2) \rightarrow (l_y - y, x, -\sigma_1, -\sigma_1 \sigma_2)$$

and

$$\overline{\frac{1}{2}\pi}: (x, y, \sigma_1, \sigma_2) \rightarrow (y, l_x - x, -\sigma_1, \sigma_1 \sigma_2) .$$

In the case of type 2 rectangles, two different symmetry transformations need to be considered. They are the reflections about the lines $x = y$ and $x = -y$ respectively. In coordinate form they may be described as

$$R: (x, y, \sigma_1, \sigma_2) \rightarrow (y, x, \sigma_1, \sigma_1 \sigma_2)$$

and

$$r: (x, y, \sigma_1, \sigma_2) \rightarrow (l_y - y, l_x - x, \sigma_1, -\sigma_1 \sigma_2) .$$

The symmetry elements are summarized in table 9, which includes a pictorial illustration of the effect of the mappings on the orientations of the triangular tiles in the square grid.

Canonical patterns are defined in the usual manner. That is, a pattern is the representative of its free equivalence class if and only if

$$\Gamma \geq \Gamma_\tau, \quad \tau \in T_f,$$

where Γ_τ , τ , and T_f have their usual meaning.

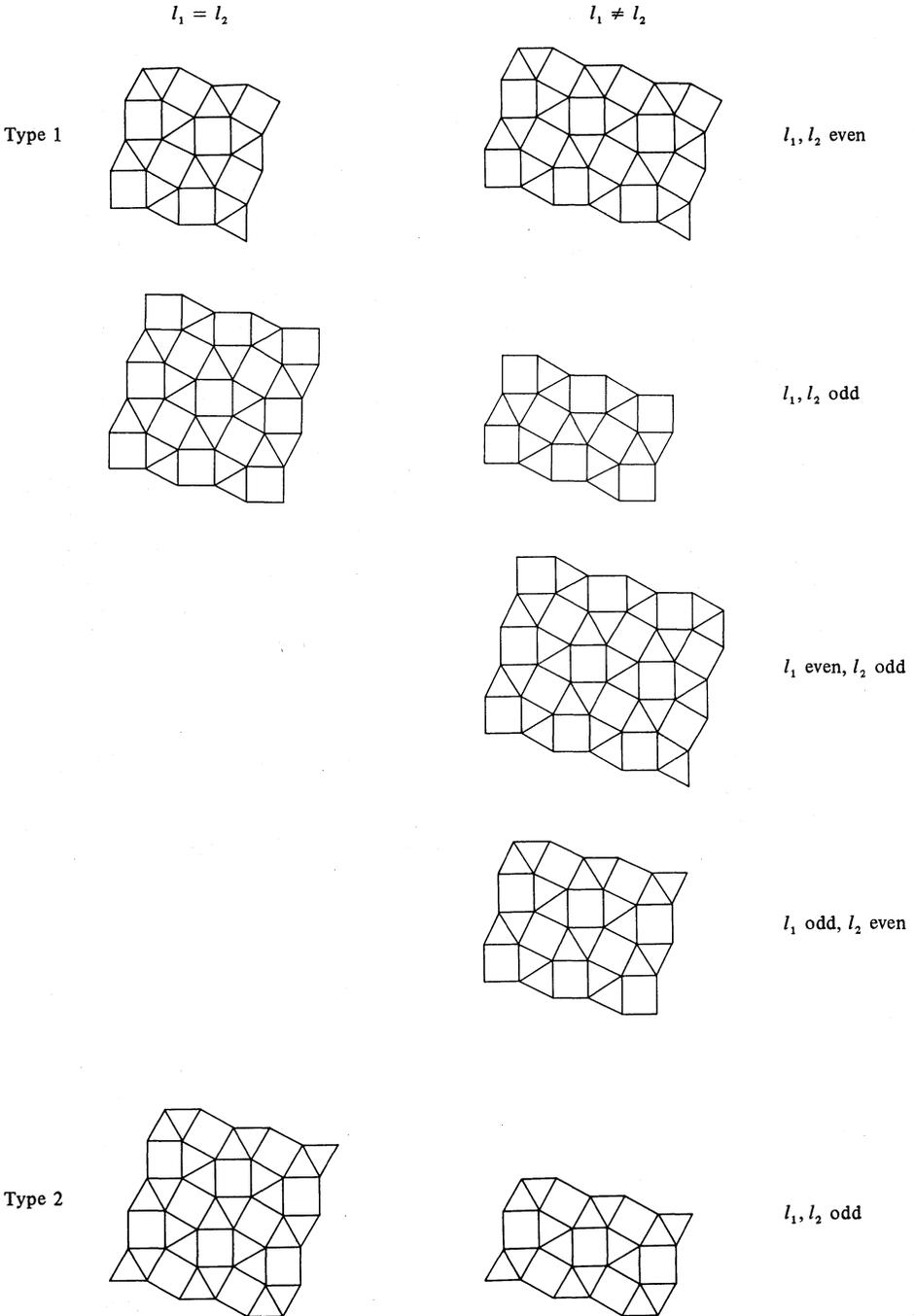


Figure 39.

Table 9.

Group element	Symbol	Coordinate-coordinate map: $(x, y, \sigma_1, \sigma_2) \rightarrow$	Permutation of triangular tiles under mapping
1 Rotation through π	π	$(l_x - x, l_y - y, \sigma_1, -\sigma_2)$	
2 Rotation through $\frac{1}{2}\pi$	$\frac{1}{2}\pi$	$(l_y - y, x, -\sigma_1, -\sigma_1\sigma_2)$	
3 Rotation through $-\frac{1}{2}\pi$	$\overline{\frac{1}{2}\pi}$	$(y, l_x - x, -\sigma_1, \sigma_1\sigma_2)$	
4 Reflection about the $x = y$ diagonal	R	$(y, x, \sigma_1, \sigma_1\sigma_2)$	
5 Reflection about the $x = -y$ diagonal	r	$(l_y - y, l_x - x, \sigma_1, -\sigma_1\sigma_2)$	

11.4 Algorithm

We have again reduced our pattern enumeration problem to problem 5.1(b). Let $S^* = \{S_1, S_2, S_3, S_4\}$ be defined as in section 10.4, except that σ is replaced by σ_1, σ_2 . The same distance criterion as in section 10.4 must be satisfied. The distance measures for the $\langle 3.3.4.3.4 \rangle$ -patterns are defined as follows. Let $u = (u_x, u_y, u_{\sigma_1}, u_{\sigma_2})$ be a vertex in the trellis of type t . Then

$$d(S_1, u) = \lceil \frac{3}{2}(u_y - u_{\sigma_1} u_{\sigma_2}) \rceil + u_{\sigma_1} u_{\sigma_2} - (t + u_x - 1) \bmod 2 ,$$

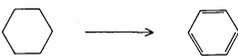
$$d(S_2, u) = \lceil \frac{3}{2}(l_y - u_y + u_{\sigma_1} u_{\sigma_2}) \rceil - u_{\sigma_1} u_{\sigma_2} - (t + l_y + u_x - 1) \bmod 2 ,$$

$$d(S_3, u) = \lceil \frac{3}{2}(u_x - u_{\sigma_2}) \rceil + u_{\sigma_2} - (t + u_y - 1) \bmod 2 ,$$

$$d(S_4, u) = \lceil \frac{3}{2}(l_x - u_x + u_{\sigma_2}) \rceil - u_{\sigma_2} - (t + l_x + u_y - 1) \bmod 2 .$$

12 $\langle 3.6.3.6 \rangle$ -patterns

Recall that the hexagonal tessellation can be associated with an integral coordinate system. Each tile is given the coordinates (x, y, z) such that $x + y + z = 0$. Moreover the tessellation may be coloured in a natural way using three colours according to whether $(x - y) \bmod 3 = 0, 1, \text{ or } 2$. Suppose that the tiles of a particular colour are designated as ‘unmarked’. Suppose further that the tiles of the other two colours are subjected to the following marking operation



with the stipulation that every ‘marked’ tile is adjacent to its unmarked neighbours through unmarked edges. The application of this marking operation is illustrated in figure 40(b). Let us now contract all the marked edges to a point; that is, apply the following reduction operation:



The result is the $\langle 3.6.3.6 \rangle$ -tessellation shown in figure 40(c). By means of these two operations we have successfully established an integral coordinate system for the $\langle 3.6.3.6 \rangle$ -tessellation. More importantly we have shown that each $\langle 3.6.3.6 \rangle$ -pattern—connected or otherwise—is derivable from some polyhex. We may therefore employ the polyhex enumeration to generate the $\langle 3.6.3.6 \rangle$ -patterns.

It is instructive to reflect upon the implications of the marking and reduction processes. First, we may notice from figure 40(b) that every marked tile is adjacent to its marked neighbours through a marked edge. Furthermore, on applying the reduction operation these tiles become triangles that share only a common point. Second, marking imposes an orientation on the hexagonal tiles in such a manner that all marked tiles of one colour reduce to an upwards triangle (Δ) and those of the other colour reduce to a downwards triangle (∇). The unmarked tiles remain unaltered. Thus marking is equivalent to defining a *marking function* which maps the colours of the hexagonal tiles to the tile types in the $\langle 3.6.3.6 \rangle$ -tessellation. Finally, marking the hexagonal tessellation results in a reduction of the adjacencies of the marked tiles by an additive factor of three. Which three of the original neighbours remain depends on the orientation of the marked tile.

We stated earlier that a $\langle 3.6.3.6 \rangle$ -pattern corresponds to a marked polyhex. However, marking a polyhex does not necessarily yield a connected $\langle 3.6.3.6 \rangle$ -pattern, for precisely the reason that marking engenders a reduction in tile adjacencies. This requires us to modify the polyhex enumeration to ensure that only those polyhexes which are 'markable' are generated.

There is another difficulty introduced by marking. This is due to the fact that marking may be carried out in three possible ways, each dependent on the colour of the hexagonal tile that is preserved as a hexagon after marking. Where M denotes a marking function, the three possible ways are listed as follows as mappings between colour and tile type:

$(x - y) \bmod 3$	M_0	M_1	M_2
0			
1			
2			

This listing may be explained as follows. Suppose tiles of colour c are preserved. The marking function is then M_c , and $M_c[(c + 1) \bmod 3]$ is an upwards triangle and $M_c[(c + 2) \bmod 3]$ is a downwards triangle.

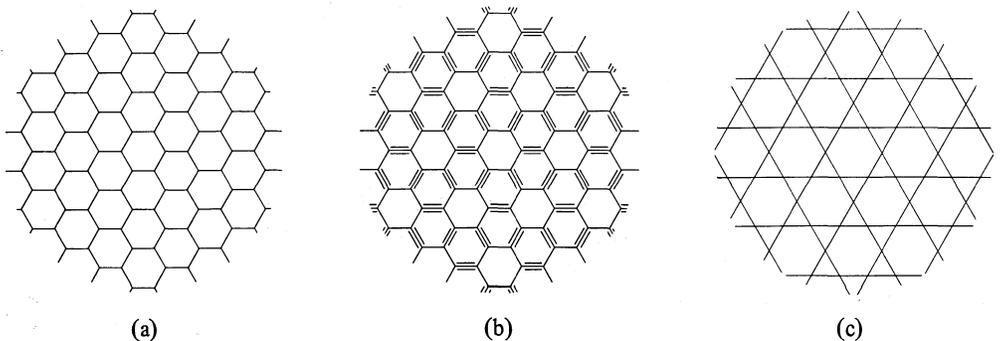


Figure 40.

For computational reasons it is convenient to represent the tile types of the (3.6.3.6)-tessellation by numerical values. A possible choice for the tags is as follows:

tile type			
tag	0	1	2

This choice for the tags allows us to define the marking functions as simple mappings from Z_3 to Z_3 :

$$M_i(c) = (c - i) \bmod 3, \quad i = 0, 1, 2.$$

The preceding discussion provides the requisite background to develop the (3.6.3.6)-pattern enumeration algorithm. One final comment is necessary. Let i indicate the marking function. Let N denote the neighbours of any tile (x, y, z) . Then the neighbours are given as follows.

12.0.1. $M_i[(x - y) \bmod 3] = 1.$

$$N(x, y, z) = \{(x, y + 1, z - 1), (x - 1, y, z + 1), (x + 1, y - 1, z)\}.$$

12.0.2. $M_i[(x - y) \bmod 3] = 2.$

$$N(x, y, z) = \{(x, y - 1, z + 1), (x + 1, y, z - 1), (x - 1, y + 1, z)\}.$$

12.0.3. $M_i[(x - y) \bmod 3] = 0.$

$$N(x, y, z) = \{(x, y \pm 1, z \mp 1), (x \mp 1, y, z \pm 1), (x \pm 1, y \mp 1, z)\}.$$

12.1 *Generating the (3.6.3.6)-patterns from the polyhex*

We have just seen how the (3.6.3.6)-pattern enumeration may be treated as a special case of the polyhex enumeration problem—though with a difference. Essentially the idea is to generate the polyhexes and apply in turn the three marking functions which

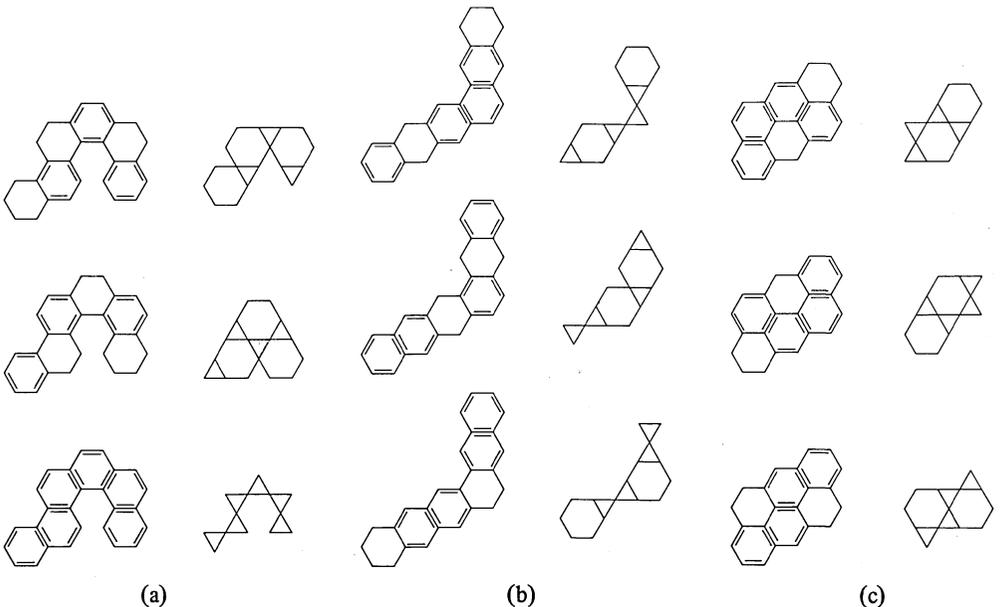


Figure 41.

yield one of the following cases:

- (a) the patterns are unconnected;
- (b) the patterns are isomorphic under a symmetry motion;
- (c) the patterns are distinct;
- (d) some combination of these three—that is, one pattern may be unconnected and the other two isomorphic, and so on.

A few examples illustrating this point are shown in figure 41.

How can we refine this process to ensure that only markable polyhexes are generated? If we recall that a polyhex is a connected subgraph of its bounding hexagon, which in turn is embedded within a triangular region (see section 7.1), then it represents a $\langle 3.6.3.6 \rangle$ -pattern provided that its graph remains connected after the adjacencies of the triangular region have been altered by the marking function. Consider the triangular region ('super trellis') for the population of 6-hexes shown in figure 42(a). The vertices are labelled by their colours. The effect of the marking functions M_0 , M_1 , and M_2 are respectively illustrated in figures 42(b), 42(c), and 42(d). Here the vertices are labelled by the tags of the tile type they represent in the $\langle 3.6.3.6 \rangle$ -tessellation. To generate the p -hexes in each of the three marked trellises by use of one of the techniques discussed in section 7.4. Therefore the $\langle 3.6.3.6 \rangle$ -pattern enumeration requires performing the polyhex enumeration thrice!

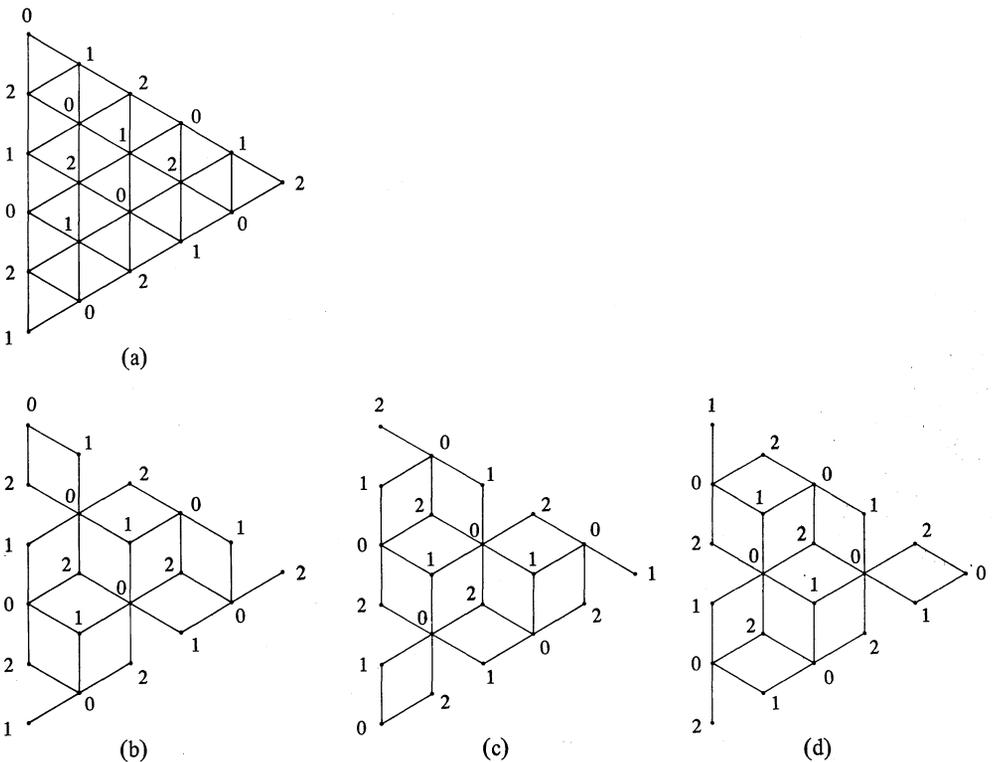


Figure 42.

12.2 Symmetries

Since we are dealing basically with polyhexes, we need only consider those symmetry motions that leave a bounding hexagon invariant in the plane. There is a problem however. The symmetry motions which preserve adjacencies of the tiles do not always preserve the colours of the tiles. They only preserve isomorphisms of the tile

colours onto themselves. That is, a tile of colour c_1 may be mapped onto a tile of colour c_2 . Consequently a symmetry motion may map a polyhex marked under M_i onto a polyhex marked under M_j , $i \neq j$. Since every hexagon in the pattern is always mapped onto a hexagon, all we need to know is the colour of the hexagonal tile after the symmetry motion has been applied, and from which the new marking function is easily determined. In other words, in addition to the list of symmetry elements summarized in tables 2 and 3, we need a list of the effect of the symmetry transformations on the indicators of the marking functions.

12.3 Coding

Figure 43 shows another peculiar phenomenon which may be attributed to the marking function. Notice that these are two isomorphic patterns that correspond to the same graph, the difference being the application of different marking functions. Hence we cannot use the standard binary representation to describe the pattern. We must be able to distinguish between the marking functions that apply. A radix 4 scheme solves this problem entirely.

Every $\langle 3.6.3.6 \rangle$ -pattern may be uniquely represented by a $(l_x + 1)$ -tuple of integers, $\Gamma = (\gamma_{-l_x}, \gamma_{-l_x+1}, \dots, \gamma_0)$, where l_x denotes the x -diameter of the bounding hexagon. Let c denote the current marking indicator. Then every vertex (x, y, z) of the trellis in the pattern contributes

$$\{3 - M_c[(x - y) \bmod 3]\}4^{-y} \text{ to } \gamma_x .$$

A pattern is canonical—that is, the representative of its free equivalence class—if and only if

$$\Gamma \geq \Gamma_\tau, \quad \tau \in T_h ,$$

where τ denotes the symmetry motion and T_h is the group of symmetries of the bounding hexagon, h . $\Gamma_\tau = (\gamma'_{-l_x}, \gamma'_{-l_x+1}, \dots, \gamma'_0)$ may be easily computed as follows. Let $\tau(c)$ denote the colour of the tile that represents the hexagon after the application of τ . Then every tile (x, y, z) in the original pattern contributes

$$\{3 - M_{\tau(c)}\{[\tau(x) - \tau(y)] \bmod 3\}\}4^{-\tau(y)} \text{ to } \gamma_{\tau(x)} .$$



Figure 43.

12.4 Algorithms

The algorithm is the same as the polyhex enumeration but with the following modifications. The initial vertex as before is chosen from the line $x = 0$. This defines the root of the search. We select in turn the marking-function indicator for this root and keep it fixed for the remainder of the search. The set of valid neighbours of any vertex used to augment the current partial graph is dictated by the sets in 12.0.1 through 12.0.3 depending upon the tag of the vertex. The pattern is coded according to section 12.3. The first occurrence of the vertex which represents the hexagon in the corresponding $\langle 3.6.3.6 \rangle$ -pattern is recorded to facilitate the symmetry transformation for canonical testing. This vertex must be either the root of the search or the second vertex chosen in the construction of the graph.

13 $\langle 3.12.12 \rangle$ -patterns

Consider the $\langle 3.12.12 \rangle$ -tessellation, shown in figure 44(a). Suppose we coalesce the triangles to a point. The result is the hexagonal tessellation. For ease of explanation we will emphasize the points in the manner shown in figure 44(b). The hexagons in the latter tessellation represent the dodecagons and the points correspond to the triangles. In a similar fashion every $\langle 3.12.12 \rangle$ -pattern corresponds to some polyhex with certain points distinguished or 'marked'. Examples of $\langle 3.12.12 \rangle$ -patterns and the corresponding marked polyhexes are shown in figure 45.

Therefore the $\langle 3.12.12 \rangle$ -pattern enumeration problem may be solved by the following two-stage process. First let us construct in toto the list of connected patterns consisting of n dodecagons, for all $n \leq p$. Then for each pattern in this list insert, if possible, the requisite number $(p - n)$ of triangles at the appropriate places. This procedure is equivalent to marking $p - n$ points on each n -hex from the list of n -hexes for all $n \leq p$. At this stage it should be remarked that any symmetry transformation that applies to the polyhexes also applies to the $\langle 3.12.12 \rangle$ -patterns, whence only the list of representative or canonical polyhexes need be considered. From section 7 we have an algorithm that generates polyhexes canonically within their bounding hexagons, which are in turn embedded within a triangular region (figure 18). The remainder of this section is devoted to the description of how the polyhexes may be marked. We will assume that $n < p$, for otherwise there is precisely one $\langle 3.12.12 \rangle$ -pattern with content p that corresponds to a p -hex, and this pattern consists of only dodecagons.

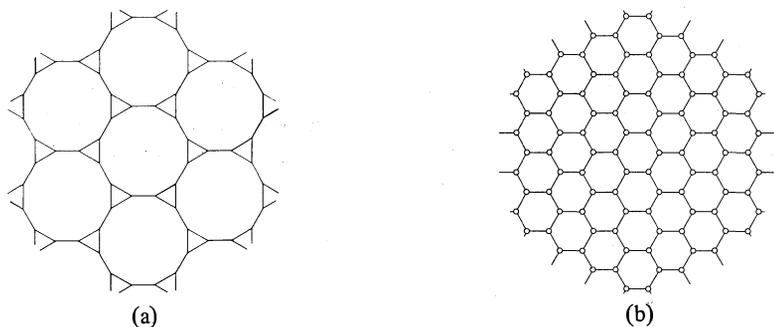


Figure 44.

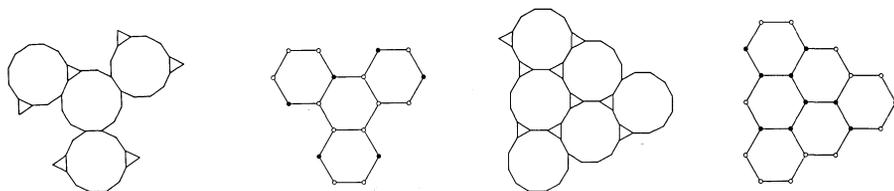


Figure 45.



Figure 46.

The point-marking problem is not the straightforward combinatorial exercise it may appear at first—namely selecting $p - n$ points from the list of points on the polyhex to be marked. Consider the two point-marked versions of the same polyhex shown in figure 46. The two $\langle 3.12.12 \rangle$ -patterns represented by these marked polyhexes are in fact isomorphic. This may be attributed to the fact that the points map onto one another under an element of the symmetry group that leaves the polyhex invariant in the plane. We may therefore say that the two sets of points are ‘indistinguishable’ or are in the same *block* of the *automorphism partition* of the points under the symmetries that leave the polyhex invariant in the plane. Therefore, in order to generate the representative or canonical $\langle 3.12.12 \rangle$ -patterns, a description for the sets of indistinguishable points for each polyhex is necessary.

13.1 *Representation of the points on the polyhex*

Recall that every polyhex with content less than or equal to p may be embedded within a graph represented by a triangular region whose vertices have integral coordinates (x, y, z) with $x + y + z = 0$, where $-p < x, y \leq 0$ and $0 \leq z < p$. This graph consists of triangular faces. Suppose we insert a vertex in each face, and adjoin these vertices by edges to the vertices at the corners of their faces. We obtain a subgraph of the graph of the $\langle 3.12.12 \rangle$ -tessellation. Each vertex in the triangular faces represents a triangle of the tessellation. Every other vertex represents a dodecagon. Effectively we may regard the original triangular graph with its vertices as the framework for describing the $\langle 3.12.12 \rangle$ -patterns. Rather than employ the vertex and edge insertion operation just described, we may allow each triangular face to stand for a triangle of the tessellation. Since every dodecagon is surrounded by six triangles we must add triangular faces along the perimeter of the triangular region to account for all possible $\langle 3.12.12 \rangle$ -patterns with content p . The result is the hexagonal region shown in figure 47(a). The trellis which houses the graphs of all possible $\langle 3.12.12 \rangle$ -patterns with content p is shown in figure 47(b).

The triangular faces may be assigned integral coordinates. For this we must first fix the relative order of the faces around a vertex. We adopt the convention indicated in figure 48. This convention may be explained as follows. Every vertex (x, y, z) has six neighbouring triangular faces designated as $N_j(x, y, z)$, $1 \leq j \leq 6$. $N_j(x, y, z)$ contains the coordinates of the triangular face in the j th position according to the

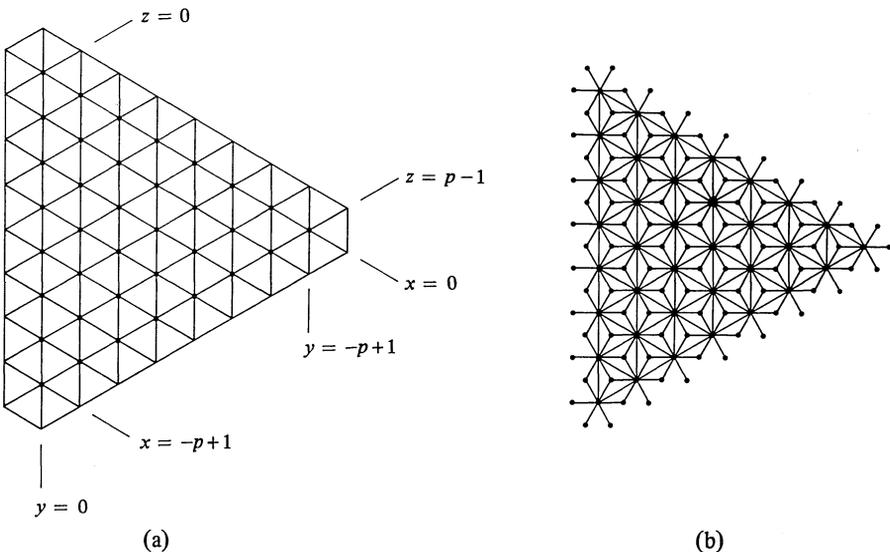


Figure 47.

relative order specified in figure 48. The coordinates of the faces around a vertex (x, y, z) are then given by:

$$\begin{aligned}
 N_1(x, y, z) &= (z, x - y) , & N_4(x, y, z) &= (z + 1, x - y) , \\
 N_2(x, y, z) &= (z, x - y - 1) , & N_5(x, y, z) &= (z + 1, x - y + 1) , \\
 N_3(x, y, z) &= (z + 1, x - y - 1) , & N_6(x, y, z) &= (z, x - y + 1) .
 \end{aligned}$$

Let loc be the function

$$\begin{aligned}
 loc(0) &= 1 , \\
 loc(z) &= loc(z - 1) + 2z + 1 , \quad z \geq 1 .
 \end{aligned}$$

Then every triangular face (x, y) in the hexagonal region [figure 47(a)] may be uniquely numbered in the range 1 through $p^2 + 4p + 1$ by the quantity

$$loc(x) + x + y + 1 .$$

Hence, given any vertex (x, y, z) , the six triangular faces surrounding it may be described by their face numbers calculated according to this formula. Conversely an inverse mapping for the triangular faces may be defined which uniquely describes each face's position in the hexagonal region. To each face (x, y) we may assign a pair $(j, \langle x', y', z' \rangle)$, which states that the face (x, y) is the j th neighbour surrounding the vertex (x', y', z') according to the convention in figure 48.

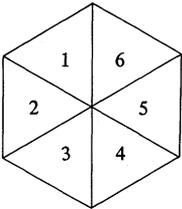


Figure 48.

13.2 Coding the triangles of a $\langle 3.12.12 \rangle$ -pattern

Suppose we are given a canonical n -hex defined within its bounding hexagon. Let the z -diameter $l_z = \hat{z} - \check{z}$. Then it is clear that the triangular faces that border this polyhex have coordinates (x, y) where x must lie in the range \check{z} through $\hat{z} + 1$. Since every pattern derived from this polyhex is a $(p - n)$ -combination of triangular faces, we must be able to represent this combination uniquely. Two methods suggest themselves: (a) a lexicographical ordering of the face numbers in the combination, or (2) an $(l_z + 1)$ -tuple of integers $E = (\epsilon_{\check{z}}, \epsilon_{\check{z}+1}, \dots, \epsilon_{\hat{z}+1})$. In the second method each face (x, y) in the combination of $p - n$ points contributes

$$2^{x-y+1} \text{ to } \epsilon_x .$$

13.3 Automorphism partition of the points and the generation of $\langle 3.12.12 \rangle$ -patterns

We have just seen how the points of a polyhex may be uniquely described as a set of face numbers calculated with respect to the hexagonal region shown in figure 47(a). We may recall from section 7.2 that a polyhex is represented by an $(l_x + 1)$ -tuple of integers Γ and that the polyhex is canonical if and only if

$$\Gamma \geq \Gamma_\tau , \quad \tau \in T_h ,$$

where Γ_τ represents the word of the isomorph under τ , which is an element of T_h , the group of symmetries that leave the bounding hexagon invariant in the plane (see tables 2 and 3).

Suppose we have just generated a n -hex with m points. Let $C = \{\tau: \Gamma = \Gamma_\tau\}$. Four cases arise.

13.3.1. Case 1: $m < p - n$.

In this case no $\langle 3.12.12 \rangle$ -pattern with content p may be derived from the polyhex. For instance a single hexagon cannot contribute to the $\langle 3.12.12 \rangle$ -patterns with content 8. All patterns with content 8 must contain at least two dodecagons.

13.3.2. Case 2: $m = p - n$.

In this case there is exactly one $\langle 3.12.12 \rangle$ -pattern that corresponds to the n -hex. Every point in the polyhex is marked.

13.3.3. Case 3: $n_\tau > p - n$ and $C = \{i\}$.

In this case the polyhex has no symmetries (other than identity, of course) and therefore every point in the polyhex is distinguishable. Each combination of $p - n$ points from the m points of the n -hex when marked yields a distinct $\langle 3.12.12 \rangle$ -pattern with content p . There are $\binom{m}{p-n}$ $\langle 3.12.12 \rangle$ -patterns with content p that correspond to the asymmetric n -hex.

13.3.4. Case 4: $m < p - n$ and $C \neq \emptyset, \{i\}$.

Here the polyhex remains invariant for every $\tau \in C$. Consider the effect of one such τ on the vertices of the graph of the polyhex. Under τ , a vertex v is permuted to occupy the position occupied by $\tau(v)$. At the same time it changes the relative order of the triangular faces that surround it. For instance the j th neighbour $N_j(v)$ of vertex v may become its k th neighbour $N_k[\tau(v)]$, $k \neq j$, when v moves into the position denoted by $\tau(v)$. In short τ defines a permutation of the triangular faces in which $N_j(v)$ maps into $N_k[\tau(v)]$ —or, $N_j(v)$ and $N_k[\tau(v)]$ are in the same block of the automorphism partition and therefore are indistinguishable. The permutations of the relative order of the triangular faces around a vertex under the symmetry motions isomorphic to the elements of D_6 are listed in table 10.

Table 10.

Group element	Symbol	Permutation of the relative order of the triangular faces surrounding a vertex
1 Identity	i	$1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4, 5 \rightarrow 5, 6 \rightarrow 6$
2 Rotation through π	π	$1 \leftrightarrow 4, 2 \leftrightarrow 5, 3 \leftrightarrow 6$
Vertical reflection about the axis that bisects:		
3 the x lines	v_x	$1 \leftrightarrow 5, 2 \leftrightarrow 4, 3 \rightarrow 3, 6 \rightarrow 6$
4 the y lines	v_y	$1 \leftrightarrow 3, 4 \leftrightarrow 6, 2 \leftrightarrow 2, 5 \leftrightarrow 5$
5 the z lines	v_z	$2 \leftrightarrow 6, 3 \leftrightarrow 5, 1 \rightarrow 1, 4 \rightarrow 4$
Horizontal reflection about:		
6 the x lines	h_x	$1 \leftrightarrow 2, 3 \leftrightarrow 6, 4 \leftrightarrow 5$
7 the y lines	h_y	$1 \leftrightarrow 6, 2 \leftrightarrow 5, 3 \leftrightarrow 4$
8 the z lines	h_z	$1 \leftrightarrow 4, 2 \leftrightarrow 3, 5 \leftrightarrow 6$
9 Rotation through $\frac{2}{3}\pi$	$\frac{2}{3}\pi$	$1 \rightarrow 3 \rightarrow 5 \rightarrow 1$ $2 \rightarrow 4 \rightarrow 6 \rightarrow 2$
10 Rotation through $-\frac{2}{3}\pi$	$\frac{2}{3}\pi$	$1 \rightarrow 5 \rightarrow 3 \rightarrow 1$ $2 \rightarrow 6 \rightarrow 4 \rightarrow 2$
11 Rotation through $\frac{1}{3}\pi$	$\frac{1}{3}\pi$	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 1$
12 Rotation through $-\frac{1}{3}\pi$	$\frac{1}{3}\pi$	$1 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$

Table 10 may be read as follows. Consider the symmetry motion π . Its permutation is described as $1 \leftrightarrow 4, 2 \leftrightarrow 5, \text{ and } 3 \leftrightarrow 6$. That is, $N_1(v)$ and $N_4[\pi(v)]$ are indistinguishable, $N_4(v)$ and $N_1[\pi(v)]$ are indistinguishable, $N_2(v)$ and $N_5[\pi(v)]$ are indistinguishable, and so on. Another example is the $\frac{1}{3}\pi$ rotation, whose permutation is described as $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 1$. Here $N_1(v)$ and $N_2[\frac{1}{3}\pi(v)]$ are in the same block, $N_2(v)$ and $N_3[\frac{1}{3}\pi(v)]$ are in the same block, and so on. Table 10 forms the basis for constructing a description of the automorphism partition of the points in the polyhex.

The computational steps that have to be employed to generate the (3.12.12)-patterns from the given polyhex are now briefly described. Initially let F_1, F_2, \dots, F_m be one-element sets containing the face numbers of the triangles bordering the polyhex. Suppose v_q is the q th vertex chosen in the search procedure (algorithm 3). We examine each triangular neighbour of v_q , and if it has not been examined before we perform the following operations. Let j be the neighbour examined. Let m denote the current number of triangular faces that border the partial graph G_q .

- Step 1: Mark $N_j(v_q)$ as examined.
- Step 2: $F_{m+\leftarrow 1} \leftarrow \{N_j(v_q)\}$.
- Step 3: Define the inverse mapping $I_{N_j(v_q)} \leftarrow (j, v_q)$.

For the purpose of backtracking every 'new' neighbour examined must be recorded.

Consider a $\tau \in C$. For each face f in the list, examine the inverse mapping $I_f = (j, v)$. That is, $f = N_j(v)$. From table 10 suppose $N_j(v)$ maps into $N_k[\tau(v)]$. We have the permutation $f \rightarrow N_k[\tau(v)]$. Let f be in set F_α and $N_k[\tau(v)]$ in set F_β . Two cases arise

- 13.3.4.1. $\alpha = \beta$. Do nothing.
- 13.3.4.2. $\alpha \neq \beta$. Perform the following set manipulations:

$$F_\alpha \leftarrow F_\alpha \cup F_\beta \quad \text{and} \quad F_\beta \leftarrow \emptyset .$$

That is, merge the sets F_α and F_β into a single set. It should be clear that if an element in F_α is indistinguishable from an element in F_β then all elements in both sets are indistinguishable.

This process is repeated for every $\tau \in C$. Let the resulting sets be renamed from 1 through m_b such that only the nonnull sets are considered. Moreover let them be arranged in order of decreasing cardinalities. That is, we have the collection $\{F_1, F_2, \dots, F_{m_b}\}$, where $|F_1| \geq |F_2| \geq \dots \geq |F_{m_b}|$. Each F_j is a block of the automorphism partition of the points.

Suppose $n = p - 1$. Then clearly there are m_b distinct patterns with content p that may be derived from the given polyhex.

On the other hand when $n < p - 1$ we have to generate the combinations of $p - n$ points from the m points which are coded according to the method suggested in section 13.2. A (3.12.12)-pattern is canonical if and only if

$$E \geq E_\tau, \quad \tau \in C,$$

where E_τ represents the combination under τ and is computed in the usual manner.

The combinations may be generated using a backtrack search strategy similar to that adopted in algorithm 3. The root of each search is a face from each set F_j , $1 \leq j \leq m_b$. Once a search has been carried out with this root, all faces in the same block of the automorphism partition of the points can be forbidden for searches with different roots. From coding considerations it increases the efficiency if the faces in each F_j are ordered according to decreasing x - and increasing y -coordinates. We can

employ the same type of forbidding mechanism as described in section 5. At any search level k , once a face has been forbidden it remains forbidden for all other search levels $j > k$. Moreover we can search for further combinations from a given level k if and only if the number of unforbidden faces is not less than $p - n - k$.

From a computational viewpoint the sets F_α refer to the names of the sets and are distinct from the elements they contain. Therefore any of the UNION-FIND algorithms for set representation may be used to perform some of the manipulations described in this section [see, for example, Aho et al (1974), Tarjan (1975), or Horowitz and Sahni (1976) for details regarding possible computational implementations].

14 Conclusions

In this paper we have presented the following.

14.1 *A computational theory for constructing spatial patterns composed of elements of tessellations*

This theory incorporates the topological and geometrical nature of these tessellations. The topology is described through graphs. The geometry is reflected within the integral coordinate systems associated with the tiles of the tessellation. In certain cases additional coordinates such as the tile designator are introduced to account for the spatial orientations of the individual tiles in the tessellation. Properties of tessellations and their imbedded patterns may then be expressed in terms of simple graphical properties. For instance, distances between tiles are simple graph distances which have simple integral formulae. In fact the integral coordinate system is necessary from a computational viewpoint since the use of traditional coordinates for these tessellations will result in expressions that may require transcendental quantities, which can only be approximated by sequences of rationals to within some topological radius of convergence on the real line defined by the word size of computer memory. This highlights the following point: to use the computer effectively one may sometimes have to distort the space in which the patterns are defined in order to represent them.

A pattern is defined within a finite section of the tessellation, termed the bounding region, as a labelled subgraph of the graph or trellis of this region. An algorithm that generates labelled subgraphs of a graph is presented in section 5. This forms the general framework for enumerating tessellation designs. Spatial transformations are expressed in terms of integral arithmetic, which in turn may be reduced to permutations of the labels of the vertices of the trellis.

14.2 *An implicit description for spatial patterns by means of data structures*

Each pattern corresponds to a data structure. The pattern properties are computationally equivalent to algebraic-type operations on these data structures. The enumeration algorithm essentially generates a set of data structures represented by vectors of numbers. In other words the computational theory provides the basis for nonnumerical descriptions for spatial patterns.

14.3 *Algorithms for patterns from the archimedean tessellations*

Many of the algorithms have been implemented—some in ALGOLW on the IBM 370/158 computer at the University of Waterloo and the rest in ALGOL68 on the IBM 370/165 computer at the University of Cambridge. The results for the regular tessellations (which are not given here) agree with those published elsewhere (Lunnon, 1971; 1972; 1975). In the case of the polycubes our results were broken down according to the bounding regions, as opposed to Lunnon's total counts. The implemented programs minimize computation by performing almost all the arithmetic calculations exactly once, and during the actual enumeration phase the computations essentially involve table look-ups (or simply memory fetches).

Each example considered in this paper attempts to illustrate slightly different aspects of spatial enumerations.

- (a) The polyomino routine describes how patterns may be represented by words over Z_2 and how these may be reduced to vectors of length $O(p)$. Moreover each vector requires $O(p)$ time to construct. Therefore the worst case complexity occurs when each pattern requires a separate path in the search tree (which is normally not the case) and is $O(pN_p)$, where N_p is the number of distinct patterns.
- (b) The routines for polyiamonds, $\langle 3.6.3.6 \rangle$ -patterns, and $\langle 3.12.12 \rangle$ -patterns demonstrate that, by suitable operations on the hexagonal tessellations, these patterns may be derived from the polyhexes. Moreover the polyhex routine shows that by an appropriate manipulation of the distance criteria the need to define explicitly the individual bounding regions may be eliminated. This is also true for the other patterns considered in this paper. (Essentially the modified distance criteria include the fixing conditions on the orientations of the bounding regions.)
- (c) The polycube and poly n -cube, $n \geq 4$, routines illustrate the extension of the theory to higher-dimensional patterns. The algorithms are defined in exactly the same manner as was done for the planar patterns.
- (d) The $\langle 3.3.3.4.4 \rangle$ - and $\langle 3.3.4.3.4 \rangle$ -patterns indicate the utility of tile designators in devising integral coordinates. Moreover they emphasize the point made earlier that to simplify computation one may sometimes have to distort the space in which the patterns are embedded.
- (e) The section on $\langle 4.8.8 \rangle$ -patterns highlights the complexity of the graph structure of the tessellation graph. It also demonstrates the need for the careful construction of the graph-distance criterion used in algorithm 3. It is interesting to note that some of the $\langle 4.8.8 \rangle$ -patterns may be obtained by point marking polyominoes in a manner similar to that for the $\langle 3.12.12 \rangle$ -patterns.
- (f) Though the $\langle 3.3.3.3.6 \rangle$ -, $\langle 3.4.6.4 \rangle$ -, and $\langle 4.6.12 \rangle$ -patterns have not been explicitly considered in this paper, these may be derived from the polyhexes by suitable point *and* line marking operations in a manner similar to that for the $\langle 3.12.12 \rangle$ -patterns. Indeed the $\langle 3.3.3.3.6 \rangle$ -patterns are a special class of the $\langle 3.4.6.4 \rangle$ -patterns, which in turn are a special class of the $\langle 4.6.12 \rangle$ -patterns.

4.4 Algorithms which satisfy the definition for efficient algorithms stated in the introduction

By theorem 1 each labelled subgraph is uniquely generated. Isomorphisms are tested for by a lexicographical comparison of two vectors of length $O(p)$. The order of groups involved is $O(n)$, where n denotes the size of the largest polygonal tile. Hence isomorphism requires at worst $O(np)$ time, where $n \leq 12$. Moreover there is a unique lexicographical ordering of the patterns in each free equivalence class. The storage is dominated by the memory required to house the trellis, which is $O(p^2)$.

The algorithms presented in this paper may be easily modified to generate patterns in which the tiles are not necessarily regular polygons, though the tessellations satisfy the valency requirement. In these cases only the group of symmetry motions that apply changes.

Acknowledgements. We thank Barbara Jones for typing the manuscript, Sally Boyle for drawing the figures, and Roger Lowry of Pion Limited for many helpful comments. This work has been carried out within the context of research programmes supported by the Open University Research Fund, the Science Research Council, and the Department of Systems Design at the University of Waterloo.

References

- Aho A V, Hopcroft J E, Ullman J D, 1974 *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, Mass)
- Bitner J R, Reingold E M, 1975 "Backtrack programming techniques" *Communications of the ACM* 18 651-656
- Bolker E D, Crapo H, 1977 "How to brace a one-story building" *Environment and Planning B* 4 125-152
- Bondy J A, Murty U S R, 1976 *Graph Theory and Its Application* (Macmillan, London)
- Cornell D G, Mathon R A, 1978 "Algorithmic techniques for the generation and analysis of strongly regular graphs and other combinatorial configurations" *Annals of Discrete Mathematics* 2 1-32
- Eden M, 1958 "A probabilistic model for morphogenesis" in *Symposium on Information Theory in Biology* Eds H P Yockey and others (Pergamon Press, New York) pp 359-370
- Eden M, 1960 "A two-dimensional growth process" *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability* 4 223-239
- Edmonds J, 1965 "Paths, trees, and flowers" *Canadian Journal of Mathematics* 17 449-467
- Fillmore J P, Williamson S G, 1974 "On backtracking: a combinatorial description of the algorithm" *SIAM Journal on Computing* 3 41-55
- Gaunt D S, Guttman A J, 1974 "Asymptotic analysis of coefficients" in *Phase Transitions and Critical Phenomena, Volume 3* Eds C Domb, M S Green (Academic Press, London) pp 181-243
- Golomb S W, 1954 "Checkerboard and polyominoes" *American Mathematical Monthly* 61 672-682
- Golomb S W, 1961a "The general theory of polyominoes, part 1, dominoes, pentominoes and checkerboards" *Recreational Mathematics Magazine* 4 3-12
- Golomb S W, 1961b "The general theory of polyominoes, part 2, patterns and polyominoes" *Recreational Mathematics Magazine* 5 3-12
- Golomb S W, 1961c "The general theory of polyominoes, part 3, pentomino exclusion by monominoes" *Recreational Mathematics Magazine* 6 3-20
- Golomb S W, 1962 "The general theory of polyominoes, part 4, extensions of polyominoes" *Recreational Mathematics Magazine* 8 7-16
- Golomb S W, 1965 *Polyominoes* (Scribner, New York)
- Golomb S W, Baumert L D, 1965 "Backtrack programming" *Journal of the ACM* 12 516-524
- Harary F, 1969 *Graph Theory* (Addison-Wesley, Reading, Mass)
- Harary F, March L, Robinson R W, 1978 "On enumerating certain design problems in terms of bicoloured graphs with no isolates" *Environment and Planning B* 5 31-43
- Harary F, Palmer E M, 1973 *Graphical Enumeration* (Academic Press, New York)
- Harary F, Palmer E M, Read R C, 1975 "The cell growth problem for arbitrary polygons" *Discrete Mathematics* 11 371-389
- Harary F, Read R C, 1970 "The enumeration of tree-like polyhexes" *Proceedings of the Edinburgh Mathematical Society* 17 1-13
- Horowitz E, Sahni S, 1976 *Fundamentals of Data Structures* (Computer Science Press, Woodland Hills, Calif.)
- Kelly J B, 1966 "Polynomials and polyominoes" *American Mathematical Monthly* 73 464-471
- Klarner D, 1965 "Some results concerning polyominoes" *Fibonacci Quarterly* 3 9-20
- Klarner D, 1967 "Cell growth problems" *Canadian Journal of Mathematics* 19 851-863
- Klarner D, 1969 "Methods for the general cell growth problem" in *Combinatorial Theory and Its Application III* Eds P Erdos, A Renyi, V T Sós (North-Holland, Amsterdam)
- Klarner D, Rivest R L, 1973 "A procedure to improve the upper bound for the number of n -ominoes" *Canadian Journal of Mathematics* 25 585-602
- Lehmer D H, 1964 "The machine tools of combinatorics" in *Applied Combinatorial Mathematics* Ed. F E Beckenbach (John Wiley, New York)
- Littlewood D E, 1931 "The groups of the regular solids in n -dimensions" *Proceedings of the London Mathematical Society* 32 10-20
- Lunnun W F, 1969 "Three combinatorial problems" PhD Thesis, Department of Mathematics, University of Manchester, Manchester
- Lunnun W F, 1971 "Counting polyominoes" in *Computers and Number Theory* Eds A O L Atkin, B J Birch (Academic Press, London) pp 347-372
- Lunnun W F, 1972 "Counting hexagonal and triangular polyominoes" in *Graph Theory and Computing* Ed. R C Read (Academic Press, New York) pp 87-100
- Lunnun W F, 1975 "Counting multidimensional polyominoes" *The Computer Journal* 18 366-367
- March L, Earl C F, 1977 "On counting architectural plans" *Environment and Planning B* 4 57-80
- Martin J L, 1974 "Computer technique for evaluating lattice constants" in *Phase Transitions and Critical Phenomena, Volume 3* Eds C Domb, M S Green (Academic Press, New York) pp 97-112

- Martin J L, Watts M C, 1971 "The endpoint distribution of self-avoiding walks in a crystal lattice" *Journal of Physics A* 4 456-463
- Palmer E M, 1972 "Variations of the cell growth problem" in *Lecture Notes in Mathematics 303. Graph Theory and Its Application* Eds Y Alavi, D R Lick, A T White (Springer, Berlin) pp 215-224
- Parkin T R, Lander I J, Parkin D R, 1967 "Polyomino enumeration results" paper presented at the Society for Industrial and Applied Mathematics (SIAM) Fall Meeting, Santa Barbara, Calif.
- Read R C, 1962 "Contributions to the cell growth problem" *Canadian Journal of Mathematics* 15 1-20
- Read R C, 1978 "'Every one a winner' or 'How to avoid isomorphism search when cataloguing combinatorial configurations'" *Annals of Discrete Mathematics* 2 107-120
- Reingold E M, Neivergelt J, Deo N, 1977 *Combinatorial Algorithms: Theory and Practice* (Prentice-Hall, Englewood Cliffs, NJ)
- Stiny G, Mitchell W J, 1978 "Counting Palladian plans" *Environment and Planning B* 5 189-198
- Sykes M F, Gaunt D S, Roberts P D, Wyles J A, 1972a "High temperature series for the susceptibility of the Ising model I. Two-dimensional lattices" *Journal of Physics A* 5 624-639
- Sykes M F, Gaunt D S, Roberts P D, Wyles J A, 1972b "High temperature series for the susceptibility of the Ising model II. Three-dimensional lattices" *Journal of Physics A* 5 640-652
- Sykes M F, Glen M, 1976 "Percolation process in two dimensions I" *Journal of Physics A* 9 87-95
- Sykes M F, Guttman A J, Watts M G, Roberts P D, 1972c "The asymptotic behaviour of self-avoiding walks and returns on a lattice" *Journal of Physics A* 5 653-660
- Tarjan R E, 1972 "Depth first search and linear graph algorithms" *SIAM Journal on Computing* 1 146-160
- Tarjan R E, 1975 "Efficiency of a good but not linear set union algorithm" *Journal of the ACM* 22 215-225
- Tarry G, 1895 "Le problème des labyrinthes" *Nouvelles Annales de Mathématiques* 14 187-190
- Tilley R C, 1970 "The cell growth problem for filaments" in *Proceedings of the Louisiana Conference on Combinatorics, Graph Theory and Computing* (Louisiana State University, Baton Rouge, La); also MA Thesis, York University, Toronto
- Walker R J, 1960 "An enumerative technique for a class of combinatorial problems" in *Combinatorial Analysis: Proceedings of Symposia in Applied Mathematics X* Eds R Bellman, M Hall Jr (American Mathematical Society, Providence, RI) pp 91-94
- Wells A F, 1977 *Three Dimensional Nets and Polyhedra* (John Wiley, New York)
- Williamson S G, 1973 "Isomorph rejection and a theorem of de Bruijn" *SIAM Journal on Computing* 2 44-59