Know When to Fold 'Em: Self-assembly of Shapes by Folding in Oritatami

Erik Demaine, Jacob Hendricks, Meagan Olsen, Matthew Patitz, Trent A. Rogers, Nicolas Schabanel, Shinnosuke Seki, Hadley Thomas

Know When to Fold 'Em: Self-Assembly of Shapes by Folding in Oritatami

Erik D. Demaine[1], Jacob Hendricks[2], Meagan Olsen[3], Matthew J. Patitz[3,4], Trent A. Rogers[3,4], Nicolas Schabanel[5], Shinnosuke Seki[6], and Hadley Thomas[7]

[1] CSAIL, Massachusetts Institute of Technology, USA. edemaine@mit.edu
[2] Department of Computer Science and Information Systems, University of Wisconsin - River Falls, River Falls, WI, USA. jacob.hendricks@uwrf.edu
[3] Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR, USA. {mo015, patitz, tar003}@uark.edu.
[4] Supported in part by NSF Grant CCF-1422152 and CAREER-1553166.
[5] CNRS, École Normale Supérieure de Lyon (LIP, UMR 5668) & IXXI, U. Lyon. perso.ens-lyon.fr/first.last Supported by Moprexprogmol CNRS MI grant.
[6] University of Electro-Communications, Tokyo, Japan. s.seki@uec.ac.jp. Supported in part by JST Program to Disseminate Tenure Tracking System, MEXT, Japan, No. 6F36, JSPS Grant-in-Aid for Young Scientists (A) No. 16H05854, and JSPS Bilateral Program No. YB29004
[7] Colorado School of Mines, Golden, CO, USA. hadleythomas88@gmail.com

Abstract. An oritatami system (OS) is a theoretical model of self-assembly via co-transcriptional folding. It consists of a growing chain of beads which can form bonds with each other as they are transcribed. During the transcription process, the δ most recently produced beads dynamically fold so as to maximize the number of bonds formed, self-assemblying into a shape incrementally. The parameter δ is called the *delay* and is related to the transcription rate in nature.

This article initiates the study of shape self-assembly using oritatami. A shape is a connected set of points in the triangular lattice. We first show that oritatami systems differ fundamentally from tile-assembly systems by exhibiting a family of infinite shapes that can be tile-assembled but cannot be folded by any OS. As it is NP-hard in general to determine whether there is an OS that folds into (self-assembles) a given finite shape, we explore the folding of upscaled versions of finite shapes. We show that any shape can be folded from a constant size seed, at any scale $n \geqslant 3$, by an OS with delay 1. We also show that any shape can be folded at the smaller scale 2 by an OS with *unbounded* delay. This leads us to investigate the influence of delay and to prove that, for all $\delta > 2$, there are shapes that can be folded (at scale 1) with delay δ but not with delay $\delta' < \delta$.

These results serve as a foundation for the study of shape-building in this new model of self-assembly, and have the potential to provide better understanding of cotranscriptional folding in biology, as well as improved abilities of experimentalists to design artificial systems that self-assemble via this complex dynamical process.

1 Introduction

Transcription is the process in which an RNA polymerase enzyme (colored in orange in Fig. 1) synthesizes the temporal copy (blue) of a gene (gray spiral) out of ribonucleotides of four types `A`, `C`, `G`, and `U`. The copied sequence is called the *transcript*.

The transcript starts folding upon itself into intricate tertiary structures immediately after it emerges from the RNA polymerase. Fig. 1 (Left) illustrates *cotranscriptional folding* of a transcript into a rectangular RNA tile structure while being synthesized out of an artificial gene engineered by Geary, Rothemund, and Andersen [11]. The RNA tile is provided with a kissing loop (KL) structure, which yields a 120° bend, at its four corners, and sets of six copies of it self-assemble into hexagons and further into a hexagonal lattice. *Structure* is almost synonymous to *function* for RNA complexes since they are highly correlated, as exemplified by various natural and artificial RNAs [6]. Cotranscriptional folding plays significant roles in determining the structure (and hence function) of RNAs. To give a few examples, introns along a transcript cotranscriptionally fold into a loop recognizable by spliceosome and get excised [17], and riboswitches make a decision on gene expression by folding cotranscriptionally into one of two mutually exclusive structures: an intrinsic terminator hairpin and a pseudoknot, as a function of specific ligand concentration [23].

What is folded is affected by various environmental factors including transcription rate. Polymerases have their own transcription rate: e.g., bacteriophage 3ms/nucleotide (nt) and eukaryote 200ms/nt [14] (less energy would be dissipated at slower transcription [8]). Changing the natural transcription rate, by adjusting, e.g., NTP concentration [19], can impair cotranscriptional processes [2,15] (note that polymerase pausing can also facilitate efficient folding [24] but it is rather a matter of gene design). Given a target structure, it is hence necessary to know not only what to fold but at what rate to fold, that is, to know when to fold 'em.

The primary goal of both natural and artificial self-assembling systems is to form predictable structures, i.e. shapes grown from precisely placed components, because the form of the products is what yields their functions. Mathematical models have proven useful in developing an understanding of how shapes may self-assemble, and self-assembling finite shapes is one of the fundamental goals of theoretical modeling of systems capable of self-assembly. e.g. in tile-based self-assembly [3, 4, 22] as well as other models of programmable matter [5, 25].

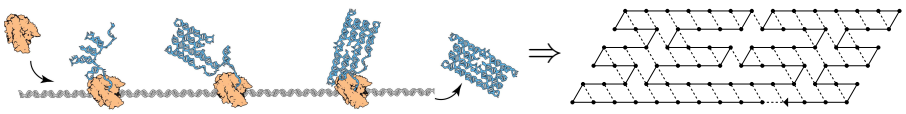


Fig. 1: (Left) RNA Origami [11]. (Right) An abstraction of the resulting RNA tile in the oritatami system, where a dot • represents a sequence of 3-4 nucleotides, and the solid arrow and dashed lines represent its transcript and interactions based on hydrogen bonds between nucleotides, respectively.

An oritatami system (abbreviated as OS) is a novel mathematical model of cotranscriptional folding, introduced by [10]. It abstracts an RNA tertiary structure as a triple of 1) a sequence of abstract molecules (of finite types) called *bead types*, 2) a directed path over a triangular lattice of *beads* (i.e. a location/bead type pair), and 3) a set of pairs of adjacent beads that are considered to interact with each other via hydrogen bonds. Such a triple is called a *configuration*. An abstraction of the RNA tile from [11] as a configuration is shown in Fig. 1 (Right). In the figure, each bead (represented as a dot) abstractly represents a sequence of 3-4 nucleotides, whose type is not stated explicitly but retrievable from the transcript's sequence of the tile (available in [11]); moreover, the interactions (or bonds) between pairs of beads are represented by dashed lines. An OS is provided with a finite alphabet B of bead types, a sequence w of beads over B called its *transcript*, and a rule ♥, which specifies between which types of beads interactions are allowed. The OS cotranscriptionally folds its transcript w, beginning from its initial configuration (*seed*), over the triangular lattice by stabilizing beads of w from the beginning one by one. Two parameters of OS govern the bead stabilization: *arity* and *delay*; arity models valence (maximum number of bonds per bead). Delay models the transcription rate in the sense that the system stabilizes the next bead in such a way that the sequence of the next bead and the $\delta - 1$ succeeding beads is folded so as to form as many bonds as possible.

Using this model, researchers have mainly explored the computational power of cotranscriptional folding (see [10] and the recent surveys [20,21]). In contrast, little has been done on self-assembly of shapes. Elonen in [7] informally sketched how an OS can fold a transcript whose beads are all of distinct types (hardcodable transcript) into a finite shape using a provided Hamiltonian path. Masuda et al. implemented an OS that folds its periodic transcript into a finite portion of the Heighway dragon fractal [16].

Our results. We initiate a systematic study of shape self-assembly by oritatami systems. We start with the formal definitions of OS and shapes in Section 2. As it is NP-hard to decide if a given connected shape of the triangular lattice contains a Hamiltonian path [1], it is also NP-hard to decide if there is an OS that folds into (self-assembles) a given finite shape. We thus explore the folding of upscaled versions of finite shapes. We introduce three upscaling schemes $\mathscr{A}_n$, $\mathscr{B}_n$ and $\mathscr{C}_n$, where n is the scale factor (see Fig. 2). We first show that oritatami systems differ fundamentally from tile-assembly systems by exhibiting a family of infinite shapes that can be tile-assembled but cannot be folded by any OS (Theorem 2, Section 3). We then show that any shape can be folded at scale factor 2 by an OS with *unbounded* delay (Theorem 3, Section 4). In section 5, we present various incremental algorithms that produce a delay-1 arity-4 OS that folds any shape from a seed of size 3, at any scale $n \geqslant 3$ (Theorems 6 and 8, Section 5). For this purpose, we introduce a universal set of 114 bead types suitable for folding any delay-1 tight OS (Theorem 4) that can be used in other oritatami designs. We then show that the delay impacts our ability to build shapes: we prove that there are shapes that can be folded (at scale 1) with delay δ but not with delay $\delta' < \delta$ (Theorem 9, Section 6).

These results serve as a foundation for the study of shape-building in this new model of self-assembly, and have the potential to provide better understanding of cotranscriptional folding in biology, as well as improved abilities of experimentalists to design artificial systems that self-assemble via this complex dynamical process.

Note that in [13] in the present proceedings, the authors study a slightly different problem: they show that one can design an oritatami transcript that folds an upscaled version of a *non-self-intersecting path* (instead of a shape). The initial path may come from the triangular grid or from the square grid. The scale of the resulting path is somewhere in between our scales 3 and 4 according to our definition. Note that the cells are only partially covered by their scheme. Combining their result with our theorem 3, their algorithm provides an oritatami transcript partially covering the upscaled version of any shape at scale 6.

2 Definitions

2.1 Oritatami System

Let B be a finite set of *bead types*. A *routing* r of a bead type sequence $w \in B^* \cup B^{\mathbb{N}}$ is a directed self-avoiding path in the triangular lattice $\mathbb{T}$,[8] where for all integer i, vertex r_i of r is labelled by w_i. r_i is the *position* in $\mathbb{T}$ of the $(i+1)$th bead, of type w_i, in routing r. A *partial routing* of a sequence w is a routing of a prefix of r.

An Oritatami system $\mathcal{O} = (B, w, \heartsuit, \delta, \alpha)$ is composed of (1) a set of bead types B, (2) a (possibly infinite) bead type sequence w, called the *transcript*, (3) an *attraction rule*, which is a symmetric relation $\heartsuit \subseteq B^2$, (4) a parameter δ called the *delay*, and (5) a parameter α called the *arity*.

We say that two bead types a and b *attract* each other when $a \heartsuit b$. Given a (partial) routing r of a bead type sequence w, we say that there is a *potential (symmetric) bond* $r_i r_j$ between two adjacent positions r_i and r_j of r in $\mathbb{T}$ if $w_i \heartsuit w_j$ and $|i-j| > 1$. A *set of bonds* H for a (partial) routing r is a subset of its potential bonds. A couple $c = (r, H)$ is called a (partial) *configuration* of w. The *arity* $\alpha_i(c)$ of position r_i in the partial configuration $c = (r, H)$ is the number of bonds in H involving r_i, i.e. $\alpha_i(c) = \#\{j : r_i r_j \in H\}|$. A (partial) configuration c is *valid* if each position r_i is involved in at most α bonds in H, i.e. if $(\forall i)\ \alpha_i(c) \leqslant \alpha$. We denote by $h(c) = |H|$ the number of bonds in configuration c.

For any partial valid configuration $c = (r, H)$ of some sequence w, an *elongation* of c by k beads (or k-*elongation*) is a partial valid configuration $c' = (r', H')$ of w of length $|c| + k$ where r' extends the self-avoiding path r by k positions

[8] The triangular lattice is defined as $\mathbb{T} = (\mathbb{Z}^2, \sim)$, where $(x, y) \sim (u, v)$ if and only if $(u, v) \in \cup_{\epsilon=\pm 1}\{(x+\epsilon, y), (x, y+\epsilon), (x+\epsilon, y+\epsilon)\}$. Every position (x, y) in $\mathbb{T}$ is mapped in the euclidean plane to $x \cdot X + y \cdot Y$ using the vector basis $X = (1, 0)$ and $Y = \text{RotateClockwise}\,(X, 120°) = (-\frac{1}{2}, -\frac{\sqrt{3}}{2})$.

and such that $H \subseteq H'$. We denote by $\mathcal{C}_w$ the set of all partial configurations of w (the index w will be omitted when the context is clear). We denote by $c^{\triangleright k}$ the set of all k-elongations of a partial configuration c of sequence w.

Oritatami dynamics. The folding of an oritatami system is controlled by the delay δ and the arity α. Informally, the configuration grows from a *seed configuration*, one bead at a time. This new bead adopts the position(s) that maximise the number of valid bonds the configuration can make when elongated by δ beads in total. This dynamics is *oblivious* as it keeps no memory of the previously preferred positions; it differs thus slightly from the hasty dynamics studied in [10] but is more prevailing in the OS research [9,12,16,18,20] because it seems closer to experimental conditions such as in [11].

Formally, given an oritatami system $\mathcal{O} = (B, w, \heartsuit, \delta, \alpha)$ and a *seed configuration* σ of the $|\sigma|$-prefix of w, we denote by $\mathcal{C}_{\sigma,w}$ the set of all partial configurations of the sequence w elongating the seed configuration σ. The considered *dynamics* $\mathscr{D} : 2^{\mathcal{C}_{\sigma,w}} \to 2^{\mathcal{C}_{\sigma,w}}$ maps every subset S of partial configurations of length ℓ, elongating σ, of the sequence w to the subset $\mathscr{D}(S)$ of partial configurations of length $\ell + 1$ of w as follows:

$$\mathscr{D}(S) = \bigcup_{c \in S} \arg\max_{\gamma \in c^{\triangleright 1}} \left(\max_{\eta \in \gamma^{\triangleright \min(\delta-1,\, |w|-|\gamma|)}} h(\eta) \right)$$

We say that a (partial) configuration c *produces* a configuration c' over w, denoted $c \vdash c'$, if $c' \in \mathscr{D}(\{c\})$. We write $c \vdash^* c'$ if there is a sequence of configurations $c = c^0, \ldots, c^t = c'$, for some $t \geqslant 0$, such that $c^0 \vdash \cdots \vdash c^t$. A sequence of configurations $c = c^0 \vdash \cdots \vdash c^t = c'$ is called a *foldable sequence over w from configuration c to configuration c'*. The *foldable configurations* in t steps of $\mathcal{O}$ are the elongations of the seed configuration σ by t beads in the set $\mathscr{D}^t(\{\sigma\})$. We denote by $\mathcal{A}[\mathcal{O}] = \cup_{t \geqslant 0} \mathscr{D}^t(\{\sigma\})$ the set of all foldable configurations. A configuration $c \in \mathcal{A}[\mathcal{O}]$ is *terminal* if $\mathscr{D}(\{c\}) = \varnothing$. We denote by $\mathcal{A}_\square[\mathcal{O}]$ the set of all terminal foldable configurations of $\mathcal{O}$. A finite foldable sequence $\sigma = c^0 \vdash \cdots \vdash c^t$ *halts* at c^t after t steps if c^t is terminal; then, c^t is called the *result* of the foldable sequence. A foldable sequence may halt after $|w| - |\sigma|$ steps or earlier if the growth is geometrically obstructed (i.e., if no more elongation is possible because the configuration is trapped in a closed area). An infinite foldable sequence $\sigma = c^0 \vdash \cdots \vdash c^t \vdash \cdots$ admits a *unique limiting configuration* $c^\infty = \sqcup_t c^t$ (the superposition of all the configurations (c^t)), which is called the *result* of the foldable sequence.

We say that the oritatami system is *deterministic* if at all time t, $\mathscr{D}^t(\{\sigma\})$ is either a singleton or the empty set. In this case, we denote by c^t the configuration at time t, such that: $c^0 = \sigma$ and $\mathscr{D}^t(\{\sigma\}) = \{c^t\}$ for all $t > 0$; we say that the partial configuration c^t *folds (co-transcriptionally) into* the partial configuration c^{t+1} deterministically. In this case, at time t, the $(t+1)$-th bead of w is placed in c^{t+1} at the position that maximises the number of valid bonds that can be made in a $\min(\delta, |w| - t - |\sigma|)$-elongation of c^t. Note that when $\alpha \geqslant 4$ the arity constraint vanishes (as a vertex may bond to at most 4 neighbors, 5 if the growth is at a dead end) and then, there is only one maximum-size bond set for every routing, consisting of all its potential bonds.

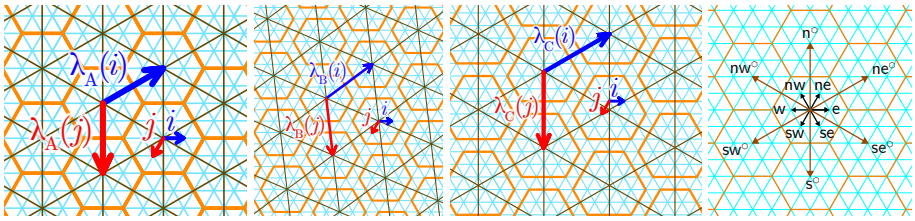


Fig. 2: The three upscaling schemes $\mathscr{A}_3$, $\mathscr{B}_3$ and $\mathscr{C}_3$ (cell boundaries are represented in orange and the upscaled triangular grid in brown); to the right: the lattice directions $\mathcal{D} = \{\mathsf{nw}, \mathsf{ne}, \mathsf{e}, \mathsf{se}, \mathsf{sw}, \mathsf{w}\}$ in $\mathbb{T}$, and the cell directions $\mathcal{D}^\circ = \{\mathsf{nw}^\circ, \mathsf{n}^\circ, \mathsf{ne}^\circ, \mathsf{se}^\circ, \mathsf{s}^\circ, \mathsf{sw}^\circ\}$.

2.2 Shape folding and scaling

The goal of this article is to study how to fold shapes. A *shape* is a connected set of points in $\mathbb{T}$. The *shape* associated to a configuration $c = (r, H)$ of an OS $\mathcal{O}$ is the set of the points $S(c) = \cup_i \{r_i\}$ covered by the routing of c. A shape S is *foldable* from a seed of size s if there is a deterministic OS $\mathcal{O}$ and a seed configuration σ with $|\sigma| = s$, whose terminal configuration has shape S.

Note that every shape admitting a Hamiltonian path is trivially foldable from a seed of size $|S|$, whose routing is a Hamiltonian path of the shape itself. The challenge is to design an OS folding into a given shape whose seed size is an *absolute* constant. One classic approach in self-assembly is then to try to fold an *upscaled* version of the shape. The goal is then to minimize the scale at which an upscaled version of every shape can be folded.

From now on, we denote by $(i, j) \in \mathbb{N}^2$ the point $i \cdot X + j \cdot Y$ of $\mathbb{T}$ in $\mathbb{R}^2$ where $X = (1, 0)$ (east) and $Y = (-\frac{1}{2}, -\frac{\sqrt{3}}{2})$ (south west) in the canonical basis.

As it turns out, there are different possible upscaling schemes for shapes in $\mathbb{T}$. A *scaling scheme* $\Lambda = (\lambda, \mu)$ of $\mathbb{T}$ is defined by a homothetic linear map λ from $\mathbb{T}$ to $\mathbb{T}$, and a shape μ containing the point $(0, 0)$, called the *cell mold*. For all $p \in \mathbb{T}$, the *cell* associated to p by Λ is the set $\Lambda(p) = \lambda(p) + \mu = \{\lambda(p) + q : q \in \mu\}$, i.e. the translation of the cell mold by $\lambda(p)$. $\lambda(p)$ is called the *center* of the cell $\Lambda(p)$. The Λ-scaling of a shape S is then the set of points $\Lambda(S) = \cup_{p \in S} \Lambda(p)$. We say that two cells $\Lambda(p)$ and $\Lambda(q)$ are neighbors, denoted by $\Lambda(p) \sim \Lambda(q)$, if they intersect or have neighboring points, i.e. if $\Lambda(p) \cap \Lambda(q) \neq \varnothing$ or there are two points $p' \in \Lambda(p)$ and $q' \in \Lambda(q)$ such that $p' \sim q'$. We require upscaling schemes to preserve the topology of S, in particular that $\Lambda(p) \sim \Lambda(q)$ iff $p \sim q$. We consider the following upscaling schemes (see Fig. 2):

Scaling $\mathscr{A}_n$: $\lambda_{\mathscr{A}_n}(i, j) = i \cdot (n-1, 1-n) + j \cdot (n-1, 2n-2)$ and $\mu_{\mathscr{A}_n} = H_n$
Scaling $\mathscr{B}_n$: $\lambda_{\mathscr{B}_n}(i, j) = i \cdot (n-1, -n) + j \cdot (n, 2n-1)$ and $\mu_{\mathscr{B}_n} = H_n$
Scaling $\mathscr{C}_n$: $\lambda_{\mathscr{C}_n}(i, j) = i \cdot (n, -n) + j \cdot (n, 2n)$ and $\mu_{\mathscr{C}_n} = H'_n$

where $H_n = \{(i, j) \in \mathbb{T} : |i| < n, |j| < n, |i - j| < n\}$ is the (filled) hexagon of radius $n - 1$ with n vertices on each side, and $H'_n = \{(i, j) \in \mathbb{T} : -n < i \leqslant n, -n < j \leqslant n, -n \leqslant i - j < n\}$ is the irregular hexagon whose sides are of alternating sizes n and $n + 1$. Note that $H_n \subset H'_n \subset H_{n+1}$. Each of these upscaling schemes have their ups and downs:

- Every cell in $\mathscr{A}_n$ is a regular hexagon. It is the most compact but, as the sides of the cells overlap, the area of $\Lambda_{\mathscr{A}_n}(S)$ scales linearly only asymptotically with the size of the original shape S. In particular empty cells are smaller than occupied cell.
- Every cell in $\mathscr{B}_n$ is a regular hexagon. It is less compact than $\mathscr{A}_n$ and twisted, but the edges of neighboring cells never overlap so the area of $\Lambda_{\mathscr{A}_n}(S)$ scales linearly with the size of the original shape S.
- $\mathscr{C}_n$ can be considered as a non-overlapping version of $\mathscr{A}_{n+1}$ where the $\mathsf{nw}^{\bigcirc}$-, $\mathsf{n}^{\bigcirc}$- and $\mathsf{ne}^{\bigcirc}$-sides of each cell have been trimmed by 1. It is isotropic as its cells are irregular hexagons, but it is untwisted and $\Lambda_{\mathscr{C}_n}(S)$ scales linearly with the size of the original shape S. One can also see the irregular hexagons as concentric spheres growing from the center of the triangles in lattice $\mathbb{T}$.

In terms of the resulting size of $\Lambda(S)$, $\mathscr{A}_n$ is strictly more compact than $\mathscr{B}_n$ which is strictly more compact than $\mathscr{C}_n$ which is as compact as $\mathscr{A}_{n+1}$ for all $n \geqslant 2$. n is referred as the *scale* for each scheme. Our goal is to find an OS with constant seed size for each of these schemes that can fold any shape at the smallest scale n.

Before we give our algorithms, we note the importance of scaling the shape in order to self-assemble it. Figure 3(a) shows an example of a shape which cannot be self-assembled by any OS (at scale 1), as it does not contain any Hamiltonian path. In fact, [1] proves that it is NP-hard to decide if a shape in $\mathbb{T}$ has a Hamiltonian path. Note that, if we are given a Hamiltonian path, there is a (hard-coding) OS that "folds" it, by simply using this path as the seed with no transcript. The existence of an OS (with unbounded seed) self-assembling a shape is thus equivalent to the existence of an Hamiltonian path. It follows that:

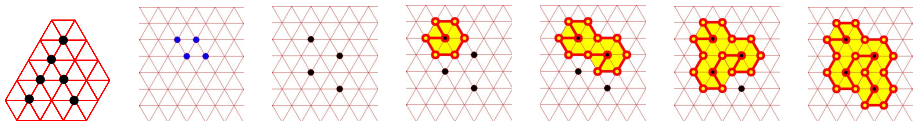
Observation 1 *Given an arbitrary shape S, it is NP-hard to decide if there is an oritatami system (with unbounded seed) which self-assembles it.*

In Section 5, we will present three algorithms building delay-1 OS that fold into arbitrary shapes at any of the scales $\mathscr{A}_n$, $\mathscr{B}_n$, and $\mathscr{C}_n$ with $n \geqslant 3$.

3 Infinite shapes with finite cut

The self-assembly of shapes in oritatami systems is fundamentally different from the self-assembly of shapes in the Tile Assembly Model due to the fact that every configuration in an OS has a routing that is a linear path of beads. To illustrate this difference, let us say an infinite shape has a *finite cut* if there is a finite subset of points K in S such that $S \setminus K$ contains at least two *infinite* connected components, S_1 and S_2. As every path going between S_1 and S_2 has to pass through the cut K of finite size, after a finite number of back and forth passes it will no longer be possible and the routing will not be able to fill at least one of S_1 or S_2. Furthermore, since any scaling of S has also a finite cut, scaling cannot help here and we conclude that:

Theorem 2. *Let S be an infinite shape having a finite cut. Then for any scaling scheme Λ and any OS $\mathcal{O}$, $\Lambda(S)$ is not foldable in $\mathcal{O}$.*



(a) (b) (c) (d) (e) (f) (g)

Fig. 3: (a) An example shape which cannot be self-assembled by an oritatami system without being scaled (b) Small example shape, (c) scaled to $\mathscr{A}_2$ and rotated version, (d) after addition of first gadget, (e) after second gadget, (f) after third gadget, (g) after fourth gadget and completion of HC.

4 Self-assembling finite shapes at scale 2 with linear delay

In this section, we show how to create an oritatami system for building an arbitrary finite shape S at scales $\mathscr{A}_2$, $\mathscr{B}_2$, and $\mathscr{C}_2$, with a delay equal to $|S|$.

The theorem below proves that: every $\mathscr{A}_2$-, $\mathscr{B}_2$- and $\mathscr{C}_2$-upscaled version of a given shape S has a Hamiltonian cycle (HC); and furthermore, presents an algorithm that outputs an OS with delay $|\Lambda(S)| = O(|S|)$ that folds into this cycle from a seed of size 3. The OS relies on set of beads following the HC and custom designed to bind to all of their neighboring beads. Using a delay factor equivalent to the size of the shape, all beads after the first three of the seed are transcribed before they then all lock into their optimal placements along the HC which allows them to form the maximum number of bonds. A schematic overview of the scaling, HC, and bead path is shown in Fig. 3.

Theorem 3. *Let S be a finite shape. For each scale $s \in \{\mathscr{A}_2, \mathscr{B}_2, \mathscr{C}_2\}$, there is an OS $\mathcal{O}_S$ with delay $|\Lambda_s(S)| = O(|S|)$ and seed size 3 that self-assembles S at scale s.*

5 Self-assembling finite shapes at scale $\geqslant$3 with delay 1

All our algorithms are *incremental* and proceed by extending the foldable routing at each step, to cover a new cell, neighboring the already covered cells. They proceed by maintaining a set of *"clean edges"* in the routing, one on every *"available side"* of each cell, from which we can extend the routing. Predictably, this is getting harder and harder as the scale gets smaller and as the edges of the cells overlap. We will present our different scaling algorithms by increasing difficulty: $\mathscr{B}_n$ for $n \geqslant 3$, then $\mathscr{C}_n$ for $n \geqslant 3$, then $\mathscr{A}_n$ for $n \geqslant 5$, then $\mathscr{A}_4$ and finally our most compact scaling $\mathscr{A}_3$.

All the scaling algorithms presented in this section have been implemented in Swift on iOS.[9] All the figures in this section have been generated by this program and reflect its actual implementation.

[9] Our app `Scary Pacman` can be freely downloaded from the app store at `https://apple.co/2qP9aCX` and its source code can be downloaded and compiled from the public Darcs repository at `https://bit.ly/2qQjzy6`.

5.1 Universal tight oritatami system with delay 1

Definition 1. *We say that an OS is* tight *if (1) its delay is* 1*, (2) every bead makes only one bond when it is placed by the folding and there is only one location where it can make a bond at the time it is placed during the folding.*

All the OS presented in this section are tight. Tight OS can be conveniently implemented using the following result:

Theorem 4. *Every tight OS can be implemented using a universal set of* $114 = 19 \times 6$ *bead types together with a universal rule, from a seed of size* 3*.*

In the next subsections, all oritatami systems are tight. We will thus focus on designing routing with a single tight bond per bead, and rely on Theorem 4 for generating the transcript from the designed routing in linear time.

5.2 Key definitions

Consider a shape S and $p_1, \ldots, p_{|S|}$ a *search* of S, i.e. a sequence of distinct points covering S such that for all $i \geqslant 2$, there is a $j < i$ such that $p_i \sim p_j$. W.l.o.g., we require that the nw-neighbor of p_1 does not belong to S so that the n-neighboring cell of $\Lambda(p)$ is empty in $\Lambda(S)$.

Starting from a tight routing covering the cell $\Lambda(p_1)$, our algorithms cover each other cell $\Lambda(p_i)$ in order $i = 2 \ldots |S|$, one by one, by extending the tight routing from a previously covered cell.

Lattice and cell directions. We denote by $\mathcal{D} = \{\mathsf{nw}, \mathsf{ne}, \mathsf{e}, \mathsf{se}, \mathsf{sw}, \mathsf{w}\}$ the set of all *lattice directions* in $\mathbb{T}$, and by $\mathcal{D}^{\hexagon} = \{\mathsf{nw}^{\hexagon}, \mathsf{n}^{\hexagon}, \mathsf{ne}^{\hexagon}, \mathsf{se}^{\hexagon}, \mathsf{s}^{\hexagon}, \mathsf{sw}^{\hexagon}\}$ the set of all *cell directions*, joining the centers of two neighboring cells (see Fig. 2). We denote by $\bar{d}$ the direction opposite to d. We denote by $\mathsf{cw}(d)$ and $\mathsf{ccw}(d)$ the next direction in $\mathcal{D}$ if $d \in \mathcal{D}$ (or in $\mathcal{D}^{\hexagon}$ if $d \in \mathcal{D}^{\hexagon}$), in clockwise and counterclockwise order respectively. For $d \in \mathcal{D}$ (resp. $d \in \mathcal{D}^{\hexagon}$), we denote by $(d)^{\hexagon}$ (resp. $(d)^{\triangle}$) the cell direction (resp. lattice direction) next to d in counterclockwise order, e.g. $(\mathsf{w})^{\hexagon} = \mathsf{sw}^{\hexagon}$ and $(\mathsf{ne}^{\hexagon})^{\triangle} = \mathsf{ne}$.

A cell $\Lambda(p)$ is *occupied* if the current routing covers it, otherwise it is *empty*. Each cell has six *sides*, its $\mathsf{nw}^{\hexagon}$-, $\mathsf{n}^{\hexagon}$-, $\mathsf{ne}^{\hexagon}$-, $\mathsf{se}^{\hexagon}$-, $\mathsf{s}^{\hexagon}$-, and $\mathsf{sw}^{\hexagon}$-sides, connecting each of its six w-, nw-, ne-, e-, se-, and sw-*corners*. Given a cell, its neighboring cell in direction $d \in \mathcal{D}^{\hexagon}$ is called its *d-neighboring cell*. At scale $\mathscr{A}_n$, the d-side of a cell is the $\bar{d}$-side of its d-neighboring cell. At scales $\mathscr{B}_n$ and $\mathscr{C}_n$, we say that the d-side of a cell and the $\bar{d}$-side of its d-neighboring cell are *neighboring sides*.

The *clockwise-most* and *second clockwise-most* edges of the d-side of a cell are the two last edges in $\mathbb{T}$ of this side in the direction $d' = \mathsf{ccw}((d)^{\triangle})$, e.g., if $d = \mathsf{nw}^{\hexagon}$, the two sw-most edges of the $\mathsf{nw}^{\hexagon}$-side of the cell.

Routing Time. At each step of our algorithms, the routing defines a *total order over the vertices* of the currently occupied cells. For every vertex p covered by the routing, we denote by $\mathsf{rtime}(p)$ its rank (from 0 to $|r| - 1$) in the current routing r. We say of two occupied vertices p and q, that p is *earlier* (resp. *later*) than q if $\mathsf{rtime}(p) < \mathsf{rtime}(q)$ (resp. $\mathsf{rtime}(p) > \mathsf{rtime}(q)$).

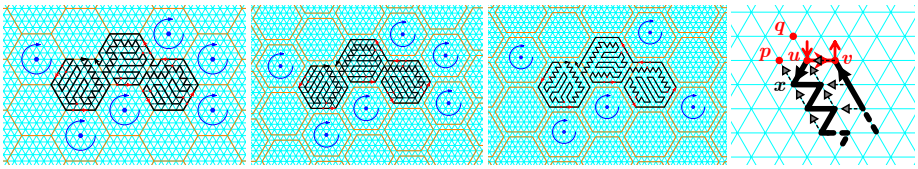


Fig. 4: *Left: Examples of clean edges at scales $\mathscr{A}_5$, $\mathscr{B}_5$ and $\mathscr{C}_5$* – the current routing is displayed in black; some clean edges are highlighted in red together with the two vertices required to be occupied, and earlier than the origin of the edge; the centers of some empty cells are highlighted in blue together with their clockwise orientation. *Right: Extending the routing from a clean edge* – the extension, drawn in black together with its tight bonds, replaces the clean edge $u \to v$ of the current routing r (in red); because p and q are occupied and earlier than u in r, the first bead of the extension is deterministically placed at x by the folding and the zigzag pattern grows south-eastwards, self-supportedly; the way back to v folds by bonding to the initial zigzag; note that all bonds are tight.

Clean edge. The d-side of an occupied cell $\Lambda(p_i)$ is *available* if its d-neighboring cell is empty. Consider an edge uv of $\mathbb{T}$ which belongs to an available d-side of an occupied cell $\Lambda(a)$. Let $\Lambda(b)$ be the empty d-neighboring cell of $\Lambda(a)$. We say that edge uv is *clean* if: (1) it belongs to the current routing; (2) uv's orientation d' in the routing is clockwise with respect to the center $\lambda(b)$ of $\Lambda(b)$, i.e. $d' = \mathsf{ccw}((d)^{\triangle})$ (e.g., $d' = \mathsf{e}$ if $d = \mathsf{s}^{\hexagon}$); and (3) the $\bar{d}'$- and $\mathsf{cw}(\bar{d}')$-neighbors p and q of its origin u are both occupied and earlier than u (e.g., the w- and nw-neighbors of u if $d = \mathsf{s}^{\hexagon}$). p and q are resp. called the *support* and the *bouncer* of the clean edge uv. Fig. 4 gives examples of clean edges for the different scaling schemes. Clean edges are a key component for our algorithms because they are the edges from which the routing is extended to cover a new empty cell. Indeed it is easy to grow a tight path from a clean edge as shown in Fig. 4.

Self-supported extension. We say that a path ρ extending a routing from a clean edge uv with support p is *self-supported* if all its bond are tight and made only with the beads at u, p or within ρ. Self-supported extensions are convenient because they fold correctly independently on their surrounding.

5.3 Design of self-supported tight paths covering pseudo-hexagons

A (a, b, c, d, e, f)-*pseudohexagon* is an hexagonal shape whose sides have length a, b, c, d, e and f respectively from the $\mathsf{ne}^{\hexagon}$- to the $\mathsf{n}^{\hexagon}$-side in clockwise order, i.e. is the convex shape in $\mathbb{T}$ encompassed in a path consisting in a steps to se, b to sw, c to w, d to nw, e to ne and f to e.

Theorem 5. *Let H be a (a, b, c, d, e, f)-pseudohexagon with $a, b, c, d, e, f \geqslant 5$. There is an algorithm* CoverPseudoHexagon *that outputs in linear time a self-supported tight routing covering H from a clean edge placed on either of the two eastmost edges above its $\mathsf{n}^{\hexagon}$-side, and such that it ends with a counterclockwise tour covering the $\mathsf{nw}^{\hexagon}$-, $\mathsf{sw}^{\hexagon}$-, $\mathsf{s}^{\hexagon}$-, $\mathsf{se}^{\hexagon}$- and finally $\mathsf{ne}^{\hexagon}$-sides.*

By Theorem 4, we conclude that all large enough pseudo-hexagons can be self-supportedly folded by a tight OS.

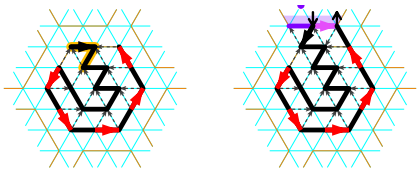


Fig. 5: *The self-supported tight routing extensions for scale $\mathscr{B}_3$:* in light purple, the clean edge used to extend the routing in this cell; in red, the ready-to-use new clean edges in every direction; highlighted in orange, the seed.

5.4 Scale $\mathscr{B}_n$ and $\mathscr{C}_n$ with $n \geqslant 3$

Cells in scaling $\mathscr{B}_n$ and $\mathscr{C}_n$ do not overlap. It is then enough to find one routing extension for the cell (with a clean edge on all of its all available side) from every possible neighboring clean edge.

Scale $\mathscr{B}_n$ is isotropic. Thus, there are only two cases to consider up to rotations: either the cell is the first, or it will plug onto a neighboring clean edge. For $\mathscr{B}_n$, the clean edges that we plug onto, are the *counterclockwise-most* of each side of an neighboring occupied cell. For $n \geqslant 7$, we rely on Theorem 5 to construct such a routing. The two routings for $n = 3$ are given in Fig. 5. We have then:

Lemma 1. *At every step, the computed routing is self-supported and tight, covers all the cells inserted, and contains a clean edge on every available side with the exception of the $\mathsf{n}^{\bigcirc}$-side of the initial cell $\Lambda(p_1)$.*

Proof. This is immediate by induction on the size of the cell insertion sequence by noticing that all the routing extensions are self-supported and tight and that every available side (but the $\mathsf{n}^{\bigcirc}$-side of the root cell) bears a clean edge. □

Note that no insertion will occur on the $\mathsf{n}^{\bigcirc}$-neighboring cell of $\Lambda(p_1)$ because it is assumed w.l.o.g. to be empty. Theorem 4 thus applies and outputs, in linear time, a corresponding OS with 114 bead types and a seed of size 3. The same technique applies at scale $\mathscr{C}_n$ with $n \geqslant 3$ (see appendix p. 33). Fig. 35 and 38 (p. 39) present a step-by-step execution of the routing extension algorithm at scales $\mathscr{B}_3$ and $\mathscr{C}_3$ respectively. It follows that:

Theorem 6. *Any shape S can be folded by a tight OS at all scales $\mathscr{B}_n$ and $\mathscr{C}_n$ with $n \geqslant 3$.*

5.5 Scale $\mathscr{A}_n$ with $n \geqslant 4$

Scale $\mathscr{A}_n$ is the most compact considered in this article. It is isotropic but its cells do overlap. For this reason, we need to provide more extension in order to manage all the cases. The cases $n \geqslant 5$ are the easiest because we can provide a routing for each situation with a clean edge on every available side. Scale $\mathscr{A}_4$ is trickier because only one available side (the latest) may contain a clean edge. Scale $\mathscr{A}_3$ requires a careful management of time and geometry in the routing to ensure that a clean edge can be exposed when needed. Scale $\mathscr{A}_3$ is presented separately in the next subsection. Scale $\mathscr{A}_4$ is deferred to the appendix p. 45.

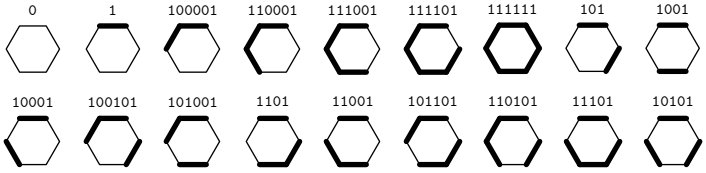


Fig. 6: List of the 18 possible signatures rooted on the clockwise-most side of a segment, placed on the $\mathsf{n}^{\circlearrowleft}$-side.

At scale $\mathscr{A}_n$ with $n \geqslant 5$, the clean edges are located at the *second counterclockwise-most* edge on all of the available sides of every occupied cell (e.g., see leftmost figure on Fig. 4). Our design guarantees this property for every possible empty cell shape. As every occupied cell covers the d-side of all its $\bar{d}$-neighboring empty cells, there are a priori $33 = 1 + 2^5$ different shapes to consider: the completely empty cell, for the first cell inserted; plus the 2^5 possible shapes corresponding to the five possible states occupied/empty for the neighboring cells on which we do not plug. For $\mathscr{A}_n$ with $n \geqslant 5$, our design can extend the routing from any clean edge, regardless of its time or location. This reduces the number of shapes to consider to 14 cases, by rotating the configuration. The following definition allows to identify conveniently the various cases.

Segment and signature. The *signature rooted on $d \in \mathcal{D}^{\circlearrowleft}$* of an empty cell $\Lambda(p)$ is the integer (written in binary) $\mathsf{sig}_d(p) = \sum_{i=0}^{5} s_i 2^i$ where $s_i = 1$ if the $\mathsf{cw}^i(d)$-neighboring cell of $\Lambda(p)$ is occupied, and $= 0$ otherwise. $\mathsf{sig}_d(p) = 0$ if and only if all the neighboring cells of $\Lambda(p)$ are empty; $\mathsf{sig}_d(p)$ is odd if and only if the d-neighboring cell of $\Lambda(p)$ is occupied. A *segment* of an empty cell $\Lambda(p)$ is a maximal sequence of consecutive sides already covered by its neighboring cells. *We will always root the signature of an empty cell on the clockwise-most side of a segment.* With this convention, the two least significant bits of the signature of an empty cell with at least one and at most 5 neighboring occupied cells is always `01`. We are then left with the following possible signatures for an empty cell, sorted by the number of segments around this cell (see Fig. 39 p. 40):

No segment: `0`
1 segment: `1`, `100001`, `110001`, `111001`, `111101`, `111111`.
2 segments: `101`, `1001`, `10001`, when both have length 1; `100101`, `101001`, `1101`, `11001`, when their lengths are 1 and 2; `101101` when both have length 2; `110101`, `11101` when one has length 3.
3 segments: `10101`.

Now, note that the following signatures define an identical cell shape up to a rotation: `10001` $\equiv$ `101`, `101001` $\equiv$ `1101`, `100101` $\equiv$ `11001`, and `110101` $\equiv$ `11101`. (note that symmetries are not allowed because they do not preserve "clockwisevity"). By rotating the patterns, we are then left with designing self-supported tight routings for 14 shapes with clean edges at the second clockwise position of every available side. For $n \geqslant 8$, the 14 pseudo-hexagons are large enough for Theorem 5 to provide the desired routings. The routing extensions for $n = 5, \ldots, 8$ are given in Fig. 40 to 43 in appendix. Scale $\mathscr{A}_4$ is handled similarly

Algorithm 1 Incremental routing algorithm for scale $\mathscr{A}_3$

```
1: procedure FillEmptyCell(centered at: λ(p))
2:     if Λ(p) has no occupied neighboring cell then
3:         Fill Λ(p) with routing 0 from Fig. 7(a), mark the n°-cell as forbidden and return.
4:     while the latest side of Λ(p) is an anomaly do
5:         Fix this anomaly in the corresponding neighboring cell according to the diagram in Fig. 7.
6:     Compute the Λ(p)'s signature rooted on the latest side and extend the path according to the corresponding basic pattern in Fig. 7(a).
```

(see appendix p. 45). We can thus conclude by an immediate induction on the size of the cell insertion sequence, as for scale $\mathscr{B}_n$, that:

Theorem 7. *Any shape S can be folded by a tight OS at scale $\mathscr{A}_n$, for $n \geqslant 4$.*

5.6 Scale $\mathscr{A}_3$

At scale $\mathscr{A}_3$, the sides of each cell have length 2, and no edge can fit in if both neighboring cells are already occupied. We must then pay extra attention to the order of self-assembly, i.e. to time. We define the *time of an occupied side* as the routing time of its middle vertex (its rank in the current routing). In $\mathscr{A}_3$, the clean edges are located at the *counterclockwise-most* position of the available sides of the occupied cells. Our routing algorithm maintains, before each insertion, an invariant for the routing that *combines time and geometry* as follows:

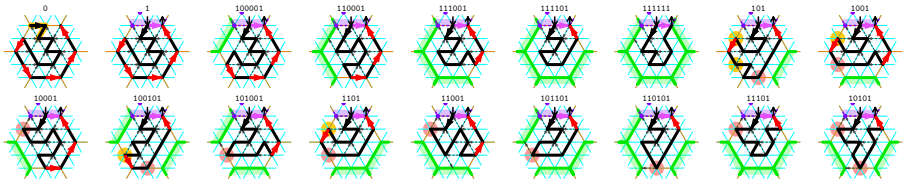
Invariant 1 (insertion) *Around an empty cell, the clockwise-most side of any segment is always the latest of that segment, and its clockwise-most edge is clean.*

As it turns out, we cannot maintain this invariant for every empty cell at every step. The middle vertex of a side violating this invariant is called a *time-anomaly*.

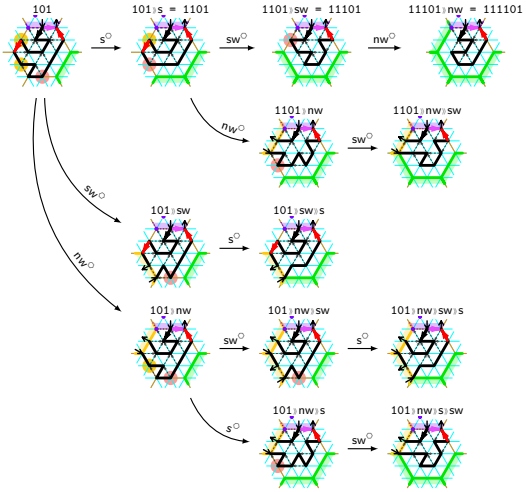
The anomalies around an empty cell are fixed *only* at the step the empty cell is covered by the algorithm. Because fixing anomalies consists in freeing the corresponding side (as if the neighboring cell was empty), without actually freeing the cell, we define the signature rooted on side d of an empty $\Lambda(p)$ slightly differently here, as: $\mathsf{sig}_d(p) = \sum_{i=0}^{5} s_i 2^i$ where $s_i = 1$ if the *vertex at the middle* of the $\mathsf{cw}^i(d)$-side is occupied, and $= 0$ otherwise.

The routing algorithm is described in Algorithm 1 and uses two series of routing extensions: the *basic* patterns in Fig 7(a), and the *anomaly-fixing* patterns in Fig. 7(b-d). There are two types of anomalies: *path-anomalies* (marked as yellow dots) only require a local rerouting inside the cell to become clean; *time-anomalies* (marked as red dots) cannot be turned into clean edge and must be freed according to the diagram in Fig. 7(b-d). Fig. 8 gives a step-by-step construction of a shape which involves fixing several anomalies.

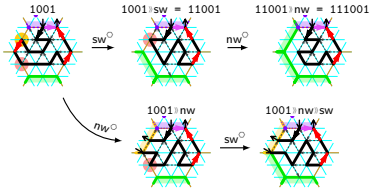
The following key topological lemma and corollary ensure that time- and path-anomalies are very limited and can be handled locally (proofs may be found in Section D.6). And the theorem follows by immediate induction:



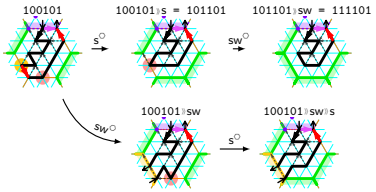
(a) The 18 basic routing extensions.



(b) Fixing anomalies in 101, 1101 and 11101.



(c) Fixing anomalies in 1001 and 11001.



(d) Fixing in 100101 and 101101.

Fig. 7: *Routing extensions at $\mathscr{A}_3$:* in purple, the latest (clockwise-most) clean edge used to extend the routing; in green, the sides already covered, earlier in the routing; in yellow, the side shared with the newly covered neighboring cell after fixing a path-anomaly; the red arrows are the new potential clean edges available to extend the routing; time- and path-anomalies, that need to be fixed to allow extension on that side, are indicated resp. by red and yellow dots; the seed is highlighted in orange in signature 0.

Lemma 2 (Key topological lemma). *At every step of the algorithm, the boundary of each empty area contains exactly one time-anomaly vertex.*

Corollary 1. *The* **while** *loop is executed at most twice, and it fixes: at most one time-anomaly, and at most one path-anomaly. After these fixes, the latest edge around the empty cell is always the clockwise-most of a segment and clean.*

Theorem 8. *Any shape S can be folded by a tight OS at scale $\mathscr{A}_3$.*

6 A shape which can be assembled at delay δ but not $< \delta$

This section contains the statement of Theorem 9 and a high-level description of its proof. For full details, see Section E.

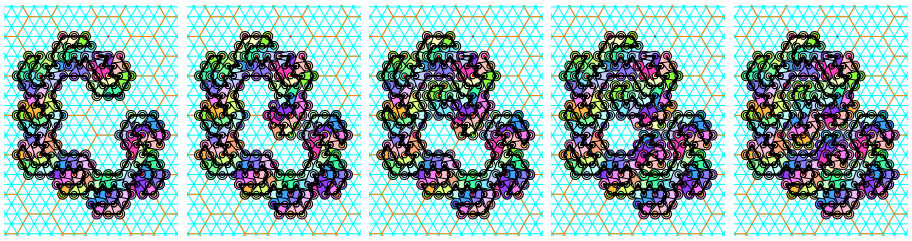


Fig. 8: The step-by-step construction of a routing folding into a shape at scale $\mathscr{A}_3$ according to Algorithm 1, involving fixing anomalies 101 → 101»nw → 101»nw»s → 101»nw»s»sw in the four last steps.

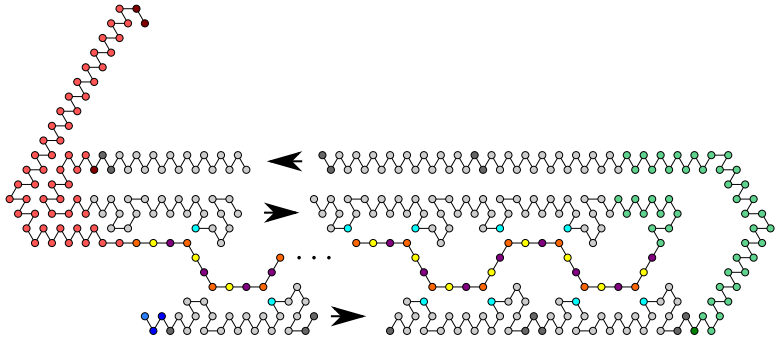


Fig. 9: A depiction of shape S_δ and a routing R'_δ for $\delta = 4$. This can be thought of as a "slice" of the shape (along with a forced routing) which cannot be self-assembled by an oritatami system with delay < 4, but can be assembled by an OS with delay 4. The arrows represent the direction of the directed path in the routing and the different colored beads represent the different gadgets in the routing.

Theorem 9. *Let $\delta > 2$. There exists a shape S_δ such that S_δ can be self-assembled by some OS $\mathcal{O}_\delta$ at delay δ, but no OS with delay δ' self-assembles S_δ where $\delta' < \delta$.*

We prove Theorem 9 by constructing a deterministic OS $\mathcal{O}_\delta$ for every $\delta > 2$, and we define $S_\delta = \mathrm{dom}(C_\delta)$ where $C_\delta \in \mathcal{A}_\square[\mathcal{O}_\delta]$. It then immediately follows that there exists a system at delay δ which assembles S_δ, and we complete the proof by showing that there exists no OS with delay less than δ which can assemble S_δ. A schematic depiction of the shape S_δ (for $\delta = 4$) can be seen in Fig. 9. $\mathcal{O}_\delta$ forms the shape as follows. First a "cave" is formed where the distance between the top and the bottom is δ at specified points. At regular intervals along the top and bottom, blue beads are placed. Once the cave is complete, a single-bead-wide path grows through it from right to left, and every δ beads is a red bead which interacts with the blue. To optimize bonds, each red binds to a blue, which is possible since the spacing between locations adjacent to blue beads is exactly δ, allowing the full transcription length to "just barely" discover the binding configuration. The geometry of S_δ ensures that any oritatami system forming it must have single-stranded portions that reach all the way across the cave. So, in any system with $\delta' < \delta$, since the minimal distance at which beads

can form a bond across the cave is δ, when the transcription is occurring from a location adjacent to one of the sides, no configuration can be possible which forms a bond with a bead across the cave. Thus, the beads must stabilize without a bond across the cave forcing their orientation and so can stabilize in incorrect locations, meaning S_δ isn't deterministically formed.

7 Finiteness of delay-1, arity-1 deterministic oritatami systems

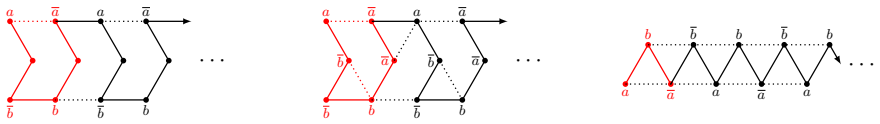


Fig. 10: Deterministically foldable infinite shapes: (Left) A glider at delay-3 and arity-1; (Middle) A glider at delay-2 and arity-2, and (Right) A zigzag at delay-1 and arity-2. Seeds are colored in red. The rule set used in common is complementary: a with $\overline{a}$ and b with $\overline{b}$.

In this section, we briefly argue that oritatami systems cannot yield any infinite conformation at delay 1 and arity 1 deterministically. For more detail, see Section F. The finiteness stems essentially from the particular settings of these parameters. The *glider* is a well-known infinite conformation foldable deterministically at delay 3 and arity 1, shown in Figure 10 (Left), and can be "widened" to adapt to longer delays. The glider can be "reinforced" with more bonds to fold at delay 2 and arity 2 as shown in Figure 10 (Middle). Even at delay 1, arity being 2 allows for the infinite zigzag conformation shown in Figure 10 (Right). Thus, we are left with just two possible settings of delay and arity under which infinite deterministic folding is impossible: arity 1 and delay at most 2. Here we set delay to 1 and leave the problem open at the other setting. Note that infinite *nondeterministic* folding is always possible at arbitrary delay and arity, as exemplified by an infinite transcript of inert beads, which can fold into an arbitrary non-self-interacting path.

At delay 1, a bead cannot collaborate with its successors so that it has to bind to as many (other) beads as possible for stabilization. It can however get stabilized without binding to any other bead only when just one point left unoccupied around. Such a non-binding stabilization requires four beads already stabilized

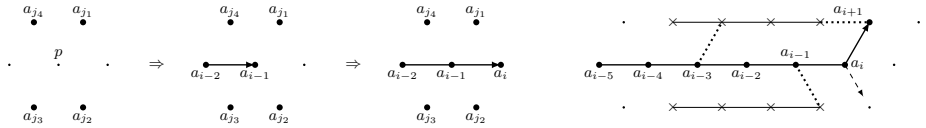


Fig. 11: Stabilization of the bead a_i through a tunnel section formed by the four beads $a_{j_1}, a_{j_2}, a_{j_3}, a_{j_4}$.

around one common point; see Figure 11, where four beads $a_{j_1}, a_{j_2}, a_{j_3}, a_{j_4}$ are at neighbors of the point p. Once the $i-2$-th bead of a transcript, say a_{i-2}, is stabilized at one of the two free neighbors of p and also the next bead a_{i-1} is stabilized at p, then the next bead a_i cannot help but be put at the sole free neighbor of p and the stabilization does not require any binding. Such a structure of four beads around one point is called a *tunnel section*. Tunnel sections can be concatenated into a longer tunnel, as shown in Figure 11 (Right). Tunnels and unbound beads, or more precisely, their *one-time* binding capabilities are resources for an oritatami system to fold deterministically at delay 1 and arity 1. Once bound, a bead cannot bind to any other bead. One tunnel consumes two binding capabilities to guide the transcript into it and to decide which way to lead the transcript to, while it can create only one new binding capability; in Figure 11, a_i does. Thus, intuitively, we can see that the number of binding capabilities is monotonically decreasing, and once they are used up, the system cannot stabilize beads deterministically any more. Formalizing this intuition brings the following theorem.

Theorem 10. *Let Ξ be an OS of delay 1 and arity 1 whose seed consists of n beads, and let w be the transcript of Ξ. If Ξ is deterministic, then $|w| \leq 9n$.*

References

1. E. M. Arkin, S. P. Fekete, K. Islam, H. Meijer, J. S. B. Mitchell, Y. Nú nez Rodríguez, V. Polishchuk, D. Rappaport, and H. Xiao, *Not being (super)thin or solid is hard: A study of grid Hamiltonicity*, Comp. Geom.-Theor. Appl. **42** (2009), no. 6–7, 582–605.
2. M. Y. Chao, M.-C. Kan, and S. Lin-Chao, *RNAII transcribed by IPTG-induced T7 RNA polymerase is non-functional as a replication primer for ColE1-type plasmids in escherichia coli*, Nucleic Acids Res. **23** (1995), 1691–1695.
3. E. D. Demaine, M. L. Demaine, S. P. Fekete, M. Ishaque, E. Rafalin, R. T. Schweller, and D. L. Souvaine, *Staged self-assembly: nanomanufacture of arbitrary shapes with $O(1)$ glues*, Natural Computing **7** (2008), no. 3, 347–370.
4. E. D. Demaine, M. J. Patitz, R. T. Schweller, and S. M. Summers, *Self-Assembly of Arbitrary Shapes Using RNAse Enzymes: Meeting the Kolmogorov Bound with Small Scale Factor (extended abstract)*, STACS 2011, LIPIcs, vol. 9, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011, pp. 201–212.
5. Z. Derakhshandeh, R. Gmyr, A. W. Richa, G. Scheideler, and T. Strothmann, *Universal shape formation for programmable matter*, SPAA 2016, ACM, 2016, pp. 289–299.
6. D. Elliott and M. Ladomery, *Molecular biology of RNA*, 2nd ed., Oxford University Press, 2016.
7. A. Elonen, *Molecular folding and computation*, Bachelor Thesis, Aalto University, 2016.
8. R. P. Feynman, *Feynman lectures on computation*, Westview Press, 1996.
9. C. Geary, P.-E. Meunier, N. Schabanel, and S. Seki, *Folding Turing is hard but feasible*, arXiv:1508.00510v2.
10. ———, *Programming biomolecules that fold greedily during transcription*, MFCS 2016, LIPIcs, vol. 58, 2016, pp. 43:1–43:14.
11. C. Geary, P. W. K. Rothemund, and E. S. Andersen, *A single-stranded architecture for cotranscriptional folding of RNA nanostructures*, Science **345** (2014), no. 6198, 799–804.

12. Y.-S. Han and H. Kim, *Ruleset optimization on isomorphic oritatami systems*, DNA 23, LNCS 10467, Springer, 2017, pp. 33–45.
13. Y-S. Han and H. Kim, *Construction of geometric structure by oritatami system*, DNA24, 2018.
14. H. Isambert, *The jerky and knotty dynamics of RNA*, Methods **49** (2009), 189–196.
15. B. T. U. Lewicki, T. Margus, J. Remme, and K. H. Nierhaus, *Coupling of rRNA transcription and ribosomal assembly in vivo: Formation of active ribosomal subunits in escherichia coli requires transcription of rRNA genes by host RNA polymerase which cannot be replaced by bacteriophage T7 RNA polymerase*, J. Mol. Biol. **231** (1993), no. 3, 581–593.
16. Y. Masuda, S. Seki, and Y. Ubukata, *Towards the algorithmic molecular self-assembly of fractals by cotranscriptional folding*, CIAA, vol. LNCS 10977, 2018.
17. E. C. Merkhofer, P. Hu, and T. L. Johnson, *Introduction to cotranscriptional RNA splicing*, Spliceosomal Pre-mRNA Splicing: Methods and Protocols, vol. 1126, Springer, 2014, pp. 83–96.
18. M. Ota and S. Seki, *Rule set design problems for oritatami systems*, Theor. Comput. Sci. **671** (2017), 26–35.
19. D. Repsilber, S. Wiese, M. Rachen, A. W. Schröder, D. Riesner, and G. Steger, *Formation of metastable RNA structures by sequential folding during transcription: Time-resolved structural analysis of potato spindle tuber viroid (-)-stranded RNA by temperature-gradient gel electrophoresis*, RNA **5** (1999), 574–584.
20. T. A. Rogers and S. Seki, *Oritatami system: A survey and impossibility of simple simulation at small delays*, Fund. Inform. **154** (2017), 359–372.
21. S. Seki, *Cotranscriptional folding: A frontier in molecular engineering – a challenge for computer scientists*, SIAM News **50** (2017), no. 4.
22. D. Soloveichik and E. Winfree, *Complexity of self-assembled shapes*, SIAM J. Comput. **36** (2007), no. 6, 1544–1569.
23. K. E. Watters, E. J Strobel, A. M. Yu, J. T. Lis, and J. B. Lucks, *Cotranscriptional folding of a riboswitch at nucleotide resolution*, Nat. Struct. Mol. Biol. **23** (2016), no. 12, 1124–1133.
24. T. N. Wong, T. R. Sosnick, and T. Pan, *Folding of noncoding RNAs during transcription facilitated by pausing-induced nonnative structures*, PNAS **104** (2007), no. 46, 17995–18000.
25. D. Woods, H.-L. Chen, S. Goodfriend, N. Dabby, E. Winfree, and P. Yin, *Active self-assembly of algorithmic shapes and patterns in polylogarithmic time*, ITCS '13, ACM, 2013, pp. 353–354.

A Omitted contents for Section 2

A.1 Omitted contents for Subsection 2.2

A scaling Λ is *valid* if it preserves the topology of any shape, that is if: (1) for all shape S and $p \in \mathbb{T}$, $p \in S$ if and only if $\lambda(p) \in \Lambda(S)$ (we do not allow a cell to be fully covered by others); (2) for all $p, q \in \mathbb{T}$, we have $p \sim q$ if and only if $\Lambda(p) \sim \Lambda(q)$ (cells are neighbors if and only if their associated points in the original shape are neighbors). We say that a scaling Λ is *fully covering* if every shape S without hole is mapped to a shape $\Lambda(S)$ without hole.[10] Upscaling schemes $\mathscr{A}_n$, $\mathscr{B}_n$ and $\mathscr{C}_n$ are all of them are valid and fully covering.

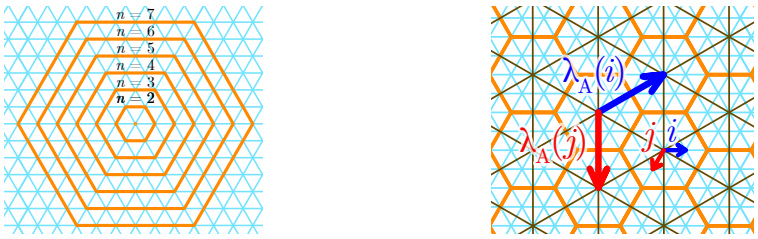


Fig. 12: *Left:* Scaling $\mathscr{A}_n$ cell shapes; they are the concentric balls centered at a vertex in $\mathbb{T}$. *Right:* the cells at scale $\mathscr{C}_3$ (in orange) and the underlying rotated triangular lattice (in brown), by $-30°$, whose vertices are located at the center vertices of the hexagons.

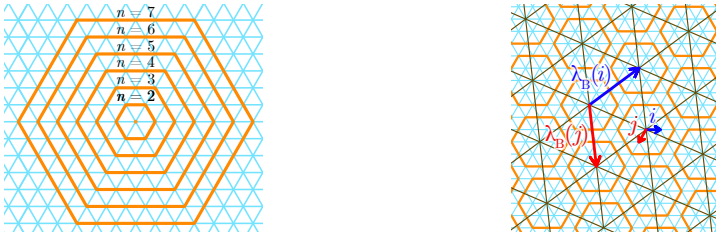


Fig. 13: *Left:* Scaling $\mathscr{B}_n$ cell shapes; they are the concentric balls centered at a vertex in $\mathbb{T}$. *Right:* the cells at scale $\mathscr{C}_3$ (in orange) and the underlying rotated triangular lattice (in brown), by $-30° + \epsilon$, whose vertices are located at the center vertices of the hexagons.

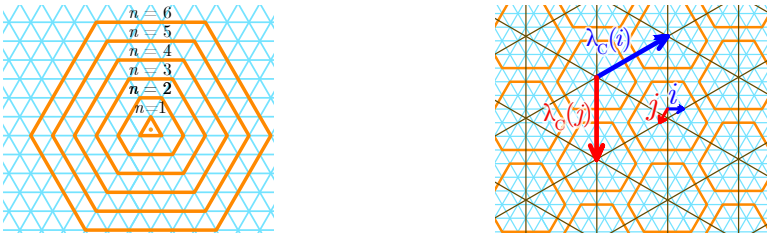


Fig. 14: *Left:* Scaling $\mathscr{C}_n$ cell shapes; they are the concentric balls, in $\mathbb{T}$, centered on the vertex at the center of the triangles of $\mathbb{T}$. *Right:* the cells at scale $\mathscr{C}_n$ (in orange) and the underlying rotated triangular lattice (in brown), by $-30°$, whose vertices are located at the center of the triangle at the center of each cell in the original triangular lattice (the orange dot in the figure to the left).

B Infinite shapes with finite cut technical details

In this section, we provide the details of the proof of Theorem 2.

[10] Recall that a hole of a shape S is a finite non-empty connected component of $\mathbb{T} \setminus S$.

Proof. Let K be a finite subset of S such that $S \setminus K$ contains two disjoint infinite connected components S_1 and S_2. For the sake of contradiction, suppose that $\mathcal{O}$ is an OS and S is foldable in $\mathcal{O}$. As S is foldable in $\mathcal{O}$ by assumption, there must exist a foldable sequence, $\overrightarrow{C} = \{C_i\}_{i=1}^{\infty}$ say, with result S and each C_i a valid foldable configuration in $\mathcal{O}$. Note that, since both S_1 and S_2 are infinite, we can find a sequence of points in S $\{q_i\}_{i=1}^{|K|+1}$ such that the following properties hold. (1) q_i is in S_1 for i odd and q_i is in S_2 for i even, and (2) for some $j \in \mathbb{N}$, q_i is a location for a bead in C_j but not a bead in C_{j-1}. Then, the routing of C_j must contain a path from a bead at location q_{i-1} to a bead at location q_i as a subpath for each i between 1 and $|K|+1$ inclusive. This subpath must must contain a point in K, for otherwise we arrive at a contradiction of the assumption that S is weakly connected, with S_1 and S_2 connected solely by K. Moreover, one can show that C_j must contain beads at i many distinct points in K. Therefore, for $i = |K|+1$, such a configuration C_j contains $|K|+1$ distinct points of K. Hence we arrive at a contradiction. Therefore, S is not foldable in $\mathcal{O}$.

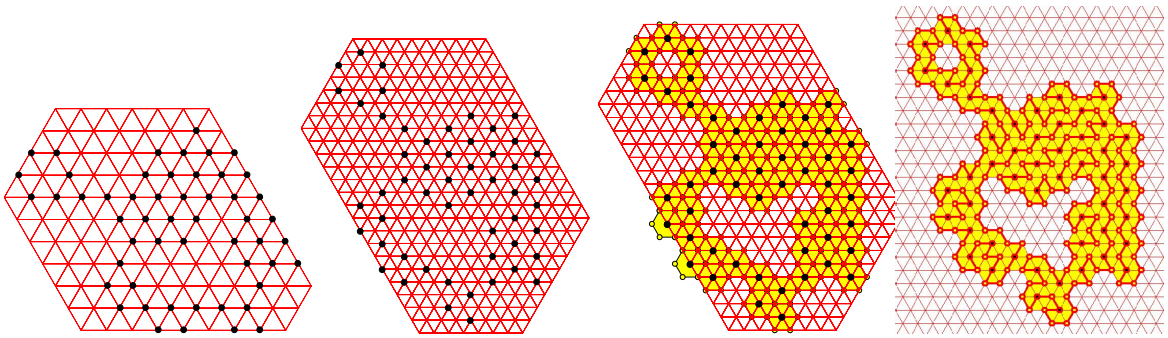
C Self-Assembling Finite Shapes At Small Scale And Linear Delay: Technical Details

In this section, we give details for the proof of Theorem 3. We do this by first proving the case for scaling $\mathscr{A}_2$, and then the cases for scalings $\mathscr{B}_2$ and $\mathscr{C}_2$ are straightforward extensions. For the case of scaling $\mathscr{A}_2$ we first show how to construct Hamiltonian cycles in the scaled shapes.

C.1 Details for construcing Hamiltonian cycles in scaled shapes

Lemma 3. *For any finite shape S in the triangular grid graph, there exists a scaling $\mathscr{A}_2$ of S, say S', such that there exists a Hamiltonian cycle through the points of S'.*

To prove Lemma 3, we give a polynomial time algorithm which, given an arbitrary shape S (i.e. a set of connected points) in the triangular grid, creates a version of S scaled by $\mathscr{A}_2$, S', and a Hamiltonian cycle through S'. (See Figure 15(a) for an example shape.) A program performing this algorithm has been implemented in Python and can be downloaded from http://self-assembly.net/wiki/index.php?title=OritatamiShapeMaker. Please note that in order to remain consistent with that code, in this section we present the scalings as rotated 90° clockwise from the formulation given in Section 2.2, which clearly results in the same shapes, just at a different rotation.



(a) (b) (c) (d)

Fig. 15: (a) Example shape S, (b) the points of example shape S rotated and scaled. It should be noted that the figure shown in (b) is not the shape S'. S' is given in Figure 15(c). (c) Example S' consisting of points of S rotated and scaled, then replaced with scaled points, i.e. pixel gadgets. S' is a 2-scaling of S. (d) A Hamiltonian cycle drawn through the points of the gadgets

Given a finite shape S, the algorithm to obtain S' and a HC in S' is composed of sub-algorithms which we outline here. For detailed algorithms, see Section C.2.

First, the `ORDER-POINTS` sub-algorithm takes a shape S as input and outputs an ordered list L of all of the points in S. `ORDER-POINTS` is as defined in Algorithm 2 (and the subroutines it utilizes are defined in Algorithms 3-8) in Section C.2. After the completion of $L =$ `ORDER-POINTS`(S), the ordered list L contains all of the points in S (this is a standard breadth-first search).

Next, the SCALE-AND-ROTATE-POINTS sub-algorithm takes an ordered list L of all of the points in S and outputs L', which is a scaled and rotated version of L. Algorithm 9 in Section C.2 formally describes this algorithm. The scaling and rotation is essentially equivalent to expanding the distances between pairs of adjacent points from 1 to $2\cos(30°)$ and rotating points 30° clockwise relative to the top-left point (which is the first point in both L and then L' by definition of ORDER-POINTS). See Figure 15(b) for an example shape and the scaled and rotated shape. While L' is an ordered list of these points, the shape S' is defined to simply be the set of points in L'.

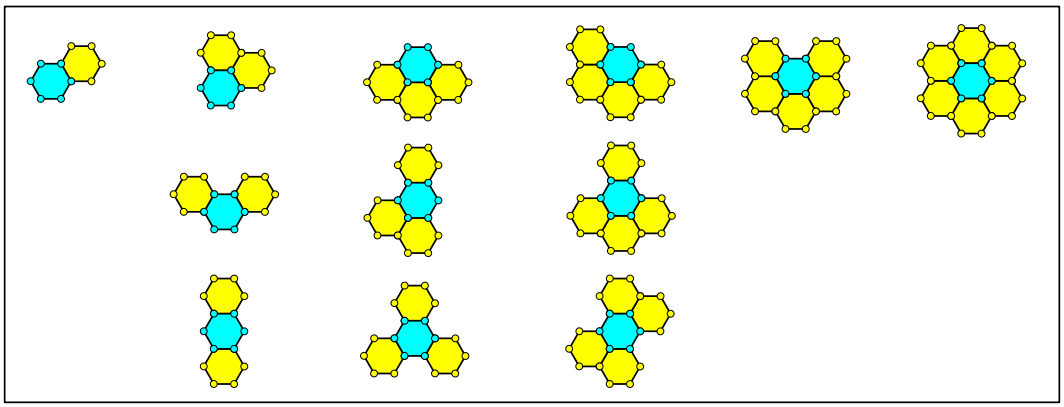


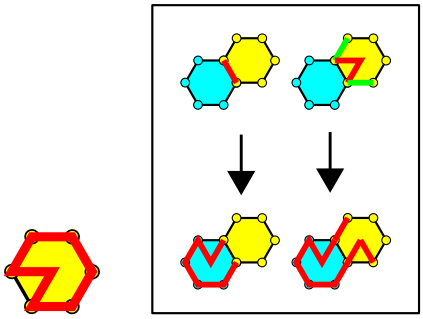
Fig. 16: All possible neighborhoods (i.e. sets of adjacent neighbors) for a newly added point gadget (blue), up to rotation and after the first. In columns from left to right, $1, 2, 3, 4, 5$, then 6 neighbors.

Now, a new shape S_G, and the Hamiltonian cycle (HC), are created by calling $(S_G, HC) =$ ADD-GADGETS(L'). S_G contains a "point gadget" for each point in S', which is simply the point and its 6 adjacent neighbor points (see Figure 15(c) for an example), and they are added in the order specified by L'. Note that adjacent point gadgets share boundary points. The HC is created by first creating a cycle through the points of the first gadget to be added, and then by extending it to include the points of each subsequently added gadget, one by one. As each gadget is added, we first note its neighborhood, which is the arrangement of any neighboring gadgets which were previously added to S_G and the edges of the HC which run through them and along the boundary of the newly added gadget. Modulo rotation and reflection, there are only 12 possible arrangements of neighboring gadgets after the placement of the first. See Figure 16 for depictions of each. We then locate the specific neighborhood scheme (again, modulo rotation and reflection) from the top rows in Figures 17(b) and Figures19-27, and then apply the depicted addition and modification of edges in the HC to extend it to cover all new points of the added gadget, while still covering all previously added points.

We now prove that S_G must have an HC and that one is correctly generated by this procedure. The specific methods for extending the HC into each new gadget are shown in Figures 17(b)-27, and the correctness is maintained due to the following facts:

1. Every gadget (after the first) is added in a location adjacent to at least one existing gadget (and this is guaranteed by the ordering of L created in Algorithm 2).
2. As each gadget is added, it is guaranteed to have on its boundary an existing edge of the HC or a "V" (an example "V" can be seen in Figure 17(a) in the direction which would be facing a neighbor in position 4, as numbered in Figure 18(b)) which includes two of its exterior points. We will refer to this as the *boundary invariant*, and it will be maintained throughout the addition of new gadgets, as will be shown.
3. Figure 16 shows all possible neighborhood configurations, modulo rotation and reflection, into which a new point gadget can be added. This is clear by inspection. Figures 17(b)-27 depict all possible scenarios, modulo rotation, for a point gadget addition. (It is important to note that, in the scenario of each figure, the full set of gadgets adjacent to the newly added gadgets are shown, i.e. empty gadget locations adjacent to the new gadgets are guaranteed to be empty, as there is a figure depicting each scenario where they are filled, up to rotation and reflection).

For each extension of the existing HC into a new point gadget, the necessity is for the replaced edge(s) to be replaced in such a way that the new series of segments (1) has the same end points as the replaced edge(s), (2) all points previously covered by the original edge(s) are covered by the new series of edges,



(a) (b)

Fig. 17: (a) The Hamiltonian cycle (HC) drawn through the first point gadget. Note that all locations adjacent to this gadget have adjacent to them a straight edge or V of the HC, thus maintaining the boundary invariant for any gadget that could be added in one of those locations, (b) The extension of the Hamiltonian cycle through a newly added point gadget (blue) which only has a single neighboring point gadget (yellow) when it is added. (In this and subsequent figures, only the edges of the existing HC which need to be observed and/or manipulated to extend it into the new gadget are shown, in red and/or green.) In the left case, the newly added gadget has a straight edge of the HC adjacent (shown in red, on the top), which can be extended into the new gadget as shown (in red, on the bottom left). Due to the boundary invariant, we know that the only other possible scenario is that shown on the right, in which a V is adjacent to the newly added gadget. Additionally, since we know that the adjacent locations in neighbor positions 0 and 2 are empty (because we're in the case with only a single occupied adjacent location relative to the glue gadget), then we also know that the additional edges colored green must be present in the HC (on the top right), because otherwise with the V present, the points which are shared with the new (blue) gadget couldn't be included in the existing HC. Therefore, the existing red and green HC edges (top) can be replaced with those on the bottom while still including all previously covered points in yellow and now covering all new points in blue. Note that both extensions result in the same previously covered points and same end points for the line segments, thus not disrupting any other portion of the HC, while covering all new points, and also maintaining the boundary invariant by exposing straight edges or V's on the boundary of the newly added gadget.

and (3) all points of the newly added gadget which weren't already included in the HC are now included. The methods for extending the HC while doing that, while also maintaining the boundary invariant, are shown for each possible point gadget addition in Figures 17(b)-27.

We prove the correctness of the generation of the HC through the points of S_G using induction. Our induction hypothesis is the following:

After n points from L' have been added to S_G, then

1. the HC at that time is a valid Hamiltonian cycle through all points in S_G, and
2. for every location l adjacent to S_G into which a point gadget could be validly placed (i.e. at the correct offset for a neighboring gadget), there is an adjacent gadget already in S_G such that the boundary it shares with l consist of either a straight edge occupying both of the shared points, or a "V" (as previously defined) which occupies both shared points. (Note that this is the previously mentioned boundary invariant.)

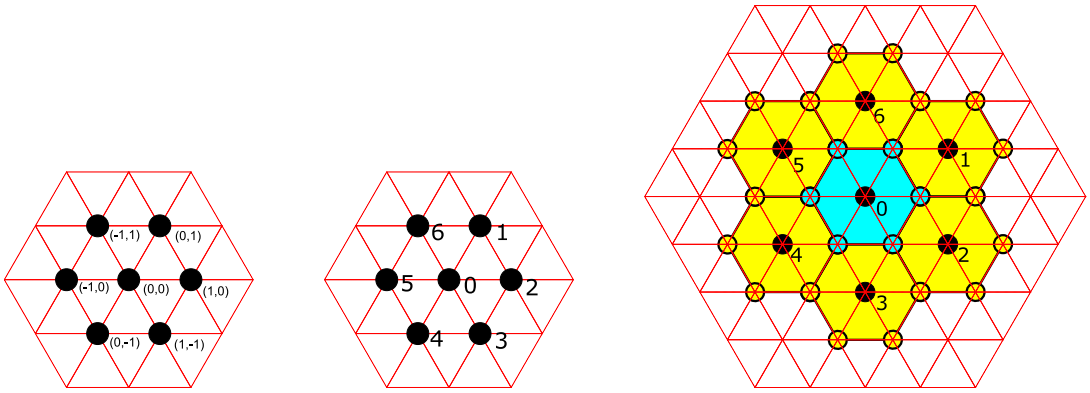
For our base case, we simply inspect the single gadget and simple HC in Figure 17(a)) which exist after the addition of the gadget for the first point from L', and note that this is a Hamiltonian cycle through all 7 points (the outer 6 and the center 1), and that for the valid locations for neighbor gadgets in positions $1, 2, 3, 5$, and 6 (as numbered in Figure 18(c)) the HC through the existing gadget has a straight edge through the potential shared points, and for that in position 4 it shares a "V" through those points. Thus, it holds for the base case.

To prove that if the induction hypothesis holds after n points from L' have been added, it must also hold after $n+1$, we rely on inspection of the scenarios depicted in Figures 17(b)-27. It is easy to verify that in each, after the addition of a gadget, no points previously covered by the HC become uncovered. It is also easy to verify that in each, the points of the newly added gadget (always depicted in light blue) which were not already included in the border of a previous gadget become covered by the HC. This ensures that all points have been covered after the addition of the $(n+1)$th gadget. Finally, it can be seen by inspection that whenever a newly added gadget causes a new neighboring location, which could potentially receive a gadget in the future, to become adjacent to the gadgets of S_G, the boundary which is adjacent to that location contains either a straight edge or a "V". (It is important to note that, oftentimes, some boundaries exposed to adjacent locations contain neither of those patterns. However, whenever that is the case, it is also the case that some other gadget in S_G was already adjacent to

that location and must have shared such a boundary. It is never the case that this previously existing boundary is switched to some other configuration, and thus the boundary invariant is maintained.) This proves that the induction holds, and thus a Hamiltonian cycle is correctly generated through the scaled and rotated points of S.

C.2 Algorithms for the proof of Theorem 3

In this section, Figure 18 gives a visual representation of the numbering schemes for the neighbors of points and gadgets. Then, the algorithms used to calculate an ordering for the points of an input shape, as well as to scale and rotate it, are presented.



(a) (b) (c)

Fig. 18: (a) The coordinate offsets of the neighbors of the point $(0, 0)$, (b) the numbering scheme for the neighbors of a point, (c) the numbering for scaled and rotated points replaced by point gadgets.

Algorithm 2 A procedure to assign an ordering to the points in a shape

```
1: procedure ORDER-POINTS(S)                          ▷ Takes a set of points S
2:     t = GET-TOP-LEFT-POINT(S)
3:     L = {t}
4:     i = 0
5:     n = |L|
6:     while i < n do
7:         p = L[i]
8:         L = L + GET-TOP-NBRS(p, L, S)
9:         L = L + GET-RIGHT-NBR(p, L, S)
10:        L = L + GET-BOTTOM-NBRS(p, L, S)
11:        L = L + GET-LEFT-NBR(p, L, S)
12:        i = i + 1
13:        n = |L|
14:    return L
```

Algorithm 3 A procedure to get the leftmost of the top points of a shape

```
1: procedure GET-TOP-LEFT-POINT(S)                    ▷ Takes a set of points S
2:     S' = {}
3:     p = NULL
4:     for all q ∈ S do
5:         if q == NULL then
6:             p = q
7:         else
8:             if q_y > p_y then
9:                 p = q
10:            else
11:                if q_y == p_y and q_x < p_x then
12:                    p = q
13:    return p
```

Algorithm 4 A procedure to get the specified neighbor of a point if it exists within the definition of a shape

```
1: NBRS = [(0,1),(1,0),(1,−1),(0,−1),(−1,0),(−1,1)]
2: procedure GET-NBR(p, i, S)          ▷ Takes a point p, a neighbor index 0 ≤ i < 6, and set of points S
3:     n = NBRS[i]
4:     q = (p_x + n_x, p_y + n_y)
5:     for all r ∈ S do
6:         if q == r then
7:             return q
8:     return NULL
```

Algorithm 5 A procedure to find the neighbor immediately left of a given point

```
1: procedure GET-LEFT-NBR(p, L, S)     ▷ Takes a point p, an ordered list L, and a set of points S
2:     L_ret = []
3:     p_1 = GET-NBR(p, S, 5)
4:     if p_1 ≠ NULL and p_1 ∉ L then
5:         L_ret = L_ret + [p_1]
6:     return L_ret
```

Algorithm 6 A procedure to find the neighbor immediately right of a given point

```
1: procedure GET-RIGHT-NBR(p, L, S)    ▷ Takes a point p, an ordered list L, and a set of points S
2:     L_ret = []
3:     p_1 = GET-NBR(p, S, 2)
4:     if p_1 ≠ NULL and p_1 ∉ L then
5:         L_ret = L_ret + [p_1]
6:     return L_ret
```

Algorithm 7 A procedure to find the set of neighbors immediately above a given point

procedure GET-TOP-NBRS(p, L, S) ▷ Takes a point p, an ordered list L, and a set of points S
  $L_{ret} = []$
  $p_1 =$ GET-NBR$(p, S, 0)$
  if $p_1 \neq NULL$ and $p_1 \notin L$ **then**
    $L_{ret} = L_{ret} + [p_1]$
  $p_2 =$ GET-NBR$(p, S, 1)$
  if $p_2 \neq NULL$ and $p_2 \notin L$ **then**
    $L_{ret} = L_{ret} + [p_2]$
  return L_{ret}

Algorithm 8 A procedure to find the set of neighbors immediately below a given point

procedure GET-BOTTOM-NBRS(p, L, S) ▷ Takes a point p, an ordered list L, and a set of points S
  $L_{ret} = []$
  $p_1 =$ GET-NBR$(p, S, 4)$
  if $p_1 \neq NULL$ and $p_1 \notin L$ **then**
    $L_{ret} = L_{ret} + [p_1]$
  $p_2 =$ GET-NBR$(p, S, 3)$
  if $p_2 \neq NULL$ and $p_2 \notin L$ **then**
    $L_{ret} = L_{ret} + [p_2]$
  return L_{ret}

Algorithm 9 A procedure to scale and rotate the points in a shape

procedure SCALE-AND-ROTATE-POINTS(L) ▷ Takes an ordered of points L
  $p = L[0]$
  $L' = [p]$
  for $0 < i < |L|$ **do**
    $q = L[i]$
    $d_x = q_x - p_x$
    $d_y = q_y - p_y$
    $s_x = (2 * d_x) + d_y$
    $s_y = d_y - d_x$
    $r = (p_x + s_x, p_y + s_y)$
    $L' = L' \cup \{r\}$
  return L'

Algorithm 10 A procedure to replace all points of an input shape with point gadgets, returning the set of points and a Hamiltonian cycle through them.

procedure ADD-GADGETS(L) ▷ Takes a list of points L
  $S_G = \varnothing$
  $HC = \varnothing$
  for $0 < i < |L|$ **do**
    $S_G = S_G \cup \{L[i]\}$
    for $0 \leq i < 6$ **do**
      $n = NBRS[i]$
      $q = (p_x + n_x, p_y + n_y)$
      if $q \notin S_G$ **then**
        $S_G = S_G \cup \{q\}$
    $HC =$ EXTEND-HC$(L[i], S_G, HC)$
  return (S_G, HC)

Explicit pseudocode is not provided for the `EXTEND-HC` procedure due to its greater complexity[11], but its general functionality is to first inspect the nodes of S_G and the current edges of the HC to determine the pattern of the edges of any gadgets neighboring the newly added gadget. After determining the number of neighbors, their relative arrangement, and the configuration of their edges, it simply compares the pattern to those seen in Figures17(a),17(b), and 19-27. Once it finds a match (perhaps after rotation and/or reflection), which it is guaranteed to find since those figures depict all possible scenarios which are possible due to the way the points are added and the HC is extended, it then extends the HC as depicted in the matching figure. After gadgets have been extended to all points in L', the HC will be correctly completed.

C.3 Extension of the HC for the proof of Theorem 3

In this section, we provide graphical representations of the methods for extending the HC into gadgets as they are added in order. Figures 27-19 show gadgets being added into neighborhoods with 2,3,4,5, and 6 existing neighbor gadgets.

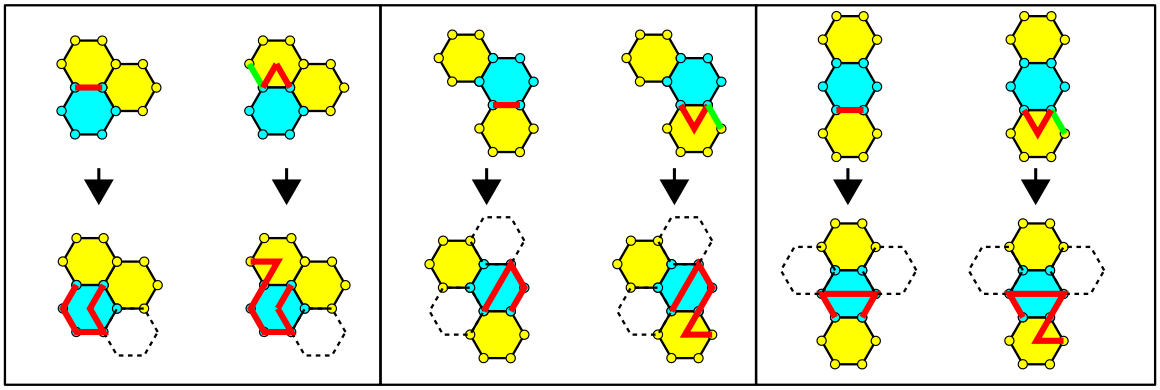


Fig. 19: The extension of the Hamiltonian cycle through a newly added point gadget (blue) which has exactly two neighboring point gadgets (yellow) when it is added. (The same principles and manipulations are used as for the cases in Figure 17(b).) For the adjacency configuration in the left box, if one of the adjacent gadgets has a solid edge on the boundary on the edge of the blue gadget, the left option is taken (symmetrically if it is the upper right neighbor). Otherwise, one of them must have a V adjacent and the right option is taken. This is also how each of the other two possible adjacency configurations are handled. Note that all necessary points are covered and line segment end points are maintained in all scenarios, so we must now verify that the boundary invariant is maintained. Adjacent locations for potential future neighboring gadgets are shown outlined with dashed boundaries if the perimeter of the newly placed blue gadget does not contain either a straight edge or a V on its boundary. However, for each such location, before the addition of the new gadget (blue), that location was already adjacent to a point gadget contained within the shape, and thus by the induction hypothesis it must already have adjacent to it one of the necessary edge configurations (none of which were modified during the current gadget addition). Therefore, the boundary invariant is maintained because at least one edge of each such location will have edges with the necessary configuration.

We now show that an HC can similarly be created through S at scaling $\mathscr{B}_2$.

Lemma 4. *For any finite shape S in the triangular grid graph, there exists a scaling $\mathscr{B}_2$ of S, say S', such that there exists a Hamiltonian cycle through the points of S'.*

The proof of Lemma 4 is a trivial modification of the proof of Lemma 3 which replaces all cases of extending the HC into a new cell with the three cases shown in Figure 28. Since the sides of cells do not share points, it is much easier to add new cells while maintaining the HC, and the only cases to be considered are handled as shown in that figure.

Finally, we show that an HC can similarly be created through S at scaling $\mathscr{C}_2$.

Lemma 5. *For any finite shape S in the triangular grid graph, there exists a scaling $\mathscr{C}_2$ of S, say S', such that there exists a Hamiltonian cycle through the points of S'.*

[11] However, a version which has been implemented in Python can be downloaded from http://www.self-assembly.net

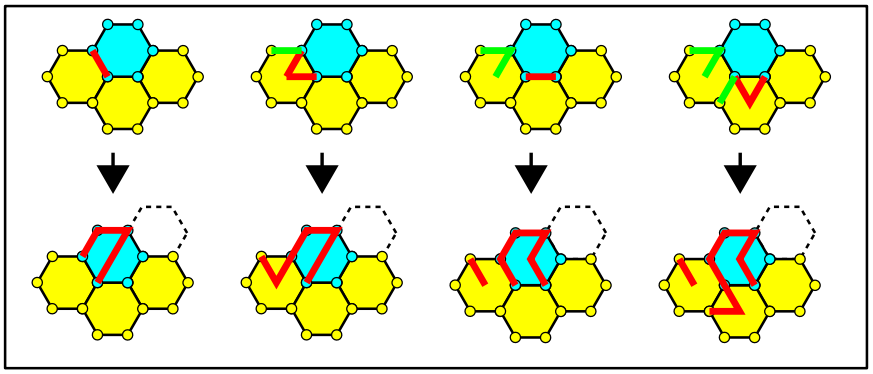


Fig. 20: The extension of the Hamiltonian cycle through a newly added point gadget (blue) which has exactly three neighboring point gadgets (yellow), in the first of three possible configurations, when it is added. (The same principles and manipulations are used as for the cases in Figure 17(b).) It is important to note that the gadget used in these scenarios are selected in preference from left to right. Note that this guarantees the existence of the green segments in the third and fourth column.

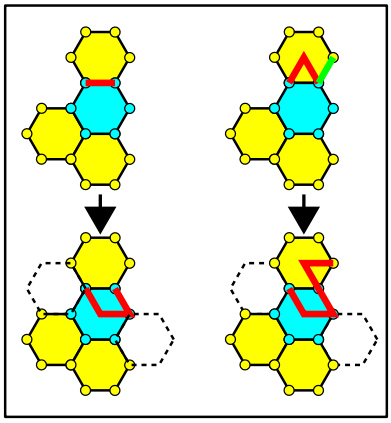


Fig. 21: The extension of the Hamiltonian cycle through a newly added point gadget (blue), which has exactly three neighboring point gadgets (yellow), in the second of three possible configurations, when it is added. (The same principles and manipulations are used as for the cases in Figure 17(b).) Note that it suffices to only consider these two case for the following reason. Assume that the yellow gadget at the top does not share a straight edge of the HC with the blue gadget. Since the yellow gadget does not have a gadget to its southeast, the edge between its southeast point and central point must be in the HC. By the same argument, the absence of a gadget to its southwest implies the inclusion of the edge between its southwest point and central point in the HC. Consequently, the yellow gadget provides a "V" towards the blue gadget. Hence we need only consider the two cases depicted here.

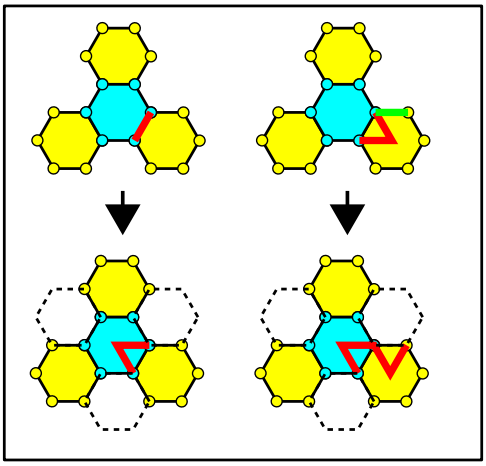


Fig. 22: The extension of the Hamiltonian cycle through a newly added point gadget (blue) which has exactly three neighboring point gadgets (yellow), in the third of three possible configurations, when it is added. (The same principles and manipulations are used as for the cases in Figure 17(b).)

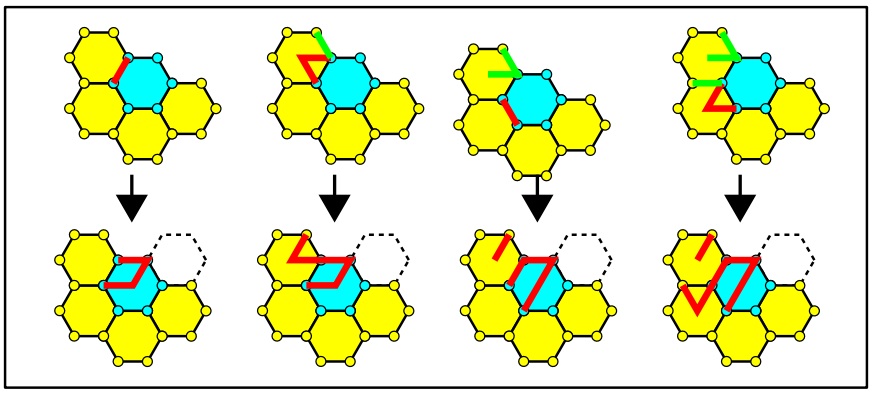


Fig. 23: The extension of the Hamiltonian cycle through a newly added point gadget (blue) which has exactly four neighboring point gadgets (yellow), in the first of three possible configurations, when it is added. Again, it is important to note that the gadget used in these scenarios are selected in preference from left to right. Note that this guarantees the existence of the green segments in the third and fourth column.

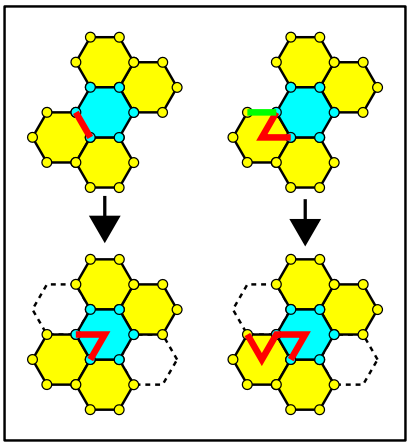


Fig. 24: The extension of the Hamiltonian cycle through a newly added point gadget (blue) which has exactly four neighboring point gadgets (yellow), in the second of three possible configurations, when it is added. (The same principles and manipulations are used as for the cases in Figure 17(b).)

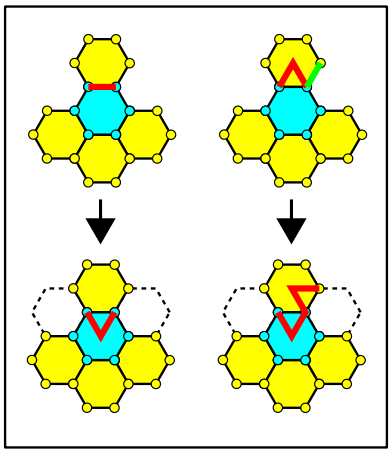


Fig. 25: The extension of the Hamiltonian cycle through a newly added point gadget (blue) which has exactly four neighboring point gadgets (yellow), in the third of three possible configurations, when it is added. (The same principles and manipulations are used as for the cases in Figure 23.) Note that it suffices to only consider the two cases depicted here by the same argument given in caption of Figure 21.

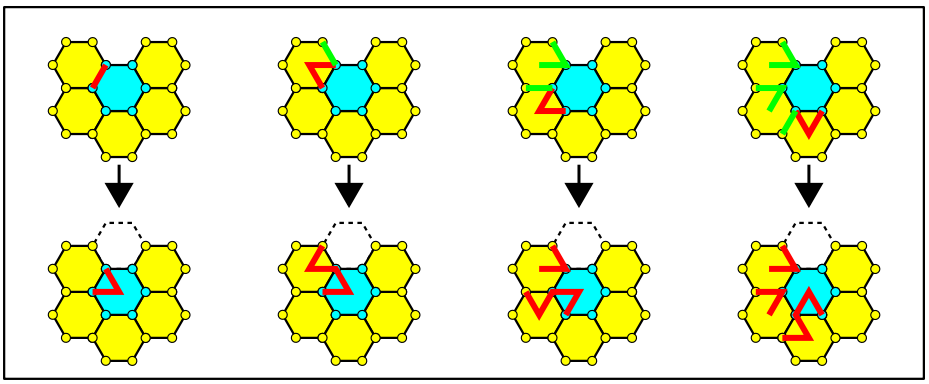


Fig. 26: The extension of the Hamiltonian cycle through a newly added point gadget (blue) which has exactly five neighboring point gadgets (yellow), in the only possible configuration, when it is added. It is important to note that the gadget used in these scenarios are selected in preference from left to left to right. Note that this guarantees the existence of the green segments in the third and fourth column. (The same principles and manipulations are used as for the cases in Figure 23.)

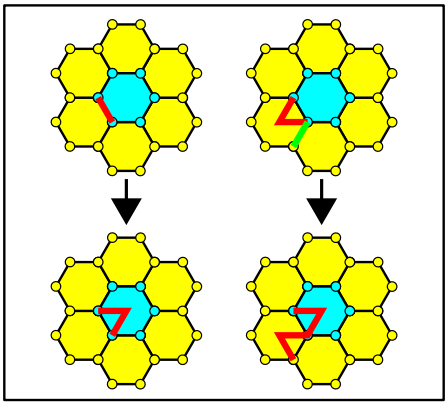


Fig. 27: The extension of the Hamiltonian cycle through a newly added point gadget (blue) which has exactly six neighboring point gadgets (yellow) when it is added. If any of the adjacent gadgets shares a straight edge of the HC on its boundary, the left option is chosen. If none of the adjacent gadgets share a straight edge, then at least one must have a V facing the new gadget. Furthermore, in at least one such gadget with a V facing the new location, an edge along the boundary of that gadget (modulo symmetry, as shown in green on the top right) must also be included in the HC. This is because otherwise, to avoid including such an edge, each V would have to connect to another V with that pattern continuing completely around the blue gadget and creating a cycle which only includes those Vs, which contradicts the fact that a single HC existing before the addition of the blue gadget. Therefore, in this scenario the red and green edges must be included in the HC and can be modified as shown to extend the HC into the single new additional point.

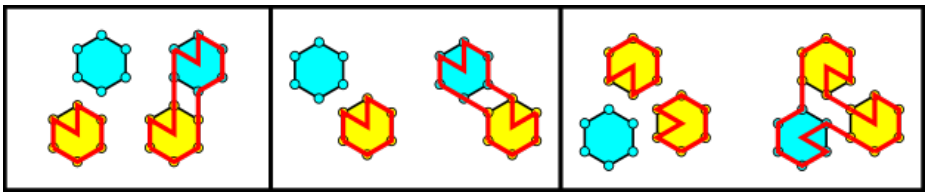


Fig. 28: The extension of the Hamiltonian cycle through a newly added point gadget (blue) in scaling $\mathscr{B}_2$. (left) The first choice is taken if at least one neighbor of a newly added pixel gadget exposes an adjacent flat side. Simply rotate the new pixel gadget so that flat walls face each other and connect them through the four points of those sides. (middle) If all neighbors have adjacent sides exposing "V"s but there are no two which are adjacent to each other, perform the extension shown which changes the exposed side of the existing neighbor which is closest from a flat side into a "V". However, since there was not a mutual neighbor for that gadget and the newly added gadget, that shared adjacent location must be empty, and if a pixel gadget is ever added there later, it can connect via the flat side of this newly added gadget. (right) If all neighbors expose "V"s and two of them are adjacent to each other, extend the HC as shown, which modifies no other exposed sides of the existing pixel gadgets.

The proof of Lemma 5 is an even more trivial modification of the proof of Lemma 3 which replaces all cases of extending the HC into a new cell with the simple observation that every pixel gadget can be of the shape shown in Figure 29, and that every new pixel gadget is able to place a flat side of its pattern adjacent to that of a flat side of a neighbor, allowing the HC to be extended into the new gadget by simply extending two parallel edges between the gadgets through the four points of those sides.

Fig. 29: The extension of the Hamiltonian cycle through a newly added point gadget (blue) in scaling $\mathscr{C}_2$ is trivial since all gadgets can mai.

C.4 Time Complexity

We note that by inspection of the algorithms given in Section C.2 that the algorithm to produce S' and the HC runs in time $O(|S|^2)$.

C.5 Self-Assembling Finite Shapes At Small Scale And Arbitrary Delay: Technical Details

We prove Theorem 3 by splitting each scale factor into its own lemma and proving each separately.

Lemma 6. *Let S be an arbitrary shape such that $|S| < \infty$. There exists OS $\mathcal{O}_S$ such that $\mathcal{O}_S$ self-assembles S at scaling $\mathscr{A}_2$.*

Proof. We prove Lemma 6 by construction. Therefore, assume S is an arbitrary shape such that $|S| < \infty$, and let S' be a 2-scaling of S, produced following the algorithm for the proof of Lemma 3. Let H be the Hamiltonian cycle (HC) through S' found by that algorithm, and for $n = |S'|$ (the number of locations in S'), let $P = p_0, p_1, \ldots, p_{n-1}$ be an ordering of the points of H such that p_0,p_1, and p_2 are points 6, 1, and 2, respectively, (as the points of a pixel gadget are labeled in Figure 18(b), i.e. NW, NE, and E) of the first pixel gadget added by the algorithm.[12] We will break H between p_{n-1} and p_0 to form our transcript sequence. We now define OS $\mathcal{O}_{S'} = (B, w, \heartsuit, \delta, \alpha)$ such that $\mathcal{O}_{S'}$ self-assembles S'. The set of bead types B will contain a unique bead type for each point in H, and thus $|B| = n$. The seed σ will be the first three bead types in locations p_0, p_1, and p_2. The transcript w will be a finite transcript, with $|w| = n - 3$, which enumerates the bead types $p_3, p_4, \ldots, p_{n-1}$, in the order of P. The delay factor δ will be $n - 3$ (i.e. the full length of the transcript), and arity $\alpha = 4$. The rule set $\heartsuit$ is defined by inspecting H overlaid with the ordering of bead types P, and adding a pair of bead types to $\heartsuit$ for every pair of adjacent beads which are not connected via the routing, therefore allowing a bond to form between every pair of adjacent beads not already connected by the routing.

We now prove that $\mathcal{O}_{S'}$ is a deterministic system whose single terminal configuration has shape S'. First, it is obvious by the design of $\mathcal{O}_{S'}$ that the beads can be placed into the shape of S' by exactly tracing the HC through S' with the n beads corresponding to the n locations in S'. We call this the *designed configuration* and note that $\mathcal{O}_{S'}$, specifically the rule set $\heartsuit$, is designed so that when the beads

[12] The only requirement for these three points is simply that they are 3 consecutive connected points in the same pixel gadget in H, and by the definition of the algorithm which creates H, these three points are guaranteed to be such since the first pixel gadget represents the leftmost of the top points of S and therefore there are no neighboring points which could cause those edges to be removed as future pixel gadgets are added.

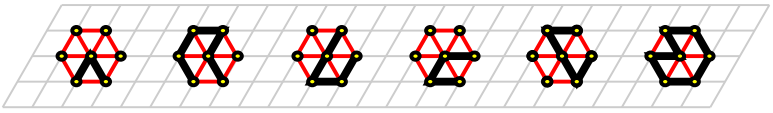


Fig. 30: The beads of pixel gadgets in various configurations, shown with portions of routing (black) connecting some of them and bonds (red) between them.

are laid out in this configuration, every neighboring pair of beads which is not connected by the transcript can form a bond, and furthermore, no bead can form a bond with any other bead other than those which are neighboring in this configuration. Therefore, the designed configuration contains the maximal number of bonds which can be formed.

To complete the proof, we must simply show that there is no configuration other than the designed configuration which can contain as many bonds. We prove this by first noting that in S', there are 7 points which form the pixel gadget corresponding to each point of S (note that points other than the center may be shared by adjacent pixel gadgets), and proving the following series of claims about the beads of each pixel gadget.

Let $k = |S|$ and $0 \leq i < k$, then g_i is the pixel gadget of S' corresponding to point i of $|S|$. In the designed configuration of $\mathcal{O}_{S'}$, there are 7 beads which correspond to each g_i.

Claim. There is exactly one configuration of the beads of each g_i, modulo rotation and reflection, which allows for the formation of the maximum number of bonds which can be formed among those beads, and that is the subset of the designed configuration corresponding to those beads, modulo rotation and reflection.

Proof. To prove this claim, we first note that the bead in the center of the designed configuration of g_i must be (1) connected to two other beads of g_i by transcript connections (by the definition of the transcript) and, in order to form the maximum number of possible bonds, it must form bonds with the other 4 beads of g_i, or (2) in the case of g_0 it may be connected via the transcript to only one other bead of g_0 due to the location where the HC was broken (to form the path of the transcript), but will then be able to form bonds with the other 5 beads of g_0. In order for this single bead to have all of these connections, it must be situated in the center of a hexagon with those beads surrounding it. This guarantees that the 7 beads of the g_i must be arranged in the shape of a pixel gadget (i.e. a hexagon) with the bead in the center matching the center bead of the designed configuration. To prove that the 6 beads around the perimeter are in the same relative locations as in the designed configuration (and can't be reordered), we consider the transcript routing and/or bonds between them. Depending on the arrangement and ordering of pixel gadgets in locations neighboring g_i, there may or may not be a connection between a pair of beads on g_i's perimeter formed by the transcript. However, for any pair that are neighbors in the designed configuration for which there is not such a connection, those two beads can form a bond. Therefore, for the transcript ordering to be maintained as well as for the maximum number of bonds to be formed, the set of all pairs of neighboring beads on the perimeter must match the set of all pairs on the perimeter of the designed configuration, which fixes the ordering of the beads (modulo rotation and reflection). This, along with the fact that the center bead matches that of the designed configuration, proves the claim that the beads of $\mathcal{O}_{S'}$ corresponding to g_i must match the designed configuration, modulo rotation and reflection.

Claim. The beads corresponding to the pixel gadget g_0 (which contains the seed), must have the same orientation as g_0 in S'.

Proof. The proof of this claim follows immediately from the fact that the placement of the first three beads of g_0 are fixed by the definition of the seed. Given that g_0 must have the same configuration as the corresponding beads in the designed configuration (by the previous claim), which matches that portion of S', the fixed location of the first 3 beads forces its orientation, i.e. rotation and reflection, to match that of S'.

Claim. For each $0 \leq i < k$, the beads corresponding to g_k must have the same orientation as g_k in S'.

Proof. We prove this claim by induction on g_i. Our inductive hypothesis is that, given that the beads corresponding to g_i have the same orientation as g_i in S', then the same must hold for those of g_{i+1}. Our base case is g_0, which holds by the previous claim. Given that the beads corresponding to g_i are oriented to match S', and that by definition of H, g_{i+1} must share 2 beads with g_i and that the beads corresponding to g_{i+1} must be in the configuration matching S' (by the first claim), then the only possible orientation for the beads of g_{i+1} is that which matches S'.

Finally, the proof of Lemma 6 follows from the fact that $\mathcal{O}_{S'}$ results in a configuration with exactly as many beads as points in S' and the previous three claims which prove that those beads must fold into a configuration in shape S'.

We now provide the statements of the lemmas for the remaining two scalings, $\mathscr{B}_2$ and $\mathscr{C}_2$, and since they are proved by constructions nearly identical to that for scaling $\mathscr{A}_2$, we just refer to that construction.

Lemma 7. *Let S be an arbitrary shape such that $|S| < \infty$. There exists OS $\mathcal{O}_S$ such that $\mathcal{O}_S$ self-assembles S at scaling $\mathscr{B}_2$.*

The proof of Lemma 7 follows immediately from Lemma 4 (which shows it is possible to form a Hamiltonian cycle H through S', which is the shape S at scaling $\mathscr{B}_2$) and the observation that an OS nearly identical to that constructed for the proof of Lemma 6 can be created to self-assemble S'.

Lemma 8. *Let S be an arbitrary shape such that $|S| < \infty$. There exists OS $\mathcal{O}_S$ such that $\mathcal{O}_S$ self-assembles S at scaling $\mathscr{C}_2$.*

The proof of Lemma 8 follows immediately from Lemma 5 (which shows it is possible to form a Hamiltonian cycle H through S', which is the shape S at scaling $\mathscr{C}_2$) and the observation that an OS nearly identical to that constructed for the proof of Lemma 6 can be created to self-assemble S'.

By Lemmas 6, 7, and 8, Theorem 3 is proved.

D Omitted contents from Section 5: Self-assembling finite shapes at scale $n \geqslant 3$ with delay 1

D.1 Omitted contents from Subsection 5.1: Universal tight OS

Proof (of Theorem 4). Let $[\![m]\!] = \{0, \ldots, m-1\}$. Let $\mathcal{D} = \{\mathsf{nw}, \mathsf{ne}, \mathsf{e}, \mathsf{se}, \mathsf{sw}, \mathsf{w}\}$ be the set of all directions in $\mathbb{T}$. Consider the following *affine* 19-*coloring* of the vertices (i, j) of $\mathbb{T}$:

$$\mathsf{color}(i, j) = (2i + 3j) \bmod 19.$$

For each $d \in \mathcal{D}$, let Δ_d be the difference of the colors (modulo 19) of a vertex and its d-neighbor (as the coloring is affine, Δ_d only depends on d): $\Delta_{\mathsf{se}} = -\Delta_{\mathsf{nw}} = 5$, $\Delta_{\mathsf{sw}} = -\Delta_{\mathsf{ne}} = 3$, $\Delta_{\mathsf{e}} = -\Delta_{\mathsf{w}} = 2$. One can check (see Fig. 31) that every of the 19 vertices of any translation of the hexagon H_2 gets a distinct color.

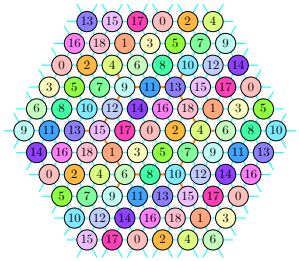


Fig. 31: Affine coloring of $\mathbb{T}$. Note that every vertex in any translation of the hexagon H_3 receives a distinct color in $[\![19]\!]$. The neighbor of a given color in a given direction, always receives the same color.

We consider the bead type set $U = \{(k, d) : k \in [\![19]\!] \text{ and } d \in \mathcal{D}\}$ together with the symmetric rule ♥ defined by: for all (k, d) and (k', d') in U,

$$(k, d) \heartsuit (k', d') \;\Leftrightarrow\; k' = (k + \Delta_d) \bmod 19 \text{ or } k = (k' + \Delta_{d'}) \bmod 19$$

that is to say, if and only if k' is the neighboring color of k in direction d, or k is the neighboring color of k' in direction d'.

Let $\mathcal{O}$ be a tight folding. Let us consider the routing r of the result of the folding of $\mathcal{O}$ starting from an arbitrary vertex in $\mathbb{T}$. We assign to each vertex (i, j) of r, the bead type (k, d) where $k = \mathsf{color}(i, j)$ and d is the direction of the unique bond it makes when it is placed during the folding $\mathcal{O}$. By construction,

the transcript obtained by reading the bead types along the routing r will exactly fold into the same shape: indeed, as the delay is 1, the to-be-placed beads might only get in touch with beads at distance at most 2 from the last placed beads; as every bead within radius 2 gets a different color, the unique location where the to-be-placed bead can make its bond, is uniquely defined by the color of the bead it will connect to, which is in turn uniquely characterized by the color of the to-be-placed bead and the direction of the bond it can make, i.e. by its bead type in U. □

Bead type representation in the figures. A bead with bead type (k, d) will be represented as a small ball of color k inside a link of the same color as the small ball, of color $k' = (k + \Delta_d) \bmod 19$, of its neighbor in direction d it is pointing to. The routing is shown as a thick translucent black line.

Fig. 32: *Bead type representation. Left:* a bead looking for its final position; *Right:* the fully folded transcript.

D.2 Omitted contents from Subsection 5.3: Pseudohexagon routing

Note that we must have: $a + b = d + e$, $b + c = e + f$ and $a + f = c + d$, since $a\,\overrightarrow{\mathsf{se}} + b\,\overrightarrow{\mathsf{sw}} + c\,\overrightarrow{\mathsf{w}} + d\,\overrightarrow{\mathsf{nw}} + e\,\overrightarrow{\mathsf{ne}} + f\,\overrightarrow{\mathsf{w}} = (a - c - d + f)\,\overrightarrow{\mathsf{e}} + (a + b - d - e)\overrightarrow{\mathsf{sw}} = 0$.

Proof (of sketch of Theorem 5). The algorithm proceeds by covering 6 areas numbered from A to F. As illustrated on Fig. 33, there are four cases depending on the parity if the sw- and s-side lengths c and d. Area A consists in a simple se-zigzag pattern, or a se-zigzag pattern with a shift depending on the relative position of the supporting clean edge. Area B consists in a simple zigzag. Area C consists in long ne/sw-switchbacks. The junction to area D is either a simple edge (c even) or a "λ"-shape (c odd). Area D consists in long ne/sw-switchbacks that stick along the shape of area C's switchbacks. The junction to the next area is either a simple edge (c and d even) or a "λ"-shape (c or d odd). Area E does not exist if c and d are even. If c or d are odd, then area E is either a long ne/sw-switchback (c and d odd), or a ne-zigzag (c and d of opposite parity). Then area F consists in a simple counterclockwise tour of the $\mathsf{nw}^{\bigcirc}$- to $\mathsf{ne}^{\bigcirc}$-sides. □

Note that as all the routings computed by CoverPseudoHexagon are self-supported and tight, Theorem 4 applies and provides an OS with 114 bead types in linear time that folds each of them correctly. Note also that in the routings generated by this algorithm, all the edges on the five sides $\mathsf{nw}^{\bigcirc}$ to $\mathsf{ne}^{\bigcirc}$ are clean, except for the five edges originating at a corner.

D.3 Omitted contents from Subsection 5.4: Scale $\mathscr{B}_n$ and $\mathscr{C}_n$ with $n \geqslant 3$

Scale $\mathscr{C}_n$ is anisotropic. Thus, there are three cases to consider up to rotations: either the cell is the first, or it will plug onto a neighboring clean edge that belongs to either a larger or a smaller side. In $\mathscr{C}_n$, the clean edges that we will plug onto, are (1) the *counterclockwise-most* of each smaller side, and (2) the *second clockwise-most* of each larger side, of an neighboring occupied cell. For $n \geqslant 7$, we rely on Theorem 5 to construct such a routing. The tight and self-supported routings for $\mathscr{C}_3$ are listed in Fig. 36 (see Fig. 37 for $n = 4 \ldots 7$).

Fig. 38 (p. 39) presents a step-by-step execution of the routing extension algorithm at scale $\mathscr{C}_3$. Theorem 6 follows, as above, from Theorem 4.

D.4 Omitted contents from Subsection 5.5: scale $\mathscr{A}_n$ with $n \geqslant 5$

We will always root the signature of an empty cell on the clockwise side of a segment. With this convention, The two least significant bits of the signature of an empty cell with at least one and at most 5 neighboring occupied cells is always 01. We are then left with the following possible signatures for an empty cell, sorted by the number of segments around this cell (see Fig. 39 p. 40):

No segment: 0

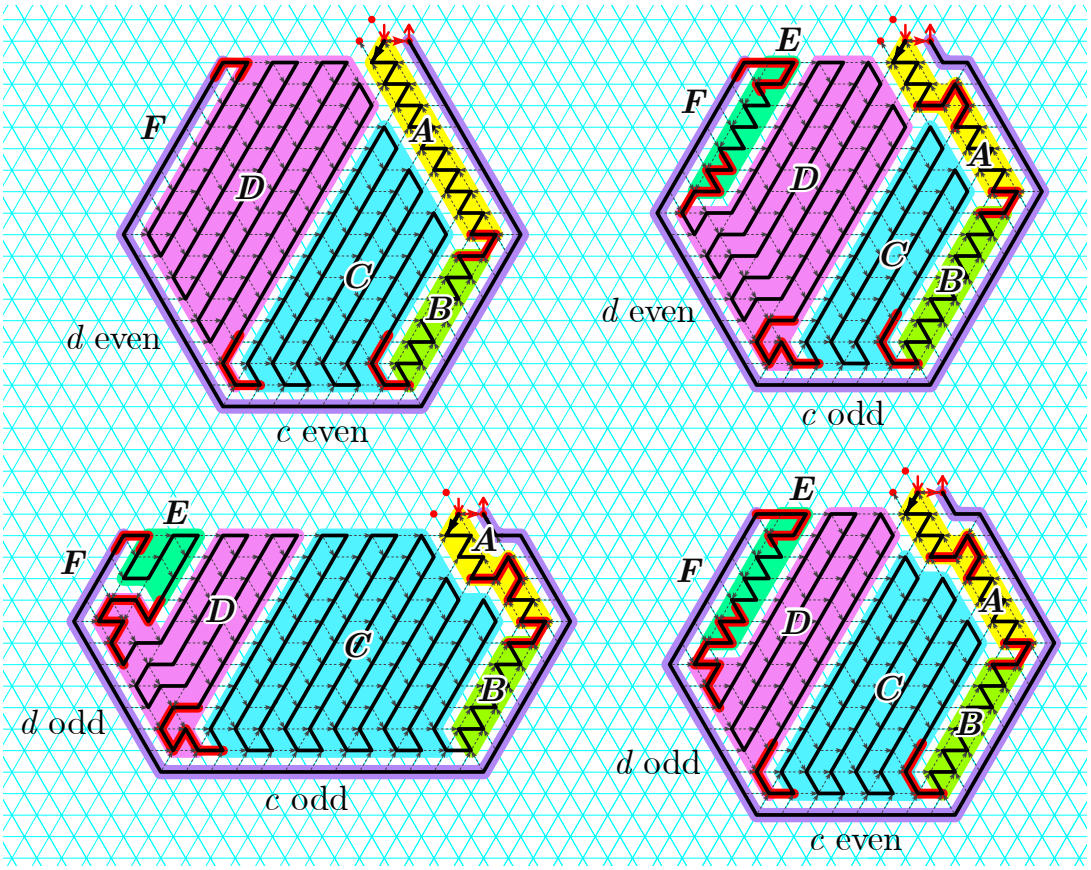
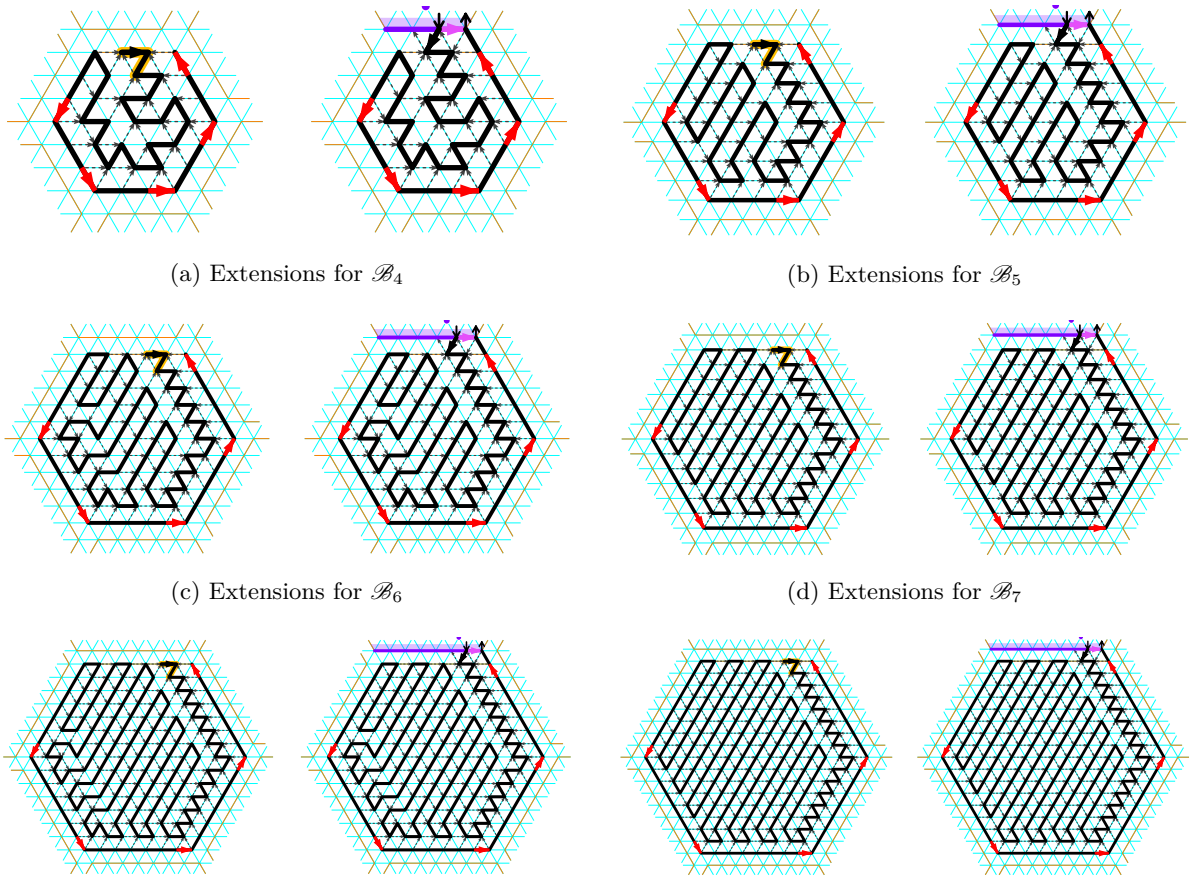


Fig. 33: The four cases to design of the self-supported tight path for large enough pseudo-hexagons. Note that the clean edge in ref is not part of the (convex) pseudo-hexagon, but is the edge upon which the pseudo-hexagon is folded.



(a) Extensions for $\mathscr{B}_4$

(b) Extensions for $\mathscr{B}_5$

(c) Extensions for $\mathscr{B}_6$

(d) Extensions for $\mathscr{B}_7$

(e) Extensions for $\mathscr{B}_8$

(f) Extensions for $\mathscr{B}_9$

Fig. 34: *The self-supported tight routing extensions for scales $\mathscr{B}_4$ to $\mathscr{B}_9$:* in purple, the clean edge used to extend the routing in this cell; in red, the ready-to-use new clean edges in every direction; highlighted in orange, the seed.

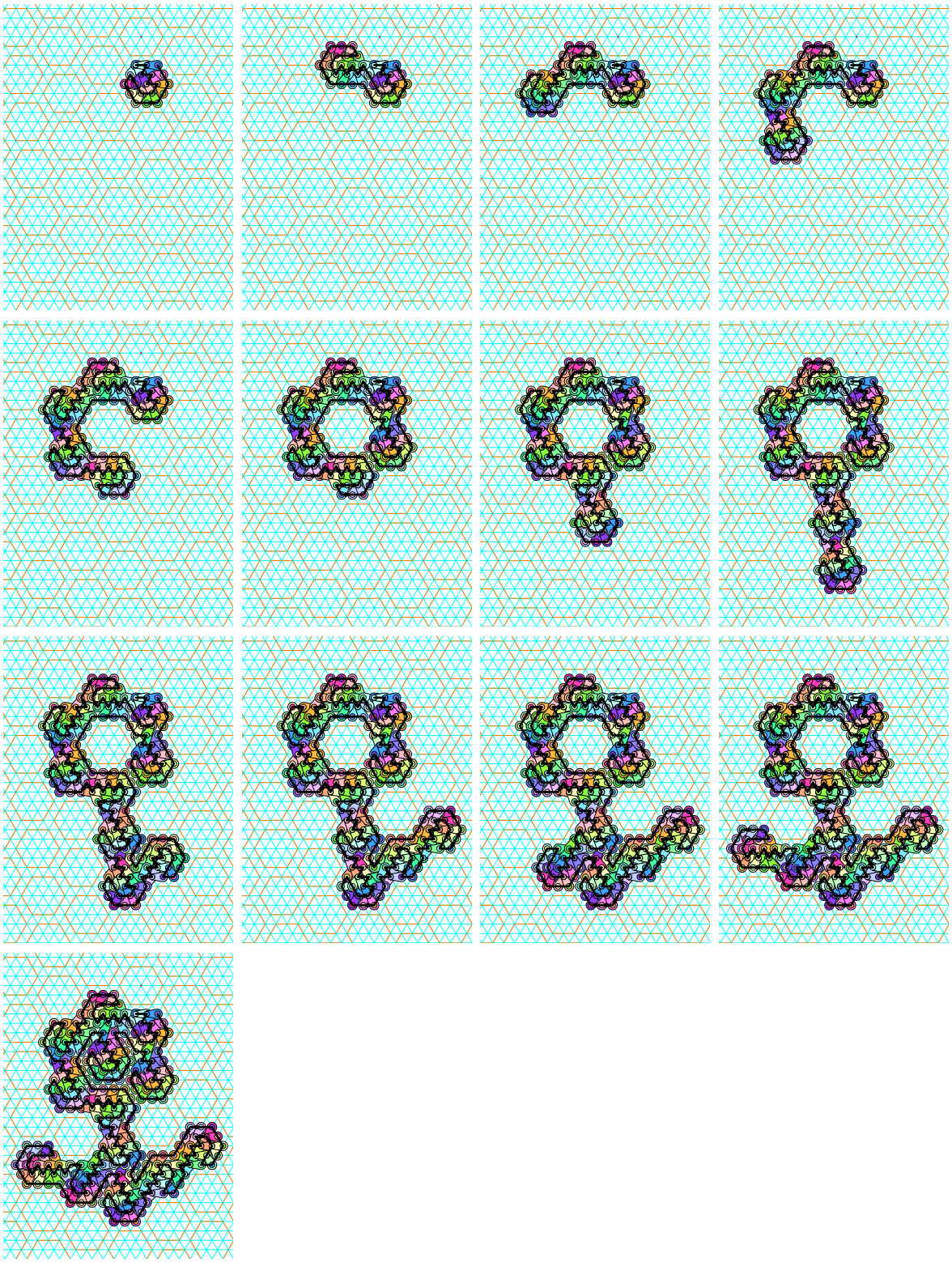


Fig. 35: The step-by-step construction of a routing folding into a flower shape at scale $\mathscr{B}_3$ according to the algorithm in Section 5.4

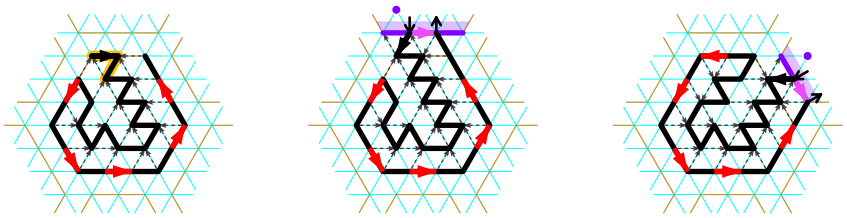
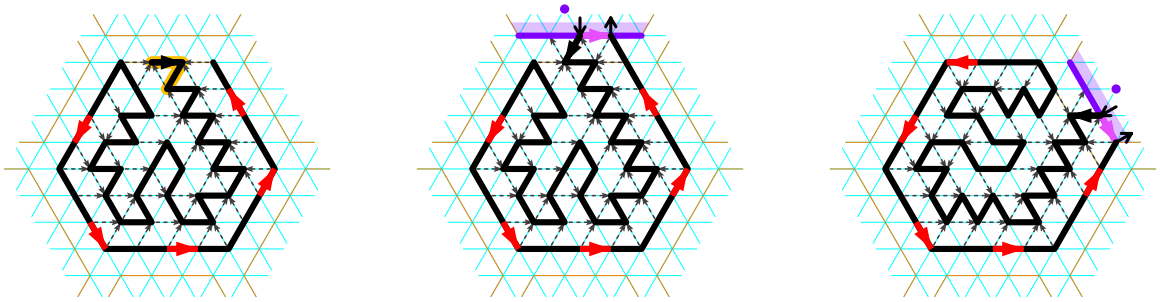
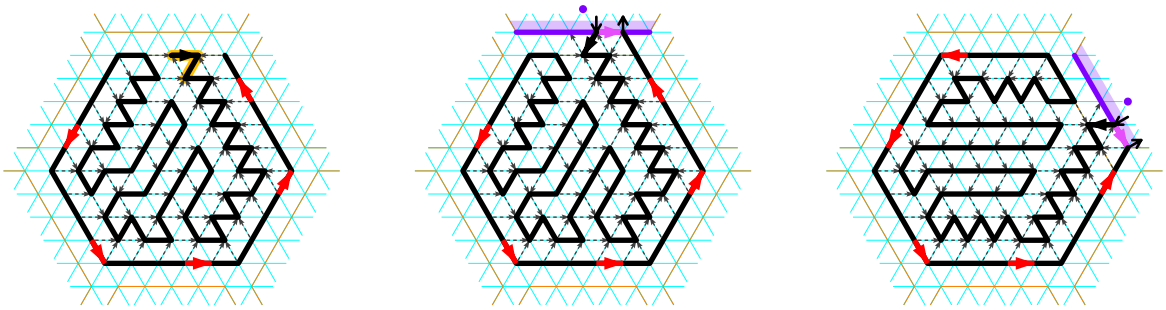


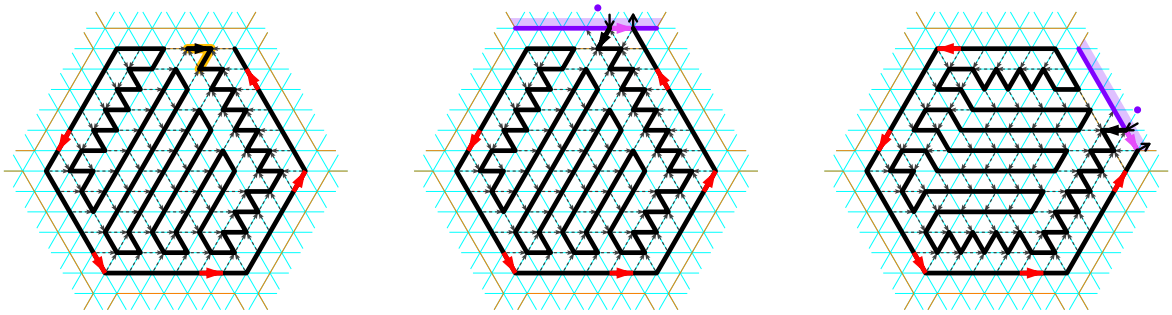
Fig. 36: *The self-supported tight routing extensions for scale $\mathscr{C}_3$:* in purple, the clean edge used to extend the routing in this cell; in red, the ready-to-use new clean edges in every direction; highlighted in orange, the seed.



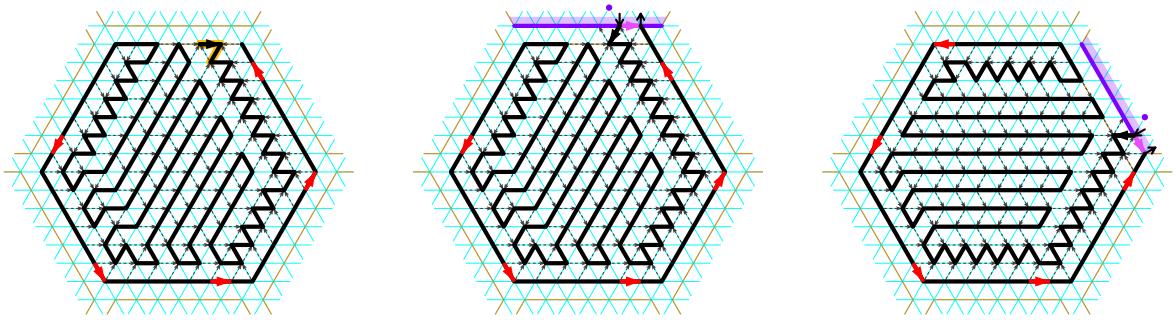
(a) Extensions for $\mathscr{C}_4$



(b) Extensions for $\mathscr{C}_5$



(c) Extensions for $\mathscr{C}_6$



(d) Extensions for $\mathscr{C}_7$

Fig. 37: *Extensions for scales $\mathscr{C}_4$ to $\mathscr{C}_7$:* in purple, the clean edge used to extend the routing in this cell; in red, the ready-to-use new clean edges in every direction; highlighted in orange, the seed.

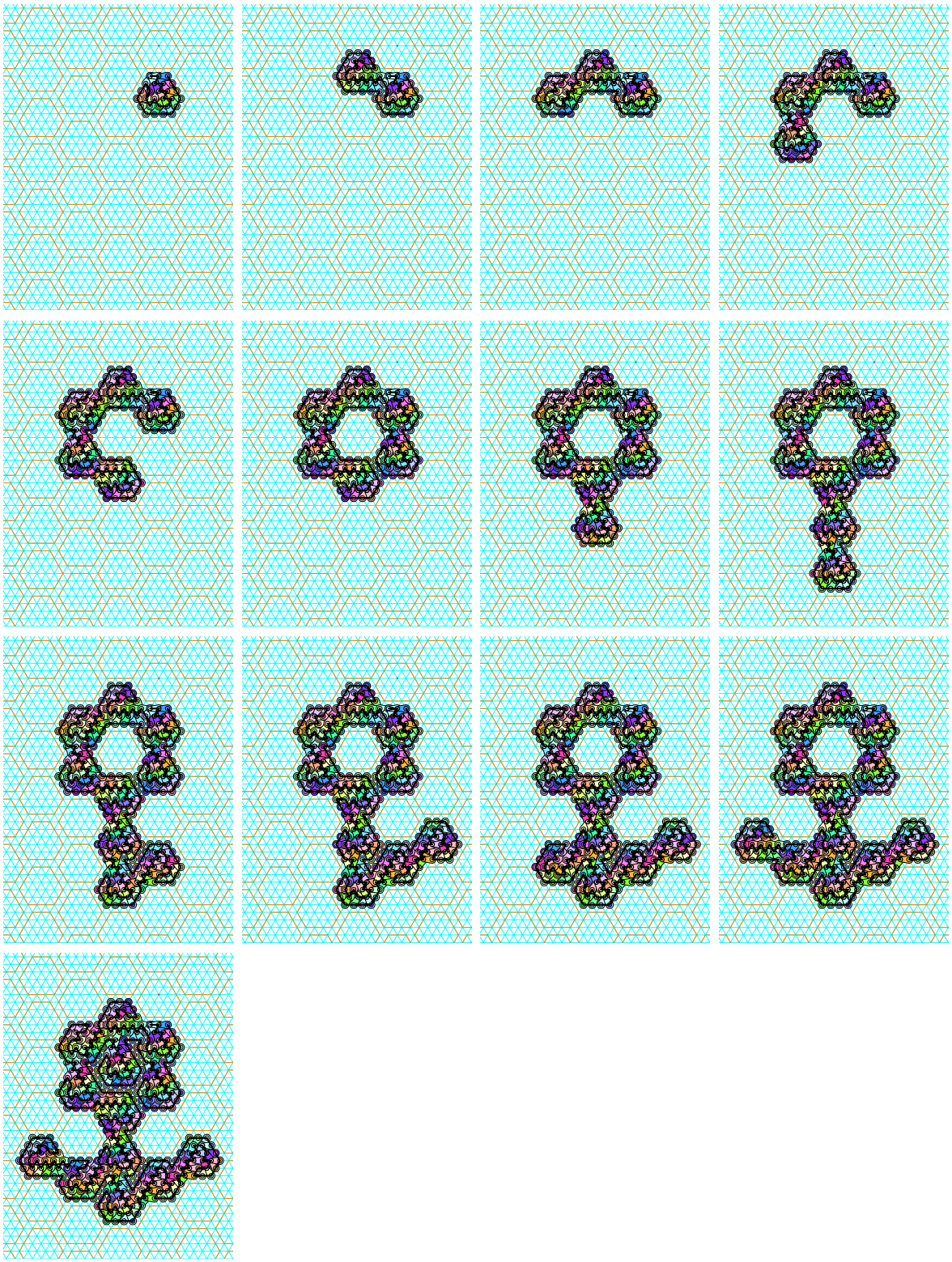


Fig. 38: The step-by-step construction of a routing folding into a flower shape at scale $\mathscr{C}_3$ according to the algorithm in Section D.3

1 segment: 1, 100001, 110001, 111001, 111101, 111111.
2 segments: 101, 1001, 10001, when both have length 1; 100101, 101001, 1101, 11001, when their lengths are 1 and 2; 101101 when both have length 2; 110101, 11101 when one has length 3.
3 segments: 10101.

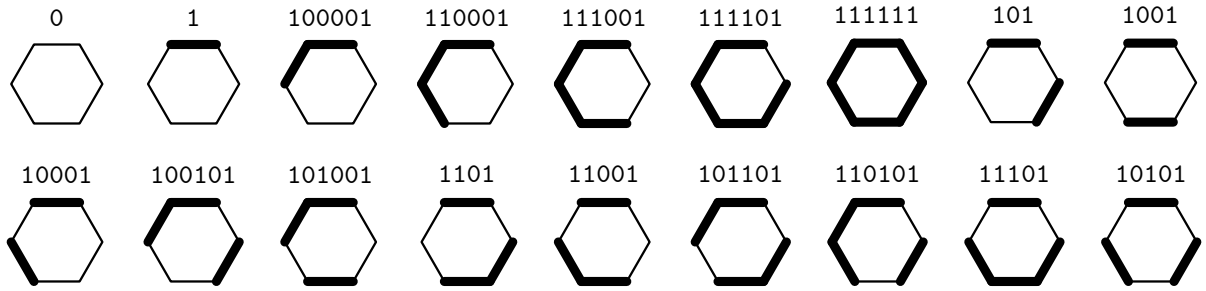


Fig. 39: List of the 18 possible signatures rooted on the clockwise-most side of a segment, placed on the n°-side.

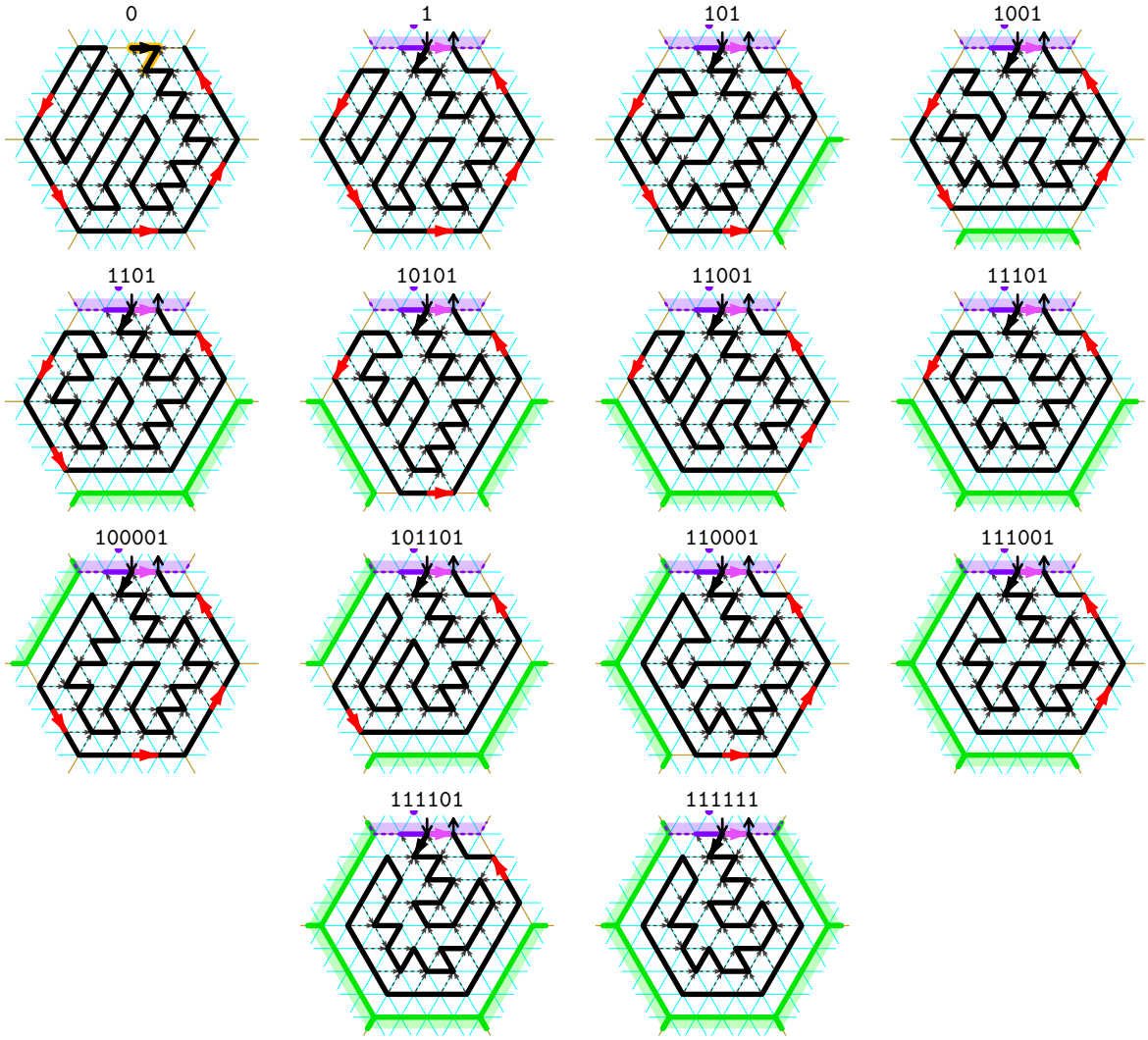


Fig. 40: *The 14 self-supported tight routings at scale $\mathscr{A}_5$:* in purple, the clean edge used to extend the routing in this cell; in green, the sides already covered by an occupied neighboring cell; in red, the new clean edges at the second clockwise-most position on every available side; highlighted in orange, the seed.

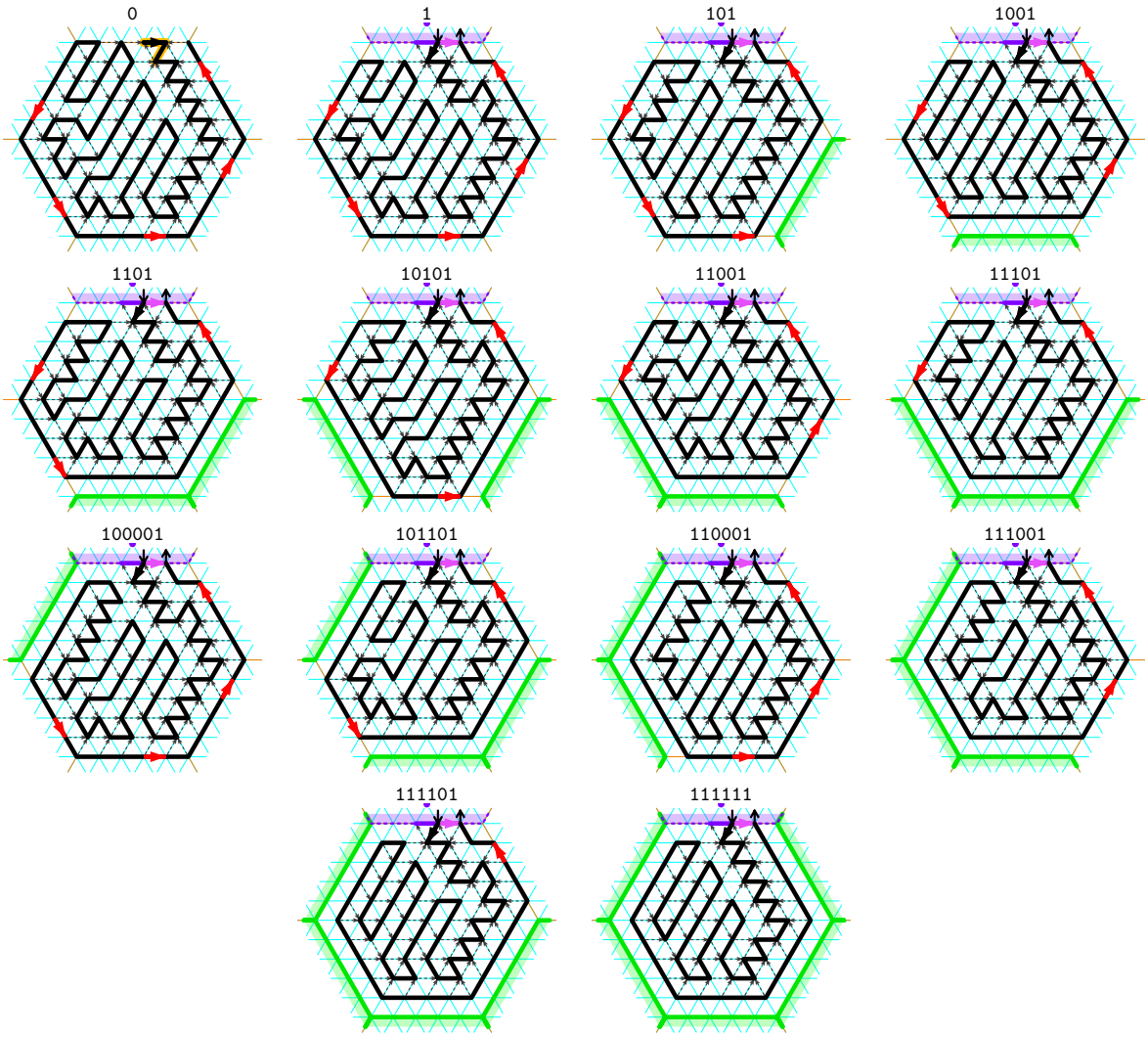


Fig. 41: *The 14 self-supported tight routings at scale $\mathscr{A}_6$:* in purple, the clean edge used to extend the routing in this cell; in green, the sides already covered by an occupied neighboring cell; in red, the new clean edges at the second clockwise-most position on every available side; highlighted in orange, the seed.

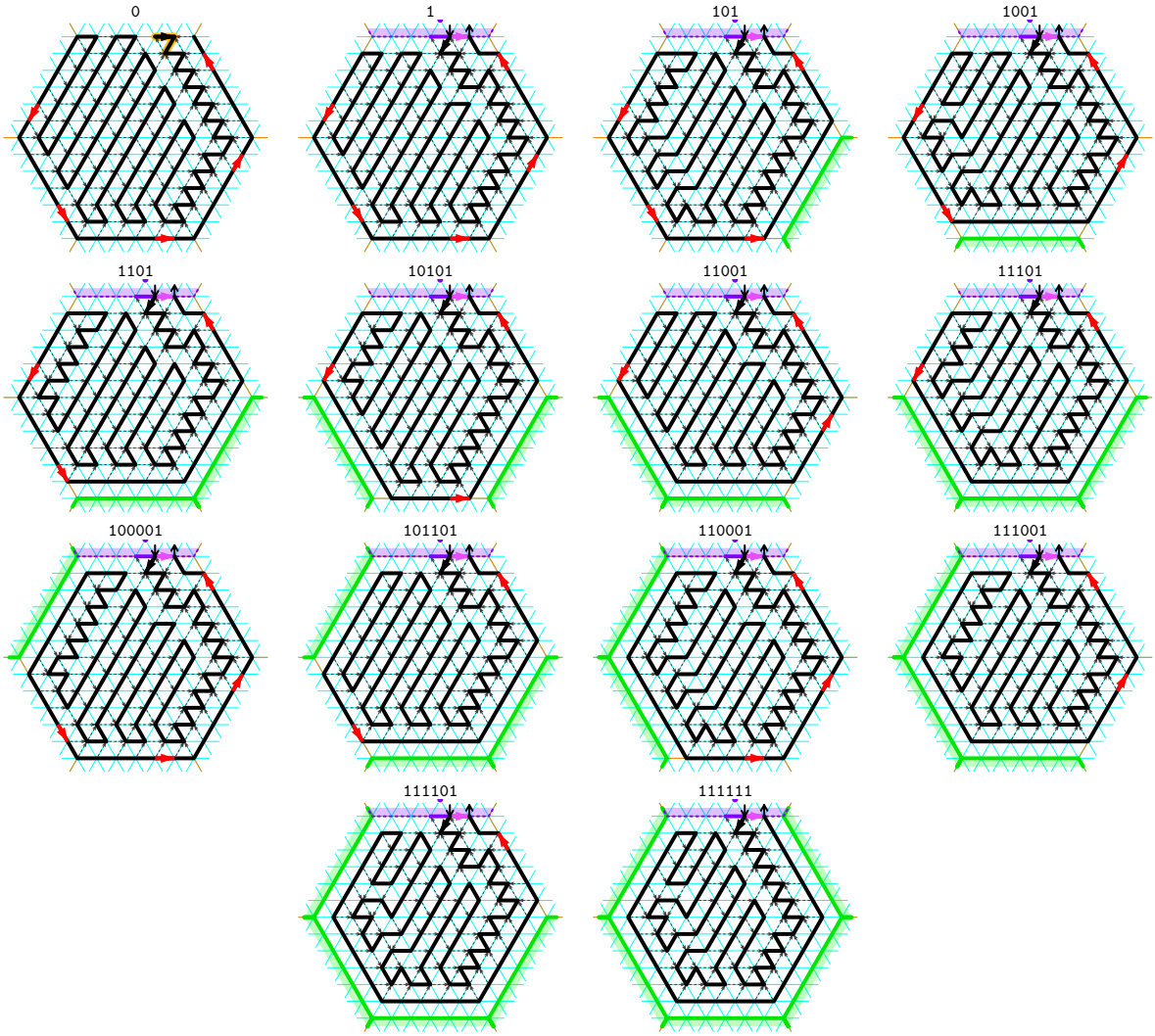


Fig. 42: *The 14 self-supported tight routings at scale $\mathscr{A}_7$:* in purple, the clean edge used to extend the routing in this cell; in green, the sides already covered by an occupied neighboring cell; in red, the new clean edges at the second clockwise-most position on every available side; highlighted in orange, the seed.

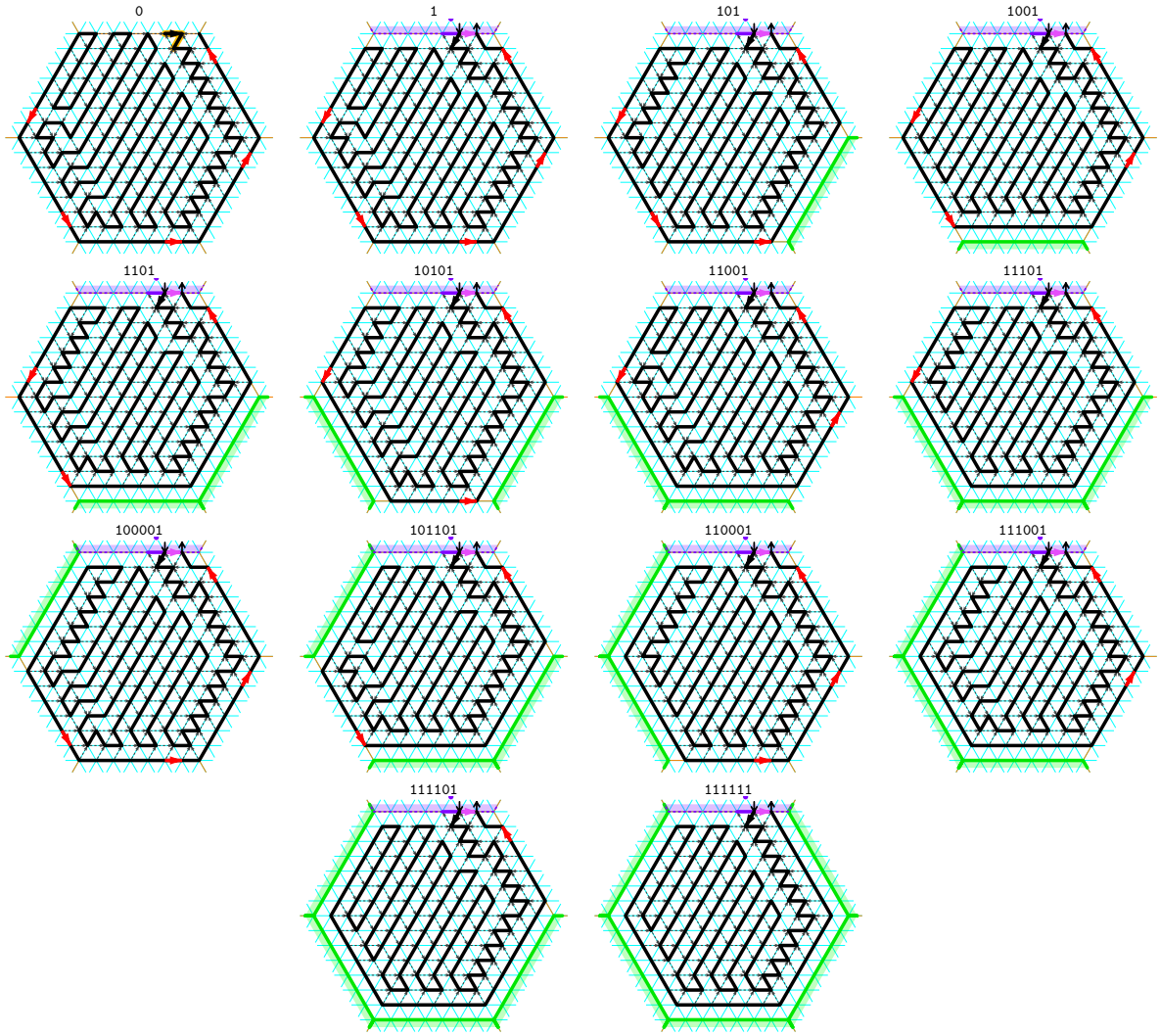


Fig. 43: *The 14 self-supported tight routings at scale $\mathscr{A}_8$:* in purple, the clean edge used to extend the routing in this cell; in green, the sides already covered by an occupied neighboring cell; in red, the new clean edges at the second clockwise-most position on every available side; highlighted in orange, the seed.

D.5 Scale $\mathscr{A}_4$

At scale $\mathscr{A}_4$, the side are 3 edges-long and the support of a clean edge may not belong to the same cell as the edge. We must then need to pay attention to the timing of the clean edges in the path. We solve this issue by always rooting the signature on the side with the *latest* clean edge, where the time of a clean edge is the time of its origin. We are then guaranteed (by an immediate induction) that the support of this clean edge will always have been placed by the folding before the clean edge is folded.

We can however no more freely rotate the signature and must then design the 33 routing extensions. At scale $\mathscr{A}_4$, the clean edges are located at the second clockwise-most edge of every available side of an occupied cell. The 33 routings are shown on Fig. 44. One can check that by immediate induction:

Lemma 9. *At every step, the computed routing is self-supported and tight, covers all the cells inserted, and contains a potential clean edge on every available side with the exception of the* n*-side of the initial cell* $\Lambda(p_1)$*. Furthermore, the potential clean edge of the latest available side of every empty cell is always clean.*

Theorem 4 allows then to conclude that:

Theorem 11. *Any shape* S *can be folded by a tight OS at scale* $\mathscr{A}_4$*.*

Fig. 44: *The 33 tight routing extensions for $\mathscr{A}_4$:* in purple, the latest edge, which is clean and can thus be used to extend the path in this cell; in red, the new potential clean edges available to extend the path for the neighboring cells (only the latest one around the empty cell might be clean); highlighted in orange, the seed for signature 0.

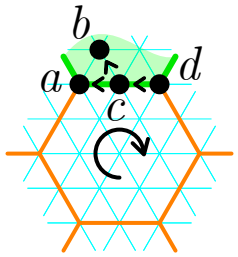


Fig. 45: At all time, the vertices a, b, c, d on and inside an available side verify the clockwise order property around an empty cell:
$$\max(\mathsf{rtime}(a), \mathsf{rtime}(b)) < \mathsf{rtime}(c) < \mathsf{rtime}(d).$$

D.6 Omitted contents of subsection 5.6: Scale $\mathscr{A}_3$

We say that an available (occupied) side of an empty cell $\Lambda(p)$ *flows clockwise* (see Fig. 45)if its 3 vertices a, c, d (taken in clockwise order around $\Lambda(p)$) plus the vertex b neighboring a and c inside the occupied neighboring cell, appear in clockwise order in the current routing, i.e. if

$$\max(\mathsf{rtime}(a), \mathsf{rtime}(b)) < \mathsf{rtime}(c) < \mathsf{rtime}(d)$$

(a and b may appear in any relative order). This property ensures that the edge $c \to d$ is clean (a and b being resp. its support and bouncer) whenever it belongs to the routing, and can thus be used to extend the path (recall Fig. 4).

Regardless of the algorithm, the following invariants are valid for any sequence of cell insertions/anomaly fixing made according to Fig. 7. These are proved by inspecting the extension patterns together with an immediate induction:

Invariant 2 After any sequence of cell insertions with patterns according to Fig. 7,

1. every vertex covered once remains covered;
2. the empty sides of an empty cell are covered by the routing one after the other in clockwise order starting from the counter-clockwise side to the clockwise side of the insertion side;
3. as the routing is extended by inserting a pattern on an edge, the relative order in the routing of the vertices outside the newly covered cell is unchanged by a insertion a new cell or fixing an anomaly (we consider that fixing an anomaly according to Fig. 7 as a new cell insertion here);
4. every available (occupied) side of an empty cell that is not marked as a time-anomaly, flows clockwise.
5. the routing enters the first time and leaves a cell for the last time from the same cell side (the latest side of the cell at the step of its insertion): it enters at its middle vertex and exits at the clockwise-most vertex;

In the patterns listed in Fig. 7, some vertices are marked with yellow or red dots, these are respectively path- and time-anomalies and they require special attention in Algorithm 1.

After step of the algorithm, the current routing covers a connected set of cells. An empty area is a connected component of not-fully-covered cells. Every empty area has a boundary which is a cycle made of neighboring cell sides (see Fig. 46). The topological lemma 2 page 14 is the key to our result.

Proof (of Lemma 2). The proof relies on applying Jordan's theorem to the current routing. Consider an empty area A. Because its boundary is a cycle, it must contain at least one time-anomaly: indeed, according to invariant (2.4), time increases clockwise along the sides of an empty area without time-anomaly; as it must decrease at some point, it must contain at least one time-anomaly. Now, consider a time-anomaly a and its clockwise and earlier neighbor e on the boundary. a was produces by one of the patterns in Fig. 7. We illustrate the proof with pattern `1101` here (see Fig. 47); the proof works identically with all patterns containing a time-anomaly, as they are all topologically identical w.r.t. this result. As e is earlier in the routing than a, the routing connects e to a by a self-avoiding path (in red on Fig. 47) that goes either (a) to the left or (b) to the right. As a and e are next to each other, together with the path in the pattern, they both "seal" this path, which thus encloses the empty area A (in its outside in (a); in its inside in (b)). According to the pattern `1101`, the routing must continue to the right after passing through a, to get back to the origin. By Jordan's theorem, the part of the routing after a (in blue) is thus entirely isolated from the empty area by the red path, and the origin must lie there as

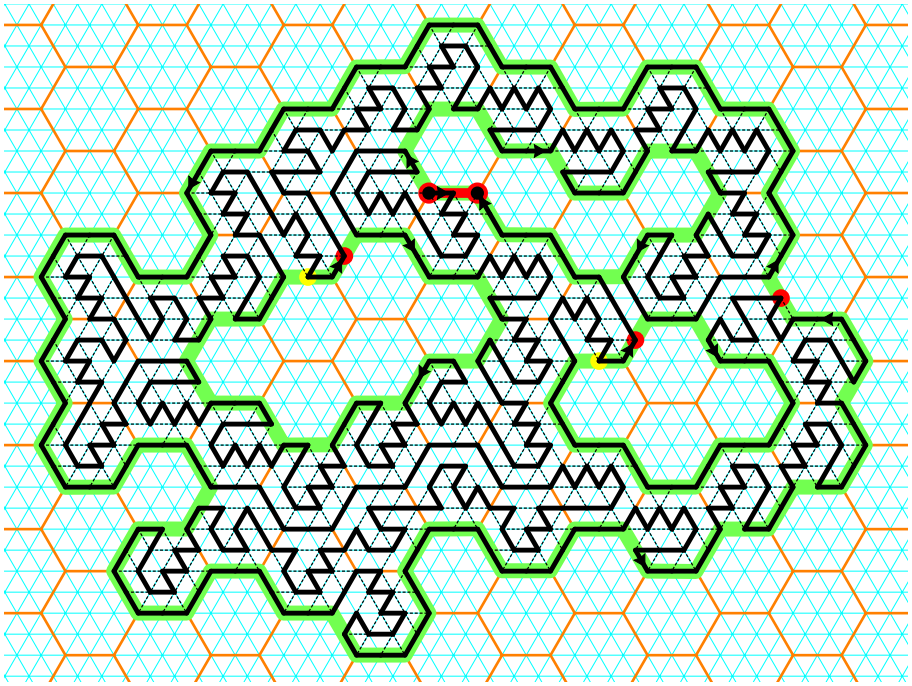
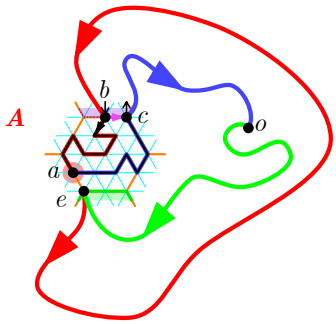
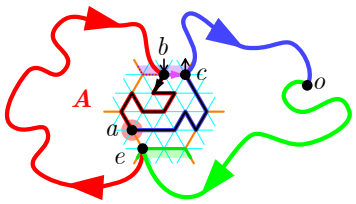


Fig. 46: *A configuration with four empty areas:* their boundaries (outlined in green) contain exactly one time-anomaly each (the red dots) (recall that we consider the originating side of the routing (in red) as one time-anomaly).



(a) The routing from e to a goes to the right



(b) The routing from e to a goes to the left

Fig. 47: In both cases, all vertices on the boundary of the empty area A must be earlier than the time-anomaly a.

well. It follows again by Jordan's theorem, that the part of the routing connecting the origin to e (in green) is also isolated from the empty area by the red path. It follows that the only occupied vertices exposed at the boundary of A belong to the red path. *All* the vertices at the boundary of A are thus *earlier* than a. There can thus not be any other anomaly on this boundary; otherwise both anomalies would be earlier than each other. □

The key lemma implies that after the while loop, the empty cell is only surrounded by "regular" edges of the boundary, which all flow clockwise by the invariant 2 above. It follows that invariant 1 is now verified and applying the pattern corresponding to the new empty cell signature extends the routing to cover the cell while ensuring its foldability. Indeed:

Proof (of Corollary 1). First, observe by invariant (2.4), that every edge which is not a time-anomaly flows clockwise. By the key topological lemma 2, this implies that every side on the boundary of an empty cell is clean, unless the empty cell is neighboring the only time-anomaly on that boundary. Furthermore, as time increases clockwise, it also implies that the latest edge around the empty cell is not only clean but located at the clockwise end of a segment.

Let us thus first focus on time-anomalies. One can observe in the patterns in Fig. 7(b-d) that the pattern fixing a time-anomaly moves the anomaly outside the sides of the empty cell, while preserving its connectivity with an already covered and earlier cell side, ensuring that we are back to the case treated above.

We can now assume that for the empty we want to cover, its latest side clean and is the clockwise most of the segment. If this side is not a path-anomaly, a simple inspection of the patterns shows that any pattern can be plugged into this side safely while preserving the tight foldability of the routing. However, if the side contains a path-anomaly, then plugging the pattern, as is, might prevent the routing from folding or create an unfixable time-anomalie (see the sw-path-anomaly in pattern 101 for instance). But, one can observe by inspecting carefully Fig. 7(b-d) that fixing a path-anomaly according to the prescribed patterns, ensures that:

- if the fixed edge is plugged into *immediately afterwards*, then the routing will be foldable. For instance, observe the pattern 101»sw fixing the sw-path-anomaly in pattern 101: it cannot be folded as is, but will be foldable if an other pattern is plugged to the swopening.
- the fixed edge will be plugged immediately afterwards by the algorithm, because it is the latest edge of the to-be-inserted empty cell (otherwise it would not have been fixed)

It follows that fixing anomalies requires fixing at most 2 anomalies and that the resulting path is foldable and tight. □

Proof (of Theorem 8). It follows from corollary 1, that every steps requires constant time computation, *once we know the rank of each vertex in the current routing.* This can be maintained using your favorite balanced search tree in $O(\log(n))$ time per vertex query after n cell insertions.

Algorithm 1 outputs then a tight routing covering any shape in time $O(n \log n)$. This tight routing is then transformed into a delay-1 OS with seed of size 3 using the universal 114 bead types by Theorem 4 in linear time, which concludes the proof.

D.7 Examples of step-by-step construction of the routing of a shape

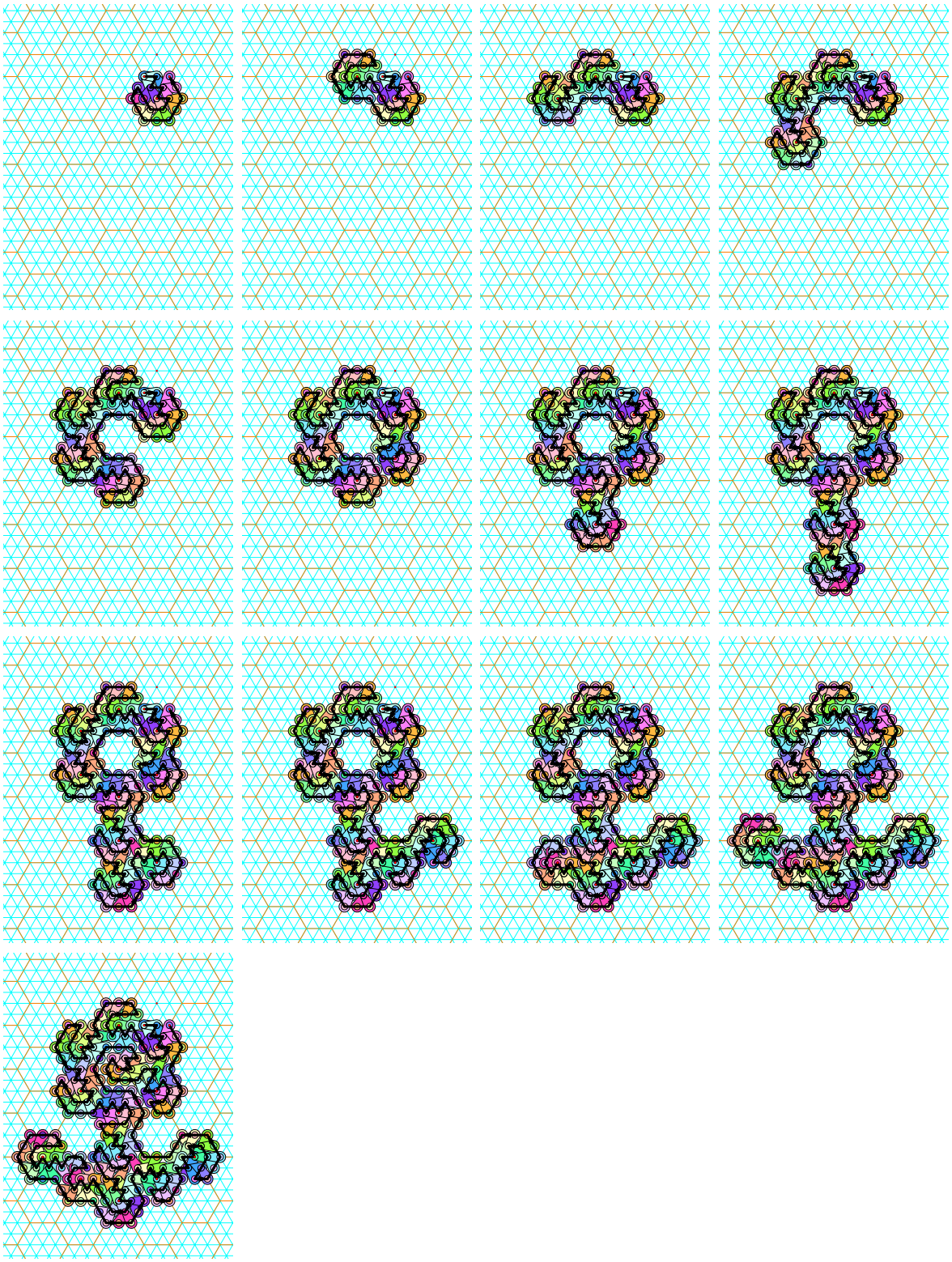


Fig. 48: The step-by-step construction of a routing folding into a flower shape at scale $\mathscr{A}_3$ according to the algorithm in Section 5.6

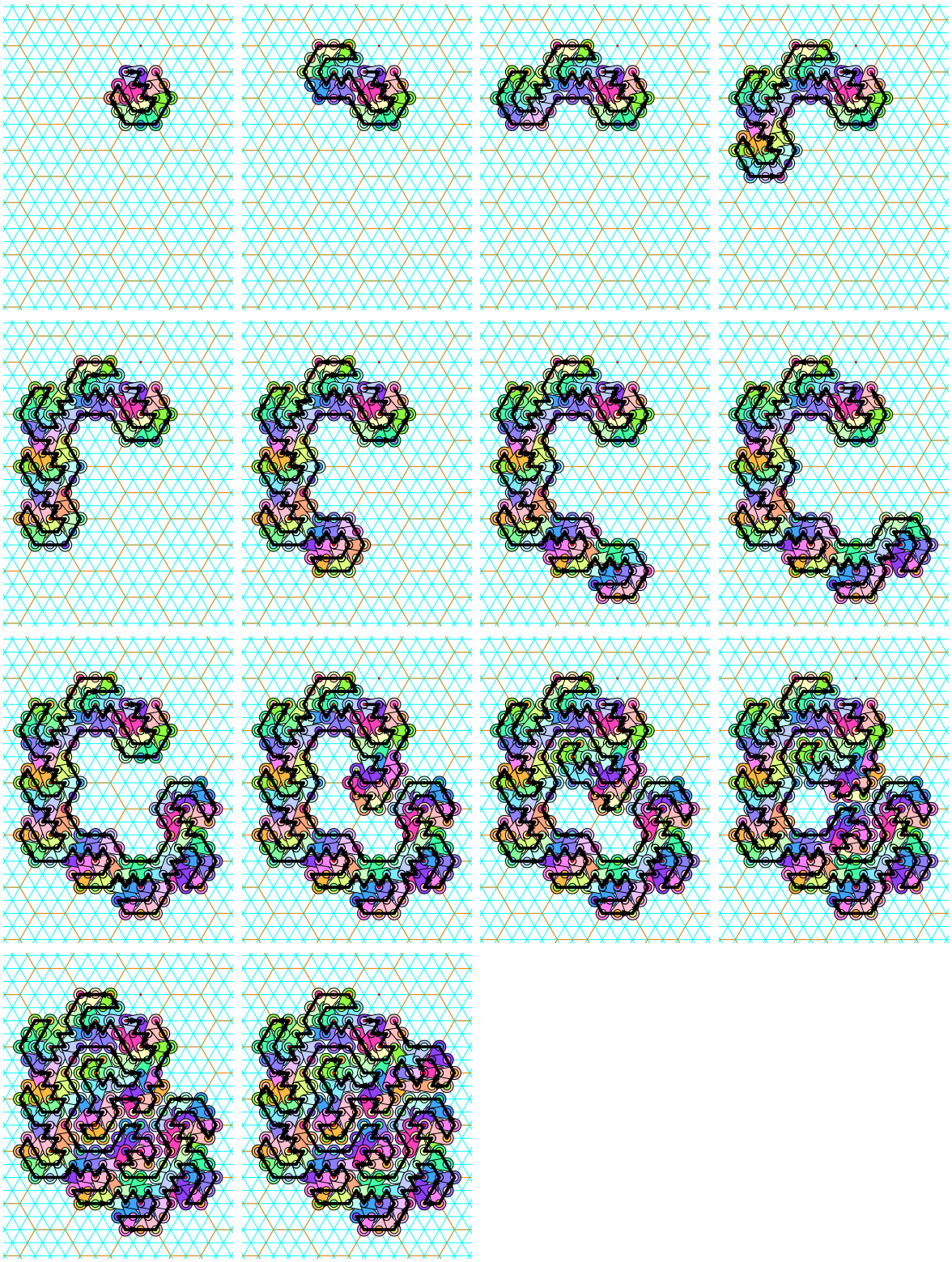


Fig. 49: The step-by-step construction of a routing folding into a shape at scale $\mathscr{A}_3$ according to Algorithm 1, involving solving anomalies $\mathsf{101} \to \mathsf{101}\text{»}\mathsf{nw} \to \mathsf{101}\text{»}\mathsf{nw}\text{»}\mathsf{s} \to \mathsf{101}\text{»}\mathsf{nw}\text{»}\mathsf{s}\text{»}\mathsf{sw}$ in the four last steps.

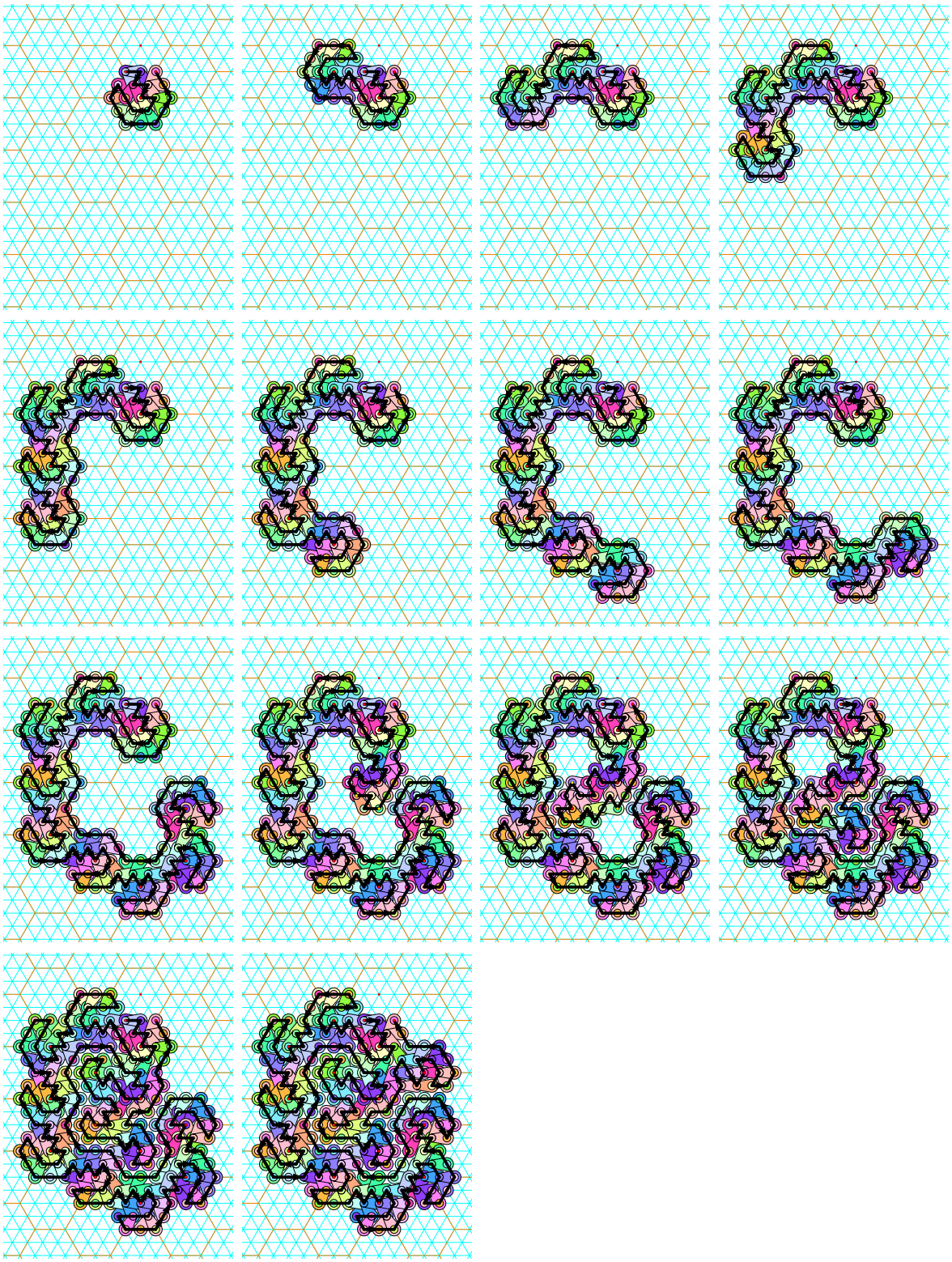


Fig. 50: The step-by-step construction of a routing folding into a shape at scale $\mathscr{A}_3$ according to Algorithm 1, involving solving anomalies $\mathsf{101} \to \mathsf{101}\text{»}\mathsf{sw} \to \mathsf{101}\text{»}\mathsf{sw}\text{»}\mathsf{s}$ and $\mathsf{11101} \to \mathsf{11101}\text{»}\mathsf{nw} = \mathsf{111101}$ (rotated clockwise) in the four last steps.

E Details for a shape which can be assembled at delay δ but not $< \delta$

We now present the full details of the proof of Theorem 9. To best fit the figures to the page, in this section we discuss configurations which are situated on a rotated triangular grid relative to $\mathbb{T}$.

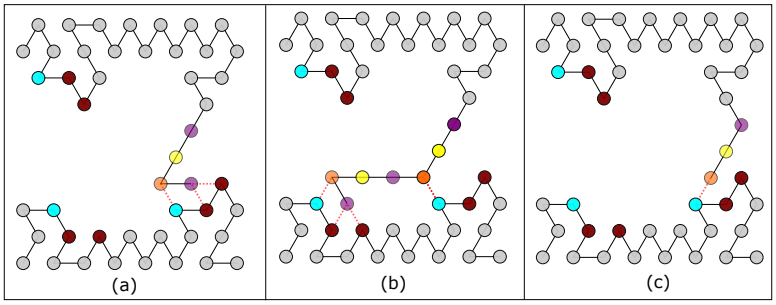


Fig. 51: Parts (a) and (b) show an example of how the rule set of Ξ^*_δ is used to assemble R'_δ when $\delta = 4$. Part (c) shows an incorrect configuration that results if only visible bonds are used to stabilize the purple bead. Transparent beads represent nascent beads and opaque beads represent stabilized beads.

E.1 Full description of S_δ

To formally describe S_δ, we first describe a finite routing R'_δ. The routing is defined so that the points in the routing are a subset of S_δ. We then show that there exists a deterministic OS Ξ^*_δ whose terminal configuration C has R'_δ as a routing. Since there exists an OS which can trivially assemble any shape with a Hamiltonian path for any δ (by encoding it as a seed), we show a straight forward way to extend Ξ^*_δ to an infinite system Ξ_δ which assembles an infinite number of copies of the routing side-by-side. The shape S_δ is then defined to be the domain of the terminal configuration of Ξ_δ

Description of R'_δ Algorithm 11 creates the routing R'_δ by calling a number of subroutines. It begins with an empty routing R and adds beads to the routing by calling subroutines. Each subroutine returns a routing which is then added to R. We say that the routing returned by each subroutine is a *gadget*. For example, we call the routing returned by the `LEFT-WALL` subroutine the `LEFT-WALL` gadget. The algorithm begins by adding three beads to the routing R. We call the routing of these three beads the `SEED` gadget due to the fact that it will act as the seed for the OS which assembles this routing.

Algorithm 11 and its subroutines make use of a few primitive subroutines which we now describe. The `PATH` subroutine defined in Algorithm 17 takes as input some length "l" and outputs a routing of width 2 and "length" l. An example of the output of the `PATH` subroutine is shown as the beads with a blue outline in Figure 54. The `SMALL-BUMP` and `BIG-BUMP` subroutines are not explicitly defined due to their simple nature. They do not take any arguments and the output of the routines is shown in Figure 53. We refer to the output of either of these routines as a `BUMP` gadget. An example of the `BUMP` gadgets are shown in Figure 54 as beads with a red outline.

Algorithm 11 then calls the `LEFT-WALL` subroutine. This subroutine adds $\lceil \frac{\delta(5(\delta-1)+1)}{4} \rceil$ repeating units to R which consist of a `SMALL-BUMP` and a `BIG-BUMP` (the `SMALL-BUMP` and `BIG-BUMP` gadgets are shown in Figure 53) spaced out using `PATH` gadgets by some amount dependent on δ. The `LEFT-WALL` subroutine passes arguments to `SMALL-BUMP` and `BIG-BUMP` so that the labels of the beads in Figure 53 have the symbol x replaced by l and $i \in \mathbb{N} \cap [1, \frac{6\delta}{4}]$. More specifically, these `BUMP` gadgets are spaced out so that the Euclidean distance between the point to the northeast of the lc_i bead in the `BIG-BUMP` gadget and the point to the southeast of the la_i bead in the `SMALL-BUMP` gadget is $\delta - 1$. The yellow beads in Figure 54 show an example of the `LEFT-WALL` gadget when $\delta = 4$. Notice that the lc_i and la_i beads in this example correspond to the beads which are adjacent to the blue beads. The spacing between the `BIG-BUMP` and `SMALL-BUMP` is such that δ beads can be placed in a straight line beginning from the northeast of the lc_i bead and ending at the southeast of the la_i bead.

The next subroutine to be called from Algorithm 11 is the `WIDE-TURN` subroutine. This subroutine adds two paths to R which form a wide "V" shape in relation to each other (as shown by the red beads in the example in Figure 54. The length of these two paths is dependent on δ. We selected the lengths of these two paths so that after all gadgets assemble, the line of beads which stretch from the `LEFT-WALL`

gadget to the RIGHT-WALL gadget (described below) consist of δ beads. For an example notice in Figure 54 how the length of the WIDE-TURN gadget (the beads shown in red) allows for a line of 4 beads to stretch from the LEFT-WALL to the RIGHT-WALL.

After the WIDE-TURN gadget is added to R, the SCAFFOLD (shown in Figure 54 as purple beads) and UTURN(shown in Figure 54 as green beads) gadgets are added to R. The SCAFFOLD gadget just consists of a long path whose length is determine by δ. The purpose of the SCAFFOLD and UTURN gadgets is to position the RIGHT-WALL gadget so that the BUMP gadgets in the RIGHT-WALL gadget lie in a particular position relative to the BUMP gadgets in the LEFT-WALL gadget. The purpose for this is to allow particular beads in the BEAD-LINE gadget (described below) to be adjacent to exactly one bead in the BUMP gadgets.

The next subroutine to be called from Algorithm 11 is the RIGHT-WALL subroutine. This subroutine adds a reflected version of the LEFT-WALL gadget to R with a couple of caveats. The RIGHT-WALL subroutine passes arguments to SMALL-BUMP and BIG-BUMP so that the labels of the beads in Figure 53 have the symbol x replaced by r and $i \in \mathbb{N} \cap [1, \frac{6\delta}{4}]$. The RIGHT-WALL gadget has a PATH gadget of length δ tacked onto the last repeating unit with a fixed size triangle adjoined to it. An example of a RIGHT-WALL gadget is shown in Figure 54 as grey beads.

The subroutine BEAD-LINE is the next subroutine to extend the routing R. The routing returned by this subroutine is $\lceil \frac{\delta(5(\delta-1)+1)}{4} \rceil$ repeating units consisting of 4 lines of beads. The first line of beads in a unit is a line of beads of length $\delta - 1$ which grows to the northwest. Attached to that is another line of beads of length $\delta - 1$ which grows to the north. Next, a line of $\delta - 1$ beads grow to the northeast. Finally, another line of length $\delta - 1$ beads grows to the north. The set of blue colored beads in Figure 54 depicts an example of the BEAD-LINE gadget when $\delta = 4$. Notice that the gadgets have been designed so that every $\delta - 1$ beads in the BEAD-LINE gadget there is a bead which lies adjacent to exactly one BUMP gadget.

The last subroutine to be called by Algorithm 11 is the SPACER subroutine. This subroutine returns a SPACER gadget which grows over the UTURN gadget using a series of PATH and TURN gadgets, and then it grows a path of length $2\delta + 4$ to the southeast. The purpose for this SPACER gadget is to allow us to attach R'_δ routings to each other in a side-by-side manner.

Algorithm 11 and its subroutines A detailed description of the shape S_δ is now provided. We begin by providing some useful notation and a list of straightforward auxiliary methods. For a finite directed path P, we use $|P|$ to denote the index of the last element in the sequence. Also, $P(i)$ denotes the i^{th} element in the sequence which in our case corresponds to a point in $\mathbb{Z}_\Delta$. We now provide a list of simple auxiliary methods. We define $D = (N, NE, SE, S, SW, NW)$ to be the set of *directions*. Given a direction d and a point p, we define the *d neighbor of p* to be the point corresponding to the direction d in Figure 52. We denote the d neighbor of p by $d(p)$. In this section, given an element of a routing $r_i = (p_i, b_i)$, we use $dom(r_i)$ to denote p_i, that is $dom(r_i) = p_i$.

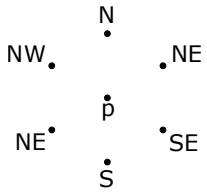


Fig. 52: A point p and all of its d neighbors for $d \in D$.

Let P be a directed path in $\mathbb{T}$. We call a point $p \in \mathbb{T}$ *empty with respect to P* provided that for all $i \in [1, n], p_i \neq p$. The method SHARED-NHBR(P, i, j) takes as input a directed path in $\mathbb{T}$, P, and two indices $i, j \in \mathbb{N}$. If $1 \leqslant i, j \leqslant |P|$ and $P[i]$ and $P[j]$ have exactly one shared neighbor which is empty with respect to P, SHARED-NHBR returns this point. Otherwise, SHARED-NHBR returns null.

We now describe some subroutines used in Algorithm 11 which are not explicitly defined. The subroutine UTURN in Algorithm 11 takes as input some routing and it returns a new routing. As shown in Figure 54, Algorithm 11 is designed so that the subroutine UTURN will always receive a routing which consists exactly of the seed (darkly shaded), the left wall (yellow), the wide turn (red) and the scaffold (purple) parts of the routing. Note that the last position in this routing is the north most bead in the scaffold. The subroutine UTURN creates the routing shown in green (where the bead types are some generic labels which are unique) and returns this routing.

The SMALL-BUMP and BIG-BUMP routines called in Algorithm 12 and Algorithm 14 each take three arguments: 1) a routing R, 2) a symbol x and 3) a number i. The result of these two routines is shown in Figure 53. The darkly shaded beads in both (a) and (b) of Figure 53 represent beads that are in the routing R which is passed as an argument. The two routines extend the routing R by adding the routing indicated by the lightly shaded beads. All non-labeled beads in Figure 53 are unique bead types with some generic labels. The labeled beads are given bead types corresponding to their label. For example, the routine call SMALL-BUMP$(R, \text{“}r'', 1)$ would return a routing where the labeled bead types would be ra_1, rb_1 and rb'_1.

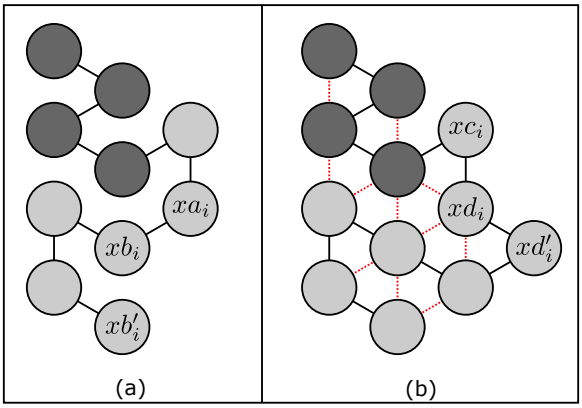


Fig. 53: The lightly shaded beads of (a) show the routing returned from the routine SMALL-BUMP and the specially labeled beads. The lightly shaded beads of (b) show the routing returned from the routine BIG-BUMP and the specially labeled beads. The red dotted lines represent attraction rules between the beads which allow them to assemble at any delay greater than two.

Algorithm 11 A procedure to build the routing R'_δ

procedure BUILD-ROUTING(δ) ▷ Takes $\delta \in \mathbb{N}$
  ▷ Initialization
  $R = \{\}$ ▷ An ordered list which holds the routing
  ▷ Add seed to the routing
  $R = R + ((0,0), b_1)$
  $R = R + ((\frac{-1}{2}, \frac{-\sqrt{3}}{2}), b_2)$
  $R = R + ((0,-1), b_3)$
  ▷ Add the yellow bead portion of the routing shown in Figure 54
  $R = R +$ LEFT-WALL(R, δ)
  ▷ Add the red bead portion of the routing shown in Figure 54
  $R = R +$ WIDE-TURN(R, δ)
  ▷ Add the purple portion of the routing shown in Figure 54. We call this part of the routing the scaffold.
  $R = R +$ PATH$(R, NI(2(\delta-3)+4+(\delta-1))+\delta)$
  ▷ Add the green portion of the routing shown in Figure 54
  $R = R +$ UTURN(R) ▷ This routine is not explicitly defined. See text for details.
  ▷ Add the grey bead portion of the routing shown in Figure 54
  $R = R +$ RIGHT-WALL(R, δ)
  ▷ Add the blue portion of the routing shown in Figure 54
  $R = R +$ BEAD-LINE(R, δ)
  ▷ Add the orange bead portion of the routing shown in Figure 54
  $R = R +$ SPACER(R, δ)
  return R

Algorithm 12 A procedure to build the `LEFT-WALL` gadget of R'_δ (shown as the yellow filled beads in Figure 54).

procedure `LEFT-WALL`(R, δ) ▷ Takes a routing R and $\delta \in \mathbb{N}$
  $S = \{\}$ ▷ Routing to be returned from this procedure.
  $sc = \lceil \frac{\delta(5(\delta-1)+1)}{4} \rceil$
  for $sc > 0$ **do**
    $S = \texttt{PATH}(R, 2(\delta - 3))$
    $S = S + \texttt{SMALL-BUMP}(S, \text{“l”}, sc)$ ▷ This routine is not explicitly defined. See text for details.
    $S = S + \texttt{PATH}(S, \delta - 1)$
    $S = S + \texttt{BIG-BUMP}(S, \text{“l”}, sc)$ ▷ This routine is not explicitly defined. See text for details.
  return S

Algorithm 13 A procedure to build the `WIDE-TURN` gadget of R'_δ (shown as the red filled beads in Figure 54).

procedure `WIDE-TURN`(R, δ) ▷ Takes a routing R and $\delta \in \mathbb{N}$
  $S = \{\}$ ▷ Routing to be returned from this procedure.
  $S = \texttt{PATH}(R, 2)$
  $S = S + \texttt{TURN}(S)$
  $S = S + \texttt{PATH}(S, \delta + 1)$
  $S = S + \texttt{TURN}(S)$
  $S = S + \texttt{PATH}(S, \delta + 1)$
  $S = S + \texttt{TURN}(S)$
  return S

Algorithm 14 A procedure to build the `RIGHT-WALL` of R'_δ (shown as the grey filled beads in Figure 54).

procedure `RIGHT-WALL`(R, δ) ▷ Takes a routing R and $\delta \in \mathbb{N}$
  $S = \{\}$ ▷ Routing to be returned from this procedure.
  $sc = \lceil \frac{\delta(5(\delta-1)+1)}{4} \rceil$
  for $sc > 0$ **do**
    $S = \texttt{PATH}(R, 2(\delta - 3))$
    $S = S + \texttt{SMALL-BUMP}(S, \text{“r”}, sc)$ ▷ This routine is not explicitly defined. See text for details.
    $S = S + \texttt{PATH}(S, \delta - 1)$
    $S = S + \texttt{BIG-BUMP}(S, \text{“l”}, sc)$ ▷ This routine is not explicitly defined. See text for details.
  $S = S + \texttt{PATH}(S, \delta)$
  $S = S + \texttt{TURN}(S)$
  $S = S + \texttt{PATH}(S, 1)$
  return S

Algorithm 15 A procedure to build the `BEAD-LINE` gadget of R'_δ (shown as the blue filled beads in Figure 54).

procedure `BEAD-LINE`(R, δ) ▷ Takes a routing R and $\delta \in \mathbb{N}$
  $S = R[|R|]$ ▷ S gets the last bead in the routing R
  $NI = \lceil \frac{\delta(5(\delta-1)+1)}{4} \rceil$
  $S = S + (NW(dom(S[|S|])), ca_i)$
  for $0 < i \leqslant NI$ **do**
    for $0 \leqslant j < \delta - 3$ **do**
      $S = S + (NW(dom(S[|S|])), l_{|S|+1})$ ▷ Recall this denotes the point to the northwest of the last position in R
    $S = S + (NW(dom(S[|S|])), cb_i)$
    $S = S + (N(dom(S[|S|])), cc_i)$
    for $0 \leqslant j < \delta - 3$ **do**
      $S = S + (N(dom(S[|S|])), l_{|S|+1})$
    $S = S + (NW(dom(S[|S|])), cd_i)$
    $S = S + (N(S[|S|]), ce_i)$
    for $0 \leqslant j < \delta - 3$ **do**
      $S = S + (NE(dom(S[|S|])), l_{|S|+1})$
    $S = S + (NW(dom(S[|S|]), cf_i)$
    $S = S + (N(dom(S[|S|])), cg_i)$
    for $0 \leqslant j < \delta - 2$ **do**
      $S = S + (N(dom(S[|S|])), l_{|S|+1})$
    $S = S + (N(dom(S[|S|])), ch_i)$
  return S

Algorithm 16 A procedure to build the `SPACER` gadget of R'_δ (shown as the blue filled beads in Figure 54) to the routing.

procedure `SPACER`(R, δ) ▷ Takes a routing R and $\delta \in \mathbb{N}$
  $S = \{\}$ ▷ Routing to be returned from this procedure.
  $S = (N(dom(R[|R|])), s_1)$
  $S = S + (N(dom(R[|R|])), s_2)$
  $S = S + (N(dom(S[|S|])), s_3)$
  $S = S + (NE(dom(S[|S|])), s_4)$
  $S = S +$ `PATH`$(S, 2(\delta - 3) + 2)$
  $S = S +$ `TURN`(S)
  $S = S +$ `PATH`$(S, 2)$
  $S = S +$ `TURN`(S)
  $S = S +$ `PATH`$(S, 2\delta + 4)$
  $S = S +$ `TURN`(S)
  return S

Algorithm 17 A procedure to build the `PATH` gadgets of R'_δ (shown as the beads with a blue outline in Figure 54).

procedure `PATH`(R, l) ▷ Takes a routing R and a length $l \in \mathbb{N}$
  $S = \{\}$ ▷ Routing to be returned from this procedure.
  $S = S + R[|R| - 1]$ ▷ Add the second to the last bead in R to S
  $S = S + R[|R|]$ ▷ Add the last bead in R to S
  for $0 \leqslant i < l$ **do**
    $S = S +$ (`SHARED-NHBR`$(S, dom(S[|S| - 1]), dom(S[|S|])), b_{|R|+|S|+1})$ ▷ Add a bead with a unique generic type in the location next to the previous two beads.
    $S = S +$ (`SHARED-NHBR`$(S, dom(S[|S| - 1]), dom(S[|S|])), b_{|R|+|S|+1})$ ▷ Add a bead with a unique generic type in the location next to the previous two beads.
  return S

Algorithm 18 A procedure to build the TURN gadgets of R'_δ (shown as the beads with a green outline in Figure 54).

1: **procedure** TURN(R) ▷ Takes a routing R
2: **return** (SHARED-NHBR($R, dom(R[|S| - 2]), dom(R[|R|])), b_{|R|+1}$)

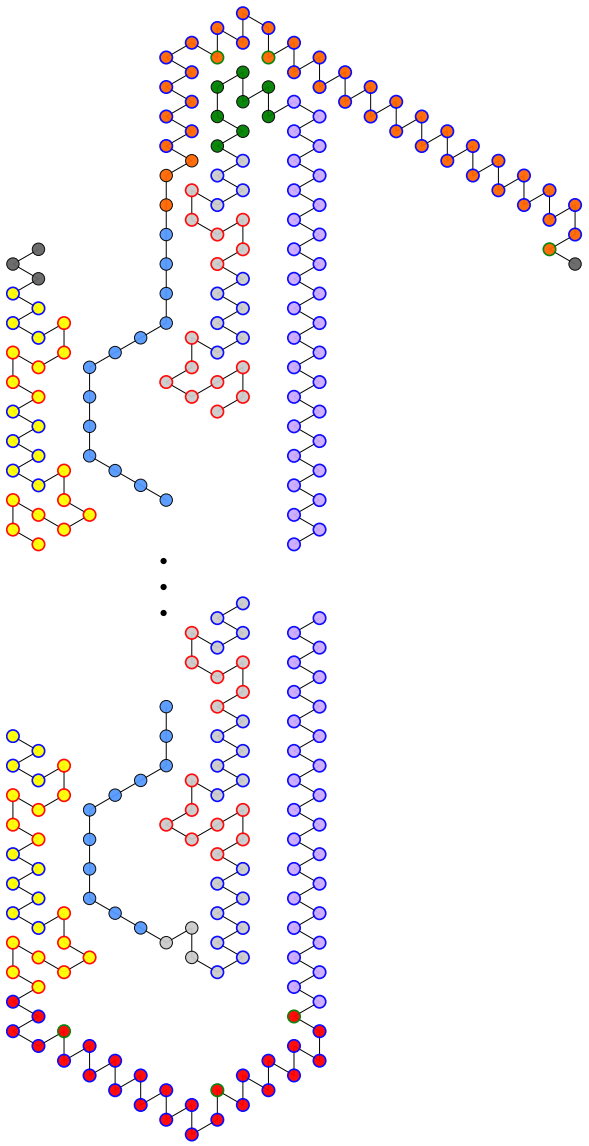


Fig. 54: An example of the routing R'_δ when $\delta = 4$. The beads are colored according to the routines from which they were returned.

A system at delay δ which assembles R'_δ We now show that there exists an OS at delay δ which assembles the routing R'_δ described in Section E.1. Formally, we say that a system Ξ assembles a routing R if for every $C \in \mathcal{A}_\square[\Xi]$ the routing of C is R. Likewise, we say that a system Ξ assembles a directed path P' provided that for every $C = (P, w, H) \in \mathcal{A}_\square[\Xi]$, $P' = P$.

Lemma 10. *Let $P = p_1p_2...p_n$ be a finite directed path in $\mathbb{T}$ with the property that there exists $j \in [4, n]$ such that for all $k \geqslant j$, p_k has two neighbors $p_r, p_s \in \{p_1, ..., p_{k-1}\}$ such that there is exactly one empty point with respect to the directed path $p_1p_2...p_{k-1}$ which is adjacent to both p_r and p_s. Then, for every $\delta \in \mathbb{N}$, there exists an oritatami system with a seed of size j and delay δ which assembles P.*

The intuition behind the proof is as follows. We construct an OS $\Xi = (\Sigma, w, \mathcal{H}, \delta, \alpha)$ with seed σ and $\alpha = 5$ based on P by first creating a hard-coded sequence of beads (i.e. every bead in the sequence is unique). We then construct $\mathcal{H}$ so that if p_r or p_s are equal to p_{i-1}, WLOG let's assume $p_r = p_{i+1}$, we add $\{p_k, p_s\}$ to $\mathcal{H}$. If that's not the case we add the rules $\{p_k, p_s\}$ and $\{p_k, p_r\}$ to $\mathcal{H}$. We argue that this assembles by inductively showing that each configuration C_{i+1} in the assembly sequence stabilizes bead type $w[i+1]$ in the correct position. To see this, note that there exists a favorable elongation of C_i which stabilizes bead type $w[i+1]$ in the correct position and this elongation makes every bond possible between the beads. Any elongation which places the bead type $w[i+1]$ in the incorrect position must "break a bond" to do so, and, consequently must not be a favorable configuration.

Proof. Let $P = p_1p_2...p_n$ be a finite directed path in $\mathbb{T}$ with the property that there exists $j \in [4, n]$ such that for all $k \geqslant j$, p_k has two neighbors $p_r, p_s \in \{p_1, ..., p_{k-1}\}$ such that there is exactly one empty point with respect to the directed path $p_1p_2...p_{k-1}$ which is adjacent to both p_r and p_s. Also, let $\delta \geqslant 2$.

We now describe a system Ξ which we claim can assemble P. Let $\Xi = (\Sigma, w, \mathcal{H}, \delta, \beta, \alpha, \sigma)$ where

- $\Sigma = \{b_i | i \in [1, |P|]\}$,
- $w = (b_i)_{i=j}^{|P|}$,
- $\beta = 1$,
- $\alpha = 5$,
- σ is the configuration $((p_i)_{i=1}^{j-1}, (b_i)_{i=1}^{j-1}, \varnothing)$.

To generate $\mathcal{H}$ we iterate over $i \in [j, |P|]$ and consider two cases for each i. The first case we consider is that there exists p_{i-1} and p_l for some $l \in [1, i-2]$ such that they are both adjacent to p_i and there is exactly one empty point with respect to the directed path $p_1p_2...p_{i-1}$ which is adjacent to both p_{i-1} and p_l. If more than one point in P satisfies this condition for p_l, we chose the one with the lowest index as a convention. In this case, we add the rule (b_{i-1}, b_l) to $\mathcal{H}$[13]. In the case that no such p_l exists, then it must be the case that exists $p_r, p_s \in \{p_1, ..., p_{k-1}\}$ such that there is exactly one empty point with respect to the directed path $p_1p_2...p_{k-1}$ which is adjacent to both p_r and p_s. If multiple such p_r and p_s exist we chose the pair of indices with the lowest lexicographical ordering as a convention. In this case, we add the rules (b_i, b_s) and (b_i, b_r) to $\mathcal{H}$. See Figure 55 for an example of the two cases we consider.

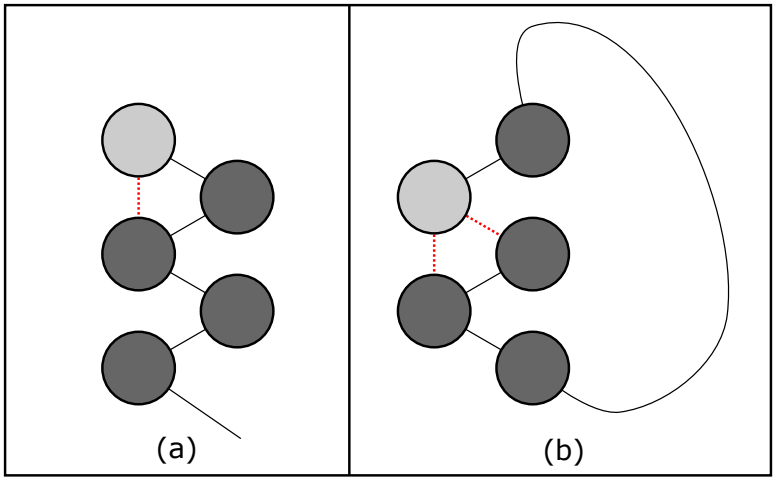


Fig. 55: The two cases we consider in the proof of Lemma 10. Part (a) corresponds to the first case in the proof and part (b) corresponds to the second case of the proof.

Let $\alpha \in [1, 5]$, let H be a rule set, let Σ be a set of bead types and let $C = (P, w, H)$ be an $\mathcal{H}$-valid configuration where $w \in \Sigma^*$. Also, let $t \in \Sigma^*$. Recall that $\mathcal{P}^{\leqslant t}_{\mathcal{H},\alpha}(C)$ is the set of $\mathcal{H}$-valid elongations by prefixes of t. We call a configuration $C' = (P', w', H') \in \mathcal{P}^{\leqslant t}_{\mathcal{H},\alpha}(C)$ a *saturated $\mathcal{H}$-valid α-δ-elongation of C by t* provided that for all other configurations $C^* = (P^*, w^*, H^*) \in \mathcal{P}^{\leqslant t}_{\mathcal{H},\alpha}(C)$, $H^* \subseteq H'$. Intuitively, a configuration C is saturated provided that even if we made a "configuration" C' by allowing bonds to

[13] Though we do not explicitly state it, when we add rule (a, b) to the set $\mathcal{H}$, we also add the rule (b, a) to ensure that $\mathcal{H}$ defines a symmetric relation.

form between nascent beads which are not adjacent (that is we remove geometry), C' would have the same bonds as C.

Observation 12 *Suppose that there exists a saturated $\mathcal{H}$-valid α-δ-elongation of C by t. Then every favorable α-δ-elongation of C is saturated.*

To prove Ξ assembles P we inductively show that at each step in the folding process, the configuration which stabilizes the next bead in the correct position is the one and only favorable configuration. Let $C \in \mathcal{A}_{\square}[\Xi]$, and let $\overrightarrow{C} = (C_i)_{i=0}^{l}$ be a foldable sequence such that $res(\overrightarrow{C}) = C$. For the base case, note that $C_0 = (P_0, w_0, \varnothing)$, which is the seed, is such that P_0 is a prefix of P. For the inductive step, assume $C_i = (P_i, w_i, H_i)$ is such that P_i is a prefix of P. We now show that the configuration $C_{i+1} = (P_{i+1}, w_{i+1}, H_{i+1})$ stabilizes bead type $w[i+1]$ at position $p[i+1]$. Consequently, this means P_{i+1} is a prefix of P. To see this, first note that by construction of Ξ, there exists $C^* = (P^*, w^*, H^*) \in \mathcal{F}_{\mathcal{H},\alpha}^{\leqslant w[i..i+\delta-1]}$ such that P^* is a prefix of P. This is due to the fact that the set $H_{sat} = \{\{l, h\} \mid \{w[l], w[h]\} \in \mathcal{H}$ and $l \in [i, i+\delta-1]$ or $h \in [i, i+\delta-1]\}$ is a subset of H^*. In other words, the beads in the "nascent portion" of C^* make every bond that they possibly can. Thus, C^* is a favorable configuration. Also, note that the fact $H_{sat} \subseteq H^*$ implies that C^* is saturated.

We now show that any configuration $\bar{C}$ which does not stabilize bead type $w[i+1]$ at location p_{i+1} is not saturated. It then follows from Observation 12 that $\bar{C}$ cannot be a favorable configuration (since there exists a saturated elongation). Thus, the only favorable configurations are those which stabilize $w[i+1]$ at location p_{i+1}.

Let $\bar{C} = (\bar{P}, \bar{w}, \bar{H}) \in \mathcal{P}_{\mathcal{H},\alpha}^{\leqslant w[i..i+\delta-1]}(C)$ be a configuration such that $\bar{P}(i+1-k) \neq p_{i+1}$ (the $-k$ expression appears due to the offset caused by the seed). If p_{i+1} falls into the first case listed above, then there was a single rule $\{b_i, b_l\}$ added to $\mathcal{H}$ for some $l \in [1, i-2]$. Note that in any configuration it is necessary that bead b_i is adjacent to bead b_{i+1} due to the fact they are next to each other on the transcript. Observe that two adjacent points in $\mathcal{T}$ have exactly two common neighbors. It now follows from the assumption that there is exactly one empty point with respect to the directed path $p_1 p_2 ... p_{i-1}$ which is adjacent to both p_{i-1} and p_l that if the bead b_{i+1} is stabilized in the incorrect position, it must be stabilized in a way such that it is not adjacent to the bead type b_l. Consequently, $\{w[i+1], w[l]\} \notin \bar{H}$. Hence, the configuration $\bar{C}$ is not saturated. A similar argument shows that the configuration $\bar{C}$ is not saturated in the event p_{i+1} falls into the second case mentioned above.

Lemma 11. *There exists a deterministic oritatami system with delay δ which assembles R'_δ.*

Proof. Let $R'_\delta = (P'_\delta, w'_\delta)$ be the routing constructed in Algorithm 11. We construct an OS Ξ^*_δ and argue that it has a single terminal configuration which has the routing R'_δ. Let $\Xi^*_\delta = (\Sigma_\delta, w^*_\delta, \mathcal{H}^*_\delta, \delta, \alpha, \sigma)$ where

- $\Sigma_\delta = \{w'_\delta[i] \mid i \in [1, |w'_\delta|]\}$,
- $w^*_\delta = w'_\delta[4..|w'_\delta|]$ (we skip the first 3 bead types since they are the seed),
- $\alpha = 5$, and
- σ is the configuration created from the routing defined in lines 5-7 of Algorithm 11 along with the empty set.

To generate $\mathcal{H}^*_\delta$, for every portion of R_δ which is not created by the `BEAD-LINE` or `BIG-BUMP` routine, we add the rules generated by the implicit algorithm in the proof of Lemma 10. In particular, for each of these portions of the routing, treat the preceding portion of the routing as the seed and generate the rule set to build the new portion of the routing using the algorithm which is implicit in the proof of Lemma 10. We note that by the construction of Algorithm 11, these portions of R'_δ meet the criteria listed in the lemma statement. Indeed, all these portions of the routing are made by placing beads in a position relative to beads currently in the routing using the subroutine `SHARED-NHBR`.

We now discuss the rules which must be added to assemble the `BEAD-LINE` and `BIG-BUMP` gadgets. The `BIG-BUMP` gadget can be assembled by adding interaction rules so that the bonds shown in part (b) Figure 53 form. Recall that the routing created in the i^{th} iteration of routines `LEFT-WALL`, `RIGHT-WALL`, and `BEAD-LINE` are translations of the routings shown in Figure 56 (which shows an example when $\delta = 4$). In order to allow the `BEAD-LINE` gadget to assemble in Ξ^*_δ we add the interaction rules (cc_i, ld_i), (cc_i, ld'_i), (cb_i, lc_i), (ce_i, lb_i), (ce_i, lb'_i), (cd_i, la_i), (cg_i, rd_i), (cg_i, rd'_i), (cf_i, rc_i), (ca_{i+1}, rb_i), (ca_{i+1}, rb'_i), (ch_i, ra_i) to $\mathcal{H}^*_\delta$.

To see that Ξ^*_δ does indeed assemble the routing R'_δ, first note that it follows from the proof of Lemma 10 that the only terminal configuration Ξ_δ folds is R'_δ provided that the `BIG-BUMP` and `BEAD-LINE` gadgets are assembled correctly. It's easy to check these interactions allow the `BIG-BUMP` gadget to form at any delay. To see that the `BEAD-LINE` gadget can assemble correctly, note that the added rules allow Ξ^*_δ to assemble the `BEAD-LINE` gadget as shown in Figure 51.

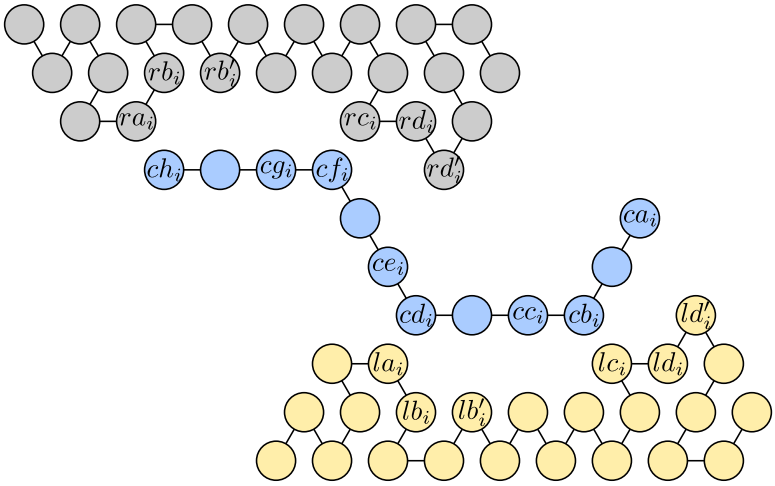


Fig. 56: This image is a 90° rotation from the actual configuration. An example of the routings created in the i^{th} iteration of routines `LEFT-WALL`, `RIGHT-WALL`, and `BEAD-LINE` when $\delta = 4$. Only the important bead types are labeled.

An infinite version of Ξ^*_δ We define an infinite version of Ξ^*_δ, which we call Ξ_δ, so that Ξ_δ assembles an infinite number of copies of R'_δ stacked on top of each other as shown in Figure 57. Let $\Xi^*_\delta = (\Sigma_\delta, w^*_\delta, \mathcal{H}^*_\delta, \delta, \alpha, \sigma)$ be the OS defined in Section E.1. Also, let $t_\delta = w^*_\delta \cdot s_1 \cdot s_2 \cdot s_3$, that is t_δ is the bead sequence w^*_δ with the three bead types in the seed concatenated onto it. Let Ξ_δ be the oritatami system defined by $\Xi_\delta = (\Sigma_\delta, w_\delta, \mathcal{H}_\delta, \delta, \alpha, \sigma)$ where w_δ is the infinite bead sequence defined by $w_\delta(i) = t(i \bmod |t| + 1)$ and $\mathcal{H}_\delta$ is the rule set $\mathcal{H}^*_\delta$ with rules added so that the bead types s_1, s_2, and s_3 (the beads in the seed σ) assemble in a position relative to the `SPACER` gadget as shown in Figure 57. To see that Ξ_δ is deterministic recall that the `SPACER` gadget grows an "arm" where the last bead in the arm is greater than δ away from other gadgets in the routing. This means that beads can not "accidently" interact with beads in other copies of the routing. This combined with the fact that Ξ^*_δ is deterministic implies Ξ_δ is deterministic. We denote the terminal assembly of Ξ_δ by C_δ. For $\delta > 2$, we define the shape S_δ by $S_\delta = dom(C_\delta)$.

E.2 S_δ cannot be assembled by any system with delay $< \delta$

Let C_δ be the terminal configuration of $\mathcal{O}_\delta$ and let C^* be the terminal configuration of $\mathcal{O}^*_\delta$ (Recall $\mathcal{O}^*_\delta$ was the system which assembled the finite routing R'_δ constructed in Algorithm 11. We call a set of points $D_{iter} \subseteq dom(C_\delta)$ an *iteration* if there exists $\overrightarrow{v} \in \mathbb{R}^2$ such that $D_{iter} = \{\overrightarrow{p} \mid \overrightarrow{p} = \overrightarrow{v} + \overrightarrow{x}$ for some $x \in \text{dom}(C^*)\}$. In other words, an iteration is just a translation of the set of points in R'_δ. We call the points added to the routing in the `BEAD-LINE` routine, (Algorithm 15), the *bead-line points*. Let B be the set of bead-line points. Let $\overrightarrow{v}$ be such that $D_{iter} = \{\overrightarrow{p} \mid \overrightarrow{p} = \overrightarrow{v} + \overrightarrow{x}$ for some $x \in \text{dom}(C^*)\}$. The *bead-line points of iteration* D_{iter} is the set $B_{iter} = \{x \mid x = \overrightarrow{v} + \overrightarrow{y}$ for some $y \in B\}$. The locations of the all the beads in Figure 54 is an example of an iteration when $\delta = 4$. Furthermore, the points of the blue beads in the figure make up the set of bead-line points in the iteration.

Let R_δ be the routing of C_δ. For convenience, we let $p(b) \subset \mathbb{T}$ be the set of points defined by $p(b) = \{\overrightarrow{x} \mid (b, \overrightarrow{x}) \in R_\delta\}$. That is $p(b)$ is the set of points where bead type b is located in the configuration C_δ. Given a specific iteration D, we define $p_D(b) = \overrightarrow{x}$ where $x \in p(b)$. Note this is well defined since the bead types placed at points in an iteration in R_δ are unique.

Lemma 12. *Let $\delta > 2$. There does not exist any system $\mathcal{O}' = (\Sigma', w', \mathcal{H}', \delta', \alpha', \sigma')$ with $\delta' < \delta$ such that $\mathcal{O}'$ assembles S_δ.*

Proof. For the sake of contradiction, suppose that there exists a system $\mathcal{O}' = (\Sigma', w', \mathcal{H}', \delta', \alpha', \sigma')$ with $\delta' < \delta$ which assembles S_δ. Let $C' = (P', w', H') \in \mathcal{A}_\square[\mathcal{O}']$ (note that $\mathcal{O}'$ isn't necessarily deterministic, so there may be more than one terminal configuration) and let $\overrightarrow{C'} = (C'_i)_{i=0}$, where $C'_i = (P'_i, w'_i, H'_i)$, be the foldable sequence of $\mathcal{O}'$ such that $res(\overrightarrow{C'}) = C'$. By assumption, $dom(C') = S_\delta$.

Intuitively, the next claim states that in at least one of the iterations, P' must pass through the set of bead-line points in a contiguous manner. That is, the path doesn't "exit" the set of bead line points and then "re-enter". Indeed, the point at which P' exits must have an adjacent neighbor with which it doesn't share an edge with in P'. Consequently, that neighbor only shares an edge with one other point

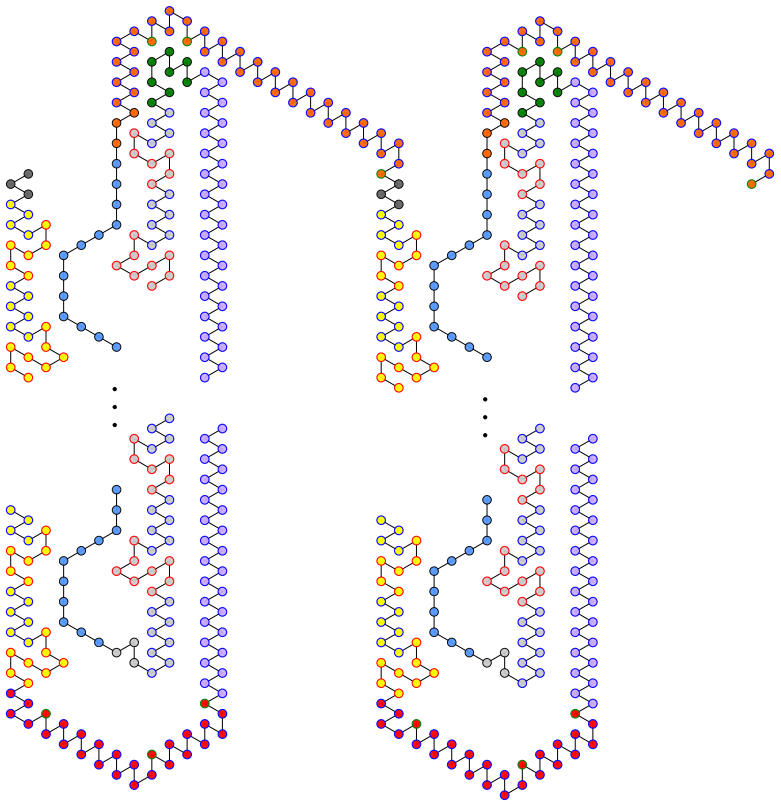


Fig. 57: A portion of the routing of the terminal configuration of Ξ_δ when $\delta = 4$.

in P' and, consequently, it is an endpoint. Since a directed path can only have two endpoints and there are an infinite number of iterations, the claim is proven.

Claim. There exists an iteration D such that the subsequence of P' consisting of exactly the bead-line points of D is a contiguous subsequence of P', and no point in D is contained in $\mathrm{dom}(\sigma')$.

Proof. Before we prove this claim we introduce the notion of an edge in a directed path. We say that there is an edge between p_i and p_j in a directed path P provided that $|i - j| = 1$. Note that for a single element p of a directed path, there are at most two elements such that there is an edge between those elements and p. And, if an element only has one edge in a directed path, then it is an endpoint.

To prove this claim, we show that any iteration D_{iter} where the subsequence of P' consisting of exactly the bead-line points of D_{iter} is not a contiguous subsequence of P' must contain an endpoint of the routing of C'. To see this, let D_{iter} be such an iteration. Let P^* be the minimal contiguous subsequence of P' which contains all the bead-line points of D_{iter}. By assumption, P^* contains points which are not in the bead-line points of D_{iter}. We assume the first point in P^* is adjacent to a point which is in S_δ but not in the set of bead-line points of D_{iter}. Otherwise, it would be the case that the first point of P^* is part of the seed σ' which would imply that D_{iter} contains an endpoint of the routing of the configuration C'.

We now consider the case where the first point in P^* which is not a point in the bead-line points of D_{iter} is in the set $\cup_{i\in[1,\lceil\frac{\delta(5(\delta-1)+1)}{4}\rceil]}\{p(ra_i), p(rc_i), p(la_i), p(lc_i)\}$ (the beads located at these points are shown in Figure 56). Without loss of generality, we assume the first point in P^* which is not a bead-line point is $p(la_i)$ for some $i \in [1, \lceil\frac{\delta(5(\delta-1)+1)}{4}\rceil]$. This means that there is not an edge in the directed path between $p(cd_i)$ and one of the neighbors which is adjacent to it in the set of bead-line points of D_{iter}, which we denote by $\overrightarrow{t}$, since it must share an edge with the point which directly preceded it and it must share an edge with $p(la_i)$. Consequently, $\overrightarrow{t}$ must be an endpoint of the directed path of the configuration C' since $\mathrm{dom}(C') = S_\delta$, $\overrightarrow{t}$ only has two neighbors in S_δ, and one of $\overrightarrow{t}$'s neighbors does not share an edge with $\overrightarrow{t}$.

The second case we consider is that the first point in P^* which is not a point in the bead-line point of D_{iter} is not in the set $\cup_{i\in[1,\lceil\frac{\delta(5(\delta-1)+1)}{4}\rceil]}\{p(ra_i), p(rc_i), p(la_i), p(lc_i)\}$. This means it is either the point in the path of the `LEFT-WALL` gadget which is adjacent to ca_1 or the point in the path of the `SPACER` gadget which is adjacent to the bead-line points of D_{iter}. In either case, it must be the case that the first bead in P^* is adjacent to one of the points in $\cup_{i\in[1,\lceil\frac{\delta(5(\delta-1)+1)}{4}\rceil]}\{p(ra_i), p(rc_i), p(la_i), p(lc_i)\}$ (by the

assumption P^* isn't a contiguous subsequence of P'). A similar argument to the first case we considered shows that one of the beads adjacent to the first bead in P^* must be an end point of the path of C'.

Since every iteration D_{iter} where the subsequence of P' consisting of exactly the bead-line points of D_{iter} is not a contiguous subsequence of P' must contain an endpoint of P', there can be at most two iterations where the subsequence of P' consisting of exactly the bead-line points of D_{iter} is not a contiguous subsequence of P'. This along with the fact C' is infinite and σ' is finite shows that there exists an iteration D_{iter} such that the subsequence of P' consisting of exactly the bead-line points of D_{iter}, denoted by $\bar{P}$, is a contiguous subsequence of P', and no point in D_{iter} is contained in $\mathrm{dom}(\sigma')$. This concludes the proof of the claim.

To reduce notation, we drop the α, δ and $\mathcal{H}$ when referring to elongations, favorable elongations and $\xrightarrow[\mathcal{H},t]{\alpha,\delta}$ since they are clear from context. That is, for the rest of this section, these terms are implicitly referring to the parameters of $\mathcal{O}'$. Another convenient piece of terminology we use is we say that a configuration C'_i in $\overrightarrow{C'}$ *stabilizes* a bead if that bead appears in C'_i but not in C'_{i-1}. Similarly, we say a configuration C'_i stabilizes a bond provided that bond appear in C'_i but not in C'_{i-1}. Let BL be a set of bead-line points in an iteration D with a contiguous routing. We know such a D exists due to Claim E.2. For the rest of this section we shorten the notation $p_D(b)$ to just $p(b)$ since D is clear from context.

Let $C_a = (P_a, w_a, H_a) \in \mathcal{A}[\mathcal{O}']$ be an element of the assembly sequence $\overrightarrow{C'}$ and let $C_b = (P_b, w_b, H_b)$ be an $\mathcal{H}$-valid α-δ-favorable elongation of C_a. We call the set of bonds in the set $H_b \setminus (H_b \cap H')$ the set of *phantom bonds of* C_a and we denote this set by $\mathcal{PH}(C_b)$. When $H_b = \mathcal{PH}(C_b)$ we say that C_b is *stabilized by only phantom bonds*. We call $H_b \setminus \mathcal{PH}(C_b)$ the set of *visible bonds of* C_b. Intuitively, the phantom bonds of a favorable elongation are the bonds which help to stabilize a bead in the next configuration in the foldable sequence, but do not show up in the terminal configuration. The set of visible bonds do show up in the terminal configuration. Note here that C' is fixed and we are always talking about the phantom bonds with respect to C'. Figure 51 shows an example of both phantom and visible bonds. Note that the bonds between the purple bead and the maroon beads are phantom bonds (since they do not appear in the terminal configuration), and the bond between the orange and aqua bead is a visible bond. Note that since any elongation can have at most δ nascent beads and a bead can have at most 5 bonds, there can be at most 5δ phantom bonds in any favorable elongation. We denote the total number of bonds made by the nascent beads in an elongation C_e by $NB(C_e)$.

Let R' be the routing of C'. Without loss of generality, we assume that the first bead to be stabilized by $\mathcal{O}'$ in BL is at location $p(ca_1)$ (the right most bead in Figure 9. Let u be the subsequence (which is not contiguous) of beads in R' constructed by adding a bead $(\overrightarrow{x}, b)$ to u if and only if $\overrightarrow{x} \in \cup_{i \in [1, \lceil \frac{\delta(5(\delta-1)+1)}{4} \rceil]} \{p(ca_i), p(cc_i), p(ce_i), p(cg_i)\}$ (these correspond to the points where the purple beads are located in the Figure 9. Let o be the subsequence of beads in R' constructed by adding a bead $(\overrightarrow{x}, b)$ to o if and only if $\overrightarrow{x} \in \cup_{i \in [1, \lceil \frac{\delta(5(\delta-1)+1)}{4} \rceil]} \{p(ra_i), p(rc_i), p(la_i), p(lc_i)\}$ (these correspond to the points where the orange beads are located in the Figure 9. Now, let $\overrightarrow{C'_u}$ be the subsequence of configurations in $\overrightarrow{C'}$ constructed by adding a configuration C'_i to $\overrightarrow{C'_u}$ provided that C'_i stabilizes a bead in the subsequence u. Similarly, let $\overrightarrow{C'_o}$ be the subsequence of configurations in $\overrightarrow{C'}$ constructed by adding a configuration C_i to $\overrightarrow{C'_o}$ provided that C_i stabilizes a bead in the subsequence o. Recall that given a bead $b = (\overrightarrow{x}, a)$ (which is a bead type along with a point), $\mathrm{dom}(b) = \overrightarrow{x}$. We define the sequence $\mathrm{dom}(u)$ and $\mathrm{dom}(o)$ to be sequences in $\mathbb{T}$ such that $\mathrm{dom}(u) = (\overrightarrow{x}_i)_{i=1}$ where $\overrightarrow{x}_i = \mathrm{dom}(u(i))$ for all i and $\mathrm{dom}(o) = (\overrightarrow{x}_i)_{i=1}$ where $\overrightarrow{x}_i = \mathrm{dom}(o(i))$ for all i

Let $\overrightarrow{C^*} = (C^*_i)_{i=0}$, where $C^*_i = (P^*_i, w^*_i, H^*_i)$, be a sequence of configurations such that C^*_i is a favorable elongation of C'_i and $C_{i+1} \sqsubseteq C^*_i$. We define $\overrightarrow{C^*_o}$ to be the subsequence of configurations in $\overrightarrow{C^*}$ such that C^*_i is in $\overrightarrow{C^*_u}$ if and only if C'_{i-1} is in $\overrightarrow{C'_o}$. Similarly, we define $\overrightarrow{C^*_u}$ to be the subsequence of configurations in $\overrightarrow{C^*}$ such that C^*_i is in $\overrightarrow{C^*_u}$ if and only if C'_{i-1} is in $\overrightarrow{C'_p}$. Intuitively, these are the favorable elongations which are projected to stabilize beads in o and u.

Claim. Suppose that i is such that $H'_{i+1} = H'_i$. Then $NB(C^*_{i+1}) \geqslant NB(C^*_i)$.

Suppose $t \in \mathbb{N}$ is such that C^*_i is an elongation of C'_i by $w[t...t+\delta-1]$. Then it must be the case that all bonds in C^*_i occur between some bead type and a bead type in $w[t+1..t+\delta-1]$ (otherwise during the projection the bond would be added to C'_i and consequently H'_{i+1} would not equal H'_i). Now, consider the favorable elongations of C'_{i+1}. Since, by definition, the favorable elongations include elongations of length shorter than δ, it also includes the favorable elongations by $w[t+1..t+\delta-1]$. Now, observe that there exists a configuration $\bar{C} = (\bar{P}, \bar{w}, \bar{H})$ with $H'_i \subset \bar{H}$ such that $\bar{C}$ is an elongation of C'_{i+1} by

$w[t+1..t+\delta-1]$. Consequently, any favorable elongation of C'_{i+1} by $w[t+1..t+\delta]$ C'' must be such that $NB(C'') \geqslant NB(\bar{C}) \geqslant NB(C^*_i)$.

Claim. For any $i \in \mathbb{N}$, $NB(C^*_{i+1}) \geqslant |\mathcal{PB}(C^*_i)|$.

Suppose $t \in \mathbb{N}$ is such that C^*_i is an elongation of C'_i by $w[t...t+\delta-1]$. Then it must be the case that all bonds in $\mathcal{PB}(C^*_i)$ occur between some bead type and a bead type in $w[t+1..t+\delta-1]$ (otherwise during the projection the bond would be added to C'_i and consequently the bond would not be a phantom bond). Now, consider the favorable elongations of C'_{i+1}. Since, by definition, the favorable elongations can include elongations of length shorter than δ, it also considers the favorable elongations by $w[t+1..t+\delta-1]$. Now, observe that there exists a configuration $\bar{C} = (\bar{P}, \bar{w}, \bar{H})$ with $\mathcal{PB}(C^*_i) \subset \bar{H}$ such that $\bar{C}$ is an elongation of C'_{i+1} by $w[t+1..t+\delta-1]$. Consequently, any favorable elongation C'' of C'_{i+1} by $w[t+1..t+\delta]$ must be such that $NB(C'') \geqslant NB(\bar{C}) \geqslant |\mathcal{PB}(C^*_i)|$.

Since S_δ was constructed so that the Euclidean distance between point $\mathrm{dom}(o(i))$ and point $\mathrm{dom}(o(i+1))$ is $\delta - 1$ and the delay factor of $\mathcal{O}$ is assumed to be $\leqslant \delta - 1$, the only way for phantom bonds to form in C_{k-1} is if the following observation holds.

Observation 13 *Let $h \in \lceil \frac{\delta(5(\delta-1)+1)}{4} \rceil$, and suppose C'_k stabilizes $u(h)$. Then $o(h) \notin \mathrm{dom}(C^*_{k-1})$.*

If this observation didn't hold, then it would not be possible for any bead in the nascent portion to be adjacent to a bead with which it could bind since the nascent portion would be "fully stretched out" as shown in part (a) of Figure 58. A similar argument also allows us to see that C^*_{k-1} must be stabilized by only phantom bonds.

Observation 14 *Let $h \in \lceil \frac{\delta(5(\delta-1)+1)}{4} \rceil$, and suppose C'_k stabilizes $u(h)$. Then C^*_{k-1} is stabilized by only phantom bonds.*

Claim. Let $\delta > 2$ and $\delta' < \delta$. For every sequence $a = (a_i)$ of length δ or greater where $a_i \leqslant \delta'$, there exists $k, l \in \mathbb{N}$ such that $\delta \cdot k < \sum_{i=1}^{l}(a_i) < \sum_{i=1}^{l+1}(a_i) \leqslant \delta \cdot (k+1)$.

Proof. To see this claim, note that since $a_i \leqslant \delta' < \delta$, $\sum_{i=1}^{\delta}(a_i) \leqslant \delta(\delta-1) = \delta^2 - \delta$. Now consider the $\delta - 1$ many sets given by $[j\delta, (j+1)\delta)$ for each $j \in \mathbb{N}$ such that $0 \leqslant j \leqslant \delta - 1$. Note that there are δ many sums $\sum_{i=1}^{m}(a_i)$ for each $m \in \mathbb{N}$ such that $1 \leqslant m \leqslant \delta$. By the pigeonhole principle, there exists a k such that at least two such sums must be numbers in the set $(j\delta, (j-1)\delta]$ for $j = k+1$. Let l be such that $\sum_{i=1}^{l}(a_i)$ is the first of these two sums. The existence of k and l prove the claim.

Observation 15 *If C'_i stabilizes a bead which has a position in BL except for the last δ' beads in BL, C^*_i must have phantom bonds.*

Observation 16 *Observation 15 implies that if C'_i stabilizes a bead which has position in BL, C^*_i at most $\delta' - 1$ (notice $\delta' - 1 \leqslant \delta - 2$) nascent beads can be in the correct position. In other words, the last bead in the nascent portion of C^*_i must always be in the incorrect position.*

This final claim allows us to say that every δ orange beads the bonds required by a favorable configuration to stabilize an orange increases. It does this by showing that there exists an "intermediate configuration" C'_k between configurations which stabilize orange beads in the foldable sequence such that C^*_k does not have a nascent orange bead in the proper position. To show that such a C'_k exists, we consider the case where C^*_k has a visible bond, but it is not in the right position. An example where there is a visible bond, but the orange bead isn't in the correct position is shown in part (b) Figure 58. In this case, the configuration which stabilizes this bead, must use an "extra bond" to cause the bead to be stabilized in the proper position. Consequently, this configuration must have one more nascent bond than the configuration which stabilized the previous orange bead. In the case where all configurations in the foldable sequence have elongations which place the orange bead in the correct position, we show that since only $\delta - 2$ beads can be stabilized in the correct positions (by Observation 15, there comes a point where an extra intermediate configuration C'_k occurs between configurations which stabilize orange beads in the foldable sequence. This configuration C'_k is required to only be stabilized using phantom bonds. Thus, the next configuration to stabilize an orange bead must "overcome" those phantom bonds by using one more nascent bond than the elongation which stabilized the previous orange bead.

Claim. For every $j \in [1, 5(\delta-1)+1]$, there exists C'_k such that $C_o(j\delta) \to C'_k$, $C'_k \to C_o((j+1)\delta)$ and $\mathcal{PB}(C^*_o(j\delta)) < \mathcal{PB}(C^*_k) < NB(C^*_o((j+1)\delta))$.

Proof. Let $j \in [1, 5(\delta-1)+1]$. Also, let $\overrightarrow{B}$ be the configuration sequence from $C_o(j\delta)$ to $C_o(j(\delta+1))$. Due to the constraints placed on how $\mathcal{O}'$ can assemble BL in Observation E.2, we know that only one such sequence exists. Without loss of generality, we assume that the first configuration C'_i in $\overrightarrow{C'}$ such that $\text{dom}(C'_i) \cap BL \neq \varnothing$ has $p(ca_1) \in \text{dom}(C_i)$. That is, the system assembles the bead line points shown as blue beads in Figure 54 starting from the bottom. We consider two cases: 1) for every configuration C'_i in $\overrightarrow{B}$ there exists a favorable elongation C^*_i of C'_i such that every visible bond in C^*_i involves a bead which has a position in BL and 2) there exists some configuration C'_i of $\overrightarrow{B}$ such that all favorable elongations of C'_i have a visible bond which involves a bead which has a position not in BL.

For the first case, we can assume that for every C^*_i which is an elongation of some configuration in $\overrightarrow{B}$, there is no bead in C^*_i which is outside BL and has a visible bond. We first show that there exists a configuration C'_l such that $C_o(j\delta) \to C'_l$, $C'_l \to C_o((j+1)\delta)$ and C'_l is stabilized by only phantom bonds. Let $d, h \in \mathbb{N}$ be such that $C'_d = C_o(j\delta)$ and $C'_{d+h} = C_o((j+1)\delta)$. Note that for $d', h \in \mathbb{N}$ if $R'(d')$ contains $\text{dom}(o(i))$ and $R'(h')$ contains $\text{dom}(o(i+1))$, $h' - d' = \delta - 1$. This means that $h = \delta(\delta-1) = \delta^2 - \delta$ which implies $\text{dom}(C'_{d+h}) \setminus \text{dom}(C'_d) = \delta^2 - \delta$.

Define C^{**} to be the subsequence of C^* such that $C^{**} = C^*_d$ and C^*_i is in C^{**} if and only if $d < i \leqslant d+h$ and C^*_i is not an elongation of any favorable elongation of C'_{i-1}. Intuitively, C^{**} includes a configuration C if that configuration is a favorable elongation which forces a bead that was in one position in a previous favorable elongation to switch to a new position due to phantom bonds, . Note that since every configuration C which stabilizes a new bead in BL must "incorrectly stabilize" the last bead (per Observation 16), this set is not empty since another configuration C' will stabilize that bead in the correct position and consequently there will not be any favorable elongation of C' which is an elongation of C. In fact, from Observation 16 it follows that C^{**} has at least δ configurations since $\frac{\delta^2-\delta}{\delta-2} \geqslant \frac{\delta(\delta-1)}{(\delta-1)} = \delta$.

Observe that in order for C^{**}_i to contain a visible bond, it must be the case that there exist r such that $\text{dom}(o(r)) \in \text{dom}(C^{**}_i)$ (by the assumption of case 1). As noted above if $R'(d')$ contains a point $\text{dom}(o(j))$ and $R'(h')$ contain $\text{dom}(o(j+1))$, then $h' - d' = \delta - 1$. Hence, there exists $\overrightarrow{c}$ such that $R'(c + (\delta-1)i')$ contains a point in o for all of $i' \in [1, 5(\delta-1)+1]$. Let $b = (b_i)$ be the sequence in $\mathbb{N}$ such that b_i is number of beads in C^{**}_i which are correctly stabilized. Recall that C^{**} has at least δ elements which implies b does as well, and note that it follows from Observation 16 that for all i, $b_i < \delta - 1$. It now follows from Claim E.2 that there exists $l', k \in \mathbb{N}$ such that $c + (\delta-1)k < \Sigma_{i=1}^{l'} b_i < \Sigma_{i=1}^{l'+1} b_i \leqslant c + (\delta-1)(k+1)$. But, this means that the elongation $C^{**}_{l'+1}$ does not contain any visible bonds since it can only potentially stabilize the beads $R'(\overrightarrow{c} + \Sigma_{i=1}^{l}(b_i)), R'(\overrightarrow{c} + \Sigma_{i=1}^{l}(b_i)+1), ..., R'(\overrightarrow{c} + \Sigma_{i=1}^{l+1}(b_i)-1)$ in the correct positions and this does not include any bead which has a position in o.

Let $l = l' + 1$, and let k be such that $C'_k = \rho_\delta(C^{**}_l)$. So far, we have established that there exists C^{**}_l such that $C_o(j\delta) \to C'_k$, $C'_k \to C_o((j+1)\delta)$ and C^*_k is stabilized by only phantom bonds. Let t be such that $C_o(t) \to C'_k$ and $C_o(t)$ is maximal. It follows from the fact that C^*_k is in the sequence C^{**} that $|\mathcal{PB}(C^*_o(t))| < NB(C^*_k)$. Indeed, suppose for the sake of contradiction that $|\mathcal{PB}(C^*_o(t))| \geqslant NB(C^*_k)$. Then there exists a favorable elongation C_{lb} of C'_k where the bonds in $\mathcal{PB}(C^*_o(t))$ "dominate" the elongation. This implies that there exists an elongation of $C_o(t)$ such that C^*_k is an elongation of $C_o(t)$. This contradicts the way we constructed C^{**}.

For the second case, let C'_r be the configuration such that all elongations of C'_r have a nascent bead involved in a visible bond which has a position outside of BL. Furthermore, let i be such that $C_o(i)$ is the minimal element such that $C'_r \to C_o(i)$. Intuitively, C^*_r makes a visible bond by placing an orange bead in a position which is next to an aqua bead, but not in S_δ as shown in part (b) of Figure 58. The configuration $C_o(i)$ is a configuration where this same orange bead is stabilized in the "correct" position. Now, we prove that $\mathcal{PB}(C^*_o(i-1)) \leqslant NB(C^*_u(i))$, $NB(C^*_p(i)) < NB(C^*_r)$, and $NB(C^*_r) < NB(C^*_o(i))$ (otherwise, the orange bead could be incorrectly stabilized).

First, note that $\mathcal{PB}(C^*_o(i-1)) \leqslant NB(C^*_u(i))$ follows directly from Claim E.2. To see that $NB(C^*_p(i)) < NB(C^*_r)$ note that if this was not the case there would exist an elongation where the bonds of $C^*_p(i)$ dominate and there would exist an elongation of C'_r with no visible bond (since $C^*_p(i)$ can't have a visible bond). To see that $NB(C^*_r) < NB(C^*_o(i))$ note that anything otherwise would mean that a malformed configuration could result. This implies $\mathcal{PB}(C^*_o(i-1)) < NB(C^*_r) < NB(C^*_o(i))$.

Since all bond strengths are natural numbers, Claim E.2 implies that for every j, $NB(C^*_o(j\delta)) = |\mathcal{PB}(C^*_o(j\delta))| + 1 < NB(C^*_o((j+1)\delta))$. It follows from the construction of S_δ that there are at least $\delta \times (5(\delta-1)+1)$ elements in o. Thus, $NB(C^*_o(\delta \times (5(\delta-1)+1))) \geqslant 5(\delta-1)+1$. But, this contradicts the fact that the total number of bonds involving nascent beads in a δ'-elongation of any configuration is $5(\delta')$ (since $\delta' < \delta$).

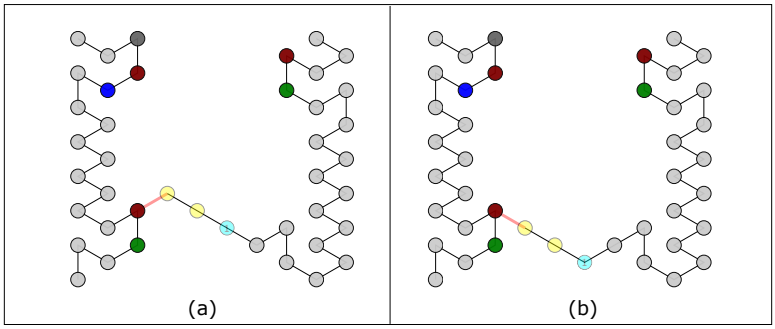


Fig. 58: Any system which uses delay less than 4 must use only phantom bonds to stabilize the first bead otherwise something bad can happen.

F Finiteness of delay-1, arity-1 deterministic oritatami systems

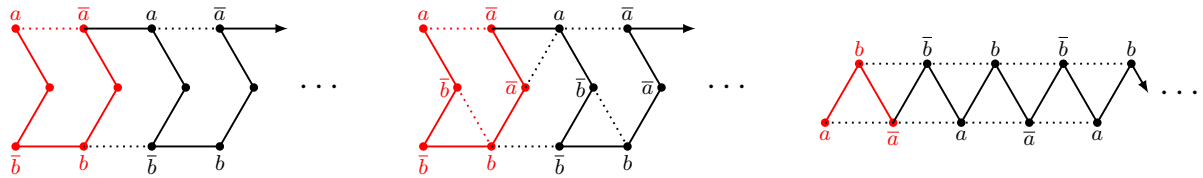


Fig. 59: Deterministically foldable infinite shapes: (Left) A glider at delay-3 and arity-1; (Middle) A glider at delay-2 and arity-2, and (Right) A zigzag at delay-1 and arity-2. Seeds are colored in red. The common bead type set consists of four letters $a, \overline{a}, b, \overline{b}$ and the rule set used in common is complementary: a with $\overline{a}$ and b with $\overline{b}$.

In this section, we prove that oritatami systems cannot yield any infinite terminal conformation at delay 1 and arity 1 deterministically. The finiteness stems essentially from the particular setting of values to delay and arity. The *glider* is a well-known infinite conformation foldable deterministically by an oritatami system at delay 3 and arity 1; see Figure 59 (Left). The glider can be "widened" in order to be folded deterministically at arbitrarily longer delays. The glider can be "reinforced" with more bonds so that it folds at a shorter delay 2 with arity 2 as suggested in Figure 59 (Middle). Even at the shortest possible delay, that is, 1, arity being 2 enables oritatami systems to fold an infinite structure deterministically, as exemplified by a zigzag conformation shown in Figure 59 (Right). These infinite conformations leave just two possible settings of delay and arity under which infinite conformations cannot be folded deterministically: arity is set to 1 and delay is set to either 1 or 2. We will show the finiteness of deterministic folding in the first case in the rest of this paper, and leave the case of delay 2, arity 1 open. Note that even at these settings infinite shapes can be folded nondeterministically; an infinite transcript of inert beads folds into an arbitrary non-self-intersecting path at an arbitrary delay, and arity does not matter because beads are inert.

Let Ξ be a deterministic oritatami system of delay 1 and arity 1. Assume its seed σ consists of n beads for some $n \geq 1$, and along its primary structure, we index these n beads as $a_{-n+1}, a_{-n+2}, \ldots, a_{-1}, a_0$. Let us denote its transcript by $w = a_1 a_2 a_3 \cdots$ for some $a_1, a_2, a_3, \ldots \in \Sigma$. For $i \geq 0$, let C_i be the unique elongation of σ by $w[1..i]$ that is foldable by Ξ. Hence, $C_0 = \sigma$. We assume that the directed path of C_i is indexed rather as $-n+1, -n+2, \ldots, 0, 1, \ldots, i$.

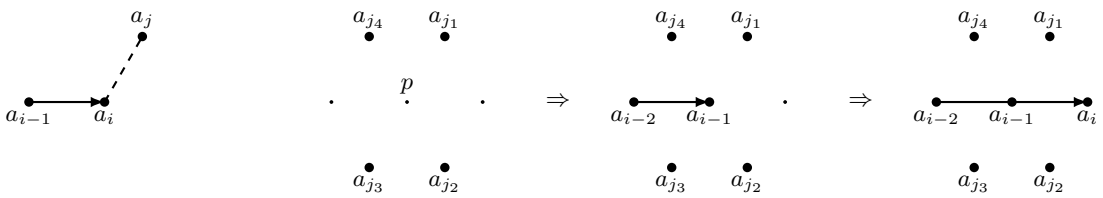


Fig. 60: The two ways for a bead to get stabilized in oritatami systems at delay 1 and arity 1: (Left) by being bound to a bead a_j for some $j \leq i-2$; and (Right) through a tunnel section formed by the four beads $a_{j_1}, a_{j_2}, a_{j_3}, a_{j_4}$.

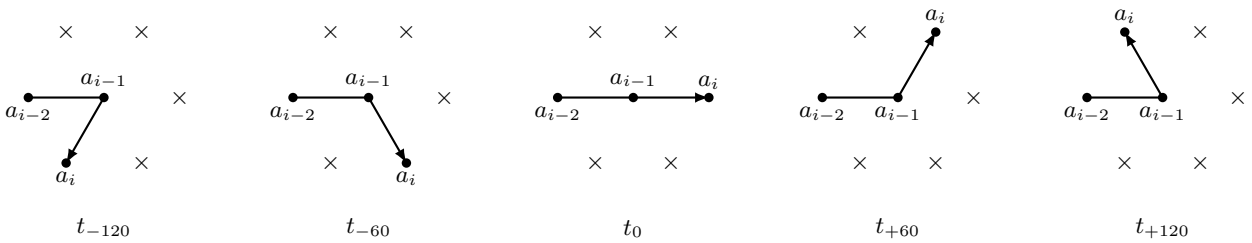


Fig. 61: All possible tunnel sections: acute right turn t_{-120}, obtuse right turn t_{-60}, straight t_0, obtuse left turn t_{+60}, and acute left turn t_{+120}.

Let us consider the stabilization of the i-th bead a_i upon C_{i-1}. The bead cannot collaborate with any succeeding beads $a_{i+1}, a_{i+2}, \ldots$ at delay 1. There are just two ways to get stabilized at delay 1. One way is to be bound to another bead, as shown in Figure 60 (Left). The other way is through a *tunnel section*. A tunnel section consists of four beads that occupy four neighbors of a point. See Figure 60 (Right). Assume that four of the six neighbors of a point p are occupied by beads $a_{j_1}, a_{j_2}, a_{j_3}, a_{j_4}$ with $-n+1 \leq j_1 < j_2 < j_3 < j_4 < i-2$ while the other two are not occupied. If the beads a_{i-2} and a_{i-1} are stabilized respectively at one of the two free neighbors and at p one after another, then the next bead a_i cannot help but be stabilized at the other free neighbor. In this way, a_i can get stabilized without being bound.

Let us now formalize the tunnel section. Four beads $a_{j_1}, a_{j_2}, a_{j_3}, a_{j_4}$ with $-n+1 \leq j_1 < j_2 < j_3 < j_4$ *form a tunnel section around a point* p if there exist an index $k \geq j_4 + 2$ and the foldable configuration $C_k = (P, u, H) \in \mathcal{A}(\Xi)$ such that

1. For all $s \in \{j_1, j_2, j_3, j_4\}$, $(P[s], p) \in E_\Delta$;
2. $(P[k-1], p) \in E_\Delta$; and
3. $P[k] = p$.

We call the four beads $a_{j_1}, a_{j_2}, a_{j_3}, a_{j_4}$ *walls* of this tunnel. The walls and the bead a_{k-1} leave at most one of the neighbors of p free in C_k. If the neighbor is not free, C_k is terminal. Otherwise, a_{k+1} is to be stabilized at the neighbor and yields C_{k+1}. The bead a_{j_4} can be regarded the newest wall because of $j_1, j_2, j_3 < j_4$. If $j_4 \geq 1$, that is, if it is transcribed, then we say the tunnel section is *created by the bead* a_{j_4}. Otherwise, we say it is *equipped in the seed*. Figure 61 exhibits all the five kinds of tunnel sections depending on which neighbors are walls (indicated by $\times$'s), modulo types and indices of wall beads.

Tunnel sections and unbound beads, or more precisely, their *one-time* capability of binding, are the resources for beads to get stabilized deterministically at delay 1 and arity 1 (at longer delays, other ways of non-binding stabilization are possible due to so-called "hidden rule," which never appears in any terminal conformation but indispensable, as argued in [18]). The seed σ of Ξ, consisting of n beads, provides at most n binding capabilities, one per bead. Claim that it can be equipped with at most n tunnel sections. If $n < 4$, it cannot be equipped with any tunnel section. For larger n, any bead a_j has its predecessor or successor or both. By definition, it cannot be a wall of any tunnel around the point where its predecessor or successor is. Therefore, the first bead a_{-n+1} and the last bead a_0 can be a wall of at most five tunnel sections, whereas any other bead a_j with $-n+2 \leq j \leq -1$ can be a wall of at most four tunnel sections. One tunnel section consists of four beads. Therefore, the seed can be equipped with at most $\lfloor (4n+2)/4 \rfloor = n$ tunnel sections.

Being bound for stabilization, a bead will not be able to bind to another bead later due to arity 1. In contrast, if it is stabilized through a tunnel section, then it can provide one-time binding capability and create tunnel sections.

Theorem 17. *Let Ξ be an oritatami system of delay 1 and arity 1 whose seed consists of n beads, and let w be the transcript of Ξ. If Ξ is deterministic, then $|w| \leq 9n$.*

Proof. Assume Ξ is deterministic. Let us represent its transcript w as $w = a_1 a_2 a_3 \cdots$ for beads $a_1, a_2, a_3, \ldots \in \Sigma$. Each of these beads is stabilized either by being bound or through a tunnel section (or by both). How they are stabilized can be described by a binary sequence S of b's (bound) and t's (tunnel section); priority is given to t, that is, $S[i] = t$ if the i-th bead a_i is stabilized not only by being bound but also through a tunnel section. For $\ell \geq 1$, we call a factor $bt^\ell b$ of S a *tunnel of length* ℓ. See Figure 62 (right) for a tunnel of length 3, where $S[i-3..i+1] = btttb$; observe that the bead a_{i-2} is stabilized both by both ways but due to the priority, $S[i-2] = t$.

If $S[i] = b$, that is, if the bead a_i is stabilized not through a tunnel section but by being bound, then it can be involved in three separate tunnel sections as a wall but no more. Indeed, two of the six neighbors

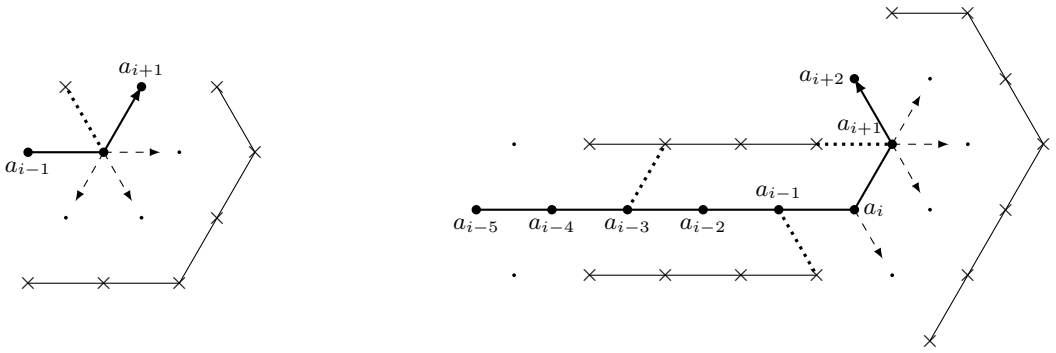


Fig. 62: Stabilization of a bead a_i (Left) by being bound, and (Right) through a tunnel of length 3. The symbol $\times$ indicates that the point is occupied, while the small dot means that the point is free. A dashed arrow indicates that the bead at its origin creates a tunnel at the pointed free point.

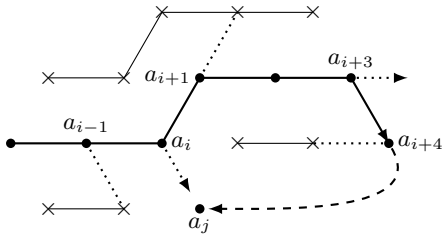


Fig. 63: A tandem of two tunnels.

of the point at which a_i is stabilized have been already occupied by its predecessor a_{i-1} and by the bead to which a_i is bound. It cannot be a wall of a tunnel section around the point where the successor a_{i+1} will be stabilized. A tunnel is of the form bt^+b by definition, that is, it consumes two binding capabilities: one for the transcript to enter it and one for the transcript to decide which way to go after exit; while only the bead stabilized by its last tunnel section can provide a new binding capability. That is, a tunnel consumes at least one binding capability in total. For instance, in Figure 62 (Right), a_{i-3} enters a tunnel of length 3 by being bound, its three successors a_{i-2}, a_{i-1}, a_i are stabilized by the tunnel, and a_{i+1} is also bound, while a_i provides a new binding capability. Since a_{i-1} wastes one unnecessary binding capability so that this tunnel consumes two binding capabilities in total. A tunnel can let the transcript create at most 4 tunnel sections, as suggested in Figure 62 (Right).

If the sequence S is free from any subsequence of the form bt^+bt^+b, then it can factorize as $S = u_1u_2u_3\cdots$ for some $u_1, u_2, u_3, \ldots \in \{b\} \cup bt^+b$. As argued above, each of these factors $u_1, u_2, \ldots$ consumes at least one binding capability. Since the seed can provide at most n binding capabilities, there exists $m \leq n$ such that $S = u_1u_2\cdots u_m$. Let m_1 be the number of tunnels among the m factors. The m_1 tunnels can create at most $4m_1$ tunnel sections in total and the remaining $m - m_1$ factors, which correspond to beads that are bound for stabilization, can create at most $3(m - m_1)$ tunnel sections. The seed is equipped with no more than n tunnel sections. The sum of the length of the m_1 tunnels is hence at most $n + 4m_1 + 3(m - m_1) = n + 3m + m_1$. Consequently, $|S| \leq n + 3m + m_1 + m - m_1 = n + 4m \leq 5n$.

Now we have to handle a subsequence of the form bt^ibt^jb of S for $i, j \geq 1$, which is a tandem of tunnels. Figure 63 shows two tunnels in tandem. The transcript diverts the binding capability which it uses to exit the first tunnel in order to enter the second. Moreover, the beads a_i and a_{i+3}, which are the last beads stabilized by the first and second tunnels, respectively, can provide one binding capability each. Thus, these two tunnels appear to lose only one binding capability *in total* by forming a tandem. This argument is incorrect unless the second tunnel turns right acutely (see Figure 61). Unless turning right acutely, the second tunnel is provided with a right wall. The index of a bead that serves as a right wall must be smaller than $i-2$, and by definition, the bead is connected to a_{i-1} by a primary structure of C_i. If a_i provided a binding capability for a future bead, say a_j $(j > i)$, then their bond once formed would close the curve along the transcript from a_i to a_j and isolate a region including the right wall from the rest of the plane, which includes a_{i-1} due to the Jordan curve theorem. This is contradictory because the primary structure of a conformation is defined to be non-self-interacting. The second tunnel should turn right acutely or a_i cannot provide any binding capability (in order for a_i to provide a binding capability rather to the left of the transcript, then the second tunnel is required to turn rather left acutely).

If the second tunnel turns right acutely, the tandem can provide two binding capabilities at the cost of three as shown in Figure 64. We cannot improve this ratio further even if another tunnel is concatenated

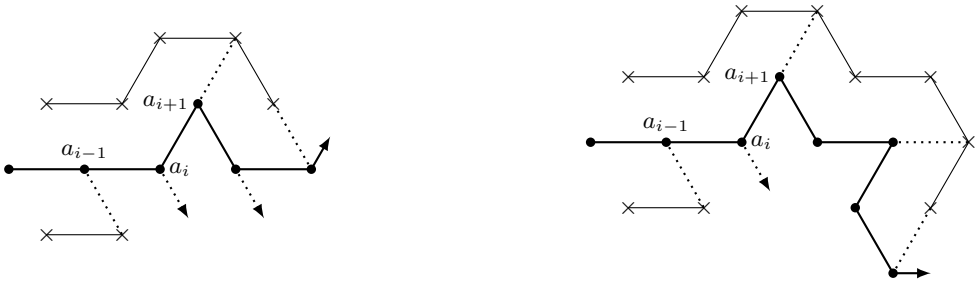


Fig. 64: A tandem of two tunnels can save binding capability but the third one just wastes a bond.

to this tandem. Not turning right acutely, the tunnel makes the binding capabilities provided by a_i or a_{i+2} useless, as discussed above based on the Jordan curve theorem. Turning right acutely, on the other hand, the third tunnel just narrows a binding region of the second tunnel so that these three tunnels in tandem can provide at most two binding capabilities at the cost of four. Therefore, the number of binding capabilities decrements every two tunnels. Let m be the number of tunnels in the sequence S, that is, $m \leq 2n$. We denote their length by $\ell_1, \ell_2, \ldots, \ell_m$. These tunnels consume at least $\lceil m/2 \rceil$ binding capabilities. Hence, at most $n - \lceil m/2 \rceil$ beads can be stabilized by being bound. These beads can create at most $3(n - \lceil m/2 \rceil)$ tunnels. The m tunnels can stabilize $\sum_{i=1}^{m} \ell_m$ beads and create at most $4m$ tunnels. Initially, the system can have at most n tunnels. Combining all of these together, the number of beads that the system can stabilize deterministically is at most

$$n - \left\lceil \frac{m}{2} \right\rceil + \sum_{i=1}^{m} \ell_i \leq n - \left\lceil \frac{m}{2} \right\rceil + n + 3\left(n - \left\lceil \frac{m}{2} \right\rceil\right) + 4m \leq 5n + 2m \leq 9n.$$

Thus, the transcript can be of length at most $9n$.

G Tribute galery

This section displays the oritatami foldings of the same iconic shape at the three scales $\mathscr{A}_3$, $\mathscr{B}_3$ and $\mathscr{C}_3$, and it is a kind of a tribute to the field.

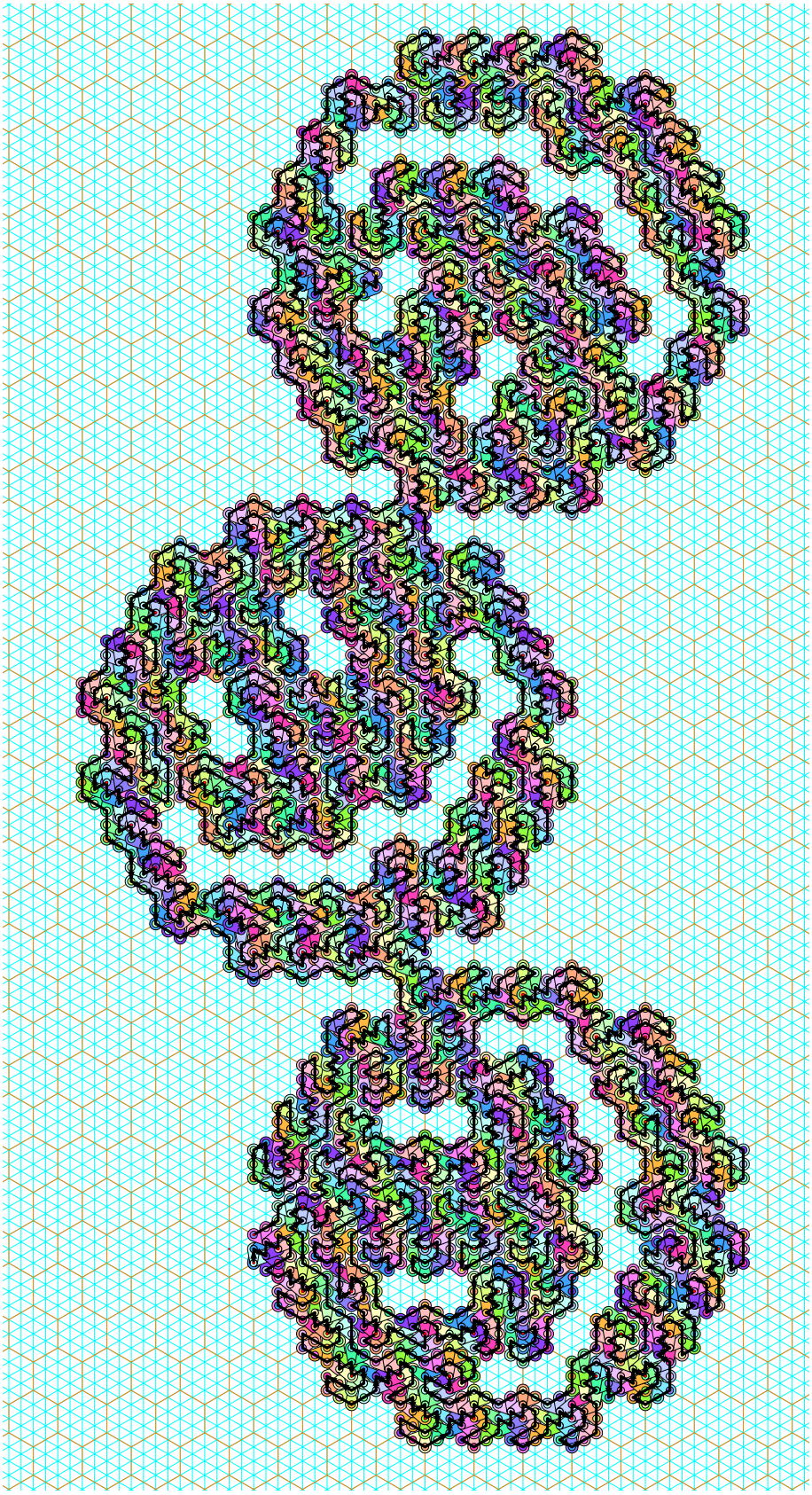


Fig. 65: Oritatami "stacking smileys" at scale $\mathscr{A}_3$.

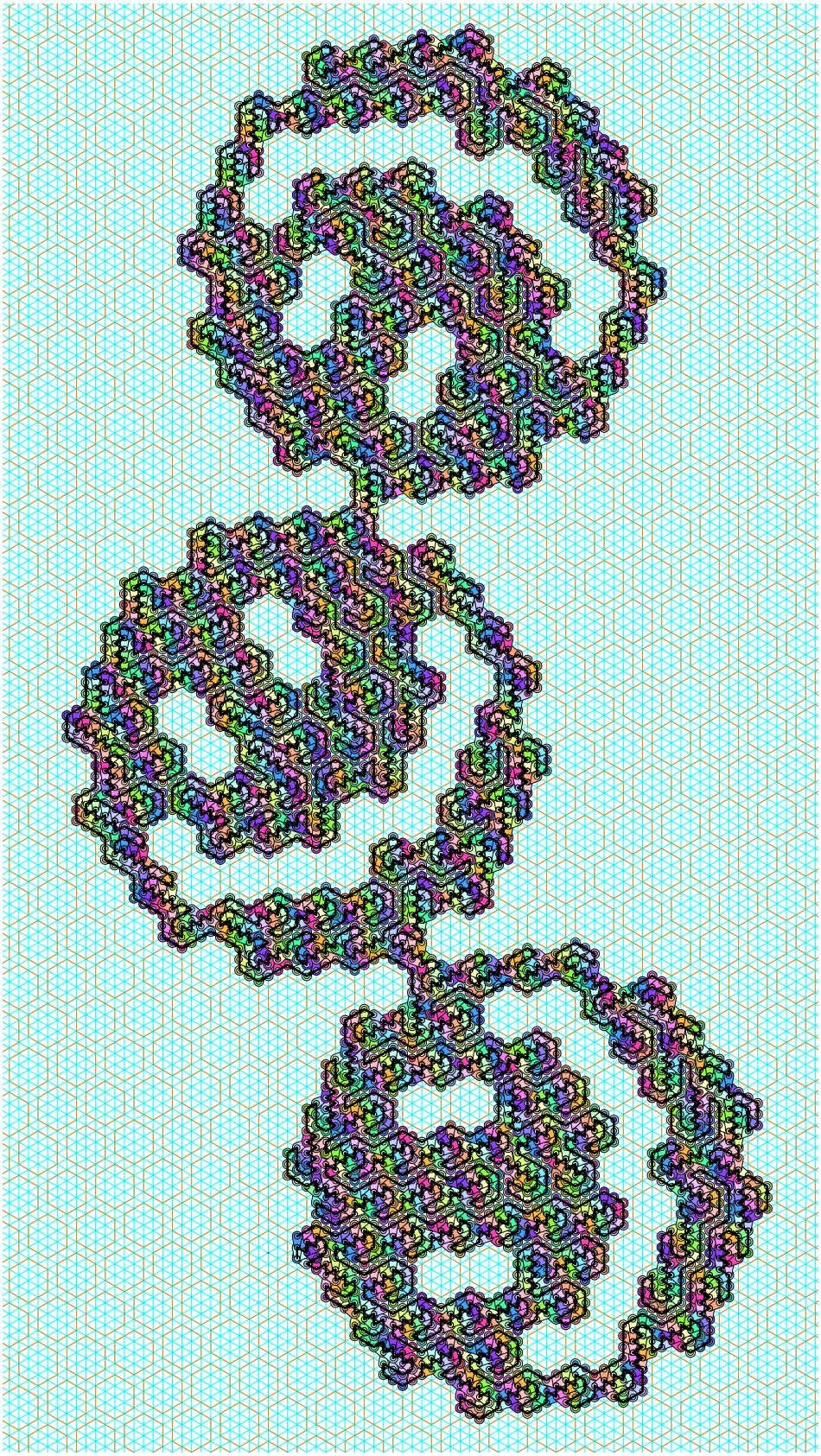


Fig. 66: Oritatami "stacking smileys" at scale $\mathscr{B}_3$.

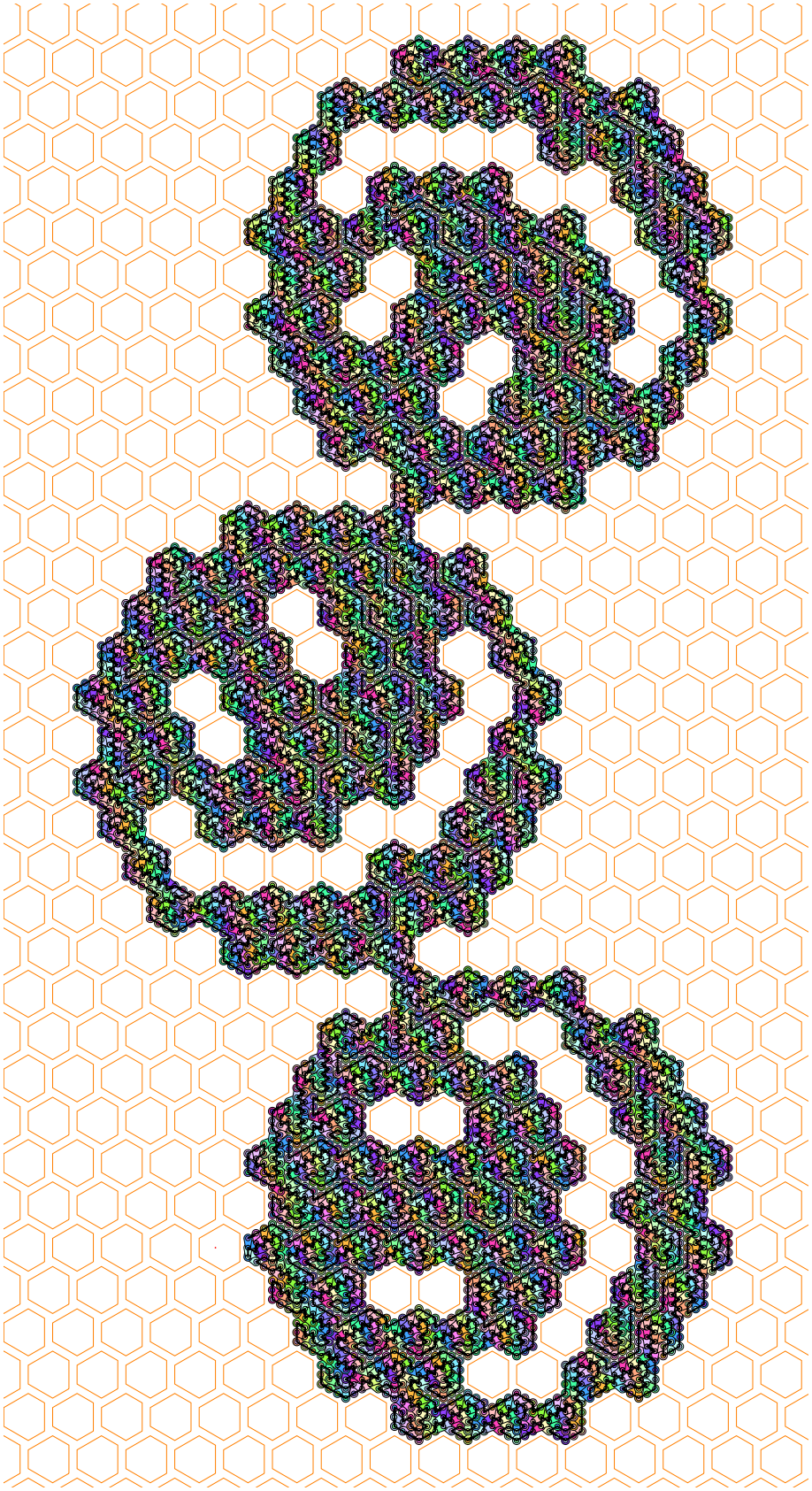


Fig. 67: Oritatami "stacking smileys" at scale $\mathscr{C}_3$.