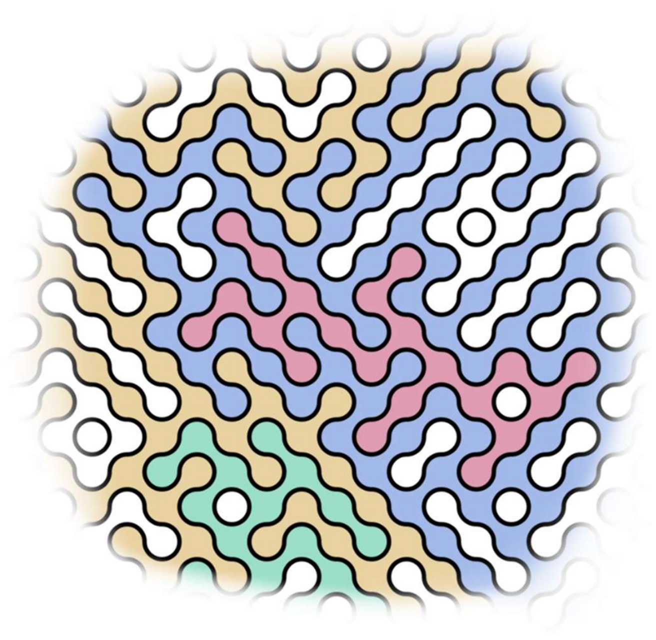
Automated reasoning – data visualization



Geometrical properties of Truchet tilings and their application in data visualization

V1.1

linkedin.com/in/parisel

Abstract

• We introduce 16 metatiles forming a permutation group of a single permutation with 6 cycles.
• We show that tiling the even lattice with this set is equivalent to tiling the plane with Truchet tiles
• We introduce saturated tilings, which exhibit interesting visual properties for representing data grouped into equivalence classes
• By automated reasoning, we prove that tiling a finite plane to produce a saturated tiling is always satisfiable
• By using the permutation group to compose 90 degree rotations, we introduce the concept of «rotor» that generalizes the process of saturated tiling to random walks
• We provide a self-avoiding random walk based on the rotor function for generating a data representation of arbitrary equivalence classes through controlled patterning (sinuosity)

This method provides a flexible and efficient way to represent arbitrary equivalence classes, offering a fresh rendering for data visualization.

Truchet Metatiles

In a 2D lattice tiled with Truchet quarter-circle tiles (or their rotations), the ball of radius √2 is a grouping of four tiles that share a common vertex at coordinates (x,y).

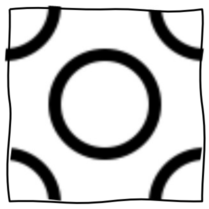
Balls can have 16 possible arrangements, depending on the orientation of each of its four constituting tiles.

These arrangements are called Metatiles. Tiling the **even** sublattice with Metatiles or the whole lattice with Truchet tiles is strictly equivalent, but we will see that Metatiles come in with more interesting features.

Here is a classification of the Metatiles:

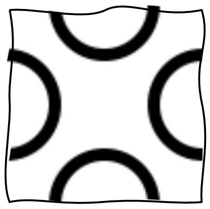
Circle (C)

Contains a single complete circle formed by four connected quarter-circles.



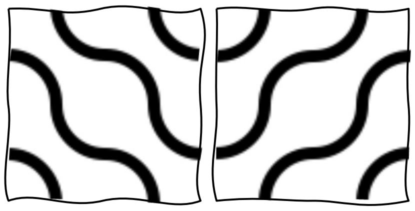
Anticircle (A)

Contains four disjoint pairs of half-circles.

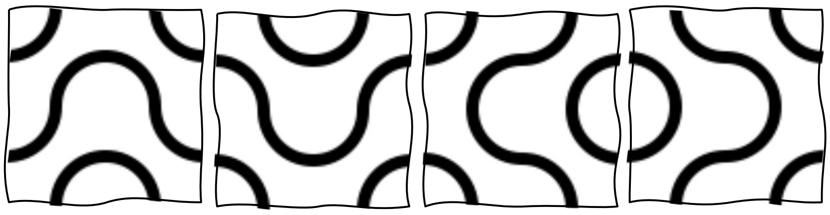


Backslash (B) and **Slash (Y)**

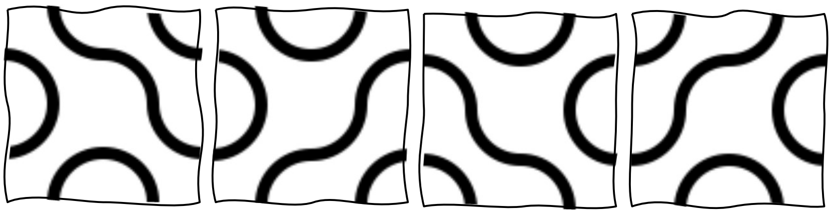
Resemble the backslash and slash characters, respectively, in their arrangement of quarter-circles.



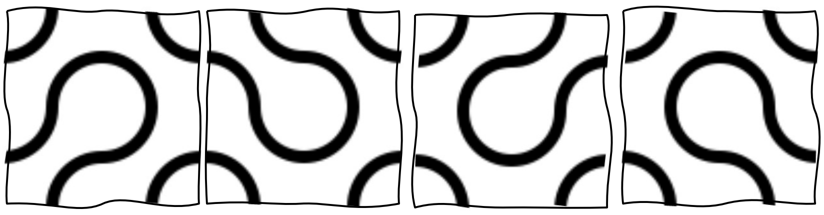
Corners (N, S, O, E): Named for their resemblance to cardinal directions, each forming a corner-like structure.



T Junctions (T, U, V, W): Resemble the letter 'T' in different orientations, representing junctions where three quarter-circles connect.



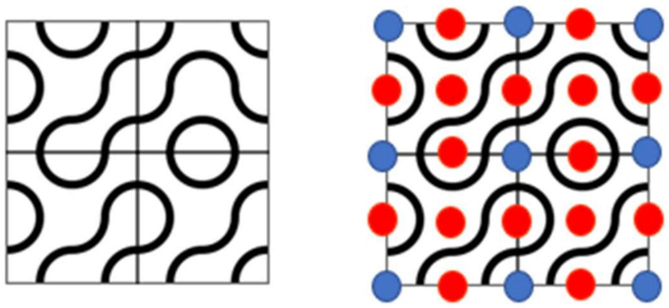
Terminators (I, J, K, L): Resemble line ends or terminations.



Plane partitioning

Note that balls on the lattice points may be either standalone (balls of the "circle" type), or connected diagonally (slash, backslash, junction, anti-circle, and corner balls) to one or more balls. So, lattice points may only connect in a **checkboard pattern**.

Left: 4 metatiles. Right: their corresponding balls on the even lattice (blue points).



With the help of Metatiles, we have just proven a well-known theorem very easily: Truchet tiles partition the plane.

More specifically, the plane is partitioned in at least two parts:

- The even lattice (blue points), where x and y coordinates are even;
- The odd lattice (red points), where at least one coordinate is odd.

Most of the time, these two parts are not continuous: they form non-overlapping **fragments** of any size, the smallest one being the circle.

Finite plane tiling

Let's first notice that the anti-circle A is the only Metatile which doesn't contain three connected quarter-circles, and that, in all other Metatiles, a "bulge" is centered in the middle of the Metatile.

Definition (Bulge): a lattice point (x,y) has a bulge iff the Metatile at (x,y) contains at least three connected quarter-circles.

Because we are aiming at a clear data visualization, we should avoid the confusion arising by the presence of anti-circles in Truchet fragments. Anti-circles should not count as class members.

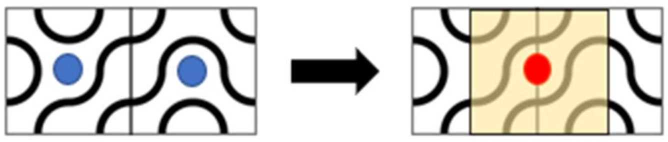
Metatiles offer a very handy way to clarify that point: rather than tiling the even sublattice with all 16 Metatiles, we restrict ourselves to the 15 ones featuring a bulge.

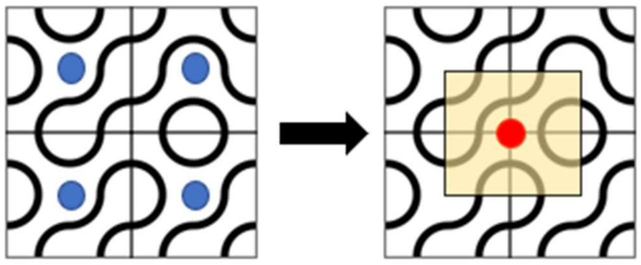
Definition (saturated tiling): a tiling is saturated if every single point in the lattice has a bulge.

In a saturated tiling, our goal will be to attempt mapping:

1. equivalence classes to Truchet fragments;
2. individual class members to individual bulges in the Truchet fragment of the class.

Unfortunately, this is not sufficient to prevent all anti-circles emergence. The reason for this is that during the process, any two adjacent Metatiles on the even lattice form another Metatile on the odd sublattice. Likewise, a square of 4 Metatiles (a "supertile") on the even lattice form another Metatile at the center of this square, on the odd sublattice.


2 adjacent Metatiles giving rise to a third Metatile (light orange) on the odd sublattice


A "supertile" (4 Metatiles) giving rise to a Metatile on the odd sublattice

We do not have direct control over the kinds of Metatiles which spring up at these odd sublattice points. So, even if anti-circles cannot emerge in the even sublattice, anti-circles may emerge in the odd sublattice during the tiling process.

Constraints imposed by the tiling process are very strong and may prevent some adjacency configurations from producing a saturated tiling. Let's prove this is not the case.

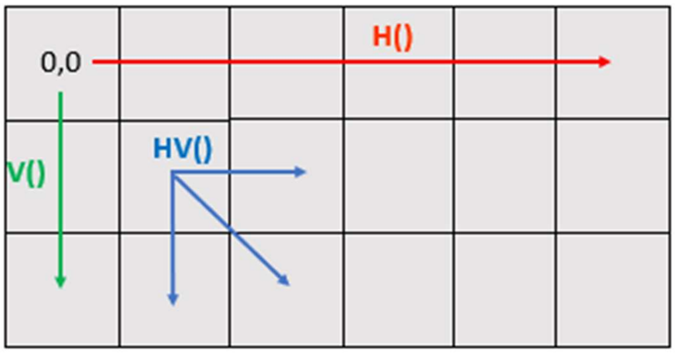
Theorem 1: all finite planes can be tiled in a saturated way.

Proof: starting at origin (0,0), we generate a saturated tiling from top to bottom and from left to right.

To handle the first row, we need a "horizontal" generator **H(m)**, which, for any Metatile m, outputs a list of eligible Metatiles to be placed to its right side. By "eligible", we mean a choice which does not give rise to an anti-circle.

When a first row is ended, to deal with the first tile on the next row, we need a "vertical" generator **V(m)**, which, for any Metatile m, outputs a list of eligible Metatiles to be placed below it.

Then, for all other situation, we need a supertile generator **HV(topLeft,topRight,bottomLeft)** which outputs a list of eligible Metatiles at location **bottomRight** given Metatiles at locations **topLeft, topRight**, and **bottomLeft**.



Sequential saturated tiling process.

To prove that any finite plane can be saturated, we must prove that, for all Metatiles topLeft, topRight, and botLeft:

H(topRight) ∩ V(botLeft) ∩ HV(topLeft,topRight,botLeft) ≠∅

This requires the analysis of 15^3=3375 configurations. We do this by automated reasoning.

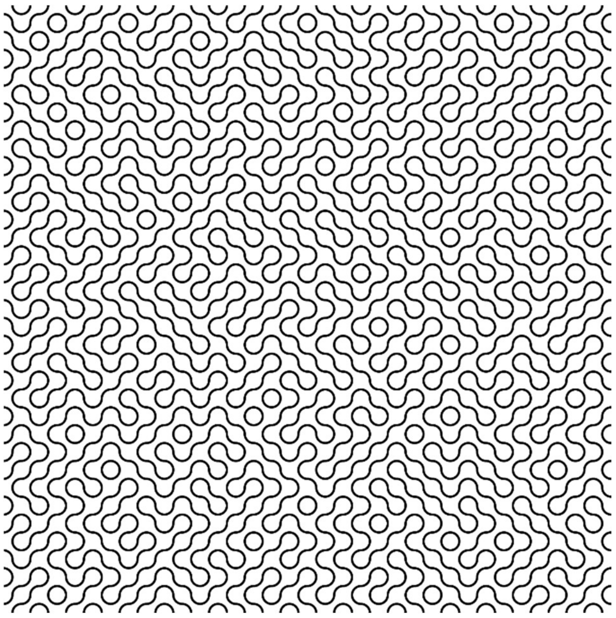
We propose a script which defines functions HV, H() and V() and enumerates HV()∩H()∩V() over its their definition domain.

Enumerating all solutions shows that the intersection always contains at lest 8 elements, proving the theorem.

The script is available at

github.com/labyrinthinesecurity/tiling-the-plane/blob/main/prover.py

Here is an example of a finite saturated tiling obtained with the tiling prover: all balls centered at even and odd lattice points have a bulge:



Tiling equivalence relations

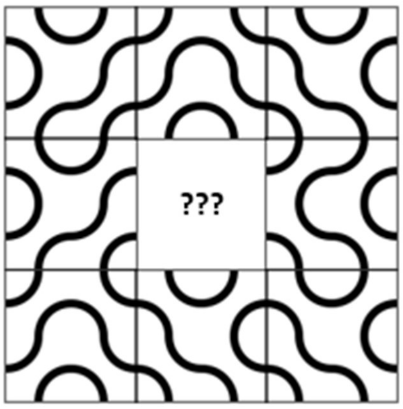
Since the union of all Truchet fragments is the lattice itself, one might ask whether the equivalence classes of any equivalence relation R could be mapped to some of these fragments?

Put it another way: *can we tile equivalence relations on the finite plane?*

The approach here is different from tiling the plane one cell at a time. We start from a blank page, and tile individual class members of each equivalence class one at a time, to fill in the finite plane.

A common practice is to resort to **random walks**, generating one walk for each equivalence class: at each step of a walk, we must determine which metatile we should place at the current even lattice location to represent a class member. The metatile must be chosen so that the walk is continuous and **self-avoiding**.

The HV() generator that we used for tiling the plane won't work here, because we do not follow a sequential process. HV() only takes 3 neighboring metatiles into account to propose a new eligible metatile. Here, we need to take into account the whole neighborhood, which is made of 8 metatiles, to propose an eligible metatile:



The configuration of 8 balls neighbors are required to produce Metatile candidates at any even lattice location.

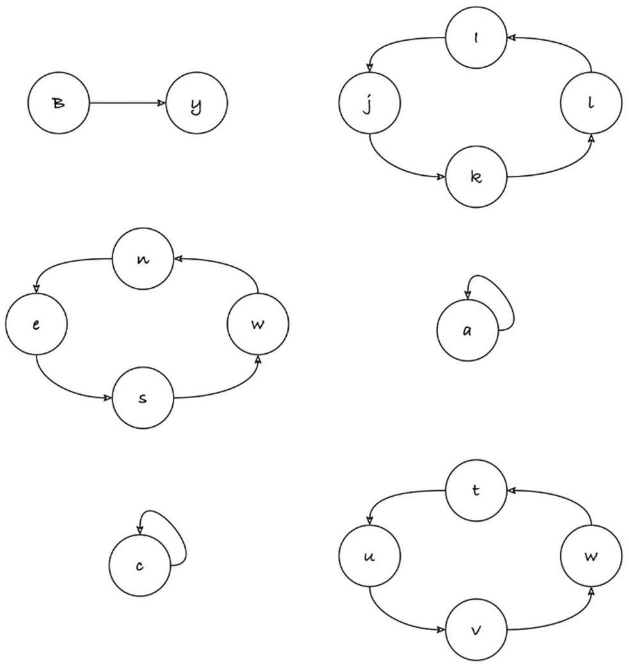
Theorem 2: the vast majority of neighborhoods configurations have a solution (i.e., they produce at least one eligible metatile).

A bruteforce approach is awkward here: there are 15^8 combinations to enumerate.

Let’s exploit the geometrical properties of our Metatiles set to find a better way. Observe that the metatiles set is a permutation group with a single permutation (if we exclude identity):

$$\sigma=(A)(C)(BY)(IJKL)(NSOE)(TUVW)$$

This permutation features 6 disjoint cycles.



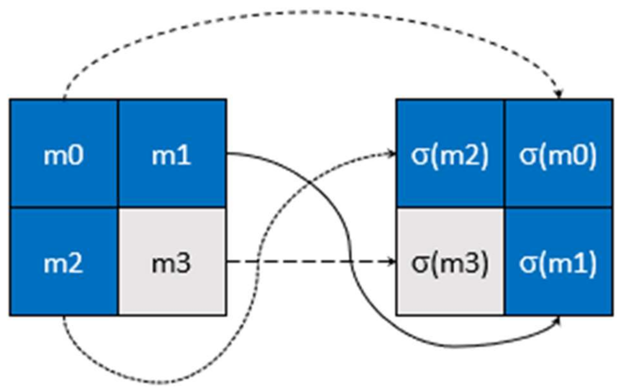
The 6 cycles of Metatiles permutation

Each permutation applies 90 degrees clockwise rotation of a Metatile.

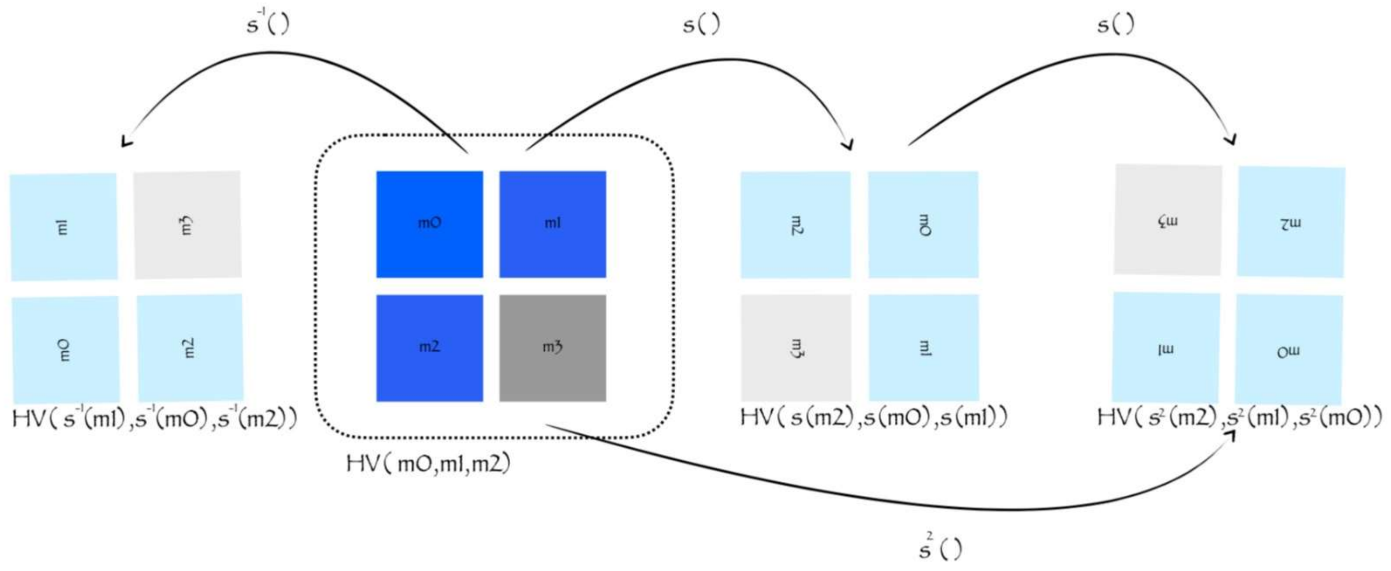
If we apply 3 different rotations to the whole supertile of HV(), we cover the 8 neighbor balls:

- HV(), the initial supertile, covers 3 neighbor balls (top and left)
- σ(HV) rotates HV by 90 degrees clockwise to cover 2 more balls (to the right)
- σ(σ(HV)) rotates HV by 180 degrees clockwise to cover 2 more balls (to the bottom right)
- σ-1(HV) rotates HV by 90 degrees anticlockwise to cover the last ball (bottom left)

As an example, here is the effect of σ( ) on the initial supertile HV():



Here is the global picture, for all 3 rotations:



s^{-1}()
s()
s()
m0
m1
m2
m3
HV(s^{-1}(m1),s^{-1}(m0),s^{-1}(m2))
HV(m0,m1,m2)
HV(s(m2),s(m0),s(m1))
HV(s^2(m2),s^2(m1),s^2(m0))
s^2()

The rotor

Let’s call the function which rotates a supertile a “rotor”.

Observe that the following equalities hold as well:

- V()=H(σ-1())

- H-1()=H(σ (σ ())
- V-1()=H(σ ())

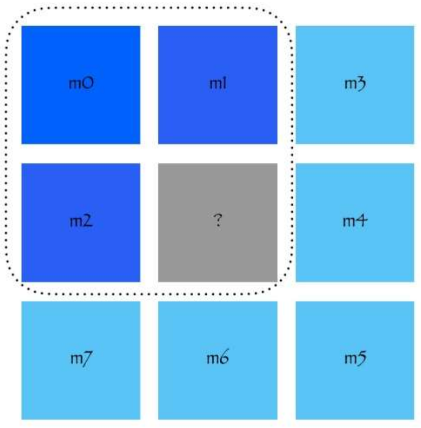
Applying the rotor at any location of the even lattice offers 2 benefits:

- We can re-use the code of HV(), and H() used for proving theorem 1
- Only HV(), H(), and their 3 rotations are sufficient to calculate eligible tiles at any location of the infinite plane. We don't need V() as it can be inferred from H().

To prove Theorem 2, we need to prove that, for all neighboring Metatiles m0 to m7:

HV(m0,m1,m2) ∩ HV(σ-1(m3), σ-1(m4), σ-1(m1)) ∩ HV(σ-2(m5), σ-2(m6),σ-2(m4)) ∩ HV(σ(m7),σ(m2),σ(m6)) ≠ ∅

Given the following layout (the rotor is depicted in its initial top left position):



We use a *Monte Carlo sampling* over all solutions to identify potential invalid configurations.

The sampling is performed with script rotor.py:

github.com/labyrinthinesecurity/metatiles-/blob/main/rotor.py

We identify **that 0.1% of configurations are not satisfiable**, proving theorem 2.

0.1% is a relief: we are confident that the tiling process is going to fail quite rarely. Now let's see how we can exploit this to generate self-avoiding random walks.

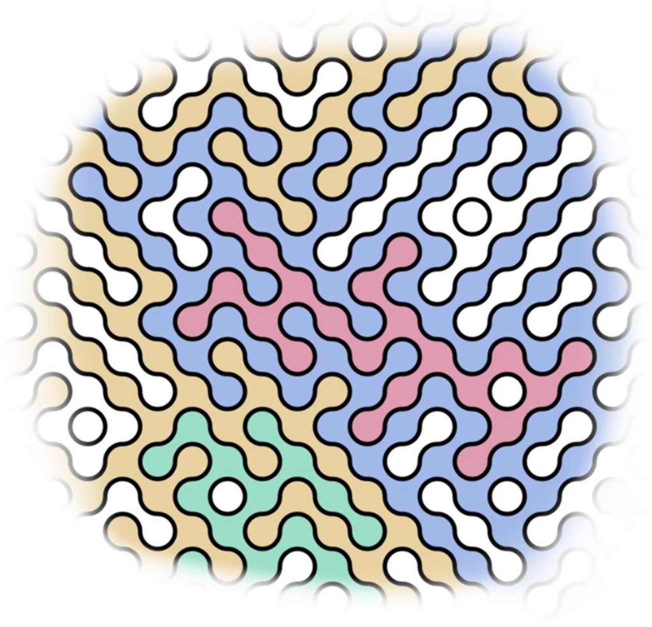
To control the tiling process, we first introduce sinuosity.

Definition (sinuosity): the sinuosity of a Truchet fragment is the ratio of the number of corner Metatiles (N, S, E, and O) in the fragment to the total number of Metatiles in the fragment, excluding terminators (I, J, K, and L).

Fragments wholly formed with corners and terminators have maximum sinuosity (1.0)

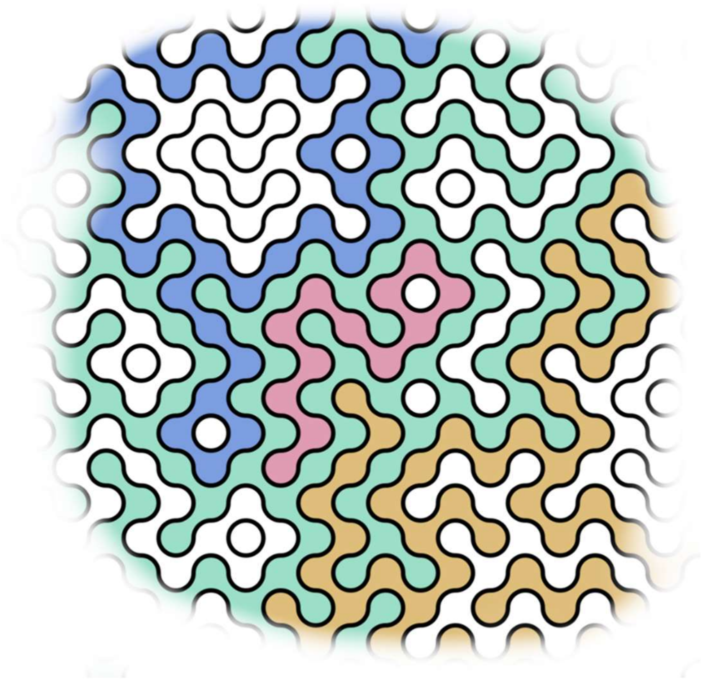
Fragments wholly formed with diagonals (slashes or backslashes), t-junctions and terminators have minimum sinuosity (0.0)

Here is an example of equivalence classes tiled with low sinuosity:



Note: corners are still spawning on the odd lattice.

Here is an example of equivalence classes tiled with high sinuosity:



Any equivalence class containing N members can be tiled on an even lattice by following a self-avoiding **random walk** derived as follows:

- We start the walk at ball (0,0);
- We call the rotor construct to get a set of eligible Metatiles at each step;
- We bias the choice of the Metatile depending on our sinuosity requirements;
- We record walk steps to backtrack if we reach a point where the walk is about to cross its own path (thus, ensuring self-avoidance);
- We assign an equivalence class member to the bulge at each step, and we decrease N by one
- When we backtrack to a fallback point, we "outgrow" the Metatile at that point, taking care of sinuosity and rotational constraints coming from the rotor;
- **Outgrowing** a Metatile means either transforming a terminator into a slash/backslash/corner, or transforming a slash/backslash/corner into a t-junction;
- The walk ends when all class members are placed (N==0) or if the backtrack cannot be satisfied due to self-avoidance constraints, or if the rotor fails (0.1% of cases).

Should we want to tile more than one equivalence class, we must add the extra constraint that walks must not accidentally cut off other existing walks. This requires keeping track of the ownership at each ball of the lattice.