Problem 1: Equilateral Polygons with Integer Vertices

Group 14

Alex Ly (c1831530), Freddy Pearce (c1835572), Alex Room (c1916814)

1 Introduction

This paper aims to solve the problem: "Given the set $L = \{(m, n) \in \mathbb{R}^2 : m, n \in \mathbb{Z}\}$, do there exist any equilateral polygons (regular or otherwise) whose vertices are all elements of L? If they do exist, explain how to construct them explicitly; if not, give a proof showing that this is impossible."

The study of geometry on our set, L (more often known in mathematics — and from here referred to in this paper — as $\mathbb{Z}^2$) is a persistent fascination. It is seen both for recreational purpose (e.g. the ancient game of Go, or number theorist Édouard Lucas' Dots and Boxes [1]) and for practical use (vector space on $\mathbb{Z}^2$ is of great interest to cryptographers [2], and 'lattice models' in physics grant ease of calculation and understanding). To our knowledge, there is little existing research into equilateral polygons on $\mathbb{Z}^2$ in general. Existing similar research focuses on regular polygons, which is uninteresting in $\mathbb{Z}^2$ (as the only regular polygon possible is a square) but far more fascinating in $\mathbb{Z}^3$ or higher $\mathbb{Z}^n$. [3]

Here we only consider non-self-intersecting (simple) polygons, i.e. ones where the lines that compose the polygon do not cross each other. This reduces the scope of the question, but we do not think much is lost from their omission. Indeed, the first algorithm we will discuss (in section 3) could potentially be weakened such that it can create self-intersecting equilateral polygons, but restricting ourselves to simple polygons gives more interesting, beautiful, and cohesive results.

It is also worth denoting some terminology we will use throughout the paper. For general polygons where the 'common' names are less common, we refer to it as an n-gon, where n denotes the number of sides. For example, we will use 'hexagon' and 'octagon' to refer to 6-gons and 8-gons, but we found it far more intuitive to say '24-gon' rather than 'icosikaitetragon'. Within examples of the algorithms themselves, we will often just use 'n-gon' instead of 'equilateral n-gon with integer vertices'. We are sure the reader will be able to infer this from context where necessary. Furthermore, composition of functions will be done left-to-right. While less popular than right-to-left convention, this suits our intuition of 'drawing lines' as a chronological task.

Throughout this paper you will see that we do not talk very much about polygons as whole entities. Rather, the methods we elaborate on rely on an idea of geometric 'paths' — that we can 'walk' between points on the coordinate grid, drawing a line behind us as we go, and if we eventually return to the start, we can step back to find we've drawn a polygon on the ground. Our initial ideas for disproof and construction were about a polygon's angle sum; while it was mostly fruitless, it gave us the knowledge to look for things which are invariant between polygons, deeper inherent properties like their nature as paths. The following section will give a general, loose idea on how we can use this 'walking' concept to elucidate the limitations on polygons we can make, then the sections after it will give a more algorithmic walking method that will allow us to construct polygons explicitly.

2 'Walking' polygons and its limitations

Our first thoughts into the question were about disproof. Can we discount certain classes of polygons as impossible to construct in $\mathbb{Z}^2$, letting us narrow down our ideas for construction? We noted early on when trying to create different polygons — searching for patterns — that it was easy to make polygons by putting triangles together. If we draw a polygon such that each side is the hypotenuse of a fixed-size right-angle triangle, length x and height y, the polygon is equilateral, with side length $\sqrt{x^2 + y^2}$. We define these hypotenuse 'paths' by way of functions:

Definition 1. *(Walking function) Let w denote a 'walking' function on the integer coordinate grid, such that:*

$$w : \mathbb{Z}^2 \to \mathbb{Z}^2$$
$$w_{a,b}(x, y) = (x + a, y + b), a, b \in \mathbb{Z}$$

We see that if we take a point (using (0,0) for a start point) and apply various $w_{a,b}$ functions to it, drawing lines for each function, that this can be used to draw a path of straight lines via applying a sequence $(w_{a_n b_n})_{n\in\mathbb{N}}$. For brevity, we will represent $w_{a,b}$ as the vector (a, b); note that adding (a, b) to a vector is the same as applying the function $w_{a,b}$, and we will also denote the set of (a_i, b_i) in a polygon as S.

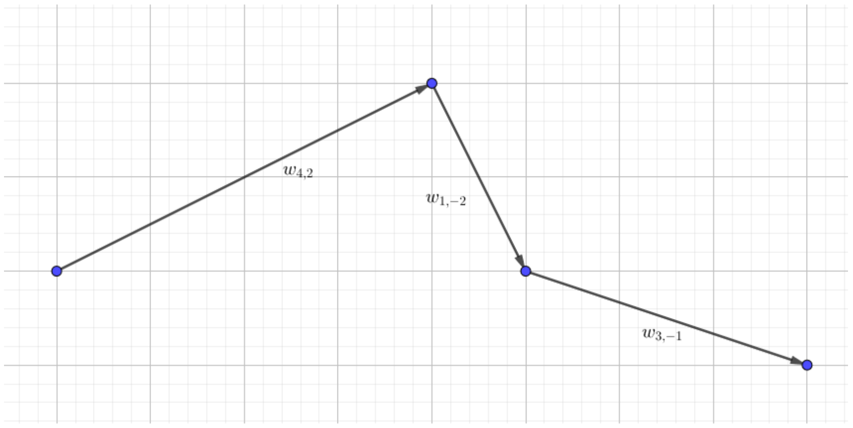

Figure 1: A walk between two points, overall the composition $w_{4,2} \circ w_{1,-2} \circ w_{3,-1}$; the associated walking functions are written by each line.

We can also reformulate the idea of an 'equilateral polygon' in this language:

Definition 2. *(Equilaterality and polygons) An equilateral walk is a sequence $(w_{a_n b_n})_{n\in\mathbb{N}}$ such that the length of each walk function, $|(a, b)| = \sqrt{a^2 + b^2}$, is equal (e.g. if we apply $w_{a,b}$, the next function applied could be $w_{-a,-b}, w_{a,-b}, w_{b,a}$, etc.), and no adjacent walk functions are parallel (scalar multiples of each other). A polygon is a sequence $(w_{a_n b_n})_{n\in\mathbb{N}}$ of at least three different walk functions such that the composition over the sequence is equal to $w_{0,0}$, the identity function on $\mathbb{Z}^2$. An equilateral polygon is an equilateral walk that is also a polygon.*

Note that the way we have defined walking functions and equilateral polygons means that an equilateral polygon created by this system has all its vertices in $\mathbb{Z}^2$ by construction. We refer to a polygon as an n-gon if it is a sequence of n walk functions.

Before we prove a theorem on certain n-gons, we create a 'minimisation lemma' — a way of reducing equilateral polygons in a way that preserves all relevant properties (equilaterality, points on $\mathbb{Z}^2$), while simplifying the kind of numbers we are dealing with.

Lemma 1. *(Minimisation lemma) Any set of points S on an equilateral polygon with can be reduced to the form S', where $a_i = 2^m a'_i$ and likewise for b'_i, for some m such that every point in S has a_i, b_i both odd, or one of a_i, b_i odd.*

Proof. Note first that an odd number squared is odd, and likewise for even numbers. Thus, we need only look at the parity of a and b. Furthermore, in cases where multiple different a and b give the same length (e.g. $|(16, 63)| = |(33, 56)| = 65$), each a and b will pairwise have the same parity. We then look at $|(a, b)|^2$; for every line in S, either $|(a, b)|^2$ is odd, in which case it is impossible for both a and b to be even. Otherwise, if $|(a, b)|^2$ is even, then either one of a or b is odd for this quantity to be even, or both of them are even, in which case we reduce them by 2^m so they are then odd. □

This lemma means that every polygon can be minimised to forcibly include odd numbers, which is essential to our theorem. Note as this is just a scaling, if we prove a theorem for all minimised equilateral polygons, we also prove it for all equilateral polygons overall. We now state our theorem:

Theorem 1. *If an equilateral polygon has all its vertices on $\mathbb{Z}^2$, then it has an even number of sides.*

Proof. Assume for contradiction we can construct an equilateral, minimised n-gon with n odd and all its points in $\mathbb{Z}^2$. By our definition of a polygon, $\sum_{i=1}^{n}(a_i, b_i) = (0, 0)$. Thus, $\sum_{i=1}^{n} a_i = 0$ and likewise for $\sum_{i=1}^{n} b_i$. Furthermore, because the polygon

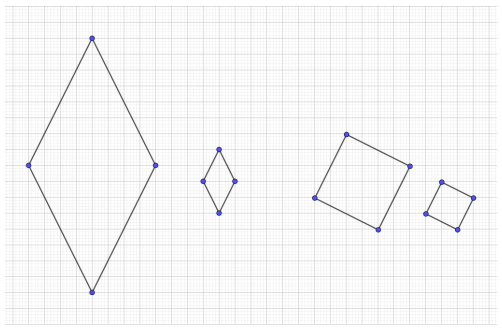
Figure 2: Two quadrilaterals and their reduced equivalents. The left is reduced by 2^2, the right by 2^1.

is equilateral, $|(a,b)|^2 = a_i^2 + b_i^2$ is equal for each a_i, b_i. If we take

$$(\sum_{i=1}^{n} a_i)^2 + (\sum_{i=1}^{n} b_i)^2 = (a_1 + a_2 + ... + a_n)^2 + (b_1 + b_2 + ... + b_n)^2 = 0$$

we get each combination of a_i's and the same for b_i's — for example, for n = 3, we get

$$a_1^2 + b_1^2 + a_2^2 + b_2^2 + a_3^2 + b_3^2 + 2a_1a_2 + 2b_1b_2 + 2a_1a_3 + 2b_1b_3 + ... = 0$$

More generally, we get

$$\sum_{i=1}^{n}(a_i^2 + b_i^2) + \sum_{\substack{i,j=1 \\ i\neq j}}^{n}(2a_ia_j + 2b_ib_j) = 0$$

$$n|(a,b)|^2 + 2\sum_{\substack{i,j=1 \\ i\neq j}}^{n}(a_ia_j + b_ib_j) = 0 \tag{1}$$

Because n is odd and 2 is even, for this expression to equal zero, $|(a,b)|^2$ must be even. It is easy to prove[1] that if $|(a,b)|^2$ is a multiple of 4, then both a_n and b_n are even — by our minimisation lemma, this is not possible. Thus, $|(a,b)|^2$ must be even while not being a multiple of 4 — if one of a_i and b_i were odd and the other even, this is not possible (as the result would be odd; again, easy to check, see footnote), so a_i and b_i must both be odd for all i. But this means each term in $\sum_{\substack{i,j=1 \\ i\neq j}}^{n}(a_ia_j + b_ib_j)$ is two odd numbers added together, which makes an even number — thus $2\sum_{\substack{i,j=1 \\ i\neq j}}^{n}(a_ia_j + b_ib_j)$ is a multiple of 4, forcing $|(a_i, b_i)|^2$ in equation (1) to be a multiple of 4 for the equality to be true, which creates a contradiction. Thus n cannot be an odd number, so any equilateral polygon with vertices on $\mathbb{Z}^2$ must have an even number of sides. $\square$

3 Construction using the knight method

When we first created our notion of 'walking', we remarked that this reminded us of chess, specifically the movement of knight pieces. By trying to create shapes by moving in L-shapes like a knight piece, we found we could create many different equilateral polygons where construction forces all their points to be in $\mathbb{Z}^2$. This was then formalised through group theory:

Definition 3. *(Knight group) Let the set of all walk functions be W. Take the following specific 'walk' functions $u, d, r, l : \mathbb{Z}^2 \to \mathbb{Z}^2$, where:*

$$\begin{aligned} u(x,y) &= (x, y+1) \qquad [= w_{0,1}] \\ d(x,y) &= (x, y-1) \qquad [= w_{0,-1}] \\ r(x,y) &= (x+1, y) \qquad [= w_{1,0}] \\ l(x,y) &= (x-1, y) \qquad [= w_{-1,0}] \end{aligned}$$

[1]The proof is done by taking a and b as $2k$, or $2k+1$ to check.

which correspond to movement up, down, right and left respectively. Now let $U, D, R, L = (u \circ u), (d \circ d), (r \circ r), (l \circ l)$, i.e. a movement of two in each direction, and let $f_g(x)$ denote $(g \circ f)(x)$. Then, for the group $(W, \circ)$ and subset $K \subset W$, we construct the knight group[2] *$\langle K \rangle$, the group generated by*

$$K = \{e, U_l, U_r, D_l, D_r, L_u, L_d, R_u, R_d\}.$$

Proposition 1. *$\langle K \rangle$ is an Abelian group.*

Proof. The inverse of any element can be found by swapping u to d and l to r wherever it occurs, for example the inverse of U_l is D_r, or the inverse of the composition $R_u U_l$ is $L_d D_r$. The identity element, e, is trivially the identity walk function $w_{0,0}$. Furthermore, as each function is simply addition in $\mathbb{Z}^2$, $\langle K \rangle$ is Abelian (although for the purpose of constructing polygons, the order in which we make moves does matter — while the functions themselves are commutative, the lines we are drawing are not). $\square$

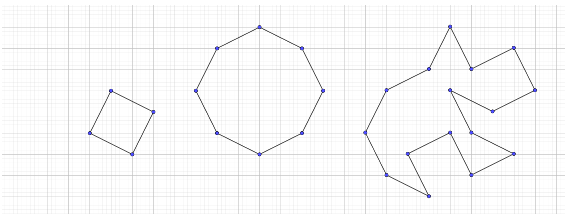

Figure 3: A square, octagon, and dog-shaped 16-gon, made using elements of the set K.

This group creates a method of moving about a coordinate grid in L-shaped paths; we can, of course, draw a line between the start and end point of each movement, and use this to construct polygons. We thus propose the following, which is not *yet* a rigorous algorithm, but rather a looser general idea of how we can use the knight group to construct equilateral polygons:

Proposition 2. *(Polygon construction using the knight group.) We can use the group $\langle K \rangle$ to construct an equilateral even-sided polygon, where all points are elements of $\mathbb{Z}^2$, using the following pseudo-algorithm:*

1. *Start with one element of K, other than e, on any point in the grid. Draw a line between the start and end points.*
2. *Compose it to the right with any element except for e, itself, or its inverse. For example, for U_l, you can then apply any element except e, U_l or D_r; it must also not intersect any existing lines. Draw a line between the start and end point of this operation.*
3. *Repeat step 2 until you reach your original point; i.e. until the composition of functions is equal to e.*

Remark. *Before proving this and giving an explicit method for a specific n-gon, it is worth noting that this theorem justifies the existence of U, D, L and R, which one may be unsure are necessary; that is, why can't we use the group $\langle \tilde{K} \rangle$, where $\tilde{K} = \{e, u_l, u_r, d_l, d_r, ...\}$? This set degenerates to $\tilde{K} = \{e, u_l, u_r, d_l, d_r\}$ as $u_l = l_u$, whereas $U_l \neq L_u$. This means we have less freedom in step 2 and thus the variety of polygons we can make is reduced — for example, $\langle \tilde{K} \rangle$ cannot be used to create an hexagon or octagon, whereas K can. An interesting note on $\langle \tilde{K} \rangle$, though, is that using it for the same algorithm creates equilateral polyominoes, elaborated on in section 4.*

Proof. If we start at any integer point (such as $(0, 0)$) then any line drawn using functions from K will produce a line to another integer point by construction, with fixed length of $\sqrt{5}$ by Pythagoras' theorem. By the restriction placed in step 2, it will always be equilateral; the only way a side can be longer than the other sides is if the same move is done twice in a row, which has been disallowed[3]. Avoiding intersection is also not strictly necessary, and can sometimes create multiple smaller polygons using one path, but can also lead to creating non-equilaterals if, say, a R_d line crosses through half of a U_r line. Finally, continuing until the composition is equal to e ensures it is a closed polygon; the overall displacement of our 'knight' is zero, so the path is continuous and starts and ends at the same point. $\square$

[2]named for its inspiration and similarity in movement, the knight chess piece.

[3]avoiding use of the inverse or e, on the other hand, is not necessary to make the theorem work, but has the added benefit of giving each n-gon a unique representation of n functions.

3.1 Creating larger n-gons from smaller ones

The next observation, and step towards an algorithm, was found soon after our discovery of using the knight group to create polygons. We found that we could stick two octagons together, intersecting at two points, to create a 14-sided equilateral polygon. We can similarly 'grow' any polygon to create a larger one.

For an n-gon where the first and second-last elements are not equal or inverse to each other[4], we can create a $2(n-1)$-gon by removing the last element, and then composing to the right by the inverse of each remaining element in order. For example, from the hexagon created from the movements

$$U_r R_u R_d D_l L_d L_u,$$

we can create the decagon

$$U_r R_u R_d D_l L_d D_l L_d L_u U_r R_u.$$

We specify the inverse rule because otherwise we would either have the same element twice, which would make the polygon non-equilateral; or an element followed by its inverse, which would degenerate the shape. However, you can cycle the elements (see example) until it fits the criteria. For example, our decagon we've created doesn't work (as the first and 9th element are both U_r), but:

$$U_r R_u R_d D_l L_d D_l L_d L_u U_r R_u = U_r R_u U_r R_u R_d D_l L_d D_l L_d L_u,$$

i.e. we moved the last two elements to the front. Then, using our method, we create the 18-gon

$$U_r R_u U_r R_u R_d D_l L_d D_l L_d D_l L_d D_l L_d L_u U_r R_u U_r R_u.$$

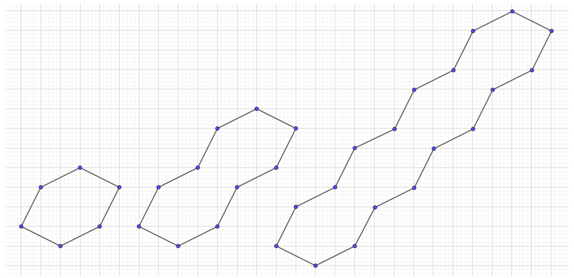

Figure 4: A hexagon, with the decagon and 18-gon 'grown' from it.

3.2 Algorithm to create a specific n-gon

While random (or guided by a human, if one wants to get faster and more reliable results) paths can create interesting and varied polygons, we need more restriction on our algorithm if we want to aim for a specific polygon size. The main issue here was self-intersection; attempting to construct a polygon by just composing $\frac{n}{2}$ elements and their inverses led to many polygons that self-intersected, giving us by no means a reliable algorithm. Naturally, self-intersection only happens if the path returns through already-created lines (as a very simple example, the pattern $U_r D_l R_d$ self-intersects) so we can avoid this by, of course, not letting it go back on itself. So, borrowing the idea from the previous subsection of using inverses to 'close' a shape, we can strengthen our algorithm:

Theorem 2. *(Specific polygon construction.) We can use the group $\langle K \rangle$ to construct an equilateral polygon with n sides, where n is even and all points are elements of $\mathbb{Z}^2$, using the following algorithm:*

1. *Start with one element of K, other than e, on any point in the grid. Draw a line between the start and end points.*
2. *Choose a direction other than either of the directions in your first element. This direction is 'forbidden' for the next step. Since our algorithm isn't affected by 90° rotation of the grid, it doesn't matter which one you choose.*
3. *Compose your first element to the right with any element except for e, itself, its inverse, or an element containing your chosen direction. Repeat this $\frac{n}{2}$ times. To ensure the polygon is n-sided, use at least one element that isn't opposite to another that you are allowed to use (e.g. if left is forbidden, use at least one of the form R_x rather than*

[4]This means $n \geq 6$, because there is no square where the first and second-last elements are not inverse to each other.

just a mix of U_x and D_x functions) — failing to do so will create multiple smaller equilateral polygons instead of one n-sided one. Furthermore, the final element cannot be the same as the first one (because then the next step will disobey our original theorem), and you must be able to draw a straight line between the first and last points such that it does not intersect the movement line (else, the inverse movement will have to pass through the existing line to get back).

4. *Compose to the right by the inverse of each of the $\frac{n}{2}$ elements in order. Naturally, by the end of this, the composition of functions should again be e.*

As an example, say we want to create a decagon (so $n = 10$). We will start with U_r, and choose left as our 'forbidden' direction. Then, we randomly compose 5 elements, to create, say, $U_rD_rR_dR_uD_r$. We then compose it with each element's inverse to get

$$U_rD_rR_dR_uD_rD_lU_lL_uL_dU_l,$$

which is an equilateral decagon with all points on $\mathbb{Z}^2$.

Remark. *As this is a restriction on Proposition 2, there is no need to re-prove that this* can *indeed create a polygon, or even a specific n-gon (see footnote 2 on the proof of Proposition 2) — only that this restriction ensures it cannot ever self-intersect. We can see the line drawn as a 'mirror', with the path back as a 'reflection'* [5] *in this line, so if the line intersects the path then this 'reflection' will do the same. We regret that we were unable to show this in a more rigorous way; all we know is that no polygon we have created with a line fitting this criteria has self-intersected. See also that this explains our previous 'growing' method; by removing the final side, we are turning the rest of the polygon into an outward path, and then building a return path based on it. This also means that, as the mirror line was originally part of the shape, the growing method is guaranteed to work.*

A Python script which performs this algorithm for given n (with random choices for each 'decision') is available at this Google Colab page.

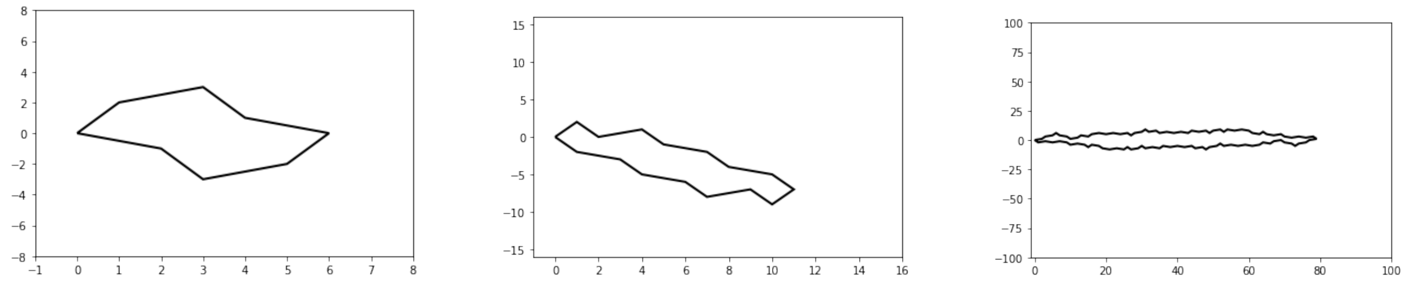

Figure 5: An octagon, 16-gon and 100-gon created using the script for this algorithm.

4 Construction using tiling

This was our very first idea for a method; it is a lot rougher (only able to produce n-gons for very specific n), but faster and guaranteed to create an equilateral n-gon first try (whereas the knight method may need you to throw out several polygons-in-progress at step 3). We put it here for comparison, as an example of a method that doesn't rely on the notion of paths that has carried through the paper so far (although, we will see that it is in fact equivalent to a variation on the knight algorithm in Proposition 3).

Definition 4. *(Polyomino) A polyomino is a polygon formed by tiling equal-sided squares together, edge to edge.*

Some polyominoes are equilateral — see Figure 6 below — and that is what we will be constructing here. Naturally, these are infinitely tilable; we can always, for example, attach an L-shaped polygon to the 12-sided cross (see image) to add 6 more sides, and then repeat this as many times as we want; thereby creating a $(12 + 6n)$-gon incredibly quickly.

Proposition 3. *Creating polygons by using the knight algorithm with the degenerate knight group $\langle\tilde{K}\rangle$ (see the remark under Proposition 2) creates equilateral polyominoes.*

[5] *Not* in the geometric sense — only within this analogy.

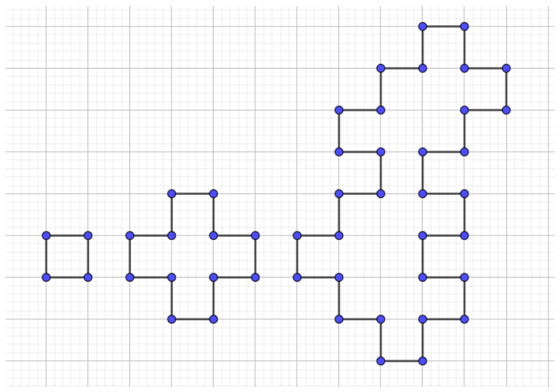
Figure 6: 4, 12, and 32-sided equilateral polyominoes.

Proof. If a polyomino is scaled up by $\sqrt{2}$, rotated 45°, and translated, all of its points will be on integer vertices, as each line will be $\sqrt{2}$ long, and the hypotenuse of a triangle at a 45° angle, and $\sqrt{2}\cos 45 = \sqrt{2}\sin 45 = 1$, so the triangle is a movement of 1 along the x-axis and 1 along the y-axis. These movements are exactly those described by the functions in $\tilde{K}$, so we can create the polyomino using a composition of these functions. □

Theorem 3. *(Polyomino construction algorithm) We can tile polyominoes to create an equilateral (12 + 8n)-gon, (12 + 4m)-gon, or (12 + 8(n-1) + 6(m+1))-gon with integer vertices like so:*

1. *Write the number of sides in the form* $12 + 8n$, $12 + 4m$, *or* $12 + 8(n-1) + 6(m+1)$, *for* $n, m \in \mathbb{N}$
2. *Draw a 12-sided cross, like on Figure 6.*
3. *Attach n ⊥-shapes (see Figure 7, left-hand polygon) on top of one of the crosses.*
4. *If* $m > 0$, *attach an L-shape (see Figure 7, middle polygon) in a concave corner of the top ⊥ (or the cross if* $n = 0$*), then* $m - 1$ *more L-shapes in the concave corner of the previous L-shape.*

Proof. Clear by construction — the tileability of polyominoes means we only need to consider what is happening locally for each addition, not globally for the whole polygon (i.e. a 4th new ⊥ at the top means nothing to the cross at the bottom). The cross, ⊥, and L were chosen as they are 'stable' — that is, adding more of them in the way described will never make the shape stop being equilateral. □

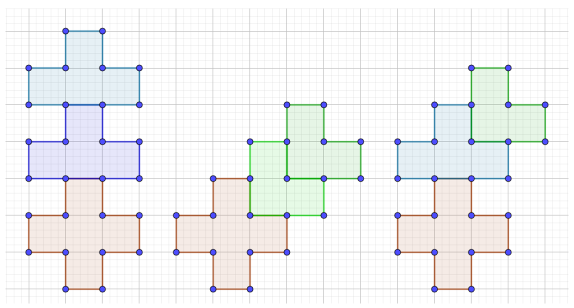
Figure 7: A 28-gon (12 + 8(2)) composed of a cross and two ⊥'s, a 20-gon (12 + 4(2)) composed of a cross and two L's, and another 24-gon (12 + 8(1-1) + 6(1 + 1)) composed of a cross, a ⊥ and an L, showing all three cases of the polyomino construction algorithm.

We can, similarly but more simply, tile convex n-gons along parallel edges to create $(2(n-1))$-gons; this method is equivalent to the polygon-growing method given in subsection 3.1. However, it depends on the notion of convexity of whole polygons rather than requirements on the knight group representation of the shape.

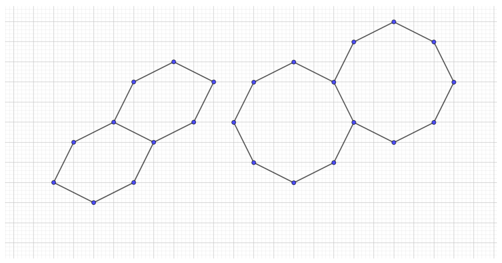
Figure 8: A hexagon and octagon tiled to create a decagon and 14-gon.

This method has benefits and drawbacks compared to the knight algorithm. On one hand, it is much more efficient; if we have some kind of 'copy and paste' functionality, we can make very large polygons very quickly. On the other hand, unlike the knight algorithm this method is not 'complete' — a machine with no knowledge of what a polygon even *is* can create any equilateral polygon with integer vertices with ease using the knight algorithm, but would need some basic shapes given to it (i.e. the 'cross', '⊥', and 'L' shapes) before it could even start computing polygons, and even then it would be unable to create some polygons (like a 14-gon) with just this method.

In essence, these two solutions represent two different problem solving strategies. The knight algorithm gives as full as possible an answer to the original problem — the closest to what we'd deem a 'solution' — but if somebody woke us up in the middle of the night and told us to make twenty equilateral polygons with integer vertices, the polyomino method would have us back in bed a lot more quickly.

Where next?

Further extensions on this report include:

- Generalisation to higher $\mathbb{Z}^n$. This would require the paper to be restructured, as our disproof no longer holds; not only does the minimisation lemma no longer hold in higher dimensions, neither does odd polygons being impossible (for example, the set of points $\{(1,0,0),(0,1,0),(0,0,1)\}$ creates an equilateral triangle in $\mathbb{Z}^3$).
- Linking to other theorems about polygons with integer vertices, such as Pick's theorem, which allows you to find the area of a polygon using the number of integer points on its perimeter and the number of the integer points inside the polygon. Any line using the knight algorithm adds exactly two integer points to the perimeter; can we thus find the area of any equilateral polygon purely from its knight-group representation?
- Linking to vector calculus; the paths we have created using the knight algorithm define a contour. Is it possible to use results from vector calculus or complex analysis to learn more about our paths?
- Looking more rigorously into the knight algorithm; for example, a more rigorous proof into why drawing a line through the shape as a check ensures the polygon works. Additionally, if there are properties inherent to polygons made this way, aside from fitting the criteria of the problem. If you look at the Python script on Google Colab, you will see that for large n, it creates long, skinny polygons. Is this a feature of our algorithm?

References

[1] Elwyn R Berlekamp, John H Conway, and Richard K Guy. *Winning Ways for Your Mathematical Plays*, volume 3, chapter 16. CRC Press, 2018.

[2] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 530–547. Springer, 2012.

[3] Hiroshi Maehara and Horst Martini. Elementary geometry on the integer lattice. *Aequationes mathematicae*, 92(4): 763–800, 2018.