Obtuse Triangular Billiards II: One Hundred Degrees Worth of Periodic Trajectories

Richard Evan Schwartz

CONTENTS

We give a rigorous computer-assisted proof that a triangle has a periodic billiard path when all its angles are at most one hundred degrees.

1. INTRODUCTION

1.1 Background

The theory of billiards on rational polygons, i.e., polygons whose angles are all rational multiples of π, is a well-studied subject with deep connections to areas such as Teichmüller theory. See [Gutkin 96, Masur and Tabachnikov 02, Tabachnikov 95] for some surveys on billiards, mainly rational. Very little is known about the existence of periodic orbits in irrational polygonal billiards. Here is a basic conjecture.

Conjecture 1.1. (Triangular billiards conjecture.) *Every triangle has a periodic billiard path.*

By *triangle* we mean a solid triangular region of the plane. I think that it is fair to say that this two-hundred-year-old problem is widely regarded as impenetrable.

In order to survey some results related to the triangular billiards conjecture, we introduce a bit of notation. Let T be a triangle, with the shortest edge labeled 1, the next-shortest edge labeled 2, and the longest edge labeled 3.

Any periodic billiard path in T gives rise to an infinite repeating word that records the succession of sides encountered by the billiard path. This periodic word is called the *combinatorial type* of the path.

In 1775, Fagnano proved that the combinatorial type 123 (repeating) describes a periodic billiard path on every acute triangle.

It is an exercise to show that 312321 (repeating) describes a periodic billiard path on all right triangles. See

2000 AMS Subject Classification: Primary 37E15

Keywords: Obtuse, triangles, periodic, billiards, orbits, dynamics

[Galperin et al. 91, Hooper 07, Troubetzkoy 04] for some deeper results on right-angled billiards. Any given rational polygon has a dense set of periodic billiard paths [Boshernitzyn et al. 98]; see also [Masur 86]. See [Veech 92] or the surveys mentioned above for the connections to Teichmüller theory. The papers [Galperin et al. 91, Halbeisen and Hungerbuhler 00] produce some infinite families of combinatorial types that describe periodic billiard paths on some obtuse triangles.

A periodic billiard path on a triangle is called *stable* if a periodic billiard path of the same combinatorial type exists on all nearby triangles. In Section 2 we explain that stability is a combinatorial property of the word. In [Hooper 07] it is shown that no right triangle has a stable periodic billiard path.

In [Schwartz 06], I prove that any triangle sufficiently close to the 30-60-90 triangle has a periodic billiard path. At the same time, I prove the following "pessimistic" result: For any $\epsilon > 0$ there is a triangle within ϵ of the 30-60-90 triangle that has no periodic billiard path of combinatorial length less than $1/\epsilon$.

In [Hooper and Schwartz 08], Pat Hooper and I prove that any sufficiently small perturbation of an isosceles triangle has a periodic billiard path. This result is deceptively hard: We require several infinite families of combinatorial types.

The purpose of this paper is to prove the following result.

Theorem 1.2. (100 degree theorem.) *Let T be an obtuse triangle whose largest angle is at most* 100 *degrees. Then T has a stable periodic billiard path.*

I discovered this result operating McBilliards, a graphical user interface that Pat Hooper and I wrote for the purpose of studying the triangular billiards conjecture. (McBilliards also inspired [Hooper 07, Schwartz 06, Hooper and Schwartz 08].)

We will prove the 100 degree theorem rigorously, using a combination of traditional mathematics and exact integer computation. We recommend that the reader operate McBilliards while reading the paper.

McBilliards really brings the ideas in the paper to life, and also allows the reader to survey the computer parts of our proof to a very fine level of detail. In Section 7 we give instructions for accessing and operating McBilliards.

For the reader who doesn't want to learn McBilliards but who still would like a visual guide to the paper, we have written a simple-to-use stand-alone Java applet that illustrates our proof. See Section 7 for details.

For the reader who does not want to use either of our programs, I have tried to make the mathematics stand on its own. Also, I have tried to explain the methods sufficiently well that the interested reader could start from scratch and reproduce the result in its entirety.

As I explain in the next section, my method of finding the needed periodic billiard paths is rather simple-minded. Someone reproducing the experiment would probably not find exactly the same list of combinatorial types I use, but I would bet that the hypothetical new list would have considerable overlap with my list. I think that my list is rather efficient.

My method of verifying that these combinatorial types do the job is rather complicated and idiosyncratic. Some of the complexity in the verification process is probably necessary, but some of it is a result of my needing to get a feasible computation. (I worked quite hard to develop an efficient method.) On much faster computers, the verification algorithms would be simpler. See Section 4.2, for instance. Perhaps the main point of my verification process is to convince the reader that the result *can* be proved by a finite computation.

One might wonder whether 100 degrees is a natural cutoff for our result. It is not. We stopped at 100 degrees because it is a nice round number. With a great deal more effort, we could perhaps get to 105 or 110 degrees. Below we will explain why 112.5 degrees ($= 5\pi/8$ radians) is a very hard barrier to pass.

To use an analogy, our approach to the triangular billiards conjecture is a bit like trying to ride a bicycle to the North Pole. It is fairly clear that the approach will come to grief, but it is hard to say in advance exactly where or how.

Results like the "pessimistic result" in [Schwartz 06] mentioned above, and also the deeper complications revealed in [Hooper and Schwartz 08], indicate some of the difficulties.

1.2 Discussion of the Experiment and the Proof

Let Δ denote the parameter space of obtuse triangles. The point $(x, y) \in \Delta$ represents a triangle with small angles of x and y radians. To each combinatorial type W, we can associate the region $O(W)$ consisting of points $(x, y) \in \Delta$ such that W is the combinatorial type of a periodic billiard path on the triangle corresponding to (x, y).

When $O(W)$ is nonempty, we call $O(W)$ an *orbit tile*, or *tile* for short. The periodic billiard path represented by W is stable if and only if $O(W)$ is a nonempty open set. When $O(W)$ is not an open set, $O(W)$ is contained in a line. See Section 2.2 for details.

Let $S_{100} \subset \Delta$ denote the region corresponding to triangles whose largest angle is at most 100 degrees. For convenience we also assume $x \leq y$ in S_{100}. Our general method of proof is to cover S_{100} with orbit tiles. We use stable words because the orbit tiles are much larger. Two problems emerge.

Problem 1.3. Consider a boundary point $(0, y) \in \partial\Delta$, with $y \in (0, \pi/2)$. A triangle very near such a point has no short periodic billiard path. See Lemma 3.1 for a proof. Thus, the covering we seek is necessarily infinite.

Problem 1.4. Our "pessimistic result" from [Schwartz 06] can be restated like this: No neighborhood of the point $p_6 := (\pi/6, \pi/3)$ has a finite covering by orbit tiles. This point corresponds to the 30-60-90 triangle.

While there is no hope of finding a finite cover of S_{100}, because of these two problems (and other problems), we nonetheless used McBilliards to find an infinite cover. McBilliards essentially does two things:

1. Given $(x_0, y_0) \in \Delta$ and some N, McBilliards finds all stable combinatorial types W of length at most N such that $(x_0, y_0) \in O(W)$. The program makes a depth-first search through the tree of words, pruning any branch of the tree as soon as it is clear that no completion of the corresponding word prefix can result in a periodic billiard path. Setting $N = 50$ gives a quick answer, and setting $N = 1000$ takes all day.
2. Given a stable combinatorial type W, McBilliards computes, to specified precision, the orbit tile $O(W)$. As we will discuss in this paper, $O(W)$ is a "finite-sided" region, bounded by analytic arcs. We think that $O(W)$ is always connected and simply connected, but we don't know how to prove it.

Our experimental method works like this. We initially set (say) $N = 50$ and sample many points in Δ. We first search for all the stable combinatorial types of length at most N corresponding to some given point. Assuming that we find some words, we then plot the corresponding tiles. Now we repeat.

Roughly speaking, at any stage of the process, we choose a point that is near the center of the largest region we have not covered by orbit tiles. When it seems that our searches for $N = 50$ are no longer meeting with any success, we increase N to 100. And so on. One could perhaps automate this process, but we have not done so.

Sometimes, guided by a hunch, we focus on a small "hole" around a particular point. (Other users are likely to develop similar hunches.) In this case, we steadily increase N while zooming in on the region of interest. Sometimes we find a finite cover; sometimes we find the initial portion of an infinite sequence of tiles whose union seems to cover the hole; and sometimes we have to give up without being able to draw any conclusion.

Let

$$p_k = \left(\frac{\pi}{k}, \frac{\pi}{2} - \frac{\pi}{k}\right) \in \partial\Delta; \qquad (1\text{–}1)$$

p_k corresponds to a right triangle. Our search reveals five features:

1. Solving Problem 1.3, we found an infinite union of tiles that covers a neighborhood P_3 of the segment $\{0\} \times [5\pi/9, \pi/2]$. Here $5\pi/9$ radians is 100 degrees.
2. The point p_4 presents a minor inconvenience. It seems that no neighborhood of this point in Δ is contained in an orbit tile. However, we cover a neighborhood P_4 of this point by a union of nine orbit tiles. (Restricting to the set $\{x \leq y\}$ as we do for S_{100}, we need only five orbit tiles.)
3. The point p_5 presents a similar inconvenience. We cover a neighborhood P_5 of this point by a union of two orbit tiles.
4. Solving Problem 1.4, we found an infinite family of orbit tiles whose union covers a neighborhood P_6 of p_6. We establish this result in [Schwartz 06].
5. Any point in $S_{100} - (P_3 \cup P_4 \cup P_5 \cup P_6)$ is contained in one of 215 orbit tiles $O(W_7), \ldots, O(W_{221})$. The maximum word length is 184.

Once we have established the listed results, we have just to show that

$$S_{100} \subset \bigcup_{i=3}^{221} P_i. \qquad (1\text{–}2)$$

We introduce "dummy" polygons P_1 and P_2 that cover $\Delta - S_{100}$. Thus, (1–2) is equivalent to

$$\Delta \subset \bigcup_{i=1}^{221} P_i. \qquad (1\text{–}3)$$

Establishing this result is a purely combinatorial result. We have a finite number of polygons and we want to see that they form a covering of Δ.

Referring to item 1 above, P_3 is a certain triangular region in the parameter space. The infinite family of words corresponding to the tiles covering P_3 grows in a very predictable way, and we will use analytic techniques to deal with all the words at once. The method we use here for P_3 is similar to the method we use for P_6 in [Schwartz 06].

The computer-aided portion of our proof deals with items 2, 3, and 5. To explain the general idea, we concentrate on item 5. For each word W_j we will produce a polygon $P_j \subset O(W_j)$. We choose P_j such that it has dyadic rational vertices. We also choose the polygons $P_1, \dots, P_6$ to have dyadic rational vertices. Choosing dyadic rational coordinates is useful for technical purposes, as we somewhat explain below.

In Section 4 we will explain how we check, with a finite amount of computation, that $P_j \subset O(W_j)$. Without going into too much detail in this introduction, perhaps we can explain why this really is a finite computation. Starting in $O(W_j)$ and moving toward $\partial O(W_j)$, we observe that the corresponding billiard path "disappears" in one of finitely many ways. Essentially, some portion of the (geometrically changing) path has to crash into a vertex of the (geometrically changing) triangle. We have just to show that none of these finitely many bad events occurs when we move around the smaller P_j.

For the purposes of giving a proof, it doesn't matter how we produce our polygons. However, it seems worthwhile to explain the general idea behind our choices. First, we plot $O(W_j)$ to high precision. Then, we select a dyadic rational polygon $P_j \subset O(W_j)$ roughly according to three criteria:

- The polygon P_j must be fairly well contained inside $O(W_j)$, so that a relatively small amount of computation reveals that $P_j \subset O(W_j)$. It is usually quite hard to verify that points very near the boundary of $O(W_j)$ are actually contained in $O(W_j)$.

- The polygon P_j should be a sufficiently close approximation to $O(W_j)$ that (considering pairs of polygons) P_i and P_j have about the same overlap as W_i and W_j. This condition guarantees that we retain the covering property when we replace $O(W_j)$ by P_j.

- The denominators of the vertex coordinates are not too large. The largest denominator we use is 2^{17}. Having fairly simple coordinates involved turns out to be useful for our rigorous calculations. The technical difficulty is that we want to make exact integer calculations, but we also need to evaluate trigonometric functions on the vertices of our polygons. To solve the problem, we create a lookup table of rational approximations to $\cos(\pi k/2^n)$ for the relevant pairs (k, n).

It takes some work to satisfy all these requirements. What helps us tremendously is that the orbit tiles, especially the small ones, are extremely close to being convex polygons. Thus, we can get very nice inner approximations. We produced the final polygons "by hand," using a special feature of McBilliards designed to help us select, manage, and modify such polygons. We got lucky in that the whole business worked out to a feasible computation. The near-convexity of the orbit tiles seems to be a general phenomenon, but we don't know how to prove any general result along these lines.

Now we explain why $5\pi/8$ is a hard barrier to pass. Our region P_3 actually covers a neighborhood of the larger segment $\{0\} \times (5\pi/8, \pi/2]$. However, we have no idea how to cover the neighborhood of any point $(0, y)$ with $y \geq 5\pi/8$. Fairly deep searches by McBilliards reveal interesting infinite patterns of orbit tiles that stop well short of covering any such neighborhood. It seems to me that the most sensible continuation of the approach here would be to find a cover of a neighborhood of $(0, 5\pi/8)$ by orbit tiles.

1.3 Plan of the Paper

This paper is organized as follows.

- In Section 2 we will present some basic material about triangular billiards. Some of this theory is well known, and some of it is (probably) new.

- In Section 3 we deal with P_3, proving that this polygonal region can be covered by infinitely many orbit tiles. This part of the proof is purely traditional, but of course is heavily inspired by computer experimentation.

- In Section 4 we explain our basic computational algorithm, which verifies an equation of the form

$$P \subset O(W),$$

where P is a polygon with dyadic rational vertices and W is a word. Running this algorithm, we verify that $P_j \subset O(W_j)$ for $j = 7, \dots, 221$.

- In Section 5 we deal with P_4 and P_5. For the most part, our treatment of P_4 and P_5 uses the algorithm presented in Section 4, but we need to intervene occasionally and do some hands-on analysis. For the sake of completeness, we briefly review how we covered the region P_6 in [Schwartz 06].

- In Section 6 we discuss the main computational issues in the paper. In particular, we explain how we verify (1–3). We also explain how we reduce all our calculations to integer arithmetic.
- In Section 7 we provide some operating instructions for McBilliards.

The list of all polygons and words we use seems too long to include in this paper. The complete list resides both in McBilliards and in our companion Java applet. Also, the publications list on my website[1] has a link to a written list of the words and polygons right next to the link to this paper.

2. UNFOLDINGS AND STABILITY

In this section we develop some of the basic theory of triangular billiards. Some of this material is well known, and some of it appears in [Schwartz 06, Hooper and Schwartz 08].

2.1 Unfoldings

We always work with even-length words. Our convention is that a finite word is meant to be a portion of an infinite periodic word. The infinite periodic word is obtained from the given portion just by repeating it endlessly.

Given a word $W = w_1, \dots, w_{2k}$ we define a sequence $T_1, \dots, T_{2k}$ of triangles by the rule that T_{j-1} and T_j are related by reflection across the w_jth edge of T_j. Here $j = 2, \dots, 2k$. The set $U(W,T) = \{T_j\}_{j=1}^{2k}$ is known as the *unfolding* of the pair (W,T). This is a well-known construction; see [Tabachnikov 95]. Figure 1 shows an example.

The *first edge* of $U(W,T)$ is the edge of T_1 labeled w_1. The *last edge* of $U(W,T)$ is the edge of T_{2k} labeled w_{2k}. In Figure 1, the first edge is the segment joining a_1 to b_1, and the last edge is the segment joining a_8 to b_8.

The word W represents a periodic billiard path in T if and only if the following hold:

1. The first and last edges of $U(W,T)$ are parallel.
2. There is a line segment L joining equivalent interior points on the first and last edges that remains entirely inside $U(W,T)$.

[1]http://www.math.brown.edu/~res.

By *equivalent points* we mean that the translation carrying the one edge to the other identifies the points. When we fold up the line segment, so to speak, it becomes a periodic billiard path on the original triangle. Conversely, a periodic billiard path unfolds into a segment as we have described.

The line segment L is never unique. Small parallel translations of L will also satisfy all the hypotheses. We call L a *centerline* for the unfolding. Sometimes we will abuse the terminology and call L *the* centerline even though it is never unique when it exists.

We now describe a labeling convention for the vertices of $U(W,T)$. We can think of $U(W,T)$ as the image in the plane of a polygonal disk $U^*(W,T)$ made by abstractly gluing together triangles as indicated above. The first and last edges of $U^*(W,T)$ make sense as we described them above. Deleting the first and last edges on $U^*(W,T)$, we have two remaining arcs on the boundary $\partial U^*(W,T)$. We call one of these arcs the "a-arc" and the other one the "b-arc." We choose so that the circuit "first-a-last-b" makes a clockwise loop around the boundary, as in Figure 1. We label the vertices of the a-arc $a_1, a_2, \dots$ going from the first edge to the last edge, and similarly for the b-arc.

In case the centerline exists, we can rotate so that the centerline is horizontal. In this case our labeling scheme is exactly as in Figure 1. All the a-vertices lie above the centerline and all the b-vertices lie below it. For certain choices of W, called *stable words*, the first and last sides of $U(W,T)$ are always parallel. In this case, we will rotate so that a horizontal translation in the positive direction carries the first edge to the last edge. In this case, a centerline exists if and only if all the a-vertices lie above all the b-vertices.

Remark 2.1. Given that our labeling is determined in a combinatorial way, we don't actually need a point in $O(W)$ in order to plot (or estimate) this tile. In practice, we always have such a point, but we don't use it.

Remark 2.2. In the end, we always rotate $U(W,T)$ so that a horizontal translation in the positive horizontal direction carries the first edge to the last. We call this *horizontal position.* However, when we compute certain quantities associated with $U(W,T)$, we initially have $U(W,T)$ in a potentially different position that we call *first position* below.

Remark 2.3. The *unfolding window* on McBilliards shows the unfoldings we have discussed in this section.

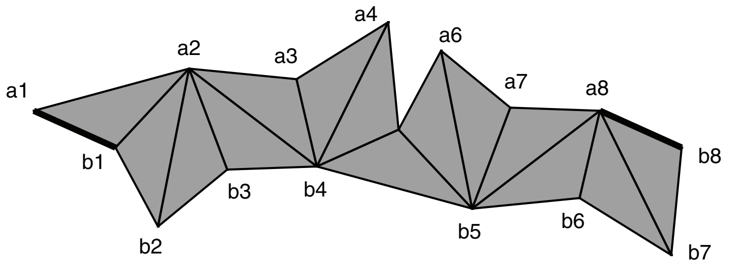

FIGURE 1. The unfolding for $W = (1232313)^2$.

Remark 2.4. Our figures of unfoldings do not always show the label of every relevant vertex. However, the required labels can easily be deduced from the labeling scheme we have described here. Essentially, one locates the vertices a1 and b1 on the left side of the picture and then "counts out to the right" to find the label of the vertex under discussion.

2.2 Stability

Recall that a periodic billiard path on a triangle is stable if nearby triangles have a periodic billiard path of the same combinatorial type. As a related notion, we say that a combinatorial type W is *stable* if the first and last sides of $U(W,T)$ are parallel for any triangle T.

Lemma 2.5. *If W is the combinatorial type for a stable billiards path, then W is a stable word. Conversely, if W is a stable word, then any periodic billiard path described by W is stable.*

Proof: Suppose that W describes a stable periodic billiard path on T. Then $U(W,T)$ has a centerline, and the first and last sides are parallel, as discussed above. When we perturb T slightly to a new triangle T', we still have a periodic billiard path with the same combinatorics. Hence, $U(W,T')$ still has a centerline. In particular, the first and last sides of $U(W,T')$ are still parallel. We have shown that the first and last sides of $U(W,*)$ are parallel for an open set of triangles. But then, by analytic continuation, these sides are always parallel.

Conversely, suppose that W describes a periodic billiard path on T and W is a stable word. When we perturb T slightly, the first and last sides are still parallel, and all the a-vertices still lie above all the b-vertices. Hence, a centerline still exists. Hence, nearby triangles still have a periodic billiard path described by W. □

Remark 2.6. In our proof, we used analytic continuation as a hammer to smash a pea. Just below, we will see that the parallelism of the first and last sides is a combinatorial condition.

Remark 2.7. One can certainly have stable words that describe no periodic billiard paths on any triangle. However, given our method of "search, then plot," the stable words we find always describe periodic billiard paths on triangles. That is, all the orbit tiles are guaranteed to be nonempty.

The well-known condition that W is a stable word is a combinatorial one. We will describe the stability condition in three equivalent ways. First, we break W into couplets, as we illustrate using the example from the previous section:

$$W = 12\ 32\ 31\ 31\ 23\ 23\ 13.$$

Let N_{ij} denote the number of couplets having type ij. In our example, we have

$$N_{12} = 1,\ \ N_{21} = 0,\ \ N_{23} = 2,\ \ N_{32} = 1,$$
$$N_{31} = 2,\ \ N_{13} = 1.$$

Lemma 2.8. *A word W is stable if and only if*

$$N_{12} - N_{21} = N_{23} - N_{32} = N_{31} - N_{13}.$$

Proof: Let $T_1, \dots, T_{2k}$ be the triangles in the unfolding $U(W,T)$. We put one more triangle T_0 at the beginning of our unfolding, so that reflection across the first edge maps T_0 to T_1. The first and last sides (both oriented the same way) are parallel if and only if T_0 and T_{2k} are related by a translation.

Let α_j be the angle of our triangle T opposite side j. Looking only at the even triangles of the unfolding, N_{12}

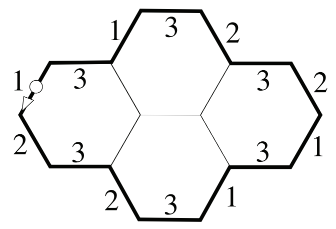

FIGURE 2. The hexpath for $W = (1232313)^2$.

counts the number of times a triangle T_j is rotated counterclockwise by $2\alpha_3$ into T_{j+2}. Likewise, N_{21} represents the same thing, but in the clockwise direction. The situation is similar for the other quantities. Thus, the angle through which we rotate T_0 to get to T_{2k} is

$$2(N_{12}-N_{21})\alpha_3+2(N_{23}-N_{32})\alpha_2+2(N_{31}-N_{13})\alpha_1. \quad (2\text{–}1)$$

The map carrying T_0 to T_{2k} is a translation if and only if the sum above is an integer multiple of 2π.

If our criterion holds, this quantity always equals an integer multiple of 2π. The point here is that $2\alpha_1+2\alpha_2+2\alpha_3 = 2\pi$. If our criterion fails, then we can produce triangles in which the corresponding sum is not an integer multiple of 2π. □

Remark 2.9. When W is unstable, setting the quantity in (2–1) to be a multiple of 2π puts a locally linear constraint on the angles. This explains why an unstable orbit tile is contained in a finite union of parallel lines. A bit more work shows that the orbit tile is contained in only one line.

Now we will describe stability a second way. Let $\mathcal{H}$ denote the 1-skeleton of the usual regular hexagonal grid in the plane. Then $\mathcal{H}$ has three parallel families of edges. Given a word, we can draw a path in $\mathcal{H}$ by following the edges as determined by the word: we move along the dth family when we encounter the digit d. We call this path the *hexpath* associated to the word. Figure 2 shows the hexpath corresponding to the example we have been considering. Note that the hexpath is closed in this case.

Lemma 2.10. *A word is stable if and only if its hexpath is closed.*

Proof: For this proof, we identify the plane with $\mathbb{C}$ and scale the picture so that opposite sides of each hexagon are one unit apart. Considering the hexpath two edges at a time, we see that the location of the final point has the same formula as in (2–1) except that we replace the angles $\alpha_1, \alpha_2, \alpha_3$ by three unit complex numbers z_1, z_2, z_3 forming the vertices of an equilateral triangle on the unit circle. An easy exercise shows that the corresponding sum vanishes exactly when our stability condition holds. □

Remark 2.11. The *word window* in McBilliards draws the hexpaths for the combinatorial types that the search engine finds.

Here we mention one last formulation of the stability condition.

Lemma 2.12. *Let $W = w_1, \dots, w_{2n}$. Let n_{dj} denote the number of solutions to the equation $w_i = d$ with i congruent to j modulo 2. Let $n_d = n_{d0} - n_{d1}$. Then W is stable if and only if $n_d(W)$ is independent of d.*

Proof: Interpreted in terms of the hexpath, the condition here again says that the hexpath is closed. □

2.3 Special Palindromes

We call W a *special palindrome* if W is stable and has the form

$$W = dwd\left(w^{-1}\right). \quad (2\text{–}2)$$

Here $d \in \{1, 2, 3\}$ and w is a subword, and w^{-1} is the reverse of w. In this case, $U(W,T)$ has bilateral symmetry, and the translation carrying the first side to the last side of $U(W,T)$ moves perpendicular to these sides.

If $U(W,T)$ has a centerline, then the centerline is necessarily perpendicular to the first and last sides. Hence, the corresponding periodic billiard path on T starts and ends perpendicular to one of the sides of T. Conversely, a *stable* periodic billiard path in T with this property has a combinatorial type that is a special palindrome.

Note that there are unstable words that satisfy some but not all of the mentioned properties. For instance, 123132 describes an unstable periodic billiard path in any right triangle, and this path starts and ends perpendicular to side 3.

2.4 Turning Angles and Turning Pairs

2.4.1 Definition. Let $U(W,T)$ be the unfolding. Let e_1 be the first edge, oriented so that it points from b_1 to a_1. We say that $U(W,T)$ is in *first position* if e_1 is parallel to $(0,1)$. That is, e_1 points in the direction of the positive Y-axis.

Given any oriented edge e of $U(W,T)$, we let $\theta(e)$ denote the angle through which one must rotate the positive

y-axis counterclockwise so that it coincides with e. Thus, $\theta(e_1) = 0$. In general, $\theta(e)$ is defined modulo 2π. The function $\theta(e)$ is really a function of $(x, y) \in \Delta$, and we write this dependence as $\theta(e; x, y)$. It is not hard to see, by induction, that there are integers $M(e)$ and $N(e)$ such that

$$\theta(e; x, y) = M(e)x + N(e)y + \epsilon\pi \bmod 2\pi.$$

Here $\epsilon \in \{0, 1\}$. It is sometimes more convenient to consider unoriented edges, in which case we have

$$\theta(e; x, y) = M(e)x + N(e)y \bmod \pi.$$

We call $(M(e), N(e))$ the *turning pair* for e. The rest of this section is devoted to explaining how one computes the turning pairs algorithmically.

2.4.2 The Angular Correspondence.

First we will give an abstract formulation of how the turning pairs are defined. There is a canonical map from the set of triangles of the unfolding to the set of vertices of the hexpath: We simply map T_i to the ith vertex v_i of the hexpath. The edge of $U(T, *)$ between T_i and T_{i+1} corresponds naturally to the midpoint of the edge joining v_i and v_{i+1}. The other two edges of T_i correspond naturally to the midpoints of the other two edges of $\mathcal{H}$ emanating from v_i. We call this correspondence the *angular correspondence*. For any object of the unfolding X, we let $\Theta(X)$ denote the point in the plane corresponding to X under the angular correspondence. It turns out that there is a real affine transformation R of the plane such that $(M(e), N(e)) = R(\Theta(e))$. The map R depends on how we coordinatize $\mathcal{H}$. This is the abstract formulation.

2.4.3 A Concrete Algorithm.

The algorithm that we describe is implemented in McBilliards. In Section 7 we explain how the reader can see this algorithm in action.

Let d be the first digit of W. For $\epsilon \in \{-1, 0, 1\}$ let $d_\epsilon \in \{1, 2, 3\}$ denote the congruence class of $(d + \epsilon)$ modulo 3. We define

$$\alpha_0(d_\epsilon) = \epsilon.$$

Suppose that we have determined $\alpha_{i-1}(1)$, $\alpha_{i-1}(2)$, and $\alpha_{i-1}(3)$. Let d be the ith digit of W. Define

$$\alpha_i(d_\epsilon) = \alpha_{i-1}(d_\epsilon) + (-1)^i 2\epsilon. \qquad (2\text{–}3)$$

In this way we produce a triple of labels for each triangle in the unfolding. If the plane is suitably coordinatized by variables (x, y, z) such that $x + y + z = 0$, then the triple associated to T_i is precisely the coordinates of $\Theta(T_i)$, the ith vertex of the hexpath.

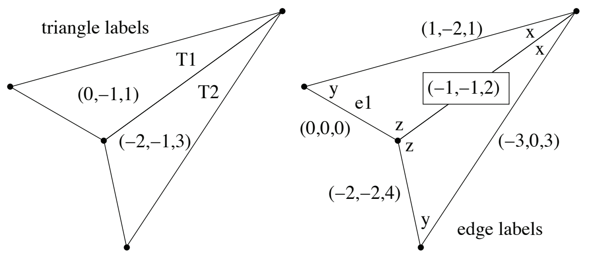

FIGURE 3. Labels for $W = 12\ldots$.

Let e be an edge of $U(W, T)$. Suppose that e is the dth edge of T_i. We define

$$\beta(e, d_\epsilon) = \alpha_i(d_\epsilon) - (-1)^i \epsilon. \qquad (2\text{–}4)$$

Note that e could also be an edge of another triangle of $U(W, T)$. This happens when T_{i-1} and T_i are related by a reflection through e. In other words, d is the ith digit of W. In this situation, (2–4) gives the same answer whether we use $i - 1$ or i in the formula. This can be seen by comparing (2–3) and (2–4).

Lemma 2.13. *We have the general formula*

$$\theta(e) = -\frac{\beta(e, 1)x + \beta(e, 2)y + \beta(e, 3)z}{3}. \qquad (2\text{–}5)$$

Here z is such that $x + y + z = 0$.

Proof: We first check our formula on the edges of T_1. If 1 is the first digit of W, then the edge labels of e_1 are $(0, 0, 0)$, and hence both sides of (2–5) are 0. The edge labels of e_2 are $(-1, -1, 2)$. In this case, (2–5) gives $\theta(e_2) - \theta(e_1) = -(-x - y + 2z)/3 = -z$, as it should. The edge labels of e_3 are $(1, -2, 1)$. In this case, (2–5) gives $\theta(e_3) - \theta(e_1) = -(x - 2y + z)/3 = y$, as it should.

Given the simple nature of the formulas in (2–3) and (2–4), it suffices to check the induction step for $i = 2$. In other words, we just have to see that (2–5) works for the edges of T_2. Again, we can suppose that 1 is the first digit of W. Suppose that 2 is the second digit. Figure 3 shows a picture of the situation. One easily checks that (2–5) holds for all these edges. When the second digit of W is a 3, the verification is similar. □

It is useful to have a formula that doesn't involve the angle z. We define

$$M(e) = \frac{\beta(e, 3) - \beta(e, 1)}{3}, \quad N(e) = \frac{\beta(e, 3) - \beta(e, 2)}{3}.$$

If follows from Lemma 2.13 that $(M(e), N(e))$ is the turning pair for e.

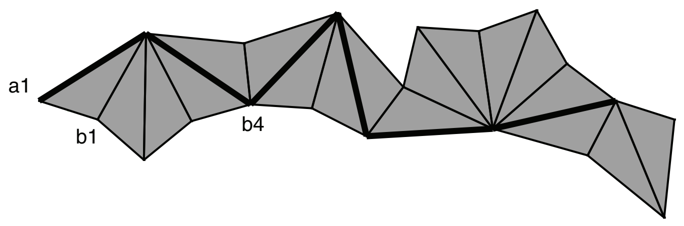

FIGURE 4. The 3-spine for $W = 123231323123232313$.

2.5 Defining Functions

Given two points $v, w \in \mathbb{R}^2$ we write $v \uparrow w$ if the y-coordinate of v exceeds the y-coordinate of w. In the opposite case, we write $v \downarrow w$. In case of equality, we write $v \updownarrow w$. We are interested in the case that v and w are vertices of an unfolding that is in horizontal position.

In this section we will explain how to define a function f_{uv} such that $f_{uv} > 0$ if and only if $u \uparrow v$. Here f_{uv} is a function of $(x, y) \in \Delta$. The function f_{uv} has a combinatorial definition in terms of the positions of u and v on the unfolding. We warn the reader in advance that we will *define* f_{uv} using constructions that make sense when $U(W,T)$ is in first position, but we will *interpret* f_{uv} as making statements about $U(W,T)$ when the unfolding is in horizontal position.

Given our functions, we are left with the following problem: To show that $Q \subset O(W)$ for some region Q, we just have to show that $f_{a_i,b_j} > 0$ throughout Q, for all pairs (a_i, b_j). This explains why the orbit tiles are "finite-sided" regions with sides defined by (as we will see) analytic functions.

Let $\widetilde{U}(W,T)$ be the bi-infinite periodic continuation of $U(W,T)$. For any $d \in \{1,2,3\}$ there is an infinite periodic polygonal path made from type-d edges in $\widetilde{U}(W,T)$. The image of this path in $U(W,T)$ is what we call the *d-spine*. Figure 4 shows the 3-spine for $U(W,T)$, where T is some triangle of no interest to us. When the first and last sides of $U(W,T)$ are glued together, the d-spine is a closed polygonal loop.

Let $e_1, \ldots, e_n$ be a complete and irredundant list of the edges that appear in the d-spine. We label so that e_1 is the leftmost edge. Let

$$g_d(x,y) = \sum_{k=1}^{n} (-1)^{k-1} \exp(i(M(e_k)x + N(e_k)y)). \quad (2\text{–}6)$$

Lemma 2.14. *Suppose that $U(W,T)$ is in first position. The translation direction of $U(W,T)$ is parallel to $\pm i g_d(x,y)$ for any $d \in \{1,2,3\}$.*

Proof: Suppose e_1 and e_2 are oriented edges of $U(W,T)$, incident to a common vertex v, and oriented the same way with respect to v. See Figure 5. Suppose that e_2 lies to the right of e_1. Then

$$Mx + Ny, \quad M = M(e_2) - M(e_1), \quad N = N(e_2) - N(e_1), \quad (2\text{–}7)$$

represents, modulo 2π, the counterclockwise angle through which e_1 is rotated to produce e_2.

We identify the plane with $\mathbb{C}$. We scale so that the vectors $e_1, \ldots, e_n$ of the spine are all unit vectors and the tail vertex of e_1 is 0. We consider two orientations on these edges. Say that the *red* orientation is the one in which the head of each edge hits the tail of the next one. Say that the *blue* orientation is the one in which incident edges both point either away or toward the vertex of incidence, as in Figure 5.

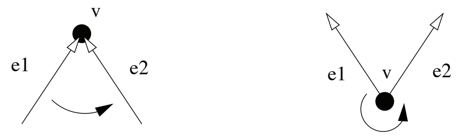

FIGURE 5. Two edges.

Let $\rho_k \in \mathbb{C}$ denote the head of e_k minus the tail of e_k when this edge is oriented according to the red orientation. Likewise define β_k for the blue orientation. We have $\beta_k = \pm\rho_k$, with the sign depending on the parity of the index. The initial point e_1 is 0. The translation we seek carries 0 to the endpoint z of e_n with the red orientation. Thus, our translation carries 0 to

$$\sum_{k=1}^{n} \rho_k = \sum_{k=1}^{n} (-1)^{k-1} \beta_k. \quad (2\text{–}8)$$

On the other hand, it follows from induction and (2–7) that

$$\rho_k = \epsilon \exp(i\theta_k), \quad \theta_k = M(e_k)x + N(e_k)y. \quad (2\text{–}9)$$

Here $\epsilon \in \{-1, 1\}$ is a global sign that does not depend on k.

Combining (2–8) and (2–9) gives us our result. □

Let v and w be two vertices of $U(W,T)$. We say that v and w are *d-connected* if there is a polygonal path of type-d edges connecting v to w, and d is as large as possible.

Lemma 2.15. *Any pair of vertices are d-connected for a unique $d \in \{1,2,3\}$.*

Proof: If d exists, then by definition d is unique. For the existence, let v and w be our vertices. Then v and w are each incident to two types of vertices. Hence, there is some type, say d, such that v and w are both incident to an edge of type d. We take d as large as possible. Either v lies on the d-spine or else we can connect v to the d-spine with an edge of type d. The same goes for w. Thus, we form our path by connecting v to the d-spine, traveling along the d-spine, and then connecting to w. □

Let $e'_1, \dots, e'_m$ be the set of type-d edges joining v to w, ordered from left to right. We define

$$h(x,y) = \sum_{k=1}^{m} (-1)^{k-1} \exp\left(i(M(e'_k)x + N(e'_k)y)\right).$$

Lemma 2.16. *Suppose that $U(W,T)$ is in first position. The vector pointing from p to q is parallel to $\pm h(x,y)$.*

Proof: This has almost exactly the same proof as Lemma 2.14. □

Letting d be such that v and w are d-connected, we set $g = g_d$.

Lemma 2.17. *Let T be the triangle corresponding to $(x,y) \in \Delta$. Suppose that $U(W,T)$ is in horizontal position. Then the function*

$$f(x,y) = \pm \Im(\overline{g}h) \qquad (2\text{–}10)$$

vanishes if and only if $v \updownarrow w$.

Proof: We observe that $v \updownarrow w$ when $U(W,T)$ is in horizontal position if and only if $g = g(x,y)$ and $g = h(x,y)$ are real multiples of each other, which happens if and only if $g\overline{h} \in \mathbb{R}$, which happens if and only if $f(x,y) = 0$. □

2.6 Getting the Sign Right

Now we discuss the sign in front of (2–10). We want to use our function to determine when a given vertex lies above another one, and not just when two given vertices are at the same height. That is, we would like to know which sign choice guarantees that $f_{vw}(x,y) > 0$ if and only if $v(x,y) \uparrow w(x,y)$. Here we emphasize that the positions of v and w depend on a parameter $(x,y) \in \Delta$.

There are two approaches we might take. One approach is to make a guess in any given case and then use a single auxiliary computation for some point (x_0, y_0) to correct the guess if necessary. By continuity, if the sign is right (or wrong) at one parameter, it is right (or wrong) at all parameters. Given that the computer-aided portion of our proof involves only a finite number of these functions, we could easily have taken this approach.

The approach we actually take is to establish some general sign conventions and follow them. After an embarrassingly huge amount of trial and error, we discovered the general rule that allows us to establish and implement our sign conventions. The proof that the sign formula is correct is a matter of induction. We omit the details.

Recall that v lies to the left of w, and $e'_1, e'_2, \dots$ are the edges of our chosen path connecting v to w. Finally, we note that there is a canonical left-to-right ordering on all the edges of the same type.

The d-spine gives a natural ordering to the edges $e_1, \dots, e_n$ (the red ordering in our proof of Lemma 2.14), and so it makes sense to speak of the left vertex of e_1. This is the tail vertex of e_1 relative to the red ordering. The left vertex of e_1 is either a_1 or b_1.

We establish the following rule:

Rule 2.18. (The sign rule.) Let s be the number of edges on the list $e_1, \dots, e_n$ that lie to the left of e'_1.

- Suppose the left vertex of e_1 is a_1. Then $(-1)^s \Im(\overline{g}h) > 0$ if and only if $v \uparrow w$.
- Suppose the left vertex of e_1 is b_1. Then $(-1)^s \Im(\overline{g}h) > 0$ if and only if $v \downarrow w$.

McBilliards uses the sign rule as a basis for establishing the following sign conventions:

1. Suppose $v = a_i$ and $w = b_j$. Then $f > 0$ if and only if $v \uparrow w$.
2. Suppose $v = a_i$ and $w = b_j$ and $i < j$. Then $f > 0$ if and only if $w \uparrow v$.
3. Suppose $v = b_i$ and $w = b_j$ and $i < j$. Then $f > 0$ if and only if $w \uparrow v$.

2.7 Notation

We introduce a shorthand notation for the function f. It suffices to list the turning pairs defining h and then the turning pairs defining g. For instance, in the example above, the defining function for the pair (a_1, b_4) is recorded as

$$\begin{array}{cccccc} 0 & 1 & 0 & 1 & (+) \\ 4 & 1 & 4 & 1 & \\ 4 & -1 & & & \\ 6 & -1 & & & \\ 6 & -3 & & & \\ 0 & -3 & & & \end{array}$$

Here $m = 2$ and $n = 6$. The $(+)$ indicates the sign choice. From the notation we read off that

$$\begin{aligned} g(x,y) &= \exp(i(y)) - \exp(i(4x+y)) + \exp(i(4x-y)) \\ &\quad - \cdots - \exp(i(-3y)), \\ h(x,y) &= (+1) \times (\exp(i(y)) - \exp(i(4x+y))). \end{aligned}$$

We call this *form* 1 for the defining function.

To arrive at a second convenient form for our function we multiply $\overline{g}$ and h together, collect terms, and use the fact that sine is an odd function. This gives us what we call *form* 2 for the defining function:

$$f(x,y) = \sum_k J_k \sin(A_k x + B_k y), \quad J_k \in \mathbb{N},\ A_k, B_k \in \mathbb{Z}. \tag{2–11}$$

Remark 2.19. Using McBilliards, the reader can see computations of the turning pairs and defining functions for any given example. See Section 7 for details.

3. THE INFINITE FAMILIES

3.1 The Region of Interest

The region $P_3 \subset \Delta$ has coordinates

$$\left(0, \frac{\pi}{2}\right), \quad \left(0, \frac{3\pi}{8}\right), \quad \left(\frac{\pi}{8}, \frac{3\pi}{8}\right).$$

The lightly shaded region in Figure 6 is S_{100}. The darkly shaded region is P_3. Note that S_{100} continues behind P_3.

Lemma 3.1. *The region P_3 has no covering by finitely many orbit tiles.*

Proof: Let T be a triangle whose largest angle is $90 + \epsilon$ and whose smallest angle is δ, where $\delta \ll \epsilon$. Any billiard path P in T must eventually hit the short side of T at a point x. But then at least one of the segments S of P, incident to x, will make an angle comparable to ϵ

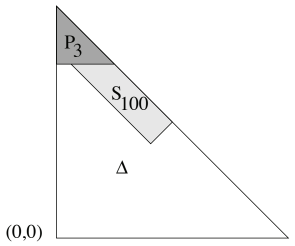

FIGURE 6. The region P_3.

with one of the long sides. Tracing P out from x in the direction of this segment, we see that P has to make about ϵ/δ bounces, moving roughly away from the short side, before its direction can change enough for it to turn around. This shows that T supports no short periodic billiard paths. Hence, we need infinitely many orbit tiles to cover P_3. □

We introduce the words A_n and B_n for $n = 1, 2, 3, \ldots$:

$$\begin{aligned} A_n &= 3w_n 3w_n^{-1}, \quad B_n = 3w_{n+1} 3w_n^{-1}, \\ w_n &= 1(32)^{n+1} 1(23)^n 2. \end{aligned}$$

We break P_3 into subregions, each of which is covered by a single tile. Let N_n denote the open triangle bounded by

- the bottom edge of P_3, namely the line $y = 3\pi/8$;
- the line through $(0, \pi/2)$ having slope $-(n+1)/2$;
- the line through $(0, \pi/2)$ having slope $-(n+2)/2$.

Let N_n' denote the open line segment that is the common boundary of N_n and N_{n+1}. We will prove that

$$N_n \subset O(A_n), \quad N_n' \subset O(B_n), \quad n = 1, 2, 3, \ldots. \tag{3–1}$$

Figure 7 shows the tiles $O(A_1), \ldots, O(A_6)$, and the right-hand side shows $O(B_1), \ldots, O(B_6)$ superimposed over the left-hand side. The tiles continue sweeping out to the left, covering P_3.

We deal with the A tiles first, then the B tiles.

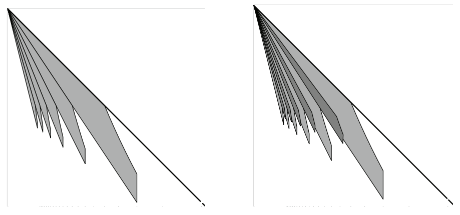

FIGURE 7. Some orbit tiles.

3.2 The A Unfoldings

Figures 8 through 10 show unfoldings for A_1, A_2, and A_3 respectively, for various choices of triangle. The pattern continues in the obvious way.

3.3 A Special Case

We will concentrate on the case $n = 2$, which is sufficiently complex to contain all the ideas in the proof. At the end we will explain the general case.

Here we analyze the vertices of $U(A_2, T)$ when T corresponds to a point in N_2. By symmetry, it suffices to consider the vertices on the right half of the unfolding. We change our labeling scheme somewhat and start counting our vertices from the center, as in Figure 9.

Lemma 3.2. *For any $T \in N_2$ and any j, we have $a_j \uparrow a_2$ or $a_j \uparrow a_7$. In other words, the lowest top vertex is either a_2 or a_7 in all cases.*

Proof: Let $\theta(e)$ denote the turning angle of an edge e, as discussed in the previous section. We write (for instance) $\theta(a_1b_2) = \theta(\overrightarrow{a_1b_2})$. We let z denote the angle opposite edge 3, so that $x + y + z = \pi$. Here are the angles of importance to us:

$$\theta(a_1a_2) = 6x+\pi, \quad \theta(a_7a_8) = -x, \quad \theta(b_7a_5) = \pi+3x+2y. \tag{3–2}$$

We derive the third equation, which is the least obvious. We rotate $\overrightarrow{b_1a_1}$ by $6x$ to get $\overrightarrow{a_2a_1}$. Then we rotate $\overrightarrow{a_2a_1}$ by $2y$ to get to $\overrightarrow{a_2b_7}$. Then we rotate $\overrightarrow{a_2b_7}$ by $-3x$ to get to $\overrightarrow{a_5b_7}$. Then we rotate by π to reverse the direction.

The conditions $(x, y) \in N_2$ give rise to the angle constraints

$$x \in \left(0, \frac{\pi}{12}\right), \quad y \in \left(\frac{3\pi}{8}, \frac{\pi}{2}\right). \tag{3–3}$$

See Figure 8.

From (3–2) we now get

$$\theta(a_1a_2) \in (\pi, \pi + \pi/2), \quad \theta(a_7a_8) = x \in (-\pi/2, 0).$$

But this means that $a_1 \uparrow a_2$ and $a_8 \uparrow a_7$.

Consider the polygonal "fan" F whose vertices are b_7 and $a_3, \ldots, a_7$. Note that $\overline{b_7a_5}$ is the line of bilateral symmetry for F. Our constraint $(x, y) \in N_2$ gives

$$\pi - \pi/4 < 2y < \pi - 3x, \quad 3x < \pi/4. \tag{3–4}$$

Combining these bounds with (3–2), we get

$$\theta(b_7a_5) \in \left(\frac{7\pi}{4}, 2\pi\right). \tag{3–5}$$

Hence $\overline{b_5a_7}$ has positive slope.

Equation (3–4) guarantees that F is contained in a half-plane. We can write $F = F_1 \cup F_2$, where F_1 is the convex hull of b_7 and the odd vertices a_3, a_5, a_7. Then F_2 is a union of two small triangles, as shown in Figure 11. Given the conditions on F, we see that $a_3 \uparrow a_7$ and $a_5 \uparrow a_7$.

Since the line of symmetry of F has positive slope and F lies in a half-plane, each even a-vertex of the fan lies above one of the adjacent odd a-vertices. The point here is that the line segments connecting b_7 to the even vertices are longer than the line segments connecting b_7 to the odd vertices. All in all, $a_j \uparrow a_7$ for $j = 3, 4, 5, 6$. $\square$

Now we deal with the bottom vertices.

Lemma 3.3. *In all cases, the highest bottom vertex is either b_6 or b_8.*

Proof: The proof is almost the same as for the top vertices. Let $(x, y) \in N_2$, as above. Since $y < \pi/2$, the line $\overline{b_1b_2}$ has positive slope. Hence $b_2 \uparrow b_1$. To understand the vertices $b_2, \ldots, b_4$, we consider the "fan" whose vertices are $a_1, b_2, b_3, b_4, b_5, b_6$. This polygon is isometric to the one considered in the previous subsection. The line of symmetry of F is $\overline{a_1b_4}$. This line has negative slope because of the fact that $3x < \pi/2$. The same argument as above now shows that $b_6 \uparrow b_j$ for $j = 2, 3, 4, 5$. The angle between $\overrightarrow{b_7b_6}$ and $\overrightarrow{b_7a_5}$ is $4x < \pi/3$. Combining this information with (3–5), we see that

$$\theta(b_7b_6) \in \left(\frac{7\pi}{4}, \frac{5\pi}{2}\right) \equiv \left(-\frac{\pi}{4}, \frac{\pi}{3}\right).$$

From this we see that $b_6 \uparrow b_7$. $\square$

To finish the proof that $N_2 \subset O(A_2)$, we prove the following result.

Lemma 3.4. *When T corresponds to a point in N_2, we have $a_i \uparrow b_j$ for $i \in \{2, 7\}$ and $j \in \{6, 8\}$.*

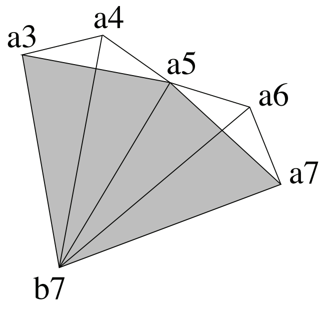

FIGURE 11. The fan.

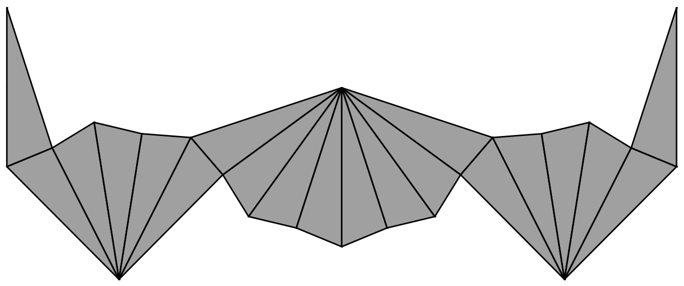

FIGURE 8. Unfolding for A_1.

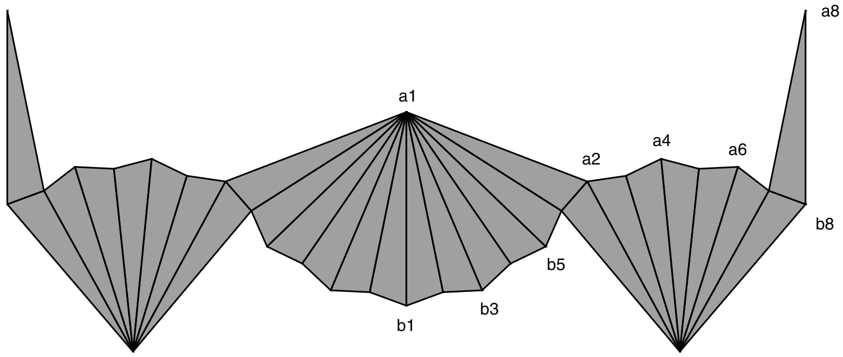

FIGURE 9. Unfolding for A_2.

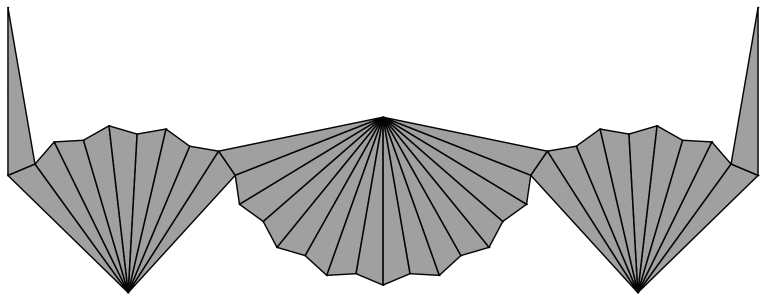

FIGURE 10. Unfolding for A_3.

Proof: Consider first (a_7, b_8). We have

$$\theta(b_8 a_7) = y \in (0, \pi/2).$$

Hence $a_7 \uparrow b_8$.

Now consider (a_2, b_6). We have $\theta(a_2 b_6) = 4x + y \in (\pi/2, \pi)$. Hence $a_2 \uparrow b_6$.

Now consider (a_2, b_8). Note that a_2 and b_8 are symmetrically located with respect to our favorite line $\overline{b_7 a_5}$. Thus a_2 and b_8 have the same height if and only if our line is vertical. From (3–2) and (3–5) we see that this happens for a point in the closure of N_2 if and only if $2y + 3x = \pi$. That is, (x, y) has to lie on the right boundary line of N_2. Equation (3–5) shows that $a_2 \uparrow b_8$ for $(x, y) \in N_2$.

Now consider (a_7, b_6). Note that a_7 and b_6 have the same height if and only if the line $\overline{b_7 a_4}$ is vertical. Essentially the same analysis as we have already done shows that our line has negative slope for $(x, y) \in N_2$ and is vertical for $2y + 4x = \pi$. Hence $a_4 \uparrow b_7$. The two points have the same height when (x, y) is in the left boundary of N_2. $\square$

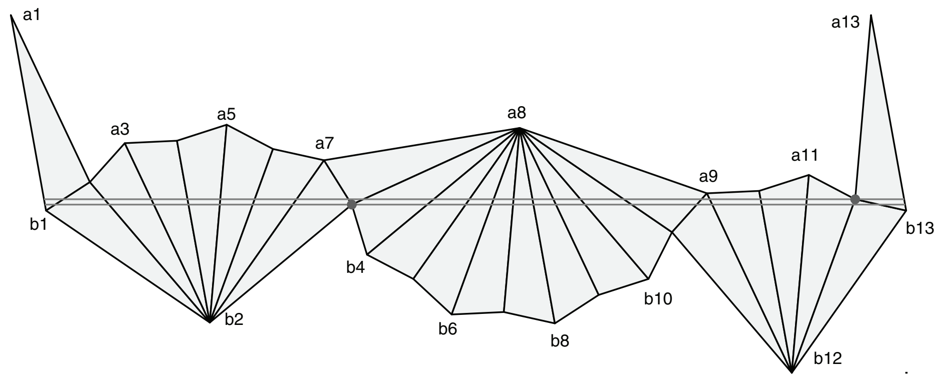

FIGURE 12. Unfolding for B_1.

3.4 The General Case

We deal with the top vertices first. The general version of (3–2) is

$$\theta(a_1a_2) = (2n+2)x + \pi, \quad \theta(a_{2n+3}a_{2n+4}) = -x,$$

and

$$\theta(b_{2n+3}a_{n+3}) = \pi + (n+1)x + 2y. \qquad (3\text{–}6)$$

Equation (3–6) eliminates a_{2n+4} and a_1 from consideration.

The conditions $(x, y) \in N_n$ give rise to the angle constraints

$$x \in \left(0, \frac{\pi}{4n+8}\right), \quad y \in \left(\frac{3\pi}{4}, \frac{\pi}{2}\right). \qquad (3\text{–}7)$$

For $(x, y) \in N_n$ we have

$$\pi - \pi/4 < 2y < \pi - (n+1)x.$$

These equations combine together with (3–6) to show that the line $\overline{b_{2n+3}a_{n+3}}$ has positive slope. This line is the center of symmetry of the fan with vertices $b_{2n+3}, a_3, \ldots, a_{2n+3}$. The same argument as above then shows that $a_2, \ldots, a_{2n+2}$ lie above $a_j \uparrow a_{2n+3}$ for $j = 2, \ldots, 2n+2$. In this way we eliminate everything but a_2 and a_{2n+3}.

Essentially the same argument eliminates all the b-vertices except b_{2n+2} and b_{2n+4}. The key point is that the line $\overline{a_1b_{n+2}}$, which is the line of symmetry for the fan with vertices $a_1; b_2, \ldots, b_{2n+2}$, has negative slope. This follows from (3–7).

The analysis of the edges is the same in the general case. The main points that need to be observed are these:

- The points a_2 and b_{2n+4} have the same height if and only if $\overline{b_{2n+4}a_{n+3}}$ is vertical, and this happens if and only if $2y + (n+1)x = \pi$.
- The points a_{2n+3} and b_{2n+2} have the same height if and only if the line $\overline{b_{2n+3}a_{n+2}}$ is vertical, and this happens if and only if $2y + (n+2)x = \pi$.

All this information assembles together in the same way as in the case $n = 2$ to show that $N_n \subset O(A_n)$.

3.5 The B Unfoldings

Now we turn to the B tiles. We will draw $U(B_1, T)$ for some triangle T, and then explain the general case.

Figure 12 shows $U(B_1, T)$ for some triangle corresponding to a point $(x, y) \in N_1'$. We return to our original convention of labeling the vertices of the unfolding, where we start all the way to the left. Such points satisfy the equation

$$3x + 2y = \pi. \qquad (3\text{–}8)$$

The (near) central edge (a_8, b_8) is parallel to both (a_1, b_1) and (a_{13}, b_{13}). Indeed, the portion of $U(B_1, A)$ to the left of (a_8, b_8) is isometric to the right half of $U(W_2, A)$, and the portion to the right of (a_8, b_8) is isometric to the left half of $U(W_1, A)$. This is fitting, because $O(B_1)$ fits “between” $O(W_1)$ and $O(W_2)$.

In general, $U(B_n)$ is obtained by splicing together the left half of $O(A_n)$ with the right half of $O(A_{n+1})$.

We will take the same approach as for the A tiles. We first consider a special case in detail and then explain the changes needed for the general case. Again, this is entirely for the sake of exposition. We first show that $N_1' \subset O(B_1)$.

3.6 An Estimate for the Rotation Angle

In the section, we prove that

$$\theta(b_{13}a_{13}) \in (0,x), \quad \theta(a_{12},a_{13}) \in (-x,0). \tag{3–9}$$

Lemma 3.5. *There is some $\epsilon > 0$ such that $\theta(a_{12},a_{13}) \in [0,\epsilon)$ is impossible.*

Proof: The conditions in (3–8) guarantee that the following lines are parallel:

$$\overline{a_{11}b_{12}}, \quad \overline{a_8b_7}, \quad \overline{a_5b_2}, \quad \overline{a_{12}a_{13}}.$$

By symmetry, the points b_{13} and a_3 are related by a reflection in $\overline{a_8b_7}$. The points a_3 and b_1 are related by a reflection in $\overline{a_2b_2}$. If $\overline{a_{12}a_{13}}$ is vertical or has negative slope, then a_3 lies below b_{13}. On the other hand, if $\overline{a_{12}a_{13}}$ is vertical and has large negative slope, then $\overline{a_2b_2}$ has negative slope. (Here we are using $3x \leq \pi/4$. Compare (3–4).) But then b_1 lies below a_3. But then b_1 lies below b_{13}, a contradiction. □

Lemma 3.6. *There is some $\epsilon > 0$ such $\theta(b_{13},a_{13}) \in (-\epsilon,0]$ is impossible.*

Proof: Condition (3–8) guarantees that $\overline{a_{10}b_{12}}$, $\overline{a_8b_8}$, $\overline{a_4b_2}$, and $\overline{a_1b_1}$ are all parallel to $\overline{a_{13}b_{13}}$. Let a_0 denote the reflection of a_2 through the line $\overline{a_1b_1}$. Our normalization puts a_0 and a_{12} at the same height. The points a_0, a_2, a_6 are successively related to each other by reflections in the lines mentioned above.

Likewise, the points a_{12}, b_{11}, b_5 are successively related to each other by reflections in the lines mentioned above. If $\overline{a_{13}b_{13}}$ either is vertical or has sufficiently large negative slope, then b_5 lies above a_6.

The points a_6 and b_3 are related to each other by a reflection through $\overline{b_2a_7}$. The points b_3 and b_5 are related to each other by reflection in the line $\overline{a_8b_4}$. If $\overline{a_{13}b_{13}}$ is either vertical or has sufficiently large negative slope, then these two last-mentioned lines both have negative slope, and hence b_5 lies below a_6. This is a contradiction. □

When x is near 0, the 1-spine of our unfolding converges to a horizontal path. Hence, the two lines $\overline{a_{13},b_{13}}$ and $\overline{a_{12},a_{13}}$ converge to vertical lines. By our previous results, these lines have opposite slopes. As we increase x and remain on N_1', this property cannot be lost, by our two results. This establishes our inequalities.

3.7 Most of the Vertices

Lemma 3.7. *For any T corresponding to points in N_1', the lowest top vertex is either a_9 or a_{12}. The highest bottom vertex is one of b_1, b_3, b_{13}.*

Proof: From (3–9) we get $\theta(a_2a_1) \in (x,2x)$. Hence $a_1 \uparrow a_2$.

We have $\theta(b_2a_5) = \theta(a_{12}a_{13})$. By (3–9) we see that $\overline{b_2a_5}$ has positive slope. This line happens to be the line of symmetry for the fan with vertices $b_2; a_3,\ldots,a_7$. The same argument as in Section 3.3 shows that $a_j \uparrow a_7$ for $j = 2,3,4,5,6$. Similarly, considering the fan with vertices $a_8, b_3, \ldots, b_{11}$, whose line of symmetry $\overline{a_8b_7}$ has positive slope, we see that $b_3 \uparrow b_j$ for $j = 4,\ldots,11$.

Equation (3–9) gives $\theta(a_7a_8) \in (0,6x) \subset (0,\pi/2)$. Hence $a_8 \uparrow a_7$. Similarly, $b_4 \uparrow b_2$ and $b_{13} \uparrow b_{12}$.

Vertices a_7 and a_9 are related by reflection through $\overline{a_8b_7}$, a line with negative slope. Hence $a_7 \uparrow a_9$.

Note that a_{12} and a_{10} are related by a reflection through $\overline{b_{12}a_{10}}$, a line that has negative slope because it is parallel to $\overline{a_{13}b_{13}}$. Hence $a_{10} \uparrow a_{12}$.

Vertices a_{12} and a_0, the point defined in the proof of Lemma 3.6, are at the same height. Moreover, a_0 and a_2 are related by a reflection through the negatively sloped $\overline{a_1b_1}$. Hence $a_2 \uparrow a_{12}$. □

3.8 The Six Pairs

It remains to deal with six pairs of vertices. Throughout our argument, we work with triangles corresponding to points in N_1'.

Lemma 3.8. *We have $a_i \uparrow b_j$ for $i \in \{9,12\}$ and $j \in \{1,13\}$.*

Proof: Note that b_1 and b_{13} are vertices related by a horizontal translation. Thus, we need not consider b_1.

We have $\theta(b_{13},a_{12}) \in (y,x+y)$. We also have $3x+2y = \pi$. Hence $\theta(b_{13},a_{12}) \in (0,\pi/2)$. Hence $a_{12} \uparrow b_{13}$.

Consider the pair (a_9,b_{13}). Since b_{13} and a_9 are related by reflection through $\overline{b_{12}a_{12}}$ and $\theta(b_{12},a_{12}) = \theta(a_{12},a_{13}) \in (-x,0)$, we have $a_9 \uparrow b_{13}$. □

It remains to consider the pairs (a_9,b_3) and (a_{12},b_3).

Lemma 3.9. *The vertex a_9 lies above the line $\overline{b_3a_{12}}$. Hence, $a_{12} \uparrow b_3$ implies $a_9 \uparrow a_3$.*

Proof: Let θ_1 denote the angle $\angle b_3a_8a_9$. Let θ_2 denote the angle $\angle b_{12}a_9a_{12}$. The point a_9 lies on $\overline{b_3a_{12}}$ if and

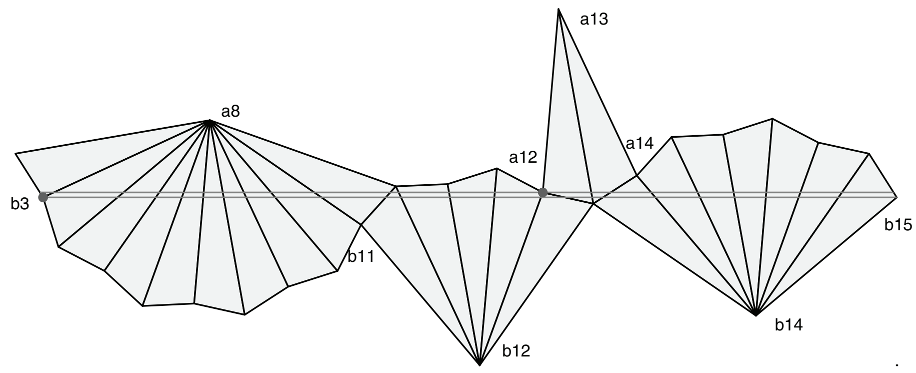

FIGURE 13. Cut and paste version of Figure 12.

only if $(\pi - \theta_2) + \theta_1 = \angle a_8 a_9 b_{12} = 2y$. Using this fact as a guide, we check signs to determine that a_9 lies above $\overline{b_4 a_{12}}$, provided that $\pi - \theta_2 + \theta_1 > 2y$.

Using the law of sines, we can normalize so that our triangles all have side lengths $\sin(x), \sin(y), \sin(z)$. Let $\theta_3 = \angle a_9 a_{12} b_{12}$. Looking at the triangle with vertices a_3, a_8, b_9 and using the law of sines, we get

$$\theta_3 = \frac{\sin(z)}{\sin(y)} \theta_1.$$

Using that the sum of the three angles in a triangle is π, together with (3–8), we get

$$\theta_1 + \theta_3 = \pi - 9x = \pi - 4 \times 3x + 3x = -3\pi + 8y + 3x = 5y - 3z. \tag{3–10}$$

The first equation uses (3–8). Solving for θ_1, we get

$$\theta_1 = \frac{(5y - 3z)\sin(y)}{\sin(y) + \sin(z)}. \tag{3–11}$$

Let $\theta_4 = \angle a_9 a_{12} b_{12}$. From the law of sines we have

$$\theta_4 = \frac{\sin(z)}{\sin(y)} \theta_2.$$

We also have

$$\begin{aligned} \theta_2 + \theta_4 &= \pi - 3x = \pi - 2 \times 3x + 3(\pi - y - z) \\ &= \pi - 2(\pi - 2y) + 3\pi - 3y - 3z \qquad (3–12) \\ &= 2\pi + y - 3z. \end{aligned}$$

(We have complicated this equation so that it readily generalizes.)

Solving for θ_2, we get

$$\theta_2 = \frac{(2\pi + y - 3z)\sin(y)}{\sin(y) + \sin(z)}. \tag{3–13}$$

Using (3–11) and (3–13), we compute

$$(\pi - \theta_2 + \theta_1) - 2b = \frac{(\pi - 2y)(\sin(z) - \sin(y))}{\sin(y) + \sin(z)}. \tag{3–14}$$

Note that $\sin(z) > \sin(y)$. The expression in (3–14) is positive as long as $y < \pi/2$, which is certainly our situation. □

Lemma 3.10. *We have* $a_{12} \uparrow b_3$.

Proof: It is useful to cycle our picture so that b_3 is all the way to the left. See Figure 13, which is a cut-and-paste equivalent to Figure 12.

Note that a_{12} lies to the left of both b_{14} and b_{15}. To see this, note that a_{14} and b_{15} are related by reflection in the nearly vertical line $\overline{b_{14} a_{17}}$, and a_{14} and a_{12} are related by reflection in the nearly vertical line $\overline{a_{13} b_{13}}$. These lines make an angle of less than x with the vertical, from (3–9). The same argument shows that b_3 lies to the left of a_{12}.

Let σ_1 and σ_2 respectively denote the slopes of $\overline{b_{15} a_{12}}$ and $\overline{b_3 b_{15}}$ when the picture is rotated so that $\overline{b_{15} b_{14}}$ is horizontal. Since a_{12} and b_3 lie to the left of both b_{14} and b_{15}, the slopes σ_1 and σ_2 are finite. We will show that that $\sigma_1 < \sigma_2$. This, together with the fact that b_3 lies to the left of a_{12}, shows that $a_{12} \uparrow b_3$, as desired.

Consider the path of eight vectors $v_1, \ldots, v_8$ defined by the vertex sequence

$$(b_{15}, b_{14}, a_{14}, a_{13}, a_{12}, b_{12}, b_{11}, a_8, b_3).$$

In the terminology of Section 4, this path is part of the 1-spine. The first vector points from b_{15} to b_{14}, and so forth. These vectors all have the same length, which we normalize to be 1.

Let θ_k denote the counterclockwise angle through which v_1 must be rotated to produce v_k. We now calculate these vectors.

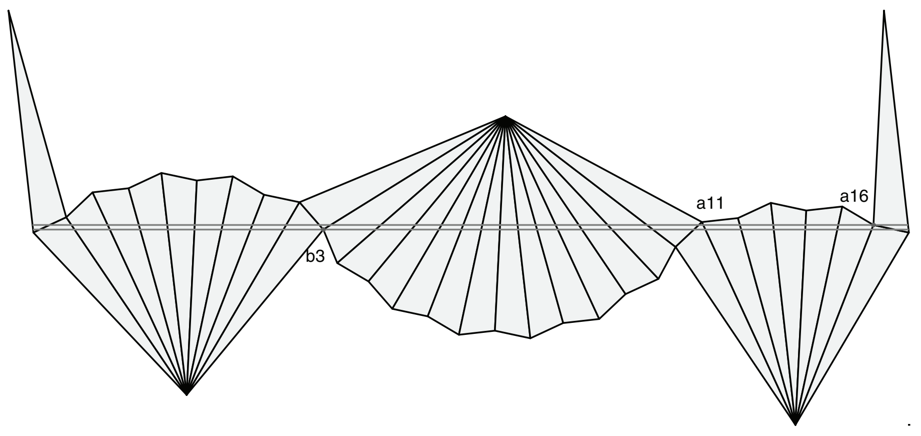

FIGURE 14. Unfolding for B_2.

Looking at Figure 13, we have $\theta_1 = 0$ and

$$\begin{aligned}\theta_2 &= 6x + \pi = -4y + \pi,\\ \theta_3 &= 6x - 2z = -4y - 2z,\\ \theta_4 &= 4x - 2z + \pi = -2y + \pi,\\ \theta_5 &= 4x - 4z = -2y - 2z,\\ \theta_6 &= 8x - 4z + \pi = -4y + \pi,\\ \theta_7 &= 8x - 2z = -4y + 2z,\\ \theta_8 &= -2z + \pi.\end{aligned}$$

In working out some of the equalities, we used the relations

$$6x = -4y, \quad 2\alpha_j = -2\alpha_{j-1} - 2\alpha_{j+1}.$$

These relations hold modulo 2π, which is all we care about. The first equation comes from (3–8). To give an example derivation, we will work out the derivations for θ_4 and θ_6:

$$\begin{aligned}4x - 2z &= 4x + 2x + 2y = 6x + 2y = -4y + 2y = -2y,\\ 8x - 4z &= 12x - 4x - 4z = -8y + (4y + 4z) - 4z = -4y.\end{aligned}$$

We want to eliminate x, because this is the approach that generalizes to the other words W'_n.

To compute the slope of a point, we divide its y-displacement by its x-displacement. We set

$$C_k = \sum_{j=1}^{k} \cos(\theta_j), \quad S_k = \sum_{j=1}^{k} \sin(\theta_j).$$

Then $\sigma_1 = S_4/C_4$ and $\sigma_2 = S_8/C_8$. Since σ_1 and σ_2 are both finite, the terms C_4 and C_8 never vanish. We compute that

$$\sigma_1 - \sigma_2 = \frac{2\sin(z)}{C_4 C_8}(\cos(z) - \cos(y)).$$

The condition $z \in (\pi/2, \pi)$ makes $\cos(z) < 0$. The condition $y \in (0, \pi/2)$ makes $\cos(y) > 0$. Hence $\sigma_1 - \sigma_2 < 0$, whence $\sigma_1 < \sigma_2$. □

This completes our proof that $N'_1 \subset O(B_1)$, the first tile in the second family.

3.9 The General Case

For N'_n we have the angle condition

$$(n + 2)x + 2y = \pi. \tag{3–15}$$

The proof of (3–9) works exactly the same way, with the same outcome. Armed with (3–9), we can use the same arguments as above to eliminate all the pairs of vertices except (b_3, a_{3n+6}) and (b_3, a_{4n+8}). Figure 14 shows the situation for $n = 2$.

Lemma 3.9 works in general with the following changes: Equation (3–10) becomes

$$\theta_1 + \theta_3 = \pi - 4 \times (n + 2)x - 3x = 8y - 3x = 5y + 3x.$$

Equation (3–12) becomes

$$\begin{aligned}\theta_2 + \theta_4 = 2 \times (n + 2)x - 3x &= 2(\pi - 2y) - 3(\pi - y - z)\\ &= 2\pi + y - 3z.\end{aligned}$$

In other words, we get the same equations! The rest of the proof is the same.

The analysis of the pair $(b_4, a_{4n+8}$ generalizes in the same way. In general, we consider the path of vectors

$$(b_{4n+11},\ b_{4n+10},\ a_{4n+10},\ a_{4n+9},\ a_{4n+8},\ b_{4n+8},\ b_{4n+7},\ a_{2n+6}, b_3).$$

The angle sequences we get are

$$\begin{aligned}
\theta_2 &= 2(n+2)x + \pi = -4y + \pi,\\
\theta_3 &= 2(n+2)x - 2z = -4y - 2z,\\
\theta_4 &= 2nx - 2z + \pi = -2y + \pi,\\
\theta_5 &= 2nx - 4z = -2y - 2z,\\
\theta_6 &= (4n+4)x - 4z + \pi = -4y + \pi,\\
\theta_7 &= (4n+4)x - 2z = -4y + 2z,\\
\theta_8 &= -2z + \pi.
\end{aligned}$$

As above, we will show the derivations for θ_4 and θ_6:

$$\begin{aligned}
2nx - 2z &= 2nx + 2x + 2y = (2n+2)x + 2y\\
&= -4y + 2y = -2y,\\
(4n+4)x - 4z &= (4n+8)x - 4x - 4z\\
&= -8y + 4y + 4z - 4z = -4.
\end{aligned}$$

The rest of the proof is the same.

4. THE VERIFICATION ALGORITHM

The publications list on my website has a link to a written list of the words $W_7, \dots, W_{221}$ and the polygons $P_7, \dots, P_{211}$. One can also see the complete list using McBilliards or the Java applet. See Section 7.

In this section we will explain how we verify computationally that

$$P_i \subset O(W_i), \quad i = 7, \dots, 221.$$

Here P_i is a given convex dyadic rational polygon, and $O(W_i)$ is the orbit tile of a word W_i. The basic algorithm works for indices $i = 30, \dots, 221$. These orbit tiles are contained in the interior of the parameter space Δ. After we describe the basic algorithm, we will explain how it is modified so as to handle the indices $i = 7, \dots, 29$. These indices correspond to orbit tiles that contain a segment on $\partial\Delta$.

Let us call a square in Δ whose sides are parallel to the coordinate axes and whose vertices have the form $x(\pi/2)$, where $x \in [0,1]$ is a dyadic rational, a *dyadic rational square*.

Our verification algorithm tries to produce a cover of P by dyadic squares $P \subset \bigcup Q_i$ such that $Q_i \subset O(W)$ for all i. To show that $Q \subset O(W)$, we need to show that all the associated defining functions f_{a_i,b_j} are positive on Q. We will sometimes write $f_{ij} = f_{a_i,b_j}$ for ease of notation.

In the first section we will explain how we verify the positivity of the defining functions. In the sections following the first one, we will explain our main algorithm.

4.1 Certificates of Positivity

Let Q be a dyadic rational square with center q and radius r. Here r denotes half the edge length of Q. Suppose that f is a defining function for a pair of vertices of the unfolding $U(W,T)$. There are two ways we try to certify that $f > 0$ on Q: the *gold* and the *silver*. The gold method is nicer.

4.1.1 The Gold Method.

Let $\nabla f = (f_x, f_y)$ be the gradient. From (2–10) we have

$$f_a = \Im(\overline{g}_a h + \overline{g} h_a), \quad a \in \{x, y\}. \tag{4–1}$$

We use (2–11) to get bounds on the second partial derivatives. Using the letters a and b to stand arbitrarily for x and y, we have bounds on the second derivatives:

$$|f_{ab}| \le F_{ab},$$

where

$$F_{xx} = \sum_k A_k^2 |J_k|, \quad F_{xy} = \sum_k A_k B_k |J_k|,$$
$$F_{yy} = \sum_k B_k^2 |J_k|.$$

We introduce the quantities

$$a_x = r(F_{xx} + F_{xy}), \quad a_y = r(F_{yx} + F_{yy}).$$

Finally, we define the rectangle

$$G(q,f) = [f_x(q) - a_x, f_x(q) + a_x] \times [f_y(q) - a_y, f_y(q) + a_y].$$

Here q is the center of Q.

It follows by integration that

$$\nabla f(x,y) \subset G(Q,f), \quad \forall (x,y) \in Q.$$

We say that f is *gold certified* if $G(Q,f)$ is disjoint from the coordinate axes in $\mathbb{R}^2$. This is to say that $G(Q,f)$ is contained in one of the standard quadrants in $\mathbb{R}^2$.

If f is gold certified, then there is some vertex v of Q such that throughout Q, the gradient ∇f is a positive linear combination of the edges of Q that emanate from Q. This means that $f(x,y) > f(v)$ for all $(x,y) \in Q$. Thus, if f is gold certified and $f(v) > 0$, then $f|_Q > 0$. We say that we have shown that $f|_Q > 0$ by the *gold method* if this situation obtains. Note that the gold method requires only a finite number of computations. The gold method works poorly if ∇f points nearly horizontally or vertically in Q.

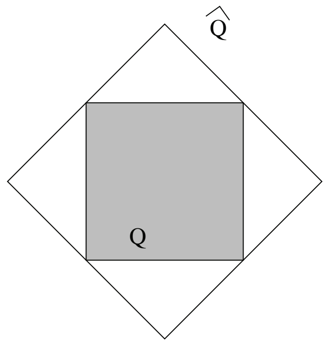

FIGURE 15. Two squares.

4.1.2 The Silver Method. Let $\widehat{Q}$ denote the square with the following property: Q is midscribed in $\widehat{Q}$, as shown in Figure 15. Note that $\widehat{Q}$ is not a dyadic rational, because its sides are not parallel to the coordinate axes. However. the vertices and center of $\widehat{Q}$ all have the form πx, where x is a dyadic rational.

We use all the same notation as in the previous section. We define the rectangle

$$S(q,f) = [f_x(q) - 2a_x, f_x(q) + 2a_x] \\ \times [f_y(q) - 2a_y, f_y(q) + 2a_y].$$

It follows from integration that

$$\nabla f(x,y) \subset S(Q,f), \quad \forall (x,y) \in \widehat{Q}.$$

We say that f is *silver certified* if $G(Q,f)$ is disjoint from the lines through the origin of slope ± 1. This is to say that $S(Q,f)$ is contained in one of images obtained by rotating the standard quadrants by 45 degrees.

If f is silver certified, then there is some vertex v of $\widehat{Q}$ such that throughout $\widehat{Q}$, the gradient ∇f is a positive linear combination of the edges of $\widehat{Q}$ that emanate from $\widehat{Q}$. This means that $f(x,y) > f(v)$ for all $(x,y) \in \widehat{Q}$. In particular, this is true for all $(x,y) \in Q$. Thus, if f is silver certified and $f(v) > 0$, then $f|_Q > 0$. We say that we have shown that $f|_Q > 0$ by the *silver method* if this situation obtains.

Remark 4.1. The silver method is not as nice as the gold method because $\widehat{Q}$ is much larger than Q. In practice, we mainly use the gold method. However, we can't get by entirely without the silver method. We need the silver method in the cases in which ∇f is nearly parallel to one of the coordinate axes.

Remark 4.2. The constant r in the formulas above has the form $r = \pi x/2$, where x is some dyadic rational number. When it comes time to do our rigorous computation, we will replace r by the larger $\widetilde{r} = 2x$ because it is a rational quantity. We will then work with the rectangles $\widetilde{G}(Q,f)$ and $\widetilde{S}(Q,f)$, which are defined as above, but with $\widetilde{r}$ in place of r. This replacement makes the functions a bit harder to certify, but helps us reduce the problem to an integer calculation.

4.2 An Inefficient First Try

As we mentioned in the introduction, a much faster computer would allow us to use a simpler verification algorithm. Here we describe a simple verification algorithm that is too slow to use on a typical computer in 2008, but easy to understand. Following this section, we will describe the algorithm we actually do use.

Let Q be a dyadic square and let W be a word. We say that W is *good* on Q if for every defining function f_{ij} we can prove that $f_{ij}|_Q > 0$ either by the gold method or by the silver method. If W is good on Q, then $Q \subset O(W)$.

Let

$$Q_0 = \left[0, \frac{\pi}{2}\right]^2.$$

For our algorithm we start with a list of squares having Q_0 as its sole member. At any point of the algorithm we have a list of dyadic rational squares. We let Q be the last square on the list. There are several options:

- If f is good on Q, we delete Q from our list and add it to our covering.
- If $Q \cap P = \varnothing$, then we delete Q from our list.
- If neither of the above is true, we replace Q on our list by the four squares obtained by dividing Q in half.

If our list ever becomes empty, then we have a covering of P by dyadic squares, each of which is contained in $O(W)$. This does the job. The problem with this algorithm is that it is too slow. We must evaluate all $O(n^2)$ defining functions for each square on the list. Our actual algorithm is similar to the one above, but enhanced so as to be much faster.

4.3 The Tournament

As above, W is a fixed word. Let Q be a dyadic rational square. A *player list* for Q will be a pair (A,B), where both A and B are lists of indices. We think of A as being a list of some distinguished a-vertices and B as being a list of some distinguished b-vertices. We say that lists $i < j \in A$ are *adjacent* if there is no index $k \in A$ such that

$i < j < k$. In this section we will make some definitions for A and at the end make the same definitions for B.

We define an A-*function* to be a defining function associated to (a_i, a_j), where i and j are adjacent indices in A. We say that a vertex $i \in A$ is an A-*loser* if one of the following two situations (when applicable) obtains:

- Let $j > i$ be the index adjacent to i. Let f be an A-function for the pair (a_i, a_j). Then $-f_Q$ can be certified positive.
- Let $j < i$ be the index adjacent to i. Let f be an A-function for the pair (a_i, a_j). Then f_Q can be certified positive.

One of the situations is not applicable if i is the first or last index in A. If i is the only index in A, then neither situation is applicable.

If $i \in A$ is an A-loser, this means that there is another index $j \in A$ such that $a_i \uparrow a_j$ throughout Q. In this case, any result $a_j \uparrow b_k$ in Q automatically implies that $a_i \uparrow b_k$ in Q. If i is not an A-loser, we call i an A-*survivor*.

We make all the same definitions for the B list, except that we reverse the signs. That is, we say that a vertex $i \in B$ is a B-*loser* if one of the following two situations (when applicable) obtains:

- Let $j > i$ be the index adjacent to i. Let f be a B-function for the pair (b_i, b_j). Then f_Q can be shown to be positive using either the gold or silver method.
- Let $j < i$ be the index adjacent to i. Let f be an B-function for the pair (b_i, b_j). Then $-f_Q$ can be shown to be positive using either the gold or silver method.

We call the following elimination process a *round* (of a tournament): We consider in order all the A-functions $f_1, \dots, f_m$. We form a new list A' consisting of the A-survivors. We call A *stable* (with respect to Q) if $A' = A$. If A is not stable, we form a sequence $A \supset A' \supset A'' \supset \cdots$ until the list stabilizes. We call this process the A-*tournament* on Q. We call the indices of the final list the A-*winners*. We carry out the same processes for the B list.

4.4 The Improved Algorithm

We start our algorithm with the list consisting of the triple (Q_0, A_0, B_0), where $Q_0 = [0, \pi/2]^2$ as above, and $A_0 = B_0 = \{1, 2, 3, \dots, k\}$ is the complete list of indices. Here k is half the length of W. During the algorithm we maintain a list of triples like this. At any stage we consider the last triple (Q, A, B) on the list.

If $Q \cap P = \varnothing$, we discard (Q, A, B) from our list and move on. Otherwise, we proceed as follows:

- We perform the A-tournament and B-tournament to produce triples (Q, A^*, B^*), where A^* consists of the A-winners and B^* consists of the B-winners.
- For each index $(i, j) \in A^* \times B^*$ we try to show, using the gold and silver methods, that $f_{ij}|_Q > 0$. If we succeed for every pair, then we add Q to our covering of P.
- Otherwise, we delete (Q, A, B) from our list, then replace by the four triples (Q_j, A^*, B^*), where Q_1, Q_2, Q_3, Q_4 are obtained by bisecting Q.

If the list becomes empty, then we have produced a covering of P by dyadic squares each of which is contained in $O(W)$. This is justified by the following result.

Lemma 4.3. *If Q is added to our cover then $Q \subset O(W)$.*

Proof: Let $(i, j) \in A_0 \times B_0$ be arbitrary indices. There is a nested sequence of squares $Q_0 \supset Q_1 \supset \cdots \supset Q_n = Q$ together with a sequence of indices $i = i_0, \dots, i_n = i'$ such that $Q \subset Q_k$ and $a_{i_k} \uparrow a_{i_{k+1}}$ for all k. Moreover, $i' \in A^*$. The same goes for j in place of i. Therefore, on Q we have $a_i \uparrow a_{i'} \uparrow b_{j'} \uparrow b_j$. □

We point out three nice features of our algorithm:

- If $P \subset P' \subset O(W)$ and the algorithm works for both P and P', then the covering produced for P' is obtained from the covering produced for P just by adding some squares.
- The gold and silver certificates are inherited: If a defining function f is gold/silver certified on a square Q, it is also gold/silver certified on a subsquare Q' of Q. We don't need to recompute the bounds.
- If Q is one of the squares in our covering, then there is a canonical sequence of squares $Q_0, \dots, Q_n = Q$, where Q_{k+1} is one of the four squares in the bisection of Q_k for all k. The presence of Q in our cover can be completely explained by looking at what happens in $Q_0, \dots, Q_n$. We don't have to look at other "branches" of the algorithm. As we will explain in Section 7, McBilliards exploits this feature to produce a nice way for the (tireless) reader to inspect the operation of the algorithm piece by piece.

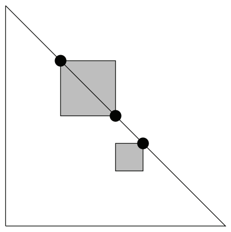

FIGURE 16. Exceptional dyadic squares.

4.5 Exceptional Pairs

Here we explain how to modify our algorithm so that it works when P_i and $O(W_j)$ both have a segment in common with the right-angled line in $\partial\Delta$

We shall say that a pair of vertices (a_i, b_j) is *exceptional* if the associated defining function vanishes along the right-angled line. We call such a defining function *exceptional* as well. For any word W there is a list A of a-vertices of $U(W,*)$ and a list B of b-vertices of $U(W,*)$ such that the set of exceptional pairs of vertices is precisely $A \times B$. For the words $W_{30}, \ldots, P_{229}$, the lists A and B are typically (though not always) empty. However, the polygons $P_{30}, \ldots, P_{221}$ are all (very) disjoint from the right-angled line, and so the lists A and B do not concern us. For the words $W_7, \ldots, W_{29}$, the lists A and B are always nonempty, and as we mentioned above, the polygons $P_7, \ldots, P_{29}$ always have an edge on the right-angled line. For this reason, we need to understand what happens with the defining functions associated with vertices in $A \times B$. It is hard to deal computationally with these defining functions, because they take arbitrarily small positive values on points in the polygons.

We shall say that a dyadic square is *exceptional* if it has one or two vertices on the right-angled line and at least one vertex in the parameter space Δ of obtuse triangles. Figure 16 shows the two kinds of exceptional dyadic squares. Let Q be an exceptional dyadic square and let f be an exceptional defining function. We say that f is *certified* on Q if the gold method shows that ∇f is contained in a quadrant throughout Q. We also insist that ∇f point into the obtuse parameter space. In this situation, the axis of the quadrant containing ∇f is perpendicular to the right-angled line, and $f > 0$ on the portion of Q that lies in Δ.

When we run our algorithm for the indices $i = 7, \ldots, 29$, we first isolate the lists A and B. We then run the algorithm as in Section 5, except that we automatically "pass" any exceptional defining function in the playoffs if the dyadic square in question is exceptional and the defining function is certified on the square. If the algorithm halts, we have a covering of P_i by a union of dyadic squares and dyadic triangles, each of which is contained in $O(W_i)$.

Now we explain how we find special pairs. Each of the exceptional words is a special palindrome. Hence, the first and last edges of $U(W,*)$ are always vertical. This allows us to predict the turning angles of the other edges solely from their turning pairs. Also, we have to worry only about the exceptional pairs involving vertices on the left half of the unfolding.

Figure 17 shows the example of W_{11}. We have not labeled the vertices in Figure 17, but the reader can easily deduce the relevant labels from the labeling scheme we have described in Section 2.1. In this case, the only exceptional pair of vertices is (a_5, b_1).

The vertices a_5 and b_1 are joined by two edges of type 3. The union of these two edges has a line of bilateral symmetry. Call this line Λ_{51}. The turning pair for Λ_{51} is $(-2,-2)$. Modulo π, the angle between the first edge, which is always vertical, and Λ_{51} is $-2x - 2y$. But $x + y = \pi/2$ on the right-angled line. Hence Λ_{51} is vertical for any unfolding with respect to a right triangle. Hence $a_5 \updownarrow b_1$ for all points on the right-angled line.

Remark 4.4. It is not actually necessary for us to show explicitly that we have obtained an exhaustive list of exceptional pairs. We just have to run the modified algorithm and see that it halts, given the exceptional pairs we have singled out. Given that the algorithm is based on finite-precision (though exact) arithmetic, another exceptional pair would cause the algorithm to get hung up, producing a list of ever smaller dyadic squares converging to the right-angled line.

4.6 Case-by-Case Analysis

We recommend that the reader read this part of our analysis while using McBilliards or the accompanying Java applet. The reader can survey all the unfoldings we discuss and verify that the analysis is correct.

4.6.1 The Easy Cases. With six exceptions, the words $W_7, \ldots, W_{29}$ have the same analysis as W_{11}. That is, they have a single exceptional pair of vertices (on the left), and the spine connecting these vertices has bilateral symmetry. In all these cases, the same analysis as for W_{11} works here word for word. Here we list these cases, together with the exceptional pairs. Referring to the example in Figure 17, the exceptional pair for the word W_{11} is (a_5, b_1). We denote this by $(11; 5, 1)$. Here are the easy cases:

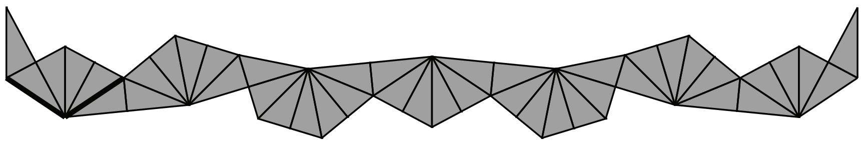

FIGURE 17. Unfolding for W_{11}.

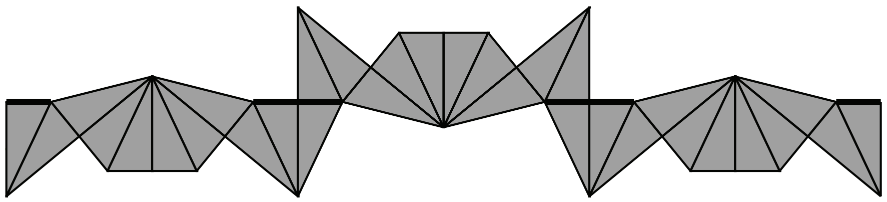

FIGURE 18. Unfolding for W_8.

$$(7;5,1),\ (9;5,10),\ (10,5,1),\ (11;5,1),\ (12;8,13),$$
$$(13;1,11),\ (14;5,1),\ (17;5,1),\ (19;19,3),\ (20;5,1),$$
$$(22;1,23),\ (24;27,5),\ (25;5,33),\ (26;42,31),$$
$$(27;38,7),\ (28;5,45),\ (29;48,11).$$

Notice that the pair (a_5, b_1) occurs quite often. In all cases, the path connecting the vertices in the exceptional pair has the same bilateral symmetry as in Figure 17, and the line Λ of bilateral symmetry has turning pair either $(2,2)$ or $(-2,-2)$ and hence is vertical when the unfolding is done with respect to a right triangle.

4.6.2 The Case W_8.

Figure 18 shows $U(W_8, T)$ for some T. We have highlighted eight line segments that are all horizontal when x lies on the right-angled line. The turning pairs for these segments are all of the form (k,k) for $k \in \{\pm 1, \pm 3, \pm 5\}$. Restricting our attention to the left-hand side, we see that the exceptional sets are $A = \{1,2,4,5\}$ and $B = \{8\}$. These are exactly the ones we single out when we run our algorithm.

4.6.3 The Case W_{21}.

For W_{21} we have $A = \{22, 23\}$ and $B = \{4\}$. In this case, the pair (a_4, b_{22}) has the same kind of bilateral symmetry as for the easy cases. Hence $a_4 \updownarrow b_{22}$ for any unfolding with respect to a right triangle. Finally, the turning pair for the edge connecting b_{22} and b_{23} is $(1,1)$. Hence, this edge is horizontal for any unfolding with respect to a right triangle.

4.6.4 The Cases W_{16} and W_{23}.

For W_{16} we have the lists $A = \{4\}$ and $B = \{7, 8, 14, 15\}$. There is an edge of $U(W_{16}, *)$ connecting a_4 and b_7, and this edge has turning pair $(1,1)$. Hence (a_4, b_7) is an exceptional pair. There is an edge connecting b_7 and b_8, and this edge has turning pair $(5,5)$. Hence $b_7 \updownarrow b_8$ on the right-angled line. Hence (a_4, b_8) is an exceptional pair. There is a path connecting b_8 to b_{14} that has bilateral symmetry. The line of symmetry contains an edge whose turning pair is $(2,2)$. Hence $b_8 \updownarrow b_{14}$ on the right-angled line. Hence (a_4, b_{14}) is an exceptional pair. Finally, there is an edge connecting b_{14} to b_{15} that has turning pair $(-1,-1)$. Hence (a_4, b_{15}) is an exceptional pair.

For W_{23} we have $A = \{12, 13, 29, 30\}$ and $B = \{9\}$. There is an edge connecting b_9 to a_{12}, and this edge has turning pair $(-5,-5)$. Hence (a_{12}, b_9) is an exceptional pair. The other three pairs are shown to be exceptional as for W_{16}.

4.6.5 The Cases W_{15} and W_{18}.

For W_{15} we have $A = \{1,2,4\}$ and $B = \{8, 9, 11, 12, 13\}$. The same arguments as in the previous section show that a_1, a_2, a_4 all lie at the same height when the unfolding is done with respect to a right triangle. The same goes for $b_8, b_9, b_{11}, b_{12}, b_{13}$. Finally, a_4 and b_8 are connected by an edge whose turning angle is $(5,5)$. Hence (a_5, b_8) is an exceptional pair. Hence all the pairs listed are exceptional.

For W_{18} we have $A = \{1,2,4,5,6,8,9\}$ and $B = \{14,16,17,18\}$. This case is essentially the same as the case of W_{15}.

5. SPECIAL CASES

5.1 The Regions of Interest

Figure 19 shows a fairly accurate picture of the parameter space Δ of obtuse triangles, as well as the regions $P_1,\dots,P_6$ discussed in the introduction. The dotted lines indicate that P_1 and P_4 continue "behind" P_2. Equation (1–3) is easier to check computationally if these two polygons continue as indicated. As we mentioned in the introduction, P_1 and P_2 are just dummy triangles. They don't correspond to periodic billiard paths in triangles. We dealt with P_3 in Section 3. We dealt with $S' = \Delta_{100} - P_3 - P_4 - P_5 - P_6$ in Section 4. Here we deal with P_4, P_5, P_6. (Actually, we already dealt with P_6 in [Schwartz 06]. Here we just recall what we did.)

We introduce the notation

$$\begin{vmatrix} n_1 & k_1 \\ n_2 & k_2 \end{vmatrix} = \frac{\pi}{2} \times \left(\frac{k_1}{2^{n_1}}, \frac{k_2}{2^{n_2}}\right)$$

An example is

$$\begin{vmatrix} 2 & 1 \\ 2 & 3 \end{vmatrix} = \left(\frac{\pi}{8}, \frac{3\pi}{8}\right).$$

To describe a dyadic polygon, we will list the vertices. For instance,

$$P_4: \quad \begin{vmatrix} 7 & 63 \\ 7 & 65 \end{vmatrix} \begin{vmatrix} 7 & 65 \\ 7 & 63 \end{vmatrix} \begin{vmatrix} 7 & 63 \\ 7 & 63 \end{vmatrix},$$

$$P_5: \quad \begin{vmatrix} 12 & 1641 \\ 12 & 2455 \end{vmatrix} \begin{vmatrix} 12 & 1637 \\ 12 & 2455 \end{vmatrix} \begin{vmatrix} 12 & 1637 \\ 12 & 2459v \end{vmatrix},$$

$$P_6: \quad \begin{vmatrix} 10 & 345 \\ 10 & 679 \end{vmatrix} \begin{vmatrix} 12 & 1380 \\ 12 & 2712 \end{vmatrix} \begin{vmatrix} 12 & 1352 \\ 12 & 2740 \end{vmatrix} \begin{vmatrix} 9 & 169 \\ 9 & 343 \end{vmatrix}.$$

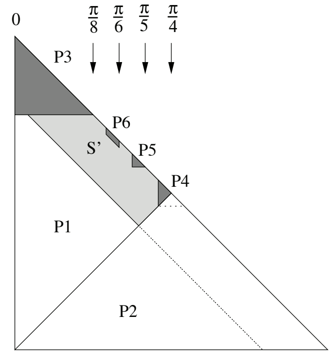
0
$\frac{\pi}{8}$ $\frac{\pi}{6}$ $\frac{\pi}{5}$ $\frac{\pi}{4}$
P3
P6
S'
P5
P4
P1
P2

FIGURE 19. Regions in the parameter space.

Here P_k is a tiny region, one of whose boundary components contains the point p_k from (1–1).

5.2 Covering P_6

Let P_6' denote the region

$$\left\{ (x,y) \in \Delta : \left|x - \frac{\pi}{6}\right| < \frac{1}{175}, \ \left|(x+y) - \frac{\pi}{2}\right| < \frac{1}{400\sqrt{2}} \right\}.$$

In [Schwartz 06] we covered P_6' by a union of two infinite families of orbit tiles. A straightforward computation shows that $P_6 \subset P_6'$.

For the record, we describe the main result here. We used two families of orbit tiles, $\{O(Y_k)\}_{k=8}^{\infty}$ and $\{O(Z_k)\}_{k=8}^{\infty}$. The words Y_k are defined for all $k \geq 1$, and the words Z_k are defined for all $k \geq 0$, but we took k fairly large to get better estimates.

We first define the Y family. Let

$$A = 3123, \quad B_1 = 23213, \quad B_2 = 23123,$$
$$C_1 = 213123, \quad C_2 = 123123.$$

We have $Y_k = 2y_k 2y_k^{-1}$. For odd indices we have

$$y_{2k+1} = AB_1(B_2B_1)^k C_1 (B_1B_1)^k, \quad k = 0,1,2,\dots.$$

For even indices we have

$$y_{2k+2} = AB_1(B_2B_1)^k C_2 (B_1B_2)^{k+1}, \quad k = 0,1,2,\dots.$$

Now we define the Z family. Define

$$A = 123, \quad B = 231, \quad C = 32, \quad D = 213.$$

Next define E_0 to be the empty word and

$$E_1 = DD, \quad E_2 = DAAD, \quad E_3 = DADDAD,$$
$$E_4 = DADAADAD,$$

and so on. Then $Z_k = 3z_k 3z_k^{-1}$, where

$$z_k = ABC3E_kABC$$

(the digit 3 included in the equation is deliberate).

Figure 20 shows the tiles $O(Y_1),\dots,O(Y_4)$. The "tips" of these tiles converge to the point $P(\pi/6)$. The largest tile $O(Y_1)$ obscures the other tiles. The left vertical gray line indicates the set $y = \pi/6$, and the right vertical gray line indicates the set $y = \pi/5$.

Figure 21 shows how the tiles $O(Y_1),\dots,O(Y_4)$ and $O(Z_0),\dots,O(Z_3)$ interlock and suggests how the neighborhood of $P(\pi/6)$ is filled.

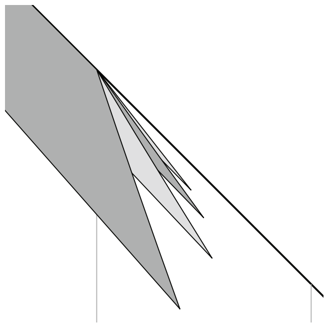

FIGURE 20. The tiles $O(Y_k)$ for $k = 0, 1, 2, 3, 4$.

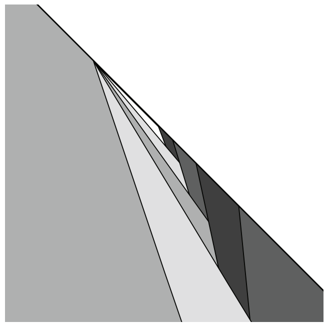

FIGURE 21. Interlocking tiles cover P_6.

5.3 Covering P_5

Let H_+ denote the half-plane given by $x \geq \pi/5$ and let H_- denote the half-plane given by $x \leq \pi/5$. We consider the words

$$F = 31232313123132323132131322321,$$
$$G = 1323123231323213213123231323213123123231323\ 21323.$$

We will show that

$$P_5 \cap H_+ \subset O(F), \quad P_5 \cap H_- \subset O(G).$$

The basic idea is to check that the verification algorithm described in the previous section halts when we ignore certain additional pairs of vertices. Then, at the end, we intervene and analyze the pairs of vertices we ignored.

5.3.1 Dealing with F. In terms of our listing, we have $F = W_7$, but $P \cap H_+$ is not contained in P_7. Indeed, $P \cap H_+$ shares a vertex with $O(F)$, and we have to work harder. From the list in Section 4.6, we see that the pair (a_5, b_1) is exceptional. When we also ignore the pairs (a_5, b_5) and (a_5, b_6), we find that our verification algorithm produces a covering of $P_5 \cap H_+$. We already know from our analysis in the previous section that $f_{51} > 0$ on P_5. The point here is that the relevant line of bilateral symmetry has turning pair $(-2, -2)$, and hence this line has positive slope throughout P_5. This positive slope forces a_5 to lie above b_1.

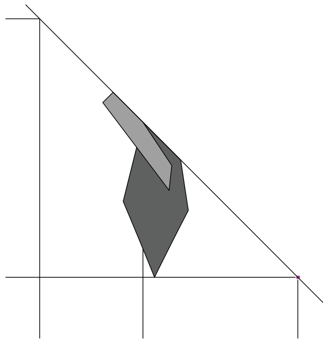

FIGURE 22. Covering P_5.

It remains to show that f_{55} and f_{56} are positive on $P_5 \cap H_+$. Figure 23 shows a picture of $U(F, T)$ when T is the right triangle corresponding to the point $p_5 \in P_5$.

The edge connecting a_5 and b_6 has turning pair $(-4, 1)$. Points $(x, y) \in P_5 \cap H_+$ have the form

$$x = \pi/5 + \epsilon, \quad y = 3\pi/10 - \epsilon - \delta.$$

Here ϵ and δ are numbers much smaller than $\pi/10$. The turning angle of the edge connecting a_5 to b_5 is therefore

$$-\pi/2 - 3\epsilon - \delta.$$

This line has negative slope throughout $P_5 \cap H_+$, and hence $a_5 \uparrow b_5$ there.

The vertices a_5 and b_6 are connected by a path of length 2 whose line of bilateral symmetry has turning pair $(-3, 2)$. The corresponding turning angle is

$$-\epsilon - 2\delta.$$

This line has positive slope for $(x, y) \in P_5 \cap H_+$, and hence $a_5 \uparrow b_6$ throughout $P_5 \cap H_+$.

5.3.2 Dealing with G. In terms of our listing, we have $G = W_{13}$. However, $P_5 \cap H_-$ is not a subset of P_{13}, so we have to do more work. When we omit the pairs (a_1, b_{11}), (a_1, b_{12}), and (a_1, b_{13}), our algorithm produces a covering of $P_5 \cap H_-$. It remains to show only that the defining functions associated with these pairs are positive on $P_5 \cap H_-$. The function $f_{1,11}$ is positive on P_5 for the symmetry reason we discussed in the previous section.

Here we explain a proof that works for all three defining functions at once. When we run our algorithm, each of these omitted defining functions gets certified on a

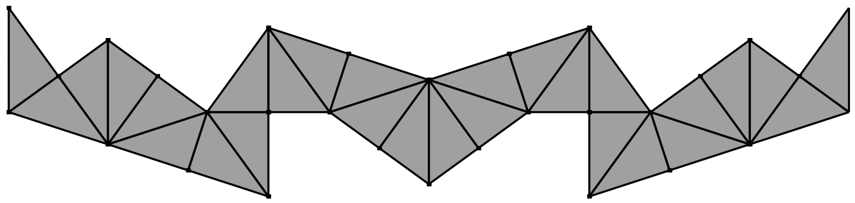

FIGURE 23. The unfolding for F.

dyadic square that contains P_5. We just check that in all three cases, the quadrant that contains the gradients is the negative quadrant. Hence ∇f_{1j} lies in the negative quadrant. Also, these functions all vanish at p_5. Every $p \in P \cap H_-$ can be joined to p_5 by a path that points from p_5 into the negative quadrant. Hence $f_{1j} > 0$ on $P \cap H_-$, as desired. Hence $P \cap H_- \subset O(G)$, as desired.

5.4 Covering P_4

Figure 24 shows a partition of $P_4 \cap \Delta$ into five regions. The regions c, d_1, d_2 are meant to be open. The segments e_1 and e_2 are meant to be open line segments. The four solid lines through p_4 have slope $-1, -1/3, 0, \infty$. The dotted line is contained in $\partial\Delta$ and bisects P_4.

Since we are taking $x \le y$ in Δ, only the left half of P_4 lies in Δ. We will use the following words:

$$C = (1232313)^2,$$
$$D_1 = 231323123231323123232132313232132313,$$
$$D_2 = 23132313231232313231323123232132313231323213\\ 23132313,$$
$$E_1 = 12323132312323213231323132321323132313231323,$$
$$E_2 = 12323132313231232321323132313231323213231323\\ 1323132313231323.$$

The left-hand side of Figure 25 shows a closeup of $O(C)$, $O(D_1)$, and $O(D_2)$. Note that $O(C)$ slops over the boundary of $P_4 \cap \Delta$. The boundary here is contained in the line through p_4 of slope 1. (See the dotted line in Figure 24.) The large tile $O(C)$ is not completely shown. The union of these three tiles covers all of $P_4 \cap \Delta$ except for two line segments. These two line segments are then covered by $O(E_1)$ and $O(E_2)$, as shown on the right-hand side of Figure 25.

We will prove that $z \subset O(Z)$, for $z \in \{c, d_1, d_2, e_1, e_2\}$.

5.4.1 Dealing with C.

In terms of our listing, we have $C = W_{30}$. Let T be the triangle corresponding to the point p_4, the right isosceles triangle.

The defining function f_{ij} vanishes at p_4 when $i \in \{1, 2\}$ and $j \in \{4, 5\}$. When we run our algorithm with these vertex pairs excepted, it produces a cover of P by four squares. Thus, all the defining functions but the excepted ones are positive on P. The algorithm in this case does not also verify that the gradients of the excepted functions lie in the negative quadrant; this is not true for f_{14} and f_{25}.

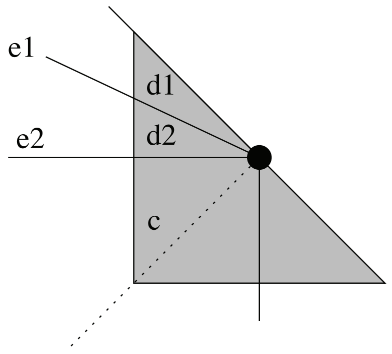

FIGURE 24. Dividing up P_4.

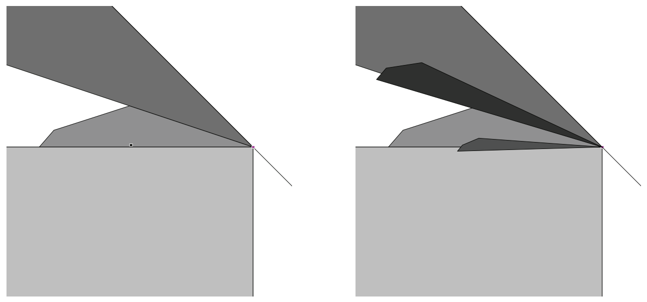

FIGURE 25. Covering P_4.

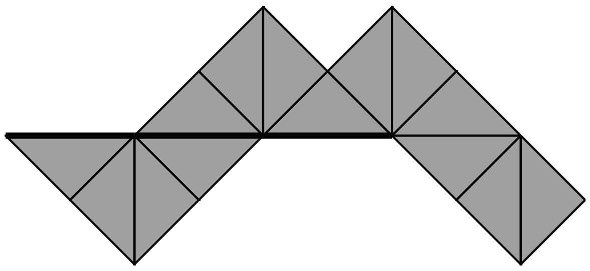

FIGURE 26. $U(C,T)$.

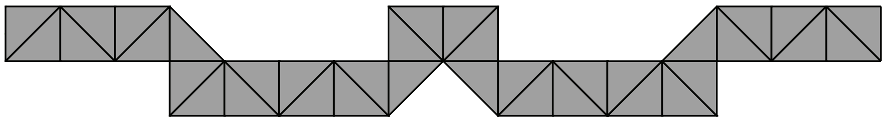

FIGURE 27. $U(D_1,T)$.

In dealing with the four exceptional defining functions, we first compute that

$$|f_{xx}|,\ |f_{xy}|,\ |f_{yy}| \le 2^6,$$

in all cases. We also note that P_4 is contained in a square of radius 2^{-6}. Hence, both $\partial_x f$ and $\partial_y f$ vary by at most two units throughout P_4.

Here is the formula for f_{15}:

$$\begin{array}{rrcrrr} 0 & 1 & \quad & 0 & 1 & (-1) \\ 4 & 1 & & 4 & 1 & \\ 4 & -3 & & 4 & -3 & \\ 0 & -3 & & & & \end{array}$$

We compute that $\nabla f_{15}(p_4) = (-8,-8)$. Hence ∇f_{15} lies in the negative quadrant throughout P_4. Hence $f_{15} > 0$ on the interior of c.

A similar computation to the one above gives $\nabla f_{24}(p_4) = (-8,-8)$. Hence $f_{24} > 0$ on c.

Here is the formula for f_{14}:

$$\begin{array}{rrcrrr} 0 & 1 & \quad & 0 & 1 & (-1) \\ 4 & 1 & & 4 & 1 & \\ 4 & -3 & & & & \\ 0 & -3 & & & & \end{array}$$

We compute that f_{14} vanishes identically along the line $y = \pi/4$. Also, we compute that $\nabla f_{14}(p_4) = (0,-16)$. Hence ∇f_{14} has positive y-coordinate throughout P_4. Hence $f_{14} > 0$ on c.

The calculation for f_{25} is just like the one for f_{14}, but with the roles of x and y switched. Hence $f_{25} > 0$ on c.

In summary, all (a,b) defining functions are positive on c. We conclude that $c \subset O(C)$.

5.4.2 Dealing with D_1.

In terms of our listing, $D_1 = W_9$. Let T be the right-angled isosceles triangle, as above.

Taking i and j on the left half of the unfolding (Figure 27), we see that the defining function f_{ij} vanishes at p_4 if and only if

$$i \in \{5,6,7,8\} \quad \text{and} \quad j \in \{1,2,3,4,10\}.$$

(The center point by convention counts as a vertex on the left half.)

When we run the algorithm with these pairs excepted, it produces a covering of P_4 by three squares. Once again, the algorithm here does not verify anything about the gradients of the exceptional defining functions.

Reflection in a certain edge e swaps a_6 and a_8. The turning pair for e is $(2,2)$. Since the leftmost edge stays vertical for all points in the parameter space, e has negative slope throughout P_4. Hence $a_6 \uparrow a_8$ throughout P_4. This eliminates a_6 from consideration.

Figure 28 shows $U(D_1,T')$, where T' is a triangle corresponding to a point of Δ between e_1 and the right-angled line. (This point isn't actually in d_1, because such points give rise to a picture that looks almost identical to Figure 27; we wanted to show the difference dramat-

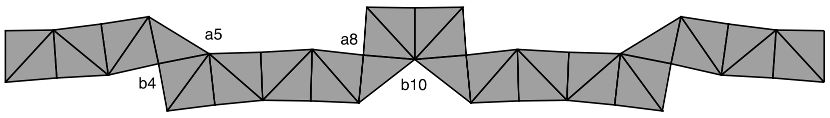

FIGURE 28. $U(D_1, T')$.

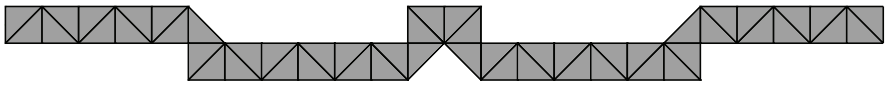

FIGURE 29. $U(D_2, T)$.

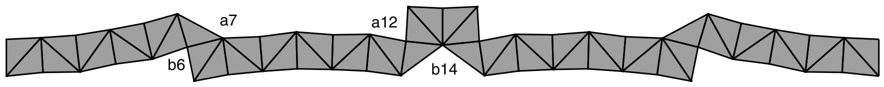

FIGURE 30. $U(D_2, T')$.

ically.) Figure 28 serves as a reality check to the arguments we give below.

The point a_6 is connected to a_7 by an edge whose turning pair is $(0, 2)$. As long as $y < \pi/4$, this edge has positive slope and $a_7 \uparrow a_6$. This condition holds in d_1. This eliminates $i = 7$ from consideration. Similar arguments show that $b_2 \uparrow b_1$, $b_3 \uparrow b_2$, and $b_3 \uparrow b_4$ throughout d_1. All in all, we have only to deal with the four defining functions f_{ij}, where $i \in \{5, 8\}$ and $j \in \{4, 10\}$. Here is the analysis:

- Points a_8 and b_4 are swapped by reflection in an edge whose turning pair is $(1, 3)$. This edge has positive slope throughout the interior of d_1, and vanishes on e_1, the line of slope $-1/3$ through p_4. Hence $a_8 \uparrow b_4$ throughout d_1. Hence $f_{84} > 0$.

- Points a_5 and b_{10} are swapped by reflection in an edge whose turning pair is $(2, 2)$. Hence $f_{5,10} > 0$ on d_1.

- Points b_4 and a_5 are connected by an edge whose turning pair is $(-2, 4)$. This edge has positive slope in d_1. Hence $f_{54} > 0$ in d_1.

- Points a_8 and b_{10} are connected by an edge whose turning pair is $(0, 2)$. This line has negative slope in d_1. Hence $f_{8,10} > 0$ in d_1.

This takes care of all the cases. Hence $d_1 \subset O(D_1)$.

5.4.3 Dealing with D_2. In terms of our listing, $D_2 = W_{87}$. The analysis of D_2 is almost identical to that of D_1. We will omit most of the details, but illustrate the main ideas with pictures. Figure 29 shows $U(D_2, T)$.

Figure 30 shows $U(D_3, T')$. Here T' is a triangle corresponding to a point that lies between the lines e_1 and e_2. (We have gone outside d_2 to get a more dramatic picture.)

When we except all the index pairs entailed by Figure 29, our algorithm produces a covering of P_4 by four squares. Using the turning pair arguments, as for D_1, we eliminate all the indices except $i \in \{7, 12\}$ and $j \in \{6, 14\}$. Figure 30 illustrates the signs of the slopes of the lines relevant to the analysis of the four remaining defining functions. These four defining functions have the same analysis as for D_1.

5.4.4 Dealing with E_1. In terms of our listing, $E_1 = W_{107}$. Recall that e_1 is the intersection of the line of

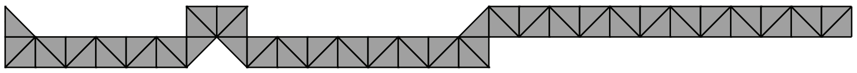

FIGURE 31. $U(E_1, T)$.

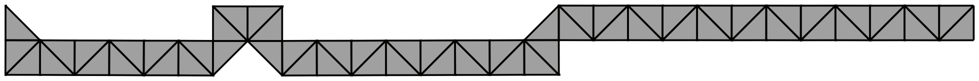

FIGURE 32. $U(E_2, T)$.

slope $-1/3$ through p_4 with P_4. Figure 31 shows a picture of $U(E_1, T)$. When we run our algorithm with all the excepted vertices, it produces a covering of P_4 by 47 squares. We also check, during the algorithm, that ∇f has positive dot product with the vector $(-3, 1)$ throughout P_4 whenever f is an exceptional defining function. This shows that all the exceptional defining functions are negative on e_1.

Hence $e_1 \in O(E_1)$.

Remark 5.1. Our gradient check is just a small tweak of the silver method. We compute ∇f, then add all the error bounds coming from the second partials, and check that the entire "error box" makes positive dot product with $(-3, 1)$.

5.4.5 Dealing with E_2. In terms of our listing, $E_2 = W_{85}$. Recall that e_2 is the intersection of the horizontal line through p_4 with P_4. Figure 32 shows a picture of $U(E_2, T)$.

When we run our algorithm with all the excepted vertices, it produces a covering of P_4 by 29 squares. We also check during the algorithm that $\partial_x f < 0$ throughout P_4 whenever f is an exceptional defining function. This shows that all the exceptional defining functions are negative on e_2. Hence $e_2 \in O(E_2)$.

6. COMPUTATIONAL DETAILS

6.1 The Covering Problem

Here we explain how we verify (1–3). Let P_j be one of the polygons on our list. Let e be an edge of P. We say that e is *good* if

$$e - \partial\Delta \subset \bigcup_{i \neq j} P_i. \tag{6–1}$$

In case $e \in \partial\Delta$, this condition is vacuous. We say that P_j is *good* if every edge of P_j is good.

Lemma 6.1. *We have $\Delta \subset \bigcup P_j$, provided that every P_j is good.*

Proof: If Δ is not covered by our polygons, then $\Delta - \bigcup P_j$ contains some open set U, and some point of ∂U is contained in some edge e of some P_j. But then e is not good. □

To make our problem easier, we scale all our polygons by the constant $2^{27}/\pi$. The result is that all the coordinates of all the polygons are positive integers between 0 and 2^{23}. Also, given the comments at the beginning of Section 2.7, we know that all the coordinates are divisible by 2^9. This fact is useful because we sometimes want to subdivide our edges in half a few times, while retaining the property that the break points are integers. We now are left with the problem of showing that a certain convex integer triangle is covered by 221 other convex integer polygons.

6.1.1 The Algorithm. Let S be some segment in the plane whose endpoints are integers. We call S an *integer segment*. We say that S is *admissible* if the midpoint of S also has integer coordinates. In this case, the two segments S_1 and S_2 formed by bisecting S are also integer segments.

Let e be an edge of P_i. To show that a given edge e is covered by our polygons, we perform the following algorithm. We start with a list of edges whose sole member is e. At any stage of the algorithm we have a finite list of integer segments. We consider the last segment S on the list.

- If we can show that $S \subset P_j$ for some $j \neq i$, then we omit S from our list. Then we continue.
- If S is admissible and we cannot show that $S \subset P_j$ for some $j \neq i$, then we omit S from our list and append S_1 and S_2 to the list. Then we continue.
- If S is not admissible and we cannot show that $S \subset P_j$ for some $j \neq i$, then we fail.
- If the list becomes empty, we have succeeded in showing that e is good.

The main step in our algorithm involves showing that an integer segment is contained in an integer convex polygon. This problem in turn boils down to checking that each of the endpoints of the segment is contained in the polygon. Showing that an integer point z is contained in an integer polygon P is an integer calculation. We just check the orientations of all the triangles obtained by coning the edges of P to z and see that they all agree. This calculation is done entirely in $\mathbb{Z}$ and produces integers that have roughly three times as many digits as the coordinates of z and P. We implement our algorithm in Java, using the BigInteger class. We discuss this in the next section. The interested reader can see and interact with the cover using McBilliards. In particular, one can rerun our algorithm, either one polygon at a time or all at once.

6.2 BigIntegers and BigIntervals

We wrote McBilliards in Java. The Java programming language has a class called BigInteger. A BigInteger is an integer with an "arbitrary" number of base-10 digits. Here "arbitrary" means "subject to the memory limitations of the machine." Once two BigIntegers are defined, they can be added, subtracted, multiplied, and even exponentiated. If the process of computing the resulting quantity does not exhaust the memory of the machine, then the result is correct. It would probably take integers billions of digits long to exhaust the memory of the machine. In our case we work with integers all of which have fewer than two hundred digits. For this reason, we are convinced that the basic arithmetic operations of the BigInteger class work without fail on the numbers we supply.

Our basic method is to convert all our calculations into integer calculations and then use BigIntegers to get the calculations exactly right. Our trick is to multiply the naturally computed quantities of interest to us by a huge integer, namely 2^{106}, and then trap these quantities inside an interval of BigIntegers. We then perform a calculation using BigInteger arithmetic, and in the end produce an interval of BigIntegers that contains 2^{106} times the quantity of interest to us.

The only real-valued functions we compute are the ones in (2–10) and (2–11). Once we have these quantities, we make some further algebraic manipulations, as discussed in connection with the gold and silver methods of Section 5. However, once we have finished with (2–10) and (2–11), we have our intervals of BigIntegers, and then we manipulate them as discussed below.

We define a *BigInterval* to be a pair (L, R) of BigIntegers with $L \leq R$. There are several basic operations that we can perform on these intervals:

$$(L_1, R_1) + (L_2, R_2) = (L_1 + L_2, R_1 + R_2),$$
$$(L_1, R_1) - (L_2, R_2) = (L_1 - R_2, L_2 - R_1),$$
$$(L_1, R_1) \times (L_2, R_2) = (L_3, R_3),$$

where

$$L_3 = \min(L_1L_2, L_1R_2, L_2R_1, L_2R_2),$$
$$R_3 = \max(L_1L_2, L_1R_2, L_2R_1, L_2R_2).$$

These operations have the following property: If $x_j \in (L_j, R_j)$ for $j = 1, 2$, then $x_j * y_j \in (L_1, R_1) * (L_2, R_3)$. Here $*$ is any of the three operations just mentioned. All our calculations boil down to showing that $x > 0$ or $x < 0$ for some real number x. We do our calculations in such a way as to produce a BigInterval (L, R) such that $2^{106}x \in (L, R)$. We would show that $x < 0$ by showing that $R < 0$, and we would show that $x > 0$ by showing that $L > 0$.

6.3 The Interval Cosine Function

Looking at (2–10) and (2–11), we see that we need some way to deal with the sine and cosine functions. When we run our subdivision algorithm, we find that it never produces a dyadic square whose side length is less than 2^{18}. For this reason, we are evaluating the sine and cosine functions only on numbers[2] of the form

$$\frac{\pi}{2} \cdot \frac{k}{2^{20}}.$$

[2]Actually, we need only 2^{18} rather than 2^{20}, but we want to give ourselves a little cushion here.

Using the identities

$$\sin(x) = \cos\left(\frac{\pi}{2} - x\right), \quad \cos(x + n\pi) = (-1)^n \cos(x),$$

we see that it suffices to consider the 2^{21} values

$$c_k := 2^{53} \cos\left(\frac{\pi}{2} \times \frac{k}{2^{20}}\right), \quad k = 0, \dots, 2^{21} - 1.$$

(There is nothing special about 2^{53}. We like it because it affords about the same precision as a double in C.)

We now explain how we produce a BigInterval I_k such that $c_k \in I_k$. Once we have I_k, we evaluate (2–10) and (4–1) using the operations discussed above. Producing I_k is quite easy. The tricky part is proving rigorously that our method really works. We know that there exist packages in Java that perform this task for the elementary functions, but we prefer to work from scratch. We want to stress that it doesn't really matter how we produce our BigInterval I_k. The important point is the proof that $c_k \in I_k$. However, it seems worth explaining our simple method.

6.3.1 Producing the Interval.

We introduce the routine `cosBestApprox`. When we evaluate this routine on the pair $(k, 20)$, it produces a BigInteger C_k. We then take $I_k = (C_k - 4, C_k + 4)$. The routine `cosBestApprox` essentially computes the "usual" cosine on the relevant point—here $n = 20$, and k is as above—and then rounds to the nearest BigInteger. Our method uses the BigDecimal class, which is just a BigInteger together with a separate integer that tells where to put the decimal point. Here is our code, all of which can be found online in the file `Deg100Trig.java`:

```
public static BigInteger cosBestApprox(int k,int n)
{
double d=Math.PI/2.0;
d=d*k/Math.pow(2.0,n);
d=Math.cos(d);
BigDecimal Y1=new BigDecimal(d);
BigInteger BIG=getBIG();
BigDecimal Y2=new BigDecimal(BIG);
Y1=Y1.multiply(Y2);
BigInteger X=Y1.toBigInteger();
return(X);
}
```

The BigInteger `BIG` is 2^{53}. Here is the routine that gets it:

```
public static BigInteger getBIG() {
BigInteger BIG=new BigInteger("9007199254740992");
return(BIG); }
```

6.3.2 Checking That the Method Works.

What we actually show is that

$$2^{357} 20! c_k \in 2^{357} 20! I_k.$$

A huge number like this appears fairly naturally because we want to clear denominators in some Taylor series approximations for cosine.

For $j = 0, 1, \dots, 10$, let L_j be the greatest integer less than

$$\frac{2^{400} 20!}{2^{40j} (2j)!} \cdot \left(\frac{\pi}{2}\right)^{2j}.$$

Let $R_j = L_j + 1$. We compute these twenty integers using Mathematica, which has a reliable arbitrary-precision evaluation of the trig functions. The reader can see our values in the file `Deg100Trig.java`. Consider the sums

$$A_k = L_0 - R_1 k^2 + L_2 k^4 - R_3 k^6 + \cdots - R_{10} k^{20},$$
$$B_k = R_0 - L_1 k^2 + R_2 k^4 - L_3 k^6 + \cdots + R_9 k^{18}.$$

Considering the Taylor series for cosine, we easily get that

$$2^{357} 20! c_k \in [A_k, B_k].$$

To verify that $c_k \in I_k$, it suffices to check that

$$2^{257} 20! (C_k - 4) < A_k, \quad B_k < 2^{357} 20! (C_k + 4).$$

This is purely a calculation involving BigIntegers. We perform the verification, and it works. As a control, we performed the verification using 2 in place of 4, and it failed at some point. The program is contained in the same file as already mentioned. The reader can launch the program right from the "100 Degree" window in McBilliards.

Remark 6.2. We found that $2^{357} 20!$ worked well for us. This choice yields the following values:

$$A_8 = 193117979382323170336391434868704,$$
$$A_9 = 1416254196461936667,$$
$$A_{10} = 8363,$$
$$A_{11} = 0.$$

Thus the choice $2^{357} 20!$ is well adapted to an approximation based on about ten terms of the Taylor series.

6.4 BigInterval Structures

As one last bit of structure, we define a *BigComplexInterval* to be a structure of the form $X + iY$, where X and Y are BigIntervals. The arithmetic on these objects is just the same as the arithmetic on ordinary complex numbers, except that we substitute the BigInterval operations for the ordinary arithmetic operations on reals. (We never have occasion to do any division, so we are just talking about addition, subtraction, and multiplication.)

Once we have our BigInterval version of sine and cosine, and the BigComplexInterval class, we plug these objects into (2–10) and (4–1), wrapping every integer in sight inside a BigInterval. We then perform all the operations described in Section 5. Our algorithm halts for all 221 polygons, and this constitutes our proof of the 100 degree theorem.

The reader can run our algorithm and survey its output using McBilliards, as discussed in the paper. In particular, the reader can run the algorithm with or without the BigInterval arithmetic, and see that the output is about the same in both cases. (The output is not exactly the same because we make some convenient but arbitrary cutoffs in the numerical version.)

6.5 Sanity Checks

In order to help ensure that we have programmed the computer correctly, we have made three additional sanity checks in our calculations.

1. We make sure that our combinatorial method of computing the defining functions, namely (2–10), is correct. We introduce a straightforward geometric method of computing the defining functions geometrically: We just take the unfolding for the word and the given triangle, rotate it so that it is horizontal, and then measure the difference in heights of the relevant vertices. For each word W_i we evaluate each defining function on the first vertex of the polygon P_i, using both methods.

As long as the geometric method yields a number that is at least 0.001, we check, up to a tolerance of 0.000001, that there is a single ratio ρ such that the ratio of the combinatorial answer to the geometric answer is always ρ. (This ratio depends on the point of evaluation.) In other words, up to an initial rescaling, the two methods agree.

We consider this to be extremely strong evidence that we have got (2–10) correct and also programmed it correctly into the computer. We do not consider the very small percentage of defining functions that evaluate to a very small number, because the roundoff error interferes with the computation of the ratio.

2. We make sure that our BigInterval versions of our functions yield essentially the same answers as our numerical versions. We make the same evaluations as for the first sanity check, except that now we compare the numerical and BigInterval implementations of the combinatorial method.

We check that the first seven digits of the left endpoint of the BigInterval version agree with the first seven digits of 2^{106} times the numerical version.

In the interest of having the check move along at a steady clip when run from the interface, we check only about four percent of the defining functions. This still comes out to a huge number of checks.

In contrast to the first check, where the point is to verify that all cases of a complicated combinatorial procedure work, here we are just checking a fairly straightforward conversion from ordinary arithmetic operations to BigInterval operations.

3. We make sure that our formula for (4–1) is correctly implemented. For this purpose we compare the partial derivatives of the defining functions with a crude version of the partial derivatives obtained by taking a difference quotient. Our value for Δx and Δy in this computation is 2^{-30}.

We check that the two computations of the partial derivatives agree up to a relative error of 0.001. By this we mean that $|X_1 - X_2|/|X_1| < 0.001$. Here X_1 and X_2 are the two computed versions of the same quantity. We also require X_1, which is the difference quotient, to be at least 0.000001.

We test about one percent of the defining functions. Given the simple nature of the passage from (2–10) to (4–1), this is overwhelming evidence that we have programmed (4–1) correctly into the computer.

The reader can run our sanity checks, either for individual words or else for all words in sequence, from the "100 Degree" window in McBilliards. The code for our sanity checks is contained in the file `Deg100SanityCheck.java`. Indeed, all our computer code pertaining to the 100 degree theorem can be launched from this window.

We also mention another sanity check. Originally we had programmed McBilliards in C and Tcl. We originally did all the computations for this paper in the C version. (We switched to Java so that the whole proof could be easily accessible right on the web, to someone without specialized computer knowledge, and also because we wanted to make a new and improved McBilliards.)

Perhaps the best sanity check of all is that McBilliards *works*. This program has many interlocking features, and the interested reader can see that they all fit together in a way that would be extremely unlikely if there were serious bugs in the program.

7. USING MCBILLIARDS

7.1 The Applet

For the reader mainly interested in seeing the results in this paper illustrated, we recommend the Java applet that we wrote. One can access this applet in several ways. One address is http://mcbilliards/sourceforge.net/Deg100/. Another address is my website: http://www.math.brown.edu/~res/Java/App46/test1.html.

This applet is a toy version of McBilliards specifically designed for the 100 degree theorem. The Java applet displays the polygons $P_3, \dots, P_{221}$. One can zoom into the picture to see the fine structure of the covering.

1. For each $j = 7, \dots, 221$, a click on the polygon P_j calls forth a display of the vertices of the polygon, and also the word W_j and its unfolding. One can drag the mouse around P_j and check visually that (modulo roundoff error) $P_j \subset O(W_j)$.
2. We break P_5 into two regions P_{51} and P_{52}. One can click on each of these regions and see the vertices and corresponding words W_{51} and W_{52} as above. (These regions are not actually named in the program.)
3. We do the same for P_4 as we do for P_5, using subregions $P_{41}, \dots, P_{45}$.
4. For $j = 3$ and $j = 6$, we break P_j into an infinite number of smaller polygonal regions. In each case, one can see the words and regions corresponding to the first ten terms in the sequence. We show enough so that the pattern is fairly clear. Again, for these first few words, one can verify visually that the polygonal region is contained in the corresponding orbit tile.

The words and polygon vertices are displayed in full, so that (modulo the reader being able to fill in several infinite patterns from a finite start) the complete record of the words and polygons resides in the applet.

7.2 The Basics of the Main Program

McBilliards can be run either as an applet or as a stand-alone Java program. To save yourself the trouble of downloading and installing the program, you should first try running McBilliards as an applet. The stand-alone version has some additional features, but these additional features would probably not be the first thing you would want to learn.

Assuming that your browser can handle Java, you can run McBilliards from my website: http://www.math.brown.edu/~res/Java/App47/A2.html. You can also find both the applet and the program on the website http://mcbilliards.sourceforge.net.

For the rest of this discussion, I will assume that you can actually access and run McBilliards, one way or another.

7.2.1 Accessing the Documentation.

The first thing to do is to see how the documentation for the program works. For example, there is a little black box with a question mark at the very bottom right of the program. If you click on this box (or any other box that has a question mark in it), the documentation window will pop up. Most of the features of McBilliards have a question box beside them, so that you can learn what they do and how to use them.

7.2.2 The Mouse Emulator.

The mouse emulator lives at the bottom right of the program. The question box we mentioned above pertains to this module, and explains how to use the mouse emulator. McBilliards is meant to be run with a three-button mouse. If you don't have one or if your browser does not interpret your mouse clicks correctly, you can get around the problem by using the mouse emulator on the bottom right. By clicking on the question box, you can see how to operate the emulator.

7.2.3 Parameter Selection.

The big central window in McBilliards is the parameter window. This window contains the region Δ that we have discussed in this paper. You select points in this window by clicking button 2 of the mouse over a point. You can also drag the mouse, again button 2, to select a point. If you want to see the triangle to which your selection corresponds, click on the "more popups" window at the top of the program. This brings up an auxiliary menu. From this menu, click on the "billiard path" option. This brings up another window, which displays the triangle corresponding to the point in parameter space you have selected. This auxiliary window also displays billiard paths in the triangle, whence the name.

7.2.4 Searching.

Once you have selected a point you like, you can press the "seek" button at the bottom of the program. This will find all the stable words of length less than the displayed length. We initially set the program to 50. We recommend that you choose a point on the obtuse side that is fairly near the right-angled line. If you pick a point too far into the obtuse region, your search won't turn up anything (unless you increase the maximum word length.) When your search is done, an auxiliary window pops up, displaying all the hexpaths of the words you have found.

7.2.5 Plotting. Once you have selected a point and done a search, you can select one of the hexpaths from the auxiliary window that has popped up. After you select one of these hexpaths, you can plot the corresponding orbit tile by pushing the plot button. The color selector allows you to change the color of the plotted tile. The bottom left portion of the program gives controls for managing the plotted tiles: deleting, raising, lowering, and recoloring.

7.2.6 Unfolding Window. Once you have plotted an orbit tile (or before), you can click on the top of the program to bring up the unfolding window. The unfolding window draws the unfolding $U(W, T)$, where W is the currently selected word and T is the currently selected triangle. By dragging the mouse around an orbit tile, you can see the unfolding change with the point selection. This is a powerful sanity check that the searching and plotting options are working correctly. Like almost all the windows in McBilliards, the unfolding window is resizable. You can see a nice large picture of the unfolding if you like.

7.2.7 Word Window. Clicking at the top of the program brings up the word window. This shows a large copy of the hexpath for the current word. The word window is animated, so one can see how the hexpath is created from the word.

The best way to get a feel for the basic features of McBilliards is to play with the program, i.e. search, plot, survey tiles, with both the word and unfolding windows open. The unfolding and word windows have many auxiliary features embedded in them, and in time, the user can learn these from the documentation. We will talk more about the unfolding window below.

7.3 The Unfolding Window: More Details

Now we will assume that you have mastered the basic features of McBilliards, discussed above. Here we discuss the unfolding window in more detail.

7.3.1 Turning Pairs. If you click the middle mouse button on an edge of the unfolding, you can see the turning pair of that edge displayed at the left. The turning pairs were discussed in Section 2.4. This will give you a better feel for the algorithm presented in that section.

7.3.2 Defining Functions. You can select a pair of vertices from the unfolding that is drawn in the unfolding window. Once you select these vertices, the unfolding window computes the defining function associated with these vertices. The defining function is then displayed at the bottom of the unfolding window, using the conventions described in Section 2.7. One peculiar feature is that positive numbers are displayed in white, and negative numbers are displayed (without signs) in black. We somewhat regret this convention now, but it does save space.

As an aid to the computation of the defining function, the unfolding window also shows the path of edges connecting the one vertex to the other, as discussed in Lemma 2.15.

7.3.3 The Leading Vertices. One can also access the defining functions in another way. One can turn on the "compute leaders" option in the unfolding window. Assuming that you have plotted a tile, the unfolding window will automatically select and display the pair of vertices on the unfolding, one top and one bottom, that come closest to having the same height. In this way, you can see which defining functions define the edges of the tile.

7.3.4 Derivative Bounds. Once a defining function is computed, the derivative bounds discussed in Section 4.1 are displayed at the bottom of the unfolding window. The documentation for this part of the unfold window has more information.

7.4 Surveying our Proof

One can survey our proof using a version of our applet that we have "embedded" into McBilliards. One accesses this embedded version of the applet by bringing up the "more popups" menu and selecting the "100 degree result" window. This brings up the embedded copy of the applet. Using the "100 degree result" window, you can survey our computational algorithm down to the last detail. (This is extremely tedious, but possible.) Here we explain how one surveys the results of the tournament algorithm, discussed in Section 4.

- To run the verify algorithm for a particular word, select the *verify single* mode in the 100 degree window interface and then click on the desired word. (These words are indexed by little square buttons on the interface.) Be sure to have the *trace verify* button off.

- Once the picture is plotted on the main McBilliards window, turn on the *trace verify button* and select your favorite dyadic square that you have just plotted by clicking inside it. Now click on the same word you just clicked.

- With the *trace verify* mode on, McBilliards reruns the algorithm, discarding any square that does not contain the selected point. This has the effect of just tracing through the part of the algorithm that deals with the selected square.

- Open up the unfolding window after the selected square has been plotted. Along the bottom of the square you will see three kinds of boxes: the top winners, the bottom winners, and the tournament record. The tournament record consists of a number of pairs of the form (p, q), where *p loses to q* on some box that contains the selected one. We call these *match boxes*.

- If you click on one of these match boxes, you will see the formulas for the defining function associated with the relevant pair of vertices. You also get to see a graphical display of the gradient and the quadrant that contains the gradient throughout the dyadic square. By moving the point around on the main interface, you can visually check that the gradient remains within the quadrant. Also, you see displayed all the quantities that go into the calculation of the certificates, so you can recompute them yourself from the information.

- If you click on every single match box and make the computations yourself, by hand, you will have given your own proof that the tournament has performed correctly. Finally, you can go through all the pairs of the form (top winner, bottom winner) and make all the same checks.

7.5 Surveying McBilliards as a Whole

We think that the internal consistency of McBilliards, the extensive testing we have done, and the exact correspondence between experimental predictions and the theoretical proofs (as in Section 3) gives extremely strong evidence that McBilliards does what we claim it does. However, we can imagine that some readers would disagree.

Some readers may feel that a rigorous computer-aided proof would include a complete description of all the computer code used in the proof. McBilliards is an enormous program, and obviously it would not be possible to give a complete description of the code here.

The interested reader can survey the McBilliards code in full. Going back to the sourceforge McBilliards web page, you can browse through our online documentation for McBilliards, which shows the details of essentially every class, method, and interface in McBilliards.

The code relevant to the 100 degree theorem takes up a only small subset of the total program. To isolate the relevant code, we have put it in files that have the `Deg100` prefix, such as `Deg100Verifier.java`. However, there are some basic classes, such as the complex number class and some graphics classes, that are required to support the code in the `Deg100` files.

ACKNOWLEDGMENTS

I did the initial experiments for this project at the Max Planck Institute in Bonn, during July 2004. I would like to thank the institute for their hospitality and generous support, and also the Guggenheim Foundation. I would like to thank Mike Boyle, Curt McMullen, Dan Rudolph, Martin Schmoll, Serge Troubetzkoy, and Sergei Tabachnikov for their encouragement, and also for helpful conversations related to this work.

I would especially like to thank Pat Hooper, who is my collaborator on McBilliards, for many, many helpful conversations about triangular billiards.

The author's research has been supported by NSF Grant DMS-0305047 and by a Guggenheim Fellowship.

REFERENCES

[Boshernitzyn et al. 98] M. Boshernitzyn, G. Galperin, T. Kruger, and S. Troubetzkoy. "Periodic Billiard Trajectories Are Dense in Rational Polygons." *Trans. AMS* 350 (1998), 3523–3535.

[Galperin et al. 91] , G. A. Galperin, A. M. Stepin, and Y. B. Vorobets. "Periodic Billiard Trajectories in Polygons." *Russian Math Surveys* 47 (1991), 5–80.

[Gutkin 96] E. Gutkin. "Billiards in Polygons: Survey of Recent Results." *J. Stat. Phys.* 83 (1996), 7–26.

[Halbeisen and Hungerbuhler 00] L. Halbeisen and N. Hungerbuhler. "On Periodic Billiard Trajectories in Obtuse Triangles." *SIAM Review* 42:4 (2000), 657–670.

[Hooper 07] W. P. Hooper. "Periodic Billiard Paths in Right Triangles Are Unstable." *Geometriae Dedicata* 125 (2007), 39–46.

[Hooper and Schwartz 08] W. P. Hooper and R. E. Schwartz. "Billiards in Nearly Isosceles Triangles." Preprint, 2008.

[Masur 86] H. Masur. "Closed Trajectories for Quadratic Differentials with an Application to Billiards." *Duke Math J.* 53 (1986), 307–314.

[Masur and Tabachnikov 02] H. Masur and S. Tabachnikov. "Rational Billiards and Flat Structures." In *Handbook of Dynamical Systems 1A*, edited by B. Hassleblatt and A. Katok, pp. 1015–1089. Amsterdam: North-Holland, 2002.

[Schwartz 06] R. Schwartz. "Obtuse Triangular Billiards I: Near the $(2, 3, 6)$ Triangle." *Experimental Mathematics* 15:2 (2006), 161–182.

[Tabachnikov 95] S. Tabachnikov. *Billiards,* SMF Panoramas et Syntheses 1. Paris: Société Mathématique de France, 1995.

[Troubetzkoy 04] S. Troubetzkoy. "Billiards in Right Triangles." Preprint, 2004.

[Veech 92] W. Veech. "Teichmüller Curves in Moduli Space: Eisenstein Series and an Application to Triangular Billiards." *Invent. Math.* 97 (1992), 341–379.

Richard Evan Schwartz, Department of Mathematics, Brown University, Providence, RI 02912 (res@math.brown.edu)

Received September 8, 2007; accepted in revised form August 26, 2008.