DATA STRUCTURES AND ALGORITHMS FOR TILINGS I.

OLAF DELGADO-FRIEDRICHS

Abstract. Based on the mathematical theory of Delaney symbols, data structures and algorithms are presented for the analysis and manipulation of generalized periodic tilings in arbitrary dimensions.

1. Introduction

A *periodic tiling* is a subdivision of a plane or a higher-dimensional space into closed bounded regions called *tiles* without holes in such a way that the whole configuration can be reproduced from a finite assembly of tiles by repeatedly shifting and copying in as many directions as needed (c.f. [GS87]).

For the purpose of classification, mathematicians have invented a variety of symbolic descriptions for periodic tilings [Hee68, DDŠ80, GS81]. Likewise, computer scientist have developed data structures and programs for storing and manipulating tilings [Cho80]. Most of these, however, have been restricted to a rather limited range of applications.

The invention of Delaney symbols has not only provided a mathematical tool for a much more systematic way of studying the combinatorial structure of tilings, thus initiating what we call *combinatorial tiling theory* (cf. [Dre85, Dre87, DH87, Hus93, MP94, MPS97, DDH+99, Del01]). Delaney symbols also form the basis for concise and efficient data structures and algorithms in what might be called *computational tiling theory*.

This article is the first in a projected series of four publications on algorithmic aspects of Delaney symbol theory. I will shortly review the mathematical concept of Delaney symbols and present and analyze some basic algorithms. In forthcoming articles, I will address the relationship between Delaney symbols and tilings in more detail and present methods for enumerating tilings.

As a teaser, please look at the two tilings shown in Figure 1, which were brought to the attention of our group by L. Collatz. They seem to be very similar in their combinatorial structure, but are they actually equivariantly equivalent, i.e., can the first be deformed into the second without breaking symmetries? This is very hard to solve without the proper tools, but becomes very easy with Delaney symbols.

2. Delaney symbols

After introducing Delaney symbols by describing their construction for a two-dimensional tiling, I will give a general characterization and review some of the most important mathematical results.

Date: 14th May 2004.
Submitted to *Theoretical Computer Science*.

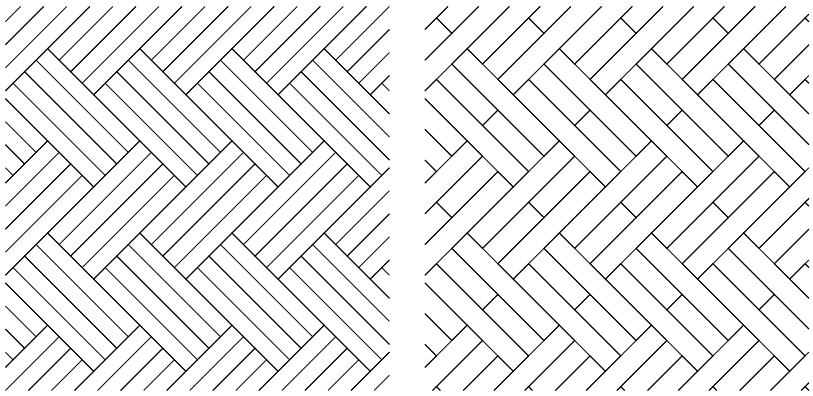

FIGURE 1. Are these tilings equivalent?

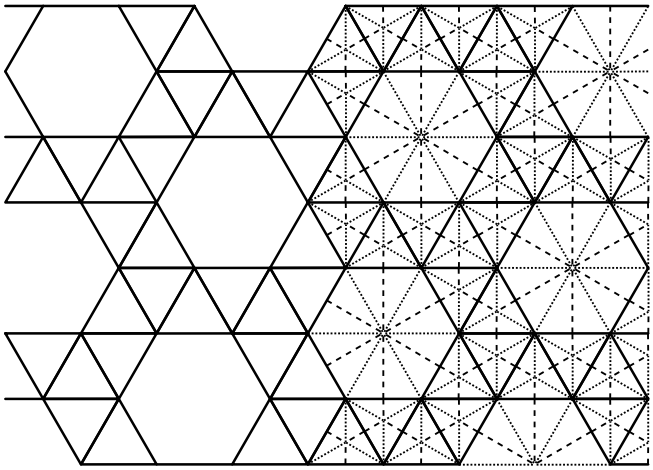

FIGURE 2. A tiling with its barycentric subdivision

Please note that in the following, the edges and vertices of a tiling will be defined in a purely combinatorial way. Thus, in Dimension 2, a vertex is a point where at least three tiles meet. An edge is a portion of the common boundary of two tiles in between two vertices.

A *barycentric subdivision* is of a given tiling is constructed as follows:

- Choose a point in the interior of each edge and each tile.
- For each tile, connect the point chosen in its interior with its vertices and all the points chosen on its edges by pairwise disjoint arcs.

Ideally, the barycentric subdivision should be constructed as to retain the symmetry of the tiling. This is always possible. In Figure 2, a small portion of a tiling is shown together with its barycentric subdivision.

Obviously, the barycentric subdivision is a triangulation. To avoid confusion, we will refer to its tiles as *chambers*. Each chamber has three types of vertices, namely one original vertex, one on an edge and one inside a tile. We label these '0', '1' and

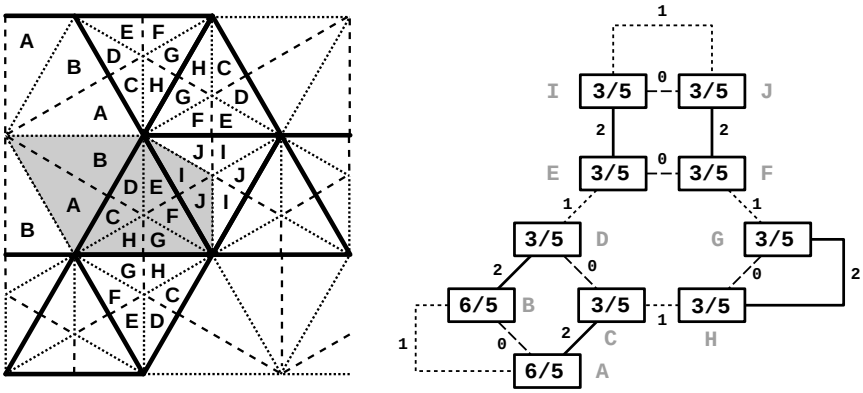

FIGURE 3. Chamber classes and the Delaney symbol.

'2', accordingly. There are also three types of edges. Edges are labelled the same as the vertices opposite to them. In Figure 2, edges labelled '0', '1' and '2' are shown dashed, dotted and solid, respectively.

Each chamber has three neighbors, which are distinguished by the type of edge they share with it. Thus, any finite portion of the triangulation can be described completely by listing for each chamber its three neighbors in order. For a given chamber t, we will denote these by $s_0(t)$, $s_1(t)$ and $s_2(t)$, respectively.

Next, we take symmetries into account. Two chambers are called symmetry equivalent if there is a symmetry of the tiling mapping one onto the other. In Figure 3, equivalent chambers are marked with a common letter. Clearly, there are 10 classes in this particular case, which bear the labels 'A' up to 'J'.

As corresponding neighbors of chambers in the same class belong to the same class again, we can assign to each class C its three "neighboring" classes $s_0(C)$, $s_1(C)$ and $s_2(C)$, respectively, where the class $s_i(C)$ consist of all the neighbors $s_i(t)$ of chambers t in class C. The set of classes together with its neighborhood relations is called the *Delaney set*[1]. It is convenient to envision the classes as nodes and the neighborhood relations as labelled edges of a graph. Consequently, the term *Delaney graph* is used as well. Note that the edges of the Delaney graph are undirected, because the neighborhood relations are reflective. Of course, the set of chambers of the barycentric subdivision can be regarded as a — possibly infinite — graph in the same way, which we will call the *chamber graph*.

It is always possible to choose a connected region containing one chamber of each type, as shown in gray in Figure 3. Such a region forms a particular *fundamental domain* for the tiling's symmetry group. A convenient way to explore the Delaney graph is to look at a fundamental domain together with its immedatiately surrounding chambers.

Clearly, the Delaney graph alone does not uniquely determine the tiling. This becomes obvious when we consider the archimedean solid, or spherical tiling, depicted in Figure 4, which has exactly the same Delaney graph as our plane example, but contains squares instead of regular hexagons.

[1]Named after M.S. Delaney, who's work [Del80] inspired the invention of Delaney symbols.

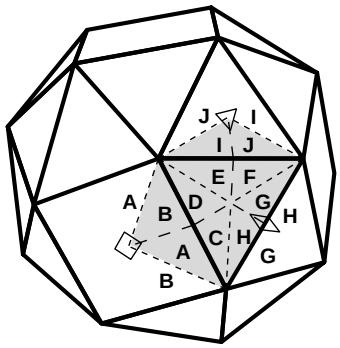

FIGURE 4. An archimedean solid.

We augment the Delaney graph by assigning to each class C of chambers the two numbers $m_0(C)$ and $m_1(C)$. The first of these gives the degree, i.e., the number of vertices, of the tiles containing chambers of this class, while the second gives the degree of the vertices adjacent to chambers of this class. By construction, neither of these numbers depends on the actual chamber chosen, so they are "well-defined". The augmented Delaney graph is called the *Delaney symbol* of the tiling in question. The Delaney symbol for the tiling in Figure 2 is shown as a labelled graph in Figure 3 to the right. The Delaney symbol for the archimedean solid of Figure 4, as a matter of fact, can be obtained by setting both $m_0(A)$ and $m_0(B)$ to 4 instead of 6.

Delaney symbols are insensitive to deformations of tilings as long as these change neither the topology nor the symmetries. More precisely, two tilings are *topologically equivalent* if there is a *homeomorphism* — a both-ways continuous transformation — mapping tiles of the first onto tiles of the second. They are *equivariantly equivalent* if there is such a transformation which in addition maps (by conjugation) the symmetry group of the first to the symmetry group of the second. By construction, Delaney symbols are invariants of combinatorial equivalence classes. The first and most important theorem reviewed here states that they are even sharp invariants.

Theorem 1 ([Dre85])**.** *Two tilings are equivariantly equivalent if and only if their respective Delaney symbols are isomorphic.*

Two Delaney symbols are isomorphic if one can be obtained from the other just by renaming the nodes. This can be efficiently tested, as will be demonstrated below.

Theorem 1 remains true in higher dimensions, where the construction of the Delaney symbol is performed in an analoguous way. In fact, it holds true whenever both spaces tiled are *simply connected manifolds*. A manifold, essentially, is a space that locally looks like an ordinary euclidean space everywhere, whereas a space is simply connected if every closed curve in it can be continuously deformed into a single point without leaving the space. An example of a manifold which is not simply connected is the surface of a doughnut, also called a torus.

Following is a characterization of "formal" Delaney symbols in arbitrary dimension, using those properties which are immediate from the construction.

Definition 2. A *Delaney symbol* of dimension n is a set $\mathbf{C}$ together with functions $s_0, \ldots, s_n$ from $\mathbf{C}$ into $\mathbf{C}$ and functions $m_0, \ldots, m_{n-1}$ from $\mathbf{C}$ into the positive integers, such that the following is true for all $C \in \mathbf{C}$ and all applicable i and j:

(DS0) The underlying Delaney graph is connected, i.e., each element can be mapped onto any other by repeatedly applying functions from the set $s_0, \ldots, s_n$.
(DS1) $s_i(s_i(C)) = C$.
(DS2) $s_i(s_j(C)) = s_j(s_i(C))$ whenever $j > i + 1$.
(DS3) $m_i(C) = m_i(s_i(C)) = m_i(s_{i+1}(C))$.
(DS4) $f_i^{m_i(C)}(C) = C$,
where $f_i^0(C) := C$ and $f_i^{k+1}(C) := s_i(s_{i+1}(f_i^k(C)))$.

For practical reasons, we will usually assume that Delaney symbols are finite. We will therefore restrict our attention to tilings which possess finite Delaney symbols, as periodic tilings do. We will refer to these as *generalized periodic tilings*. Among the generalized periodic tilings are tilings of spheres and also certain tilings of hyperbolic spaces.

The following notations will be useful:

Definition 3. Let $\mathbf{C}$ be an n-dimensional Delaney symbol. For $C \in \mathbf{C}$ and $0 \leq i < j \leq n$, define $r_{i,j}(C)$ as the smallest positive number r such that $f_{i,j}^r(C) = C$, where $f_{i,j}^0(C) = C$ and $f_{i,j}^{k+1}(C) = s_i(s_j(f_{i,j}^k(C)))$. Define $v_{i,j}(C)$ as the fraction $m_i(C)/r_{i,j}(C)$ if $j = i + 1$ and as $2/r_{i,j}(C)$ otherwise.

Every *face*, i.e., in the two-dimensional case, every vertex, edge and tile, of a tiling is represented by a unique vertex of its barycentric subdivision. This vertex is labelled i if it lies in an i-dimensional face. Two chambers t and t' share a common i-vertex if and only if they lie in the same connected component of the graph obtained by removing all i-edges from the chamber graph. These are exactly those chambers which have a non-empty intersection with the given face. This relationship carries over to the Delaney symbol, where symmetry equivalence classes of i-faces are represented by connected components of the partial Delaney graph obtained by removing all i-edges.

These connected components are an important tool in combinatorial tiling theory, especially in dimensions 3 and higher. Together with the relevant 'm'-functions, we refer to them as *subsymbols*. An I-subsymbol of an n-dimensional Delaney symbol $\mathbf{C}$ is defined by an element $C \in \mathbf{C}$ and a subset $I \subset \{0, \ldots, n\}$. It consists of all elements of $\mathbf{C}$ which can be reached from C by repeatedly applying only functions s_i with $i \in I$. A $\{0, \ldots, n-1\}$-subsymbol, for example, represents the combinatorial structure of a tile.

The Delaney symbol in Figure 3 contains the three $\{0, 1\}$-subsymbols $\{A, B\}$, $\{C, D, E, F, G, H\}$ and $\{I, J\}$.

Next, we consider how to determine whether a given formal Delaney symbol actually is derived from a tiling of, say, the euclidean plane. In dimension 2, this can be done using a simple numerical invariant:

Definition 4. Let $\mathbf{C}$ be a two-dimensional Delaney symbol. The *curvature* of $\mathbf{C}$ is defined as the sum

$$K(\mathbf{C}) := \sum_{C \in \mathbf{C}} k(C)$$

where

$$k(C) := \frac{1}{m_0(C)} + \frac{1}{m_1(C)} - \frac{1}{2}.$$

Theorem 5. *Let* $\mathbf{C}$ *be a two-dimensional Delaney symbol. Then* $\mathbf{C}$ *encodes a tiling of*

- *the hyperbolic plane if and only if* $K(\mathbf{C}) < 0$*,*
- *the euclidean plane if and only if* $K(\mathbf{C}) = 0$*,*
- *the sphere if and only if* $K(\mathbf{C}) > 0$ *and for all* $i, j \in \{0, 1, 2\}$ *and all* $C \in \mathbf{C}$*, the quantity*

$$\frac{4}{v_{i,j}(C) \cdot K(\mathbf{C})}$$

is a natural number,
- *no tiling at all otherwise.*

To illustrate this, for the Delaney symbol in Figure 3, we have

$$k(A) = k(B) = \frac{1}{6} + \frac{1}{5} - \frac{1}{2} = \frac{5 + 6 - 15}{30} = \frac{-4}{30}$$

and

$$k(C) = \cdots = k(J) = \frac{1}{3} + \frac{1}{5} - \frac{1}{2} = \frac{10 + 6 - 15}{30} = \frac{1}{30},$$

thus

$$K(\mathbf{C}) = \frac{2 \cdot (-4) + 8 \cdot 1}{30} = 0,$$

whereas for the Delaney symbol of the archimedean solid shown in Figure 4, we have

$$k(A) = k(B) = \frac{1}{4} + \frac{1}{5} - \frac{1}{2} = \frac{5 + 4 - 10}{20} = \frac{-1}{20} = \frac{-3}{60},$$

thus

$$K(\mathbf{C}) = \frac{2 \cdot (-3) + 8 \cdot 2}{60} = \frac{10}{60} = \frac{1}{6} > 0.$$

Moreover, we have $4/K(\mathbf{C}) = 24$, which implies that $v_{i,j}(C)$ must divide 24 for all applicable C, i and j. This is indeed the case.

Unfortunately, there is no complete, easily accessible proof of Theorem 5 available from the literature. A complete, but rather involved proof is contained in [Bal90]. A proof for the Euclidean case based on ideas by Dress appears in [Hus89]. There is a rather neat constructive proof easily following from [Hus93] together with some well-established facts on two-dimensional *orbifolds* (also called *space-forms*) [Vin93]. This material is somewhat beyond the scope of this article, though, and thus will appear in a forthcoming publication.

Here, I will only shortly indicate why the conditions stated for the spherical case are necessary. It is easy to see that for the chamber graph $\mathbf{C}_0$ of a spherical tiling, the number $K(\mathbf{C}_0)$ is just twice $F - E + V$, where F, E and V are the numbers of tiles, edges and vertices of that tiling, respectively. Indeed, this is true for any tiling of a closed surface. By Euler's well-known theorem on polyhedra, it follows that $K(\mathbf{C}_0) = 2 \cdot 2 = 4$. Now the size n of each chamber class must coincide with the size of the symmetry group and two chambers in the same class must have the same k-value. Consequently, for the Delaney symbol $\mathbf{C}$ of the same tiling, $4 = K(\mathbf{C}_0) = n \cdot K(\mathbf{C})$, which implies $n = 4/K(\mathbf{C})$ and $K(\mathbf{C}) = 4/n > 0$. It is, furthermore, obvious from the definitions that a value of $v_{i,j}(C)$ larger than 1

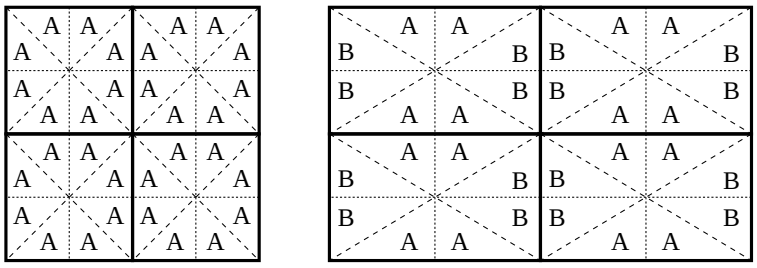

FIGURE 5. Two topologically, but not equivariantly equivalent tilings.

indicates a rotation of that order fixing the vertex — or, in general, the face of co-dimension 2 — of any chamber in class C at the intersection of its two co-dimension 1 faces labelled i and j. The order of such a rotation must divide the size of the symmetry group, i.e., $4/(K(\mathbf{C}) \cdot v_{i,j}(C))$ must be a natural number.

Theorem 5 relies on the Euler characteristic of surfaces. Since the Euler characteristic is always 0 in odd dimensions, no analogous result is available for three-dimensional Delaney symbols. In [Del01], a partial algorithm is described for the recognition of Delaney symbols of three-dimensional tilings.

To complete this section, let us consider tilings which are topologically, but not equivariantly equivalent. Figure 5 shows a simple tiling by squares and a topologically equivalent one by rectangles, both with barycentric subdivisions and letters marking respective chamber classes. In the rectangle tiling, the reflections at the diagonals are no longer symmetries of the tiling, so its Delaney symbol has two elements instead of just one in the case of the square tiling.

The rectangle tiling is called a *symmetry breaking* of the square tiling. There is a homeomorphism which maps tiles of the rectangle tiling onto tiles of the square tiling and symmetries of the rectangle tiling to symmetries of the square tiling, but not all symmetries of the square tiling are obtained in this way. Such a homeomorphism also takes chambers and chamber classes of the first tiling onto chambers and chamber classes of the second one, thereby inducing a well-behaved mapping between their Delaney symbols, which we call a *Delaney map* or just a map.

Definition 6. A function $f\colon \mathbf{C} \to \mathbf{C}'$ between Delaney symbols $\mathbf{C}$ and $\mathbf{C}'$ is called a *Delaney map* if and only if for each $C \in \mathbf{C}$ and for all applicable i:

- $f(s_i(C)) = s_i(f(C))$
- $m_i(f(C)) = m_i(C)$.

A Delaney map is called an *isomorphism* if it is one-to-one. A Delaney symbol is called *minimal* if every Delaney map defined on it is an isomorphism.

Every isomorphism has a reverse map which is a Delaney map, which justifies its name. For finite Delaney symbols, a Delaney map is an isomorphism if both symbols have the same size. A finite Delaney symbol is minimal if it can not be mapped onto a smaller one.

If $f\colon \mathbf{C} \to \mathbf{C}'$ is a Delaney map and $\mathbf{C}'$ is the Delaney symbol of a tiling, then $\mathbf{C}$ is the Delaney symbol of a symmetry breaking of that tiling. If $\mathbf{C}$ corresponds to some tiling, then a tiling corresponding to $\mathbf{C}'$ would have to have more symmetries, which might not always be possible. It has been shown, however, that for all $\mathbf{C}$

corresponding to either a two-dimensional or a euclidean three-dimensional tiling, $\mathbf{C}'$ will correspond to a tiling in the same geometry [Del01].

As will be shown in the next section, there is for every Delaney symbol a unique minimal image, which can be computed efficiently. This means that, at least for two-dimensional and euclidean three-dimensional tilings, every topological equivalence class has a unique representative with maximal symmetry.

3. Algorithms

In this section, I will present algorithms for finding subsymbols, for testing whether two Delaney symbols are isomorphic and for determining the minimal image of a Delaney symbol. Instead of pseudocode, I will use the programming language Python created by G. van Rossum (c.f. [Pyt00, Bea99]). This has the advantage that the code shown can actually be run, although it looks almost like pseudocode. A complete working demo program containing the code shown below can be obtained from the author's website as

`http://www.mathematik.uni-bielefeld.de/~delgado/TCS/code.py`

Note that lists in Python are indexed starting at 0. Negative indices count backwards, i.e., `a[-1]` is equivalent to `a[len(a)-1]`, which is the last entry of `a`. The command `del a[i]` removes a particular entry. Python's `range` function produces a list of consecutive integers. In particular, `range(n)` produces a list with first entry 0 and last entry $n-1$. As in C, the double equal sign `==` tests for equality, while the single equal sign `=` is used for assignment.

A Delaney symbol will be represented by four entities: its dimension, a list of elements and two dictionaries `s` and `m`. Instead of dictionaries, two-dimensional arrays or arrays of arrays can be used, of course.

As an example, the Delaney symbol for the rectangle tiling shown in Figure 5 can be created as follows:

```
dimension = 2
elements = [ 'A', 'B' ]

s = {}           # this creates an empty dictionary
s[0, 'A'] = 'A'; s[0, 'B'] = 'B'
s[1, 'A'] = 'B'; s[1, 'B'] = 'A'
s[2, 'A'] = 'A'; s[2, 'B'] = 'B'

m = {}
m[0, 'A'] = m[0, 'B'] = 4
m[1, 'A'] = m[1, 'B'] = 4
```

Please note that additional information as, for example, vertex coordinates, can be added easily. I will elaborate on extensions of Delaney symbols to present actual, i.e., geometrically realized, tilings in a forthcoming paper.

The dimensions of Delaney symbols of interest are usually small, so we may treat the dimension as constant.

Clearly, conditions (DS1) up to (DS4) are straightforward to test for. To check whether condition (DS0) holds, any method to explore connected components of a graph can be used. The following method proves useful for this and several other purposes.

Algorithm 7 (Index priority depth-first traversal)**.**

Input:

- A dictionary `s`, representing a Delaney graph.
- A list `indices` of valid indices, i.e., edge labels, for `s`.
- A list `seeds` of nodes of the Delaney graph.

Output: A spanning forest of the graph obtained by removing all edges with labels not in `indices`, including only those components which contain an element of `seed`. This forest is represented as a list of entries of the form (i, C), where C is a node and i is either the special object `None`, in which case C is the root of a new component, or else an element of `indices`, in which case it is the label of an edge from C towards the root of the current component.

Method:

```
def index_priority_depth_first_traversal(s, indices, seeds):
    seen = {}
    result = []

    for seed in seeds:
        if not seen.has_key(seed):
            result.append((None, seed))
            seen[seed] = 1
            stack = [seed]

            while stack:
                C = stack[-1]
                del stack[-1]
                for i in indices:
                    Ci = s[i, C]
                    if not seen.has_key(Ci):
                        result.append((i, Ci))
                        seen[Ci] = 1
                        stack.append(Ci)

    return result
```

For a Delaney symbol of size n and m indices given, Algorithm 7 obviously has time complexity $n * m$ and space complexity n. For appropriate inputs, its output can be analyzed in linear time to check connectivity or extract connected components and subsymbols.

The index priority depth-first traversal can be used to construct a canonical labelling for Delaney symbols. First, we must define a function `compare` (not shown here) to compare two labelled symbols of the same size and dimension. This is most easily done by converting the data for each symbol into a consecutive string of numbers and comparing the results lexicographically. The particular choice of comparison function is irrelevant here as long as it is used consistently. Following is an algorithm to construct a canonical form:

Algorithm 8 (Canonical form)**.**

Input: A Delaney symbol.

Output: A Delaney symbol in canonical form, with nodes labelled consecutively from 0. The algorithm produces identical output for isomorphic Delaney symbols.

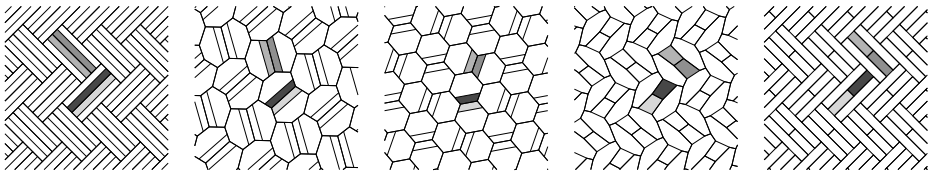

FIGURE 6. A continuous deformation between tilings.

Method:

```
def canonical_form(dimension, elements, s, m):
    best_s = best_m = None
    indices = range(dimension + 1)
    n = len(elements)

    for seed in elements:
        edges = index_order_depth_first_traversal(s, indices, seed)
        old2new = {}
        new2old = {}
        for k in range(n):
            (i, C) = edges[k]
            old2new[C] = k
            new2old[k] = C
        s_new = {}
        m_new = {}
        for C_new in range(n):
            C_old = new2old[C_new]
            for i in range(dimension + 1):
                s_new[i, C_new] = old2new[s[i, C_old]]
            for i in range(dimension):
                m_new[i, C_new] = m[i, C_old]
        if (best_s is None or
            compare(dimension, n, s_new, m_new, best_s, best_m) < 0):
            best_s = s_new
            best_m = m_new

    return (dimension, range(n), best_s, best_m)
```

Clearly, Algorithm 8 has quadratic time and linear space complexity. For each node, the symbol is relabelled in index priority depth-first order starting at that node. Of all these numberings, the one leading to the best, i.e. smallest, labelled symbol with respect to the `compare` function is chosen. Because all possible starting nodes are used, the outcome does not depend on the inital labelling and the canonical form is the same for isomorphic symbols.

As an example, we can now easily solve the question of whether the tilings shown in Figure 1 are equivalent. It turns out that they are, and, indeed, Figure 6 indicates that they can even be deformed continuously into one another.

Finally, consider the problem of finding the minimal image of a given n-dimensional Delaney symbol $\mathbf{C}$. Let C and D be two arbitrary elements of $\mathbf{C}$. Assume that there is some unknown Delaney map f which maps C and D to a common element C'. Then, by the Definitions, we must have $m_i(C) = m_i(D)$

for $i \in \{0, \dots, n-1\}$ and, moreover, for $i \in \{0, \dots, n\}$, $s_i(C)$ and $s_i(D)$ must be mapped to the common element $s_i(C')$. By performing a parallel traversal of the Delaney graph starting with the pair (C, D), we can partition $\mathbf{C}$ into equivalence classes $\mathbf{C}^1, \dots, \mathbf{C}^k$ such that two elements in the same class must have the same image under f. Moreover, if a contradiction appears during the traversal, we know that there is no such Delaney map.

The following algorithm does the trick. To keep track of equivalence classes, it uses a so-called *union-find* or *partition* data structure (cf. [Sed88]). We represent this as an instance `p` of a class `Partition` with two methods `union` and `find`. The `find` method returns a representative for the equivalence class its argument is in. Initially, every item is in a class of its own. The `union` method unites the classes of its two argument. In the code, a third method `copy` is used, which produces a copy of the given structure. This is necessary because in Python, assignment of complex objects produces a new reference, but does not copy the data. Partition data structures are a well-known subject, so the code for the class `Partition` is not shown. Note however, that it can be implemented as to require space $O(n)$ and accumulated running time $O(m \cdot a(n))$ for any sequence of $m > n$ `find` or `union` operations on a total set of size n, where a is the inverse Ackermann function.

Algorithm 9 (Checking and propagating equivalence)**.**

Input: A Delaney symbol, two elements C and D and a partition.
Output: Nothing (`None`), if C and D can not be set equivalent, else the partition resulting from setting them equivalent and drawing all consequences.
Method:

```
def try_to_set_equivalent(dimension, elements, s, m, C, D, p):
    for i in range(dimension):
        if m[i,C] != m[i,D]:
            return None
    if p.find(C) == p.find(D):
        return p

    p = p.copy()
    p.union(C, D)
    stack = [(C, D)]

    while stack:
        (C, D) = stack[-1]
        del stack[-1]
        for i in range(dimension+1):
            Ci = s[i,C]
            Di = s[i,D]
            for j in range(dimension):
                if m[j,Ci] != m[j,Di]:
                    return None
            A = p.find(Ci)
            B = p.find(Di)
            if A != B:
                p.union(A, B)
                stack.append((Ci, Di))
    return p
```

If Algorithm 9 returns a new partition, all the m-functions are constant on classes and each s_i maps the members of one class into the same "neighbor"-class. Thus, the s_i and m_i can be defined classwise and the set of classes forms a Delaney symbol.

By calling Algorithm 9 with C staying the same and D ranging through the elements of $\mathbf{C}$, every element that can have the same image as C at all will eventually be found. As above, the final collection of equivalence classes will form a Delaney symbol $\mathbf{C}_0$ which will necessarily be minimal. To see this, note that a Delaney map is one-to-one whenever there is an element in its image with a unique preimage. Thus, if $\mathbf{C}_0$ were not minimal, it would have been possible to unite the class of C with some other class. Moreover, every image of $\mathbf{C}$ by a Delaney map f must map onto $\mathbf{C}_0$ by the unique Delaney map which maps $f(C)$ to the class of C.

Here is the code that constructs $\mathbf{C}_0$:

Algorithm 10 (Minimal image)**.**

Input: A Delaney symbol.

Output: The unique minimal image of the given Delaney symbol by a Delaney map.

Method:

```
def minimal(dimension, elements, s, m):
    p = Partition()
    C = elements[0]
    for D in elements:
        q = try_to_set_equivalent(dimension, elements, s, m, C, D, p)
        if q is not None:
            p = q

    old2new = {}
    new2old = {}
    k = 0
    for C in elements:
        D = p.find(C)
        if not old2new.has_key(D):
            old2new[D] = k
            new2old[k] = D
            k = k + 1
        old2new[C] = old2new[D]

    s_new = {}
    m_new = {}
    for C_new in range(k):
        C_old = new2old[C_new]
        for i in range(dimension + 1):
            s_new[i, C_new] = old2new[s[i, C_old]]
        for i in range(dimension):
            m_new[i, C_new] = m[i, C_old]

    return (dimension, range(k), s_new, m_new)
```

Clearly, Algorithm 9 performs at most $n-1$ `union` operations for a d-dimensional Delaney symbol of size n. A pair of elements is pushed onto the stack only following a union operation, so the total number of find operations is $O(nd)$. The copy

operation clearly takes time $O(n)$, while all other operations run in time $O(nd)$, thus the total running time of Algorithm 9 is $O(nd \cdot a(n))$.

As Algorithm 9 is called n times in Algorithm 10 and everything else there runs in time $O(nd)$, we obtain a total running time of $O(n^2 d \cdot a(n))$. Thus, for all practical purposes, we may assume a quadratic time bound for Algorithm 10.

It would be interesting to know whether the construction of canonical forms or minimal images or the test for minimality can be done in worst case running time less than $O(n^2)$.

References

[Bal90] L. Balke. Diskontinuierliche Gruppen als Automorphismengruppen von Pflasterungen. Bonner mathematische Schriften, Bonn, Germany, 1990. (master thesis).

[Bea99] David Beazley. *Python essential reference*. New Riders, 1999.

[Cho80] W.W. Chow. Automatic generation of interlocking shapes. *Computer Aided Design*, 12:29–34, 1980.

[DDH+99] O. Delgado Friedrichs, A.W.M. Dress, D.H. Huson, J. Klinowski, and A.L. Mackay. Systematic enumeration of crystalline networks. *Nature*, 400:644–647, 1999.

[DDŠ80] B.N. Delone, N.P. Dolbilin, and M.I. Štogrin. Combinatorial and metric theory of planigons. *Proc. of the Steklov Inst. of Math.*, 4:111–141, 1980.

[Del80] M.S. Delaney. Quasisymmetries of space group orbits. *Match*, 9:73–80, 1980.

[Del01] O. Delgado-Friedrichs. Recognition of flat orbifolds and the classification of tilings in R^3. *Discrete and Computational Geometry*, 26(4):549–571, 2001.

[DH87] A.W.M. Dress and D.H. Huson. On tilings of the plane. *Geometriae Dedicata*, 24:295–310, 1987.

[Dre85] A.W.M. Dress. Regular polytopes and equivariant tessellations from a combinatorial point of view. In *Algebraic Topology, Göttingen 1984*, number 1172 in Lecture Notes in Math., pages 56–72. Springer, Berlin, 1985.

[Dre87] A.W.M. Dress. Presentations of discrete groups, acting on simply connected manifolds. *Adv. in Math.*, 63:196–212, 1987.

[GS81] B. Grünbaum and G.C. Shephard. A hierarchy of classification methods for patterns. *Zeitschrift f. Kristallographie*, 154:163–187, 1981.

[GS87] B. Grünbaum and G.C. Shephard. *Tilings and Patterns*. W.H. Freeman and Company, New York, 1987.

[Hee68] H. Heesch. *Reguläres Parkettierungsproblem*. Westdeutscher Verlag, Köln-Opladen, 1968.

[Hus89] D.H. Huson. *Patches, Stripes and Net-Like Tilings*. PhD thesis, University of Bielefeld, 1989.

[Hus93] D.H. Huson. The generation and classification of tile-k-transitive tilings of the euclidean plane, the sphere and the hyperbolic plane. *Geometriae Dedicata*, 47:269–296, 1993.

[MP94] E. Molnár and I. Prok. Classification of solid transitive simplex tilings in simply connected 3-spaces. I. The combinatorial description by figures and tables, results in spaces of constant curvature. In *Intuitive geometry (Szeged, 1991)*, pages 311–362. North-Holland, Amsterdam, 1994.

[MPS97] E. Molnár, I. Prok, and J. Szirmai. Classification of solid transitive simplex tilings in simply connected 3-spaces. II. Metric realizations of the maximal simplex tilings. *Period. Math. Hungar.*, 35(1-2):47–94, 1997.

[Pyt00] Python language website. `http://www.python.org`, 2000.

[Sed88] R. Sedgewick. *Algorithms*. Addison-Wesley, Reading Mass., 1988.

[Vin93] E.B. Vinberg, editor. *Geometry II: Spaces of Constant Curvature*, volume 29 of *Encyclopaedia of Mathematical Sciences*. Springer Verlag, Berlin, 1993.

Olaf Delgado-Friedrichs, Universität Bielefeld, Fakultät für Mathematik, Postfach 100131, 33501 Bielefeld, Germany

E-mail address: `delgado@mathematik.uni-bielefeld.de`