Improved Lower and Upper Bounds on the Tile Complexity of Uniquely Self-Assembling a Thin Rectangle Non-Cooperatively in 3D

David Furcy · Scott M. Summers · Logan Withers

Abstract
We investigate a fundamental question regarding a benchmark class of shapes in one of the simplest, yet most widely utilized abstract models of algorithmic tile self-assembly. More specifically, we study the directed tile complexity of a \( k \times N \) thin rectangle in Winfree’s ubiquitous abstract Tile Assembly Model, assuming that cooperative binding cannot be enforced (temperature-1 self-assembly) and that tiles are allowed to be placed at most one step into the third dimension (just-barely 3D). While the directed tile complexities of a square and a scaled-up version of any algorithmically specified shape at temperature 1 in just-barely 3D are both asymptotically the same as they are (respectively) at temperature 2 in 2D, the (nearly tight) bounds on the directed tile complexity of a thin rectangle at temperature 2 in 2D are not currently known to hold at temperature 1 in just-barely 3D. Motivated by this discrepancy, we establish new lower and upper bounds on the directed tile complexity of a thin rectangle at temperature 1 in just-barely 3D. The proof of our upper bound is based on the construction of a novel, just-barely 3D temperature-1 self-assembling counter. Each value of the counter is comprised of \( k - 2 \) digits, represented in a geometrically staggered fashion within \( k \) rows. This nearly optimal digit density, along with the base of the counter, which is proportional to \( N^{\frac{k-1}{k-2}} \), results in an upper bound on the directed tile complexity of a thin rectangle at temperature 1 in just-barely 3D of \( O \left( N^{\frac{k-1}{k-2}} + k \right) \), and is an asymptotic improvement over the previous state-of-the-art upper bound. On our way to proving our lower bound, we develop a new, more powerful type of specialized Window Movie Lemma that lets us bound the number of “sufficiently similar” ways to assign glues to a set (rather than a sequence) of fixed locations. Consequently, our lower bound on the directed tile complexity of a thin rectangle at temperature 1 in just-barely 3D of \( \Omega \left( N^{\frac{1}{k-1}} \right) \), is also an asymptotic improvement over the previous state-of-the-art lower bound.
Keywords  Self-Assembly · Algorithmic Self-Assembly · Tile Self-Assembly · Window Movie Lemma · Lower Bounds · Directed Tile Complexity

1 Introduction

A key objective in algorithmic self-assembly is to characterize the extent to which an algorithm can be converted to an efficient self-assembling system comprised of discrete, distributed and disorganized units that, through random encounters with, and locally-defined reactions to each other, coalesce into a terminal assembly having a desirable form or function. In this paper, we study a fundamental theoretical question regarding a benchmark class of shapes in one of the simplest yet most popular abstract models of algorithmic self-assembly.

Ubiquitous throughout the theory of tile self-assembly, Erik Winfree’s abstract Tile Assembly Model (aTAM) [1] is a discrete mathematical model of DNA tile self-assembly [2] that augments classical Wang tiling [3] with a mechanism for automatic growth. In the aTAM, a DNA tile is represented by a unit square (or cube) tile type that may neither rotate, reflect, nor fold. Each side of a tile type is decorated with a glue consisting of both a non-negative integer strength and a string label, the symbols of which are drawn from some fixed alphabet. A tile set is a finite set of tile types, from which infinitely many tiles of each type may be instantiated. If one tile is positioned at an unoccupied location Manhattan distance 1 away from another tile and their opposing glues are equal, then the two tiles bind with the strength of the adjacent glues creating a tile assembly of size two. A special seed tile type is designated and a seed tile, which defines the seed-containing assembly, is placed at some fixed location.

During the process of self-assembly, a sequence of tiles bind to and never detach from the seed-containing assembly, provided that each one, in a non-overlapping fashion, binds to one or more tiles in the seed-containing assembly with total strength at least a certain positive integer value called the temperature. If the temperature is greater than or equal to 2, then it is possible to enforce cooperative binding, where a tile may be prevented from binding at a certain location until at least two adjacent locations become occupied by tiles. Otherwise, only non-cooperative binding is allowed (temperature-1 self-assembly). A fundamental theoretical question in tile self-assembly is determining the effect of the value of the temperature on the computational and geometric expressiveness of tile self-assembly. To that end, temperature-1 self-assembly has been shown to hinder the efficient self-assembly of shapes when tile assemblies are required to be fully connected [4] or contain no glue mismatches [5]. Temperature-1 self-assembly is also neither intrinsically universal [6, 7], nor capable of bounded Turing computation [7]. Recently and quite remarkably, Meunier, Regnault and Woods [8] established a general pumping lemma for temperature-1 self-assembly, nearly proving a conjecture by Doty, Patitz and Summers [9] on the computational weakness of temperature-1 self-assembly.

Interestingly, temperature-1 self-assembly does not limit the computational or geometric expressiveness of generalizations of the aTAM [10–14]. This is also true even when the generalization only adds a small number of additional features to the model, like a single negative glue [15], duple tiles [16], or another plane in which (cubic)
3D tiles are allowed to be placed [17–20]. The latter variant is colloquially known as “just-barely” 3D self-assembly. In this paper, we study the limitations of temperature-1 self-assembly for unique shape-building in the just-barely 3D aTAM. We are specifically interested in studying the directed tile complexity of a given target shape, or the size of the smallest directed tile set, which is a tile set where, regardless of the order in which the tiles thereof bind to the seed-containing assembly, the result is a unique terminal assembly in which tiles are placed on and only on points of a given target shape. Although temperature-1 self-assembly cannot enforce cooperative binding, there is a striking resemblance of its computational and geometric expressiveness in just-barely 3D, to that of temperature-2 self-assembly in 2D, with respect to the directed tile complexity of two benchmark shapes: a square and a scaled-up version of any algorithmically specified shape.

Adleman, Cheng, Goel and Huang [21] proved, using optimal base conversion, that the directed tile complexity of an \(N \times N\) square at temperature 2 in 2D is \(O\left(\frac{\log N}{\log \log N}\right)\), matching a corresponding lower bound for all Kolmogorov-random \(N\) and all positive temperature values, set by Rothemund and Winfree [4]. The lower bound also holds in just-barely 3D. An \(O\left(\frac{\log N}{\log \log N}\right)\) upper bound for the directed tile complexity of an \(N \times N\) square at temperature 1 in just-barely 3D was established by Furcy, Micka and Summers [18] via a just-barely 3D, optimal encoding construction at temperature 1. Just-barely 3D, optimal encoding at temperature 1 was inspired by, achieves the same result as, but is drastically different from the 2D optimal encoding at temperature 2 developed by Soloveichik and Winfree [22], who proved that the directed tile complexity of a scaled-up version of any algorithmically specified shape \(X\) at temperature 2 is \(\Theta\left(\frac{K(X)}{\log K(X)}\right)\), where \(K(X)\) is the size of the smallest Turing machine that outputs the list of points in \(X\). This tight bound for temperature-2 self-assembly in 2D was shown to hold for temperature-1 self-assembly in just-barely 3D by Furcy and Summers [20].

Another benchmark shape, for positive integers \(k, N\), is the \(k \times N\) rectangle, where \(k < \frac{\log N}{\log \log N - \log \log \log N}\), making it “thin”. A thin rectangle is an interesting testbed because its restricted height creates a limited channel through which tiles may propagate information, for example, the current value of a self-assembling counter. In fact, Aggarwal, Cheng, Goldwasser, Kao, Moisset de Espanés and Schweller [23] used an optimal, base-\(\lceil N^{\frac{1}{k}}\rceil\) counter that uniquely self-assembles within the confines of a thin rectangle to derive an upper bound of \(O\left(N^{\frac{1}{k}} + k\right)\) on the directed tile complexity of a \(k \times N\) thin rectangle at temperature 2 in 2D. They then leveraged the limited bandwidth of a thin rectangle in a counting argument for a corresponding lower bound of \(\Omega\left(\frac{N^{\frac{1}{k}}}{k}\right)\).

The previous theory for a square and an algorithmically specified shape would suggest that these thin rectangle bounds should hold at temperature 1 in just-barely 3D. Yet, we currently do not know if this is the case. Thus, the power of temperature-1 self-assembly in just-barely 3D resembles that of temperature-2 self-assembly in 2D,
with respect to the directed tile complexities of a square and a scaled-up version of any algorithmically specified shape, but not a thin rectangle.

Motivated by this theoretical discrepancy, we prove new lower and upper bounds on the directed tile complexity of a thin rectangle at temperature 1 in just-barely 3D, where \( R_{k,N}^3 \) is a just-barely 3D \( k \times N \) rectangle if it satisfies \( \{0, 1, \ldots, N - 1\} \times \{0, 1, \ldots, k - 1\} \times \{0\} \subseteq R_{k,N}^3 \subseteq \{0, 1, \ldots, N - 1\} \times \{0, 1, \ldots, k - 1\} \times \{0, 1\} \). See Table 1 for a quick summary of our results and Table 2 for how our results compare with previous state-of-the-art results.

First, we have our main upper bound:

**Theorem 1** For \( k > 2 \), the directed tile complexity of a just-barely 3D \( k \times N \) rectangle at temperature 1 is \( O\left(N^{\frac{1}{k-1}+k}\right)\).

Theorem 1, which we prove in Section 3, is an asymptotic improvement over the previous state-of-the-art upper bound: \( O\left(N^{\frac{1}{\lfloor k/3 \rfloor}+\log N}\right)\) [19]. The latter bound is based on the self-assembly of a just-barely 3D counter that uniquely self-assembles at temperature 1, but whose base \( M \) depends on the dimensions of the target rectangle, where each digit is represented geometrically and in binary within a just-barely 3D region of space comprised of \( \Theta(\log N) \) columns and 3 rows. In a construction like this, the number of rows used to represent each digit affects the base of the counter, which, for a thin rectangle, turns out to be the asymptotically-dominating term in the tile complexity.

For example, in the Furcy, Summers and Wendlandt construction, the number of rows per digit is 3, so the base is set to \( \Theta\left(N^{\frac{1}{\lfloor k/3 \rfloor}}\right)\). Intuitively, “squeezing” more digits into the counter for the same rectangle of height \( k \) will result in a decrease in the base and therefore the tile complexity. Our construction for Theorem 1 is based on the self-assembly of a just-barely 3D counter similar to the Furcy, Summers and Wendlandt construction, but the geometric structure of our counter is organized according to value regions, or just-barely 3D regions of space comprised of \( k \) rows and \( \Theta\left(N^{\frac{1}{k-1}}\right) \) columns, in which \( k - 2 \) base-\( \Theta\left(N^{\frac{1}{k-1}}\right) \) digits are represented in a staggered fashion. This increase in digit density is the main reason why the “\( \left\lfloor\frac{k}{3}\right\rfloor \)” term from the Furcy, Summers and Wendlandt upper bound is replaced by a “\( k - 1 \)” term in Theorem 1.

Second, we have our main lower bound:

**Theorem 2** For positive \( k \), the directed tile complexity of a just-barely 3D shape that extends at most \( k \) points vertically and at most \( N \) points horizontally at temperature 1 is \( \Omega\left(N^{\frac{1}{k}}\right)\).

Theorem 2, which we prove in Section 4.3, yields as a corollary an asymptotic improvement over the previous state-of-the-art lower bound for a just-barely 3D \( k \times N \) rectangle: \( \Omega\left(N^{\frac{1}{k^2}}\right)\). Technically, the latter bound is not explicitly proved (or even stated) and therefore cannot be referenced, but it can be derived via a straightforward
Table 1  In the right half of this table, we give our improved lower and upper bounds on the directed tile complexity of rectangles, the main contributions of this paper, and compare them with corresponding bounds in 2D at temperature 2

|                        | 2D Temperature 2 |                                      | Just-barely 3D Temperature 1 |
|------------------------|------------------|--------------------------------------|-----------------------------|
|                        | Lower bound      | Upper bound                          | Lower bound                  |
| $k \times N$ rectangle, for $k > 2$ | $\Omega \left( \frac{N^{1/k}}{k} \right)$ [23] | $O \left( N^{1/k} + k \right)$ [23] | $\Omega \left( N^{1/k} \right)$             |
|                        |                  |                                      | $O \left( N^{1/k-1} + k \right)$ |
Table 2  Directed tile complexity for benchmark shapes in the aTAM

| Shape                          | 2D Temperature 2 | 3D Temperature 1 |
|--------------------------------|------------------|------------------|
|                                | Lower bound      | Upper bound      | Lower bound      | Upper bound      |
| $N \times N$ Square            | $\Theta \left( \frac{\log N}{\log \log N} \right)$ [4] | Same as 2D Temperature 2 |
| Algorithmically-defined shape $X$ | $\Theta \left( \frac{K(X)}{\log K(X)} \right)$ [22] | Same as 2D Temperature 2 |
| $k \times N$ rectangle, for $k > 2$ | $\Omega \left( \frac{N^2}{k} \right)$ [23] | $O \left( N^k + k \right)$ [23] | $\Omega \left( \frac{N^k}{k} \right)$ | $O \left( N^{\left\lfloor \frac{k}{3} \right\rfloor} + \log N \right)$ [19] |

Here, $K(X)$ is the size of the smallest Turing machine that outputs the list of points in $X$. The unreferenced lower bound in the last row follows from an application of the Window Movie Lemma from [6].
application of the standard Window Movie Lemma (WML) introduced in [6]. This lower bound holds even when the tile set is not directed, meaning that the tile set produces more than one terminal assembly (similarly for the 2D lower bound in [23]).

On our way to proving Theorem 2, in Section 4.1, we prove Lemma 2, which is essentially a new, more powerful type of Window Movie Lemma technique, specifically designed for temperature-1 self-assembly of a just-barely 3D shape. Lemma 2 lets us develop a more refined counting argument based on upper bounding the number of “sufficiently similar” ways for an assembly sequence to assign glues to a fixed set of locations abutting a plane. Intuitively, two assignments are sufficiently similar if, up to translation, they respectively agree on: the set of locations to which glues are assigned, the local order in which certain consecutive pairs of glues appear, and the glues that are assigned to a certain set (of roughly half) of the locations. After we prove Theorem 2, in Section 4.4, we prove via construction, that Lemma 2, unlike the standard WML, requires the tile set to be directed.

2 Formal Definition of the Abstract Tile Assembly Model

In this section, we briefly sketch a strictly 3D version of Winfree’s abstract Tile Assembly Model (see also [4, 24, 25]).

All logarithms in this paper are base-2. Fix an alphabet $\Sigma$. $\Sigma^*$ is the set of finite strings over $\Sigma$. A grid graph is an undirected graph $G = (V, E)$, where $V \subset \mathbb{Z}^3$, such that, for all $\{\vec{a}, \vec{b}\} \in E$, $\vec{a} - \vec{b}$ is a 3-dimensional unit vector. The full grid graph of $V$ is the undirected graph $G^f_V = (V, E)$, such that, for all $\vec{x}, \vec{y} \in V$, $\{\vec{x}, \vec{y}\} \in E \iff ||\vec{x} - \vec{y}|| = 1$, i.e., if and only if $\vec{x}$ and $\vec{y}$ are adjacent in the 3-dimensional integer Cartesian space.

A 3-dimensional tile type is a tuple $t \in (\Sigma^* \times \mathbb{N})^6$, e.g., a unit cube, with six sides, listed in some standardized order, and each side having a glue $g \in \Sigma^* \times \mathbb{N}$ consisting of a finite string label and a nonnegative integer strength. We assume a finite set of tile types, but an infinite number of copies of each tile type, each copy referred to as a tile. A tile set is a set of tile types and is usually denoted as $T$.

A configuration is a (possibly empty) arrangement of tiles on the integer lattice $\mathbb{Z}^3$, i.e., a partial function $\alpha : \mathbb{Z}^3 \rightarrow T$. Two adjacent tiles in a configuration bind, interact, or are attached, if the glues on their abutting sides are equal (in both label and strength) and have positive strength. Each configuration $\alpha$ induces a binding graph $G^b_\alpha$, a grid graph whose vertices are positions occupied by tiles, according to $\alpha$, with an edge between two vertices if the tiles at those vertices bind. An assembly is a connected, non-empty configuration, i.e., a partial function $\alpha : \mathbb{Z}^3 \rightarrow T$ such that $G^f_{\text{dom } \alpha}$ is connected and $\text{dom } \alpha \neq \emptyset$. Given $\tau \in \mathbb{Z}^+$, $\alpha$ is $\tau$-stable if every cut-set of $G^f_\alpha$ has weight at least $\tau$, where the weight of an edge is the strength of the glue it represents. When $\tau$ is clear from context, we say $\alpha$ is stable. Given two assemblies $\alpha, \beta$, we say $\alpha$ is a subassembly of $\beta$, and we write $\alpha \subseteq \beta$, if $\text{dom } \alpha \subseteq \text{dom } \beta$ and, for all points $\vec{p} \in \text{dom } \alpha$, $\alpha(\vec{p}) = \beta(\vec{p})$. 

 Springer
Let \( \mathcal{A}^T \) denote the set of all assemblies of tiles from \( T \), and let \( \mathcal{A}^T_{\infty} \) denote the set of finite assemblies of tiles from \( T \). A sequence of \( k \in \mathbb{Z}^+ \cup \{ \infty \} \) assemblies \( \vec{\alpha} = (\alpha_0, \alpha_1, \ldots) \) over \( \mathcal{A}^T \) is a \( T \)-assembly sequence if, for all \( 1 \leq i < k, \alpha_{i+1} \rightarrow^T \alpha_i \).

The result of an assembly sequence \( \vec{\alpha} \), denoted as \( \text{res}(\vec{\alpha}) \), is the unique limiting assembly (for a finite sequence, this is the final assembly in the sequence). We write \( \alpha \rightarrow^T \beta \), and we say \( \alpha \) \( T \)-produces \( \beta \) (in 0 or more steps), if there is a \( T \)-assembly sequence \( \alpha_0, \alpha_1, \ldots \) of length \( k \) such that (1) \( \alpha = \alpha_0 \), (2) \( \beta = \bigcup_{0 \leq i < k} \alpha_i \), and (3) for all \( 0 \leq i < k, \alpha_i \subseteq \beta \). We say \( \alpha \) is \( T \)-producible if \( \alpha \rightarrow^T \beta \), and we write \( \mathcal{A}[T] \) to denote the set of \( T \)-producible assemblies. An assembly \( \alpha \) is \( T \)-terminal if \( \alpha \) is \( \tau \)-stable and the set of empty locations at which a tile could stably attach to \( \alpha \) is empty. We write \( \mathcal{A}_{\square}[T] \subseteq \mathcal{A}[T] \) to denote the set of \( T \)-producible, \( T \)-terminal assemblies. If \( |\mathcal{A}_{\square}[T]| = 1 \) then \( T \) is said to be directed.

In general, a 3-dimensional shape is a set \( X \subseteq \mathbb{Z}^3 \), such that \( G^f_X \) is connected. For a finite shape \( X \subseteq \mathbb{Z}^3 \), define \( x^- = \min \{ x \mid (x, y, z) \in X \} \) and \( x^+ = \max \{ x \mid (x, y, z) \in X \} \). The quantities \( y^-, y^+, z^- \) and \( z^+ \) can be defined similarly. Then, we say that the horizontal extent of \( X \) is \( x^+ - x^- + 1 \). The vertical and stacked (vertical with respect to the \( z \)-axis) extents are defined similarly. We say \( X \) is just-barely 3D if it has stacked extent 2.

A tile-assembly system (or TAS) \( T \) is a triple \( (T, \sigma, \tau) \), where \( T \) is a tile set, \( \sigma \) is a seed tile, and \( \tau \) is a temperature. We say that a TAS \( T \) self-assembles \( X \) if, for all \( \alpha \in \mathcal{A}[T] \), dom \( \alpha \) = \( X \), i.e., if every terminal assembly produced by \( T \) places a tile on every point in \( X \) and does not place any tiles on points in \( \mathbb{Z}^3 \setminus X \). We define the tile complexity of a shape \( X \) at temperature \( \tau \), denoted by \( K^{T}_{SA}(X) \), as the minimum number of distinct tile types of any TAS that self-assembles it, i.e.,

\[
K^{T}_{SA}(X) = \min \{ n \mid T = (T, \sigma, \tau), |T| = n \text{ and } T \text{ self-assembles } X \}.
\]

We say that a tile-assembly system (or TAS) \( T = (T, \sigma, \tau) \), where \( T \) is a tile set, \( \sigma \) is a seed tile, and \( \tau \) is a temperature, self-assembles \( X \) if, for all \( \alpha \in \mathcal{A}[T] \), dom \( \alpha \) = \( X \), i.e., if every terminal assembly produced by \( T \) places a tile on every point in \( X \) and does not place any tiles on points in \( \mathbb{Z}^3 \setminus X \). We define the tile complexity of a shape \( X \) at temperature \( \tau \), denoted by \( K^{T}_{SA}(X) \), as the minimum number of distinct tile types of any TAS that self-assembles it, i.e.,

\[
K^{T}_{SA}(X) = \min \{ n \mid T = (T, \sigma, \tau), |T| = n \text{ and } T \text{ self-assembles } X \}.
\]

We say that a TAS \( T \) uniquely self-assembles a shape \( X \subseteq \mathbb{Z}^3 \) if \( \mathcal{A}[T] = \{ \alpha \} \) and dom \( \alpha \) = \( X \). The directed tile complexity of a shape \( X \) at temperature \( \tau \) is the minimum number of distinct tile types of any TAS that uniquely self-assembles (USA) \( X \), denoted by \( K^{T}_{USA}(X) = \min \{ n \mid T = (T, \sigma, \tau), |T| = n \text{ and } T \text{ uniquely self-assembles } X \} \).

### 3 Upper Bound

In this section, we give an overview of a construction that proves Theorem 1, our main upper bound on the directed tile complexity of a just-barely 3D rectangle. Recall that, for positive integers \( k \) and \( N \), \( R^3_{k,N} \) is said to be a just-barely 3D rectangle if it satisfies \( \{0, 1, \ldots, N - 1\} \times \{0, 1, \ldots, k - 1\} \times \{0\} \subseteq R^3_{k,N} \subseteq \{0, 1, \ldots, N - 1\} \times \{0, 1, \ldots, k - 1\} \times \{0\} \).
Theorem 1 says that \( K_{USA}^1 \left( R^3_{k,N} \right) = O \left( N^{\frac{1}{k-1}} + k \right) \). To prove Theorem 1, we construct a TAS that uniquely self-assembles a sufficiently large rectangle (of any height \( k \geq 3 \)) \( R^3_{k,N} \). Specifically, we construct a TAS \( T = (T, \sigma, 1) \) so that it simulates a base \( B = \left\lceil N^{\frac{1}{k-1}} \right\rceil, W = k - 2 \) digit counter, henceforth referred to as the counter, that starts counting at a specified starting value and stops just after the maximum value is incremented and rolls over to 0.

The remainder of this section is organized into subsections as follows. In Section 3.1, we describe the self-assembly of the counter, assuming a given configuration of tiles that represents the initial value of the counter. In Section 3.2, we describe the self-assembly of the initial value of the counter. Finally, in Section 3.3, we discuss the tile complexity and correctness of our construction.

### 3.1 Self-Assembly of the Counter

The self-assembly of the counter generally proceeds in a west to east fashion. However, within certain regions of space within the counter, paths of tiles do self-assemble from east to west. The reason for this back-and-forth self-assembly pattern is to carry out basic counter functionality such as “writing” the current value of the counter as a certain geometric pattern of tiles (west to east), and then going back (east to west) and “reading” the previously written value to increment it.

Each \( W \)-digit, base-\( B \) value of the counter is represented in a corresponding just-barely 3D rectangular region of space called a value region. There are three types of value regions, namely copy and increment regions that correspond to the two types of counter steps, and a special case initial value region. A copy region duplicates the value from the previous increment region and the latter increments the value from the previous copy region. The counter alternates between increment and copy steps. The initial value of the counter is in a special region that mimics the geometry of, but is technically not, a copy region.

Counter digits are represented geometrically and in binary, using the bit bump technique by Cook, Fu and Schweller [17]. Each digit is comprised of \( b + 2 = \left\lceil \log B \right\rceil + 2 \) bit bumps that protrude from a row of tiles. Each bit bump geometrically encodes one bit. The two most significant (westernmost) bits of a digit (drawn in light gray in Fig. 1) are its indicator bits: 10 – most significant, 01 – least significant, 00 – neither, 11 – both. The rest of the bits (the two eastmost white bits within each digit in Fig. 1) represent a base-\( B \) value. Bumps of a digit in increment (copy and initial value) regions protrude to the south (north). If the bump is in the \( z = 0 \) (\( z = 1 \)) plane, then it represents a 0 (1). The \( W \) digits in a value region are staggered like the steps of a staircase, descending (ascending) in a copy or initial value (increment) region. Figure 1 shows the layout of a copy and an increment value region, positioned consecutively as they would be in the counter, which is self-assembling to the east.

**Value region overview** (Fig. 1): Before proceeding, let us recall the following variables, their definitions and meanings, assuming positive height \( k > 3 \) and large width \( N \):
- \( B = \) the base of the counter = \( \left\lceil N^\frac{1}{k-1} \right\rceil \)
Fig. 1 A copy or initial value region (north) and an increment region (south). Each numeric quantity below or above each region indicates a number of columns, while the $d_i$, for $i = 1, \ldots, W$, indicate the digits of the counter’s value, whose west-to-east relative order alternates between regions. Note that $W = 3$ (so $k = 5$), $B = 3$, and $b = 2$ (not drawn to scale). The value $e$ is explained below.

- $W$ = the number of digits of the counter = $k - 2$
- $b$ = the number of bit bumps in a digit = $\lceil \log B \rceil$

We assume that the most significant bit of a digit is represented by the westernmost bit bump (located immediately to the east of the indicator bits), which means that in our example we have $d_1 = 2$ (least significant digit), $d_2 = 2$ and $d_W = d_3 = 1$ (most significant digit). Thus, the base-3 value 122 in the copy (or initial value) region is incremented to 200 in the increment region. When $B$ is greater than or equal to the combined width of two consecutive digit regions, or $2W(3(b + 2)) + 50$, then the column with diagonal lines represents a variable number of extra columns, which are inserted to ensure that the combined width of two consecutive value regions is $B$. In this case, we set the value of $e$ to be $B - (2W(3(b + 2)) + 50)$. The idea is that the terminal assembly of $T$ starts with an initial value region, followed by a sequence of copy-increment value region pairs with one pair per value of the counter. Since each value of the counter corresponds to a pair of consecutive value regions and $e$ is such that the width of two consecutive value regions is $B$ columns, assuming the counter starts counting at 0, the width of the $k$-row terminal assembly that $T$ will produce is $B \cdot B^W = \left\lceil N^{\frac{1}{k-1}} \right\rceil \cdot \left\lceil N^{\frac{1}{k-1}} \right\rceil^{k-2} \geq N^{\frac{1}{k-1}} \cdot N^{\frac{k-2}{k-1}} \geq N$. Then, $T$ can be modified to produce a unique terminal assembly of height $k$ and width $N$, by using a positive

Fig. 2 The initial value configuration, with no glues shown for the sake of clarity. Only the tiles that are needed and expected by the subsequent increment step are shown. The indicator bits are shown in light gray, the rest of the bits are in dark gray, and the blocker tiles are in black. For each $W$-digit, base-$B$ value, there exists a corresponding initial value configuration. Note that the depicted configuration also corresponds to a copy region of the counter, since the geometry of the initial value and copy regions are both what is expected and needed to be present at the start of the subsequent increment step.
starting value and $O(N \mod B) = O(B)$ additional tile types. The case where $B$ is less than the combined width of two consecutive digit regions is handled similarly, with $e = 0$, and choosing an appropriately (higher) starting value.

Figures 2 through 9 illustrate the self-assembly of an increment step for an artificial example with $B = 3$, $k = 5$ and starting value 122, from which a general construction can be easily derived. We describe the self-assembly of the counter in terms of the logical behavior of a series of gadgets. Gadgets are groups of tiles that self-assemble into a specified assembly via a series of deterministic tile attachments and in doing so propagate information (e.g., the value of a digit and/or the value of a bit indicating the presence of an arithmetic carry) from an input glue to one or more output glues. Throughout our discussion, we visually highlight the tiles that are attaching with non-white tiles (e.g., black and grayscale). Tiles that have already been placed are white. The input glue of the first non-white tile to attach is either explicitly described or can be inferred from where the first non-white tile attaches to a white tile. In the subsequent figures in this section, unless noted otherwise, we use big (small) squares to represent tiles placed in the $z = 0$ ($z = 1$) plane. A glue between a $z = 0$ tile and $z = 1$ tile is denoted as a small disk. Glues between $z = 1$ ($z = 0$) tiles are denoted as thin (thick) lines.

The initial value (Fig. 2): The initial value is represented inside a region that mimics the geometry of, but is technically not, a copy region. In the following discussion, we represent the initial value with a configuration of tiles, not an assembly, with no glues given for the sake of simplicity. In the initial value region (and, in general, in every copy region), the two black tiles in the $z = 1$ plane immediately east of the least significant bit of digit $d_i$ for $i > 1$ are blockers that will eventually block the self-assembly of a subsequent path of repeating tiles (for example, the path of dark gray tiles repeating to the west depicted in Fig. 7). The two easternmost black tiles in the $z = 0$ plane are also blockers and will block the self-assembly of a subsequent path of repeating tiles (for example, the path of black tiles repeating to the east depicted in Fig. 5). The tiles that traverse the extra columns represent paths of $e$ tiles. While, in general, we can hard-code a path of tiles that uniquely self-assembles a corresponding initial value region, where the glues of each tile type along the path encode the relative location of the tile in the path, we do not adopt this approach in our construction because such a path could contribute $\Omega(e + kb) = \Omega\left(e + k \log \left\lceil \frac{N}{e^{k+1}} \right\rceil \right) = \Omega(B + \log N)$ tile types to $T$. Later, in Section 3.2, we describe a “decoding” scheme for the initial value that will self-assemble into an initial value assembly during an initial value step and contribute only $O(B + k) = o(B + \log N)$ tile types to $T$. Given the initial value or copy region, the counter executes an increment step, where each digit in the initial value or copy region is read and incremented accordingly. In the following discussion, we describe how an increment step works, given an initial value configuration, but an increment step works the same way for the configuration of tiles resulting from the previous copy step. In our construction, we guarantee that the resulting geometry (positions and order of tile placements) of the initial value region and a copy region is what the subsequent increment step expects and needs in order to correctly self-assemble.

Conceptual depiction of the assembly sequence of an increment step (Fig. 3): The thick gray (thin black) lines represent paths of tiles in the $z = 0$ ($z = 1$) plane. The
lines with south-pointing notches are paths of repeating tiles and the south-pointing notches represent paths of tiles that attempt to self-assemble but are blocked from doing so by a previous portion of the assembly. The black dashed lines represent the portion of the assembly sequence that “weaves” through a previous portion of the assembly in which the value of a digit is represented geometrically and in binary and in doing so accumulates the digit’s value. The gray circles represent locations at which the assembly sequence transitions from placing tiles in one plane (either $z = 0$ or $z = 1$) to the other (either $z = 1$ or $z = 0$). The assembly sequence depicted here starts at the black square in the initial value region and proceeds to the east, indicated by the dashed line. After reading all the bits of the least significant digit, a repeating path of tiles self-assembles in the $z = 0$ plane, propagating the value of the least significant digit, until it gets blocked by a corresponding previous portion of the assembly. Eventually, an increment step concludes after the bits of $d_W$ are written in the increment region. The conclusion of the assembly sequence depicted here basically occurs at the most significant bit bump of the most significant digit in the increment region, indicated by the bit bump with the northwest-to-southeast diagonal lines. Note that, if there is a positive carry out after writing $d_W$ in the increment region (i.e., the value of the most significant digit changed from $B - 1$ to 0, and the current value of the counter is thus zero), then the counter stops counting.

**Read a digit (Fig. 4):** The black tiles are reading the four bits of $d_1$ in the initial value or copy region, starting with its most significant (westernmost) bit, which is an indicator bit. In this example, the south-facing glue of the westernmost black tile in the $z = 1$ plane is bound to what will eventually be the last tile to attach in the initial value or copy region. The westernmost black tile in the $z = 1$ plane initiates reading the value of digit $d_1$. In general, a similar westernmost black tile in the $z = 1$ plane initiates reading the value of digit $d_i$. Below this initial reader tile, another black initial reader tile attaches in the $z = 0$ plane. The initial reader tiles propagate a bit indicating the presence of an arithmetic carry, the carry bit. This bit is propagated from the south-facing glue of the $z = 1$ initial reader tile to the east-facing glues of both the $z = 1$ and $z = 0$ initial reader tiles. The value of the carry bit is initially set.

**Fig. 3** A conceptual depiction of the complete self-assembly of an increment region (bottom), given the initial value (top)

**Fig. 4** The black tiles are reading $d_1$ of the initial value or copy region, from most to least significant bit (west to east)
The path of repeating black tiles propagates the bits of digit $d_1$ and the carry bit to the next increment region. In general, a similar path of repeating tiles propagates the bits of digit $d_i$ and the carry bit to the next increment region.

In general, the east-facing glue on the $z = 1$ ($z = 0$) initial reader tile encodes 0 (1), but, due to the presence of the westernmost bit bump of $d_i$, only one is exposed. For each $1 \leq j < b + 2$ and $x \in \{0, 1\}^{j-1}$, there corresponds a reader gadget that reads a 0 in bit position $j$ (for notational convenience, $j = 1$ is the position of the most significant bit of a digit, i.e., the leftmost indicator bit). Such a reader gadget self-assembles a horizontal path of three tiles in the $z = 1$ plane and a tile below the easternmost $z = 1$ tile (in the $z = 0$ plane), such that, the west-facing input glue of its westernmost tile encodes $x0$, the east-facing output glue of its easternmost $z = 0$ tile encodes $x01$ and the east-facing output glue of its easternmost $z = 1$ tile encodes $x00$.

A reader gadget that reads a 1 is defined similarly. All reader gadgets also propagate the carry bit. A reader gadget that reads the bit in position $b + 2$, i.e., the least significant bit of the digit’s value, is similar to previous reader gadgets but has only one output glue, which is on a tile in the $z = 0$ plane and encodes the value of its input glue. It also initiates the self-assembly of a path of repeating tiles, along with the bits of the digit that was just read and the carry bit. The presence of all $b + 2$ bit bumps ensures a unique assembly sequence of $b + 2$ corresponding reader gadgets.

In general, for each $1 \leq i \leq b + 2$, we use $O \left( 2^i \right)$ reader gadgets that read a bit in position $i$. Thus, in general, all the reader gadgets, along with the initial reader tiles, contribute $O \left( 2^b \right) = O(B)$ tile types to $T$. Our reader gadgets are inspired by the “simulation macro tiles” depicted in Figures 3 and 6 of [17]. After the bits of a digit have been read from the initial value or copy region, they, along with the carry bit, are propagated to the next increment region, where the result is written.

**Propagate a digit an arbitrary distance (Fig. 5):** After the bits of any digit in the initial value or copy region are read, as previously described, a black tile type with equal west and east glues self-assembles in a path of repeating tiles to the east, propagating the bits of the digit that was just read along with the carry bit. A previous portion of the assembly blocks the self-assembly of this path (the two easternmost white tiles in the $z = 0$ plane in Fig. 5 block the path of repeating tiles that self-assemble after $d_1$ was read), at which point the south-facing glue of the easternmost black tile in the path is exposed. The path of repeating black tiles, all of the same type, propagates the bits of the digit that was just read, and the carry bit, thus, in general, contributing $O(B)$ tile types to $T$.

**Write the least significant digit (Fig. 6):** After the repeating tiles carrying the bits of $d_1$ (and the carry bit) are blocked by a previous portion of the assembly, the south-facing glue on the easternmost tile in this blocked path is exposed (top), to which the input glue of the first tile in a hard-coded path of dark gray tiles attaches. The value of $d_1$ is incremented from 2 and rolls over to 0, resulting in a carry out, which
Fig. 6 The value of $d_1$ that was read in Fig. 4 is incremented (dark gray tiles), and then written in the next increment region (black tiles). The new $d_1$ value is $0 = (2 + 1) \mod 3$ (with the indicator bits left unchanged)

is propagated to the output glue of the last tile to attach in the dark gray path of tiles (bottom). The hard-coded dark gray path of tiles shown here is used specifically for the least significant digit. Nevertheless, it propagates one of $O(B)$ possible values, along with a carry bit, a distance of length $O(1)$ tiles, thus, in general, contributing $O(B)$ tile types to $T$. The bits are written using a general set of fixed size writer gadgets and are depicted with black tiles (bottom). The writer gadgets are used for the self-assembly of every digit and work similarly to the reader gadgets, where we have a corresponding writer gadget for each bit position and each one propagates the carry bit. Thus, just like the reader gadgets, the writer gadgets, in general, contribute $O(B)$ tile types to $T$.

**After writing a non-most significant digit (Fig. 7):** After a non-most significant digit is written in an increment region, a blocker gadget self-assembles, the two easternmost $z = 0$ black tiles (top) of which will eventually block the self-assembly of a subsequent path of repeating tiles (similar to the black tiles in Fig. 5 but after reading $d_i$ for $i > 1$). The path of black tiles, starting with the black tile immediately east of the least significant bit in the $z = 0$ plane and ending at the westernmost $z = 1$ black tile is hard-coded, propagates the carry bit only (but no digit value) and self-assembles across a distance of $O(b)$ tiles, thus, in general, contributing $O(b) = O(B)$ tile types to $T$.

Then, we use a dark gray tile type with equal west and east glues to self-assemble a path of repeating tiles to the west, starting six tiles to the west of the most significant bit of the digit that was just written in the increment region (bottom). A previous portion of the assembly, namely the bump in the $z = 1$ plane that is east of the least significant bit of each digit $d_i$ for $i > 1$ in the initial value or copy region blocks the self-assembly of the path of repeating dark gray tiles, at which point the south-facing glue of the westernmost dark gray tile in the path is exposed, to which the input glue of the first tile in a hard-coded path of light gray tiles self-assembles, ending at the westernmost light gray $z = 0$ tile. This light gray $z = 0$ tile, and the $z = 1$ tile to

Fig. 7 Return from the increment region (top) to the copy (or the initial value) region (bottom) to read the next digit, $d_2$
which it attaches, start the reading of the next digit from the initial value or copy region and both propagate the carry bit to their output glues, to which the input glue of the correct reader gadget will ultimately attach.

The dark gray tile types propagate the carry bit, thus, in general, contributing \( O(1) \) tile types to \( T \). The hard-coded path of light gray tiles also propagates the carry bit only (but no digit value) and self-assembles across a distance of \( O(b) \) tiles, thus, in general, contributing \( O(b) = O(B) \) tile types to \( T \). Note that the tile types described here are used for the self-assembly of similar paths that self-assemble after writing every digit except \( d_w \) in an increment region. In general, the same tile types are used for the self-assembly of similar blockers, paths of repeating tiles and hard-coded paths that self-assemble after writing every digit except \( d_w \) in an increment region.

**Write a non-least significant digit (Fig. 8):** Here, the value of \( d_2 \) is incremented from 2 and rolls over to 0 (due to the presence of a positive carry bit). The bits are written using the same writer gadgets that we use to write the bits of any other digit. The main difference between the process of writing \( d_1 \) and \( d_i \) for \( i > 1 \) is that before the bits of the latter are written, the dark gray hard-coded gadget self-assembles, a portion of which is in the \( z = 1 \) plane, and will ultimately serve to block a path of repeating tiles in the \( z = 1 \) plane (repeating to the west from the next copy region). In this example, the light gray tiles are a repeating path of tiles that are propagating the value of \( d_2 \), along with the carry bit. Assuming the digit that was just written is not the most significant digit, the counter self-assembles back to the copy region to read another digit, similar to the process described in Fig. 7. In general, the dark gray hard-coded gadget propagates the result of incrementing a digit value, along with a carry bit a distance of \( O(1) \) tiles, thus contributing \( O(B) \) tile types to \( T \). Then, the bits of the resulting digit are written using the same general set of fixed-size writer gadgets that were previously discussed.

**After writing the most significant digit (Fig. 9):** However, if the digit that was just written is the most significant digit, then we use a hard-coded path of black tiles to initiate the reading of these same bits, starting with the black tile immediately east of the least significant bit in the \( z = 0 \) plane and ending at the westernmost \( z = 1 \) black tile. The reading of the bits of the most significant digit is the beginning of a copy step because, after the bits are read, they are propagated to and written in the next copy region. Note that if there is a positive carry out when incrementing the
most significant digit in an increment step, then the counter stops. The path of black tiles here self-assembles across a distance of $O(b)$ tiles, thus, in general, contributing $O(b) = O(B)$ tile types to $T$.

**Overview of the assembly sequence of a copy step** (Fig. 10): After an increment step concludes, a copy step is executed. A copy step is carried out in a fashion similar to an increment step, using a specific set of gadgets dedicated to the self-assembly of a copy region. In a copy step, the digits are read from the previous increment region (see the top half of Fig. 10 and written (essentially copied) to the next copy region in the order $d_W$ to $d_1$ (reverse of an increment step). We first give a brief list of high-level activities carried out by the counter during a copy step:

1. The value of each digit in the previous increment region is read, in a west-to-east fashion (this is depicted with dotted lines in Fig. 10). The first digit read is the most significant digit in the previous increment region, an example of which is the bit filled with down-and-to-the-right diagonal lines in the upper right portion of Fig. 10.
2. After each digit in the previous increment region is read, its value is propagated (west-to-east), in the $z = 0$ plane, to and written at a specific location in the current copy region (the propagation of the value just read is depicted by the thick grey lines in Fig. 10).
3. After the least significant bit of any digit is written in the copy region, a blocking structure immediately to the east of the digit that was just written self-assembles. This structure has an intricate geometry, places tiles in both $z$ planes and is used in the previous propagation step to effectively mark the location at which each digit, except the least significant digit, from the previous increment region is to be written in the copy region.
4. After the blocking structure self-assembles, a path of tiles self-assembles (east-to-west) back to the previous increment region to read another digit. This path of tiles is depicted by a solid black line in Fig. 10, and self-assembles in the $z = 1$ plane.
5. After the least significant digit is written in the copy region, the next increment step is initiated. For example, see the bit in the lower right portion of Fig. 10) filled with the up-and-to-the-right diagonal lines.
We now give a more detailed description of the self-assembly of a copy step, which, in general is carried out by the following gadgets:

- **Reader gadgets** read the value of each digit in the previous increment region. The reader gadgets used in a copy step work exactly the same as the reader gadgets that are used in an increment step. In general, we use $O(B)$ reader gadgets and each reader gadget is a fixed size. We use a fixed set of reader gadgets to read the value of every digit in a copy step (technically, the reader gadgets used in a copy step are reading the values of digits that are located in the previous increment region). The set of all such reader gadgets contributes $O(B)$ tile types to $T$.

- After the value of the most significant digit in the previous increment region is read, a hard-coded, fixed-length path of tiles self-assembles in the $z = 0$ plane to the location where the value of the most significant digit in the copy region will be written. In general, we use $O(B)$ such paths, one for each possible most significant digit value, and each path is a fixed length. The set of all such paths contributes $O(B)$ tile types to $T$.

- **Writer gadgets** write the value of each digit that was read, where the bump for each copy step writer gadget points north. Otherwise, the writer gadgets used in a copy step work similarly to those used in an increment step. In general, we use a fixed set of $O(B)$ writer gadgets and each writer gadget is a fixed size. The set of all such writer gadgets contributes $O(B)$ tile types to $T$.

- After the least significant bit of any digit is written, a corresponding blocker gadget self-assembles. We use two different types of blocker gadgets for this purpose. The blocker gadget that corresponds to a non-least significant digit is a special double blocker gadget in the sense that it will ultimately block the self-assembly of two repeating paths of tiles (one self-assembling from the west and the other from the east). The blocker gadget that corresponds to the least significant digit will block the self-assembly of one repeating path of tiles, self-assembling from the west after reading the value of the least significant digit in the next increment step. Both types of blocker gadgets are similar in the following way: after they block the self-assembly of a repeating path of tiles from continuing to self-assemble in one direction, they allow another path of tiles to start self-assembling in a different direction. In general, the double blocker gadget that self-assembles after a non-least significant digit is a fixed size and we use the same gadget after all non-least significant digits. This gadget contributes $O(1)$ tile types to $T$. Then, we use a fixed size blocker gadget after the least significant digit, which contributes $O(1)$ tiles types to $T$. The next two bullet points discuss the double blocker gadget after a non-least significant digit and the blocker gadget after the least significant digit, respectively.

- After the least significant bit of a non-least significant digit is written, a special double blocker gadget self-assembles. The geometry of the first portion of this gadget resembles that of a bit bump that represents the bit value 1. The purpose of this portion of the double blocker gadget is to block the self-assembly of the repeating path of tiles (from the east) used to read and possibly increment the value of the digit to which the double blocker corresponds. The geometry of the second portion of this gadget will ultimately block the self-assembly of the repeating path of tiles.
path of tiles (from the west) propagating the value of a non-least significant digit from the previous increment region to the current copy region. After a double blocker gadget for a non-least significant digit self-assembles, a hard-coded path of tiles self-assembles to the west in the \( z = 1 \) plane across a distance of \( O(b) \) tiles to a specific location where the self-assembly of a repeating path of tiles will be initiated. In general, we use a fixed hard-coded path of tiles for all non-least significant digits. This fixed, hard-coded path of tiles contributes \( O(b) \) tile types to \( T \). In general, we use a fixed double blocker gadget that self-assembles after all non-least significant digits. This fixed-size blocker gadget corresponding to a non-least significant digit in a copy region, along with the aforementioned path of tiles, in general, contribute \( O(b) \) tile types to \( T \).

- After the least significant bit of the least significant digit is written, a blocker gadget self-assembles. This blocker will ultimately block the self-assembly of the repeating path of tiles (from the west) propagating the value of the least significant digit from this copy region to the next increment region. First, a path of tiles self-assembles to the east in the \( z = 0 \) plane across a distance of \( O(e) \) tiles (across the extra columns of tiles). Then, the blocker itself self-assembles, which is a fixed size. Finally, after the blocker gadget for the least significant digit self-assembles, a hard-coded path of tiles self-assembles in the \( z = 1 \) plane across a distance of \( O(e + b) \) tiles. After this path self-assembles, a copy step effectively concludes and the next increment step proceeds, starting with the self-assembly of a series of increment step reader gadgets. Note that this hard-coded path of tiles that self-assembles after the blocker gadget for the least significant digit in the copy step is different from the hard-coded path of tiles that self-assembles after the double blocker gadget for a non-least significant digit. Nevertheless, this hard-coded path of tiles contributes \( O(e + b) = O(B) \) tile types to \( T \). Note that \( e = O(B) \), which means that this blocker gadget corresponding to the least significant digit in a copy region, along with the two aforementioned paths of tiles, in general, contribute \( O(B) \) tile types to \( T \).

- The westernmost glue of the previous path of tiles initiates the self-assembly of a repeating path of tiles in the \( z = 1 \) plane, to the west, that is comprised of a single tile type. Each tile along the path attempts to continue self-assembling to the west as well as initiating the self-assembly of another path of tiles to the north. However, the latter is blocked until westward self-assembly is blocked by the corresponding blocker gadget in the increment region. This blocker gadget was described in Fig. 8. More specifically, if \( d_i \), for \( i > 1 \) was just written in the copy region, then the blocker gadget in the increment region that hinders the westward self-assembly of the repeating path of tiles will correspond to \( d_i \). This blocker gadget will not only block the westward self-assembly of the repeating path of tiles but will also allow the path of tiles to self-assemble to the north. In general, we use one tile type for all non-least significant digits, which contributes one tile type to \( T \).

- In general, the previous path self-assembles three tiles to the north, then three tiles to the west, then three tiles to the south and finally follows a hard-coded path to the west that self-assembles across a fixed distance of \( O(b) \) tiles. This hard-coded path of tiles, which we use for all non-most significant digits, contributes \( O(b) \)
tile types to $T$. Note that this path eventually initiates the self-assembly of a series of copy step reader gadgets that read the most significant bit of the next non-most significant digit, as the digits in the previous increment step are being read from most significant to least significant.

- After the value of a non-most significant digit in the previous increment region is read, a repeating path of tiles self-assembles in the $z = 0$ plane, to the east, and is comprised of a single tile type. In doing so, the repeating path propagates the value of the digit that was just read. Each tile along the path attempts to continue self-assembling to the east as well as initiating the self-assembly of another path of tiles to the north. However, the latter is blocked until eastward self-assembly is blocked by the double blocker gadget in the copy region that self-assembled after the next-most significant digit. More specifically, if the repeating path of tiles is propagating $d_i$, for $i < W$, then it will be blocked by the double blocker gadget in the copy region that corresponds to $d_{i+1}$. This double blocker gadget will not only block the eastward self-assembly of the repeating path of tiles but will also allow the path of tiles to self-assemble to the north. In general, we use one such tile type for all non-most significant digits and we must have one tile type for each possible digit value. The set of all such tile types contributes $O(B)$ tile types to $T$.

- After the repeating path of tiles in which the value of a non-most significant digit is being propagated (from the increment region to the copy region) gets blocked by the corresponding blocker gadget (in the copy region), a fixed-length path of tiles self-assembles to the north and then to the specific location where it initiates the self-assembly of a series of copy step writer gadgets that write the value of the digit that was just read. In general, we use one such fixed-length path for each possible digit value. The set of all such fixed-length paths contributes $O(B)$ tile types to $T$.

Based on the above itemized list, the additional tile types used for a copy step contribute $O(B)$ tile types to $T$. Moreover, all the tile types used for an increment step are explicitly defined to be disjoint from those for a copy step, which at most doubles the size of $|T|$, having no effect on the asymptotic size of $T$. To complete the proof of our upper bound, we show, in the next subsection, that the initial value self-assembles using $O(B + k)$ additional tile types.

### 3.2 Self-Assembly of the Initial Value

We self-assemble the initial value of the counter in an initial value step, which is comprised of two phases and is carried out within a westernmost initial value region of our thin rectangle. To avoid confusion, in the remainder of this section, we will refer to the counter from the previous subsection as the main counter.

**Initial value region overview (Fig. 11):** The first phase in the initial value step is the self-assembly of the initial value counter (abbreviated IVC from here on), which occupies the westernmost portion of the initial value region. The IVC counts from 1 to $k - 2$ (i.e., $W$), where each value of the IVC is represented geometrically and in binary using modified bit bumps. The IVC is a non-zigzag (no copy row) modified binary counter. We use a counter rather than hard code a path of tiles to self-assemble the
geometric representations of the values 1 to $W$ because such a path could contribute $\Omega(k \log k)$ tile types to $T$ and our goal is to self-assemble the initial value for the main counter using only $O(B + k)$ tile types. The values of the IVC will be staggered because we will need to be able to read and then re-read each value later on to self-assemble the digits of the initial value of the main counter. Note that each IVC value takes up 3 tiles vertically (except for the last value, which only takes up 2 tiles). This works out nicely to exactly fill up $k$ rows because:

- The IVC counts through a total of $k - 2$ values.
- Each subsequent value of the IVC only takes up one more row than the previous one (because of the staggering of the values and the fact that the last value is only two tiles vertically). This gives us a total of $(k - 2) + 1 = k - 1$ rows.
- There is an extra row (at the bottom), which we need in order to read the bits of the first value of the IVC twice (once to increment this value inside the IVC and once to generate $d_1$ in the initial value of the main counter). The reading of the other values of the IVC also take up one row each but these rows are shared with the tiles for the previous value of the IVC (thanks to the staggering of the values).

Then, in the second phase of the initial value step, each value of the IVC is re-read and the corresponding digit for the initial value of the main counter is generated.

Finally, after the initial value of the main counter self-assembly, the first increment step of the main counter is executed, followed by subsequent copy and increment steps of the main counter (as discussed previously).

In what follows, we will discuss the general self-assembly of the initial value step. We will frame our discussion within the context of the concrete example that was specified in the previous subsection (i.e., for parameters $W = 3$, $k = 5$, $B = 3$ and $b = 2$). We continue using the same conventions for figures in this subsection that we established in the previous subsection.
Conceptual depiction of the assembly sequence of the IVC (Fig. 12): In the first phase of the initial value step, the IVC self-assembles. The assembly sequence depicted in Fig. 12 starts at the location of the single seed tile, indicated by the westernmost gray square and proceeds to write the bits of the first value. After the bits of each value are written, the value that was just written is read and incremented, and the resulting value is written. This pattern is repeated until the last value is written, at which point the IVC effectively stops and the generation of the digits of the initial value for the main counter from each value of the IVC begins.

In Figs. 13 through 18, we give a more general and detailed account of the self-assembly of the IVC, while still framed within the context of our concrete example.

Write a non-last value (Fig. 13): Every value of the IVC, except the last value, is written via the self-assembly of a series of “double” bit bump IVC writer gadgets, each of a fixed-size, and pointing to the north (as opposed to the writer gadgets shown in Figs. 6 and 8, used in the self-assembly of an increment region, which point to the south and have vertical extent two). We use taller bit bumps with vertical extent three to represent each value (except the last one) in the IVC, to allow each IVC value to be read twice. In general, $O(W) = O(k)$ generic IVC writer gadgets suffice to self-assemble all values of the IVC, excluding its last value, thus contributing $O(k)$ tile types to $T$. Indeed, for each bit position $1 \leq i \leq \lceil \log W \rceil$ of a value of the IVC other than the last value, we use $2^{\lceil \log W \rceil - i + 1}$ IVC writer gadgets, because an IVC writer gadget for bit position $i$ effectively takes as input a bit string of length $\lceil \log W \rceil - i + 1$, outputs a bit string of length $\lceil \log W \rceil - i$ and self-assembles the geometric, double bit bump representation for the bit at position $i$.

After writing a value other than the last value (Fig. 14): After each value of the IVC self-assembles, the IVC does not explicitly “remember” the value that just
self-assembled. In other words, this value is not encoded in the east-facing output glue of the final (easternmost) initial value writer gadget to self-assemble, which means the IVC cannot know the next (incremented) value to self-assemble. For this reason, we use a hard-coded path of tiles that self-assembles from the easternmost initial value writer gadget back to the most significant bit of the (non-last) value that just self-assembled, ready to read this value for the purpose of incrementing it. The same such path of hard-coded tiles self-assembles after writing each value, other than the last value, and self-assembles across a distance of $O(k)$ tiles. Thus, in general, this hard-coded path contributes $O(k)$ tile types to $T$.

Read a value (Fig. 15): Reading a value of the IVC is carried out in a manner similar to reading a value of a digit in the main counter, for example during an increment step (see Fig. 4 and the corresponding discussion thereof). In our construction, reading a value of the IVC is accomplished via the self-assembly of a series of IVC reader gadgets that are complementary to their writer counterparts. Based on a reasoning similar to that used in the discussion of the writer gadgets in Fig. 13, $O(k)$ generic IVC reader gadgets suffice to read an arbitrary value of the IVC, thus contributing $O(k)$ tile types to $T$.

After reading a value other than the last value (Fig. 16): After any value of the IVC (but the last one) is read, it will be incremented and subsequently written using a series of IVC writer gadgets (as previously described). Here, we depict the situation where the value that was just read is not the last value of the IVC, but is in fact the first value. In this case, we use a hard-coded path of tiles that effectively takes as input (via its westernmost glue) the value that was just read, say $x$, and effectively outputs (via its easternmost glue) the binary representation of the value $x + 1$. This hard-coded path of tiles self-assembles across a fixed length and we must have a different hard-coded path for each possible value of the IVC. Thus, in general, the set of all such hard-coded paths contributes $O(k)$ tile types to $T$. Note that, if $x + 1$ is equal to $W$.
The value of the IVC that was previously written, assuming it is not the last value, was just read and the result of incrementing it is propagated via the hard-coded path of black tiles depicted here. In this example, the easternmost glue of the path of black tiles would encode the value 2 (i.e., the maximum possible value of the IVC), then the last IVC value is about to be written, after which the IVC will stop, as explained in the next figure.

**Write the last value (Fig. 17):** The bits of the last value of the IVC are written via the self-assembly of a series of special IVC writer gadgets. These writer gadgets are special because 1) their vertical extent is only two (in contrast to the previous writer gadgets shown in Fig. 13) and 2) the easternmost (output) glue of the IVC writer gadget that self-assembles the least significant bit of the last value (i.e., the glue on the east side of the easternmost black tile in the figure) carries an extra bit of information indicating that the last value of the IVC just self-assembled and that the IVC must now stop. Just like the previous IVC writer gadgets, the special IVC writer gadgets that write the last value of the IVC contribute $O(k)$ tile types to $T$.

**After writing the last value (Fig. 18):** After the bits of the last value of the IVC have been written, we use a fixed, hard-coded path of tiles to self-assemble back to the most significant bit of the last value of the IVC. This path effectively stops the IVC, thus ending the first phase of the initial value step, and initiates the second phase in the initial value step, where each digit in the initial value of the main counter will be generated from a corresponding value of the IVC. This path, used only after writing the last value of the IVC, self-assembles across a distance of length $O(k)$ tiles, and thus, in general, contributes $O(k)$ tile types to $T$.

We now turn our attention to the second phase of the initial value step, in which each digit of the initial value for the main counter is generated from a corresponding value in the IVC.

**Conceptual depiction of the assembly sequence of the initial value (Fig. 19):** In the second phase of the initial value step, the fully assembled IVC is used to assemble the initial value of the main counter in the easternmost portion of the initial value region, east of the IVC. The self-assembly of the initial value uses dedicated gadgets that re-read each value of the IVC and then generate the bit bumps of the corresponding digit in the initial value of the main counter. The digits in the initial value are generated...
Fig. 18 After the bits of the last value have been written, the hard-coded path of black tiles depicted here self-assembles back to the most significant bit of the last value that just self-assembled, ready to read this value. However, this last value is not read for the purpose of incrementing it, but rather to begin the second phase of the initial value step.

In conclusion, the initial value step self-assembles using \( O(B + k) \) tile types.

### 3.3 Statement of Main Upper Bound

It is straightforward to convert the preceding discussion into a formal construction that takes as input sufficiently large \( k, N \in \mathbb{Z}^+ \) and outputs a 3D TAS \( T = (T, \sigma, 1) \) in which \( R_{k,N}^3 \) uniquely self-assembles, with \( \left| T \right| = O(B + k) = O \left( N^{1 \frac{1}{k+1}} + k \right) \). Then, one can show that \( T \) is directed using the method of Conditional Determinism by Lutz and Shutters [26]. Thus, we have our main upper bound:

**Theorem 1** \( K_{USA}^1 \left( R_{k,N}^3 \right) = O \left( N^{1 \frac{1}{k+1}} + k \right) \).
4 Lower Bound

In this section, we prove Theorem 2, our main lower bound, from which \( K_{\text{SA}}^1 (R_{E,N}^3) = \Omega \left( N^\frac{1}{12} \right) \) follows as a corollary. To that end, in Section 4.1, we develop a new type of Window Movie Lemma that applies to directed, temperature-1 self-assembly, followed by its proof in Section 4.2. Then, in Section 4.3, we prove our lower bound. Finally, in Section 4.4, we show that our new Window Movie Lemma does need the directedness assumption but that, when the latter is dropped, it still gives us a better bound than the original Window Movie Lemma.

4.1 New Window Movie Lemma for Directed, Temperature-1 Self-Assembly

In this section, we will develop a new type of Window Movie Lemma that applies to directed, temperature-1 self-assembly. It will end up being the cornerstone of our lower bound machinery from which we derive Theorem 2. We first give some notation. In this section, we will develop a new type of Window Movie Lemma that applies to directed, temperature-1 self-assembly. It will end up being the cornerstone of our lower bound machinery from which we derive Theorem 2. We first give some notation.

Given a window \( w \) and an assembly \( \alpha \), a window that intersects \( \alpha \) is a partitioning of \( \alpha \) into two configurations (i.e., after being split into two parts, each part may or may not be disconnected). In this case we say that the window \( w \) cuts the assembly \( \alpha \) into two non-overlapping configurations \( \alpha_L \) and \( \alpha_R \), satisfying, for all \( \bar{x} \in \text{dom} \ \alpha_L \), \( \alpha(\bar{x}) = \alpha_L(\bar{x}) \), for all \( \bar{x} \in \text{dom} \ \alpha_R \), \( \alpha(\bar{x}) = \alpha_R(\bar{x}) \), and \( \alpha(\bar{x}) \) is undefined at any point \( \bar{x} \in Z^3 \) \( \backslash \) (\( \text{dom} \ \alpha_L \cup \text{dom} \ \alpha_R \)). Given a window \( w \), its translation by a vector \( \Delta \), written \( w + \Delta \) is simply the translation of each one of \( w \)'s elements (edges) by \( \Delta \). All windows in this paper are assumed to be induced by some translation of the \( yz \)-plane. Each window is thus uniquely identified by its \( x \) coordinate. For a window \( w \) and an assembly sequence \( \tilde{\alpha} \), we define a glue window movie \( M \) to be the order of placement, position and glue type for each glue that appears along the window \( w \) in \( \tilde{\alpha} \), regardless of whether the glue (eventually) forms a bond. Given an assembly sequence \( \tilde{\alpha} \) and a window \( w \), the associated glue window movie is the maximal sequence \( M_{\tilde{\alpha},w} = (\tilde{v}_1, g_1), (\tilde{v}_2, g_2), \ldots \) of pairs of grid graph vertices \( \tilde{v}_i \) and glues \( g_i \), given by the order of appearance of the glues along window \( w \) in the assembly sequence \( \tilde{\alpha} \). We write \( M_{\tilde{\alpha},w} + \Delta \) to denote the translation by \( \Delta \) of \( M_{\tilde{\alpha},w} \), yielding \( (\tilde{v}_1 + \Delta, g_1), (\tilde{v}_2 + \Delta, g_2), \ldots \).

If \( \tilde{\alpha} \) follows \( s \), then the notation \( M_{\tilde{\alpha},w} \upharpoonright s \) denotes the restricted glue window submovie (restricted to \( s \)), which consists of only those steps of \( M_{\tilde{\alpha},w} \) that place glues that form positive-strength bonds that cross \( w \) at locations belonging to the simple path \( s \). Let \( \tilde{v} \) denote the location of the starting point of \( s \) (i.e., the location of \( \sigma \)). Let \( \tilde{v}_i \) and \( \tilde{v}_{i+1} \) denote two consecutive locations in \( M_{\tilde{\alpha},w} \upharpoonright s \) that are located across \( w \) from each other. We say that these two locations define a crossing of \( w \), where a crossing has exactly one direction. We say that this crossing is away from \( \tilde{v} \) (or away from \( \tilde{v} \) itself, depending on the sign).
Fig. 20 An assembly, a simple path, and two types of glue window movies in 2D. Here, we have $M_{\vec{\alpha},w} = ((\vec{v}_1, g_1), (\vec{v}_2, g_2), (\vec{v}_3, g_3), (\vec{v}_4, g_4), (\vec{v}_5, g_5), (\vec{v}_6, g_6), (\vec{v}_7, g_7), (\vec{v}_8, g_8), (\vec{v}_9, g_9), (\vec{v}_{10}, g_{10}), (\vec{v}_{11}, g_{11}), (\vec{v}_{12}, g_{12}), (\vec{v}_{13}, g_{13})$, where $g_1 = g_2, g_3 = g_4, g_6 = g_7, g_8 = g_9, g_{11} = g_{12}$ and $g_{13} = g_{10}$. Note that $M_{\vec{\alpha},w} | s$ only includes the location-glue pairs where the glues actually form bonds between locations in $s$. For example, $\vec{v}_{10}$ and $\vec{v}_{13}$ are excluded from $M_{\vec{\alpha},w} | s$ because the glues that connect them are not part of the path of glues that follow $s$.

We say that two restricted glue window submovies are “sufficiently similar” if they have the same (odd) number of crossings, the same set of crossing locations (up to horizontal translation), the same crossing directions at corresponding crossing locations, and the same glues in corresponding “away crossing” locations.

**Definition 1** Assume: $T = (T, \sigma, 1)$ is a 3D TAS, $\alpha \in A[T]$, $s$ is a simple path in $G^b_{\alpha}$ starting from the location of $\sigma$, $\vec{\alpha}$ is a sequence of $T$-producible assemblies that follows $s$, $w$ and $w'$ are windows, $\sigma$ is not located between $w$ and $w'$, $\vec{\Delta} \neq \vec{0}$ is a vector satisfying $w' = w + \vec{\Delta}$, $e$ and $e'$ are two odd numbers, and $M = M_{\vec{\alpha},w} | s = (\vec{v}_1, g_1), \ldots, (\vec{v}_{2e}, g_{2e})$ and $M' = M_{\vec{\alpha},w'} | s = (\vec{v'}_1, g'_1), \ldots, (\vec{v'}_{2e'}, g'_{2e'})$ are both non-empty restricted glue window submovies.

We say that $M$ and $M'$ are **sufficiently similar** if the following constraints are satisfied:

1. same number of crossings: $e = e'$,
2. same set of crossing locations (up to translation): \( \{ \vec{v}_i + \Delta \mid 1 \leq i \leq 2e \} = \{ \vec{v}_j \mid 1 \leq j \leq 2e \} \).

3. same crossing directions at corresponding crossing locations:
   \( \{ \vec{v}_{4i-2} + \Delta \mid 1 \leq i \leq \frac{e+1}{2} \} = \{ \vec{v}_{4j-2} \mid 1 \leq j \leq \frac{e+1}{2} \} \), and

4. same glues in corresponding “away crossing” locations:
   for all \( 1 \leq i, j \leq \frac{e+1}{2} \), if \( \vec{v}_{4j-2} = \vec{v}_{4i-2} + \Delta \), then \( g_{4j-2} = g_{4i-3} \).

See Fig. 21 for a 2D example of two restricted glue window submovies that are sufficiently similar.

The following result is a new and specialized type of Window Movie Lemma, called the “Sufficiently Similar Window Movie Lemma”, restricted to directed, temperature-1 self-assembly and is the cornerstone of our lower bound machinery. It basically says that if, for some directed TAS \( T \), two distinct restricted glue window submovies are sufficiently similar, then \( T \) does not self-assemble a specified just-barely 3D shape.

First, we define the furthest extreme column of a shape \( X \subseteq \mathbb{Z}^3 \) from a location \( (x, y, z) \in X \) to be \( \{ (\vec{x}^-) \times \mathbb{Z} \times \mathbb{Z} \} \cap X \), if \( |x^- - x| \geq |x^+ - x| \) and \( \{(x^+) \times \mathbb{Z} \times \mathbb{Z} \} \cap X \) otherwise.

**Lemma 2** Assume: \( T \) is a directed, temperature-1, 3D TAS, \( k \in \mathbb{Z}^+ \), \( X_k \subseteq \mathbb{Z}^3 \) is any finite just-barely 3D shape with vertical extent \( k \) and finite horizontal extent, \( s \subseteq X_k \) is a simple path in the full grid graph of \( X_k \) from the location of the seed of \( T \) to some location in the furthest extreme column of \( X_k \) such that there exists a \( T \)-assembly sequence \( \vec{a} \) that follows \( s \), \( w \) and \( w' \) are windows, such that, \( \Delta \neq \vec{0} \) is a vector satisfying \( w' = w + \Delta \), and \( e \) is an odd number satisfying \( 1 \leq e < 2k \). If \( M = M_{\vec{a},w} \mid s = (\vec{v}_1, g_1), \ldots, (\vec{v}_{2e}, g_{2e}) \) and \( M' = M_{\vec{a},w'} \mid s = (\vec{v}'_1, g'_1), \ldots, (\vec{v}'_{2e}, g'_{2e}) \) are sufficiently similar non-empty restricted glue window submovies, then \( T \) does not self-assemble \( X_k \).

See Fig. 21 for a 2D example of the hypothesis of Lemma 2.

We now give some notation that will be useful for proving Lemma 2. The definitions and notation in the following paragraph are inspired by notation that first appeared in [6].

For a \( T \)-assembly sequence \( \vec{a} = (\alpha_i \mid 0 \leq i < l) \), we write \( |\vec{a}| = l \). We write \( \vec{a}[i] \) to denote \( \vec{x} \mapsto t \), where \( \vec{x} \) and \( t \) are such that \( \alpha_{i+1} = \alpha_i + (\vec{x} \mapsto t) \). We write \( \vec{a}[i] + \Delta \), for some vector \( \Delta \), to denote \( (\vec{x} + \Delta) \mapsto t \). If \( \alpha_{i+1} = \alpha_i + (\vec{x} \mapsto t) \), then we write \( \text{Pos}(\vec{a}[i]) = \vec{x} \) and \( \text{Tile}(\vec{a}[i]) = t \). Assuming \( |\vec{a}| > 0 \), the notation \( \vec{a} = \vec{a} + (\vec{x} \mapsto t) \) denotes a tile placement step, namely the sequence of configurations \( (\alpha_i \mid 0 \leq i < l + 1) \), where \( \alpha_i \) is the configuration satisfying, \( \alpha_i(\vec{x}) = t \) and for all \( \vec{y} \neq \vec{x} \), \( \alpha_i(\vec{y}) = \alpha_{i-1}(\vec{y}) \). Note that the “+” in a tile placement step is different from the “+” used in the notation “\( \beta = \alpha + (\vec{p} \mapsto t) \)”. However, since the former operates on an assembly sequence, it should be clear from the context which operator is being invoked. The definition of a tile placement step does not require that the sequence of configurations be a \( T \)-assembly sequence. After all, the tile placement step \( \vec{a} = \vec{a} + (\vec{x} \mapsto t) \) could be attempting to place a tile at a location that is not even adjacent to (a location in the domain of) \( \alpha_{l-1} \). Or, it could be attempting to place a tile.
Fig. 21 A 2D example of the hypothesis of Lemma 2 for $k = 10$ and $e = 9$. Since the example is 2D, we use $R_{k,N} = \{0, 1, \ldots, N - 1\} \times \{0, 1, \ldots, k - 1\}$, rather than $R_3^{k,N}$, even though $X_k$ is assumed to be a just-barely 3D shape. Note that $\vec{\alpha}$ follows a simple path $s$ from the location of $\sigma$ to a location in the furthest extreme column. The restricted glue window movies are sufficiently similar because their glues are at the same locations (up to horizontal translation), oriented in the same direction (away or toward $\sigma$), and each pair of glues that are placed by $\vec{\alpha}$ at an “away crossing” of one of the windows is equal to its translated counterpart in the other window, e.g., the two topmost glues that touch $w$ and $w'$ are both light gray. The same constraint holds for all glue pairs shown with a solid shade of gray or a striped pattern. On the other hand, the glues adjacent to $w'$ that are placed by $\vec{\alpha}$ at a “toward crossing”, for example $g'_{11}$ and $g'_{12}$, are decorated with a letter in order to represent the fact that we do not assume that these glues are equal to their translated counterparts that touch $w$ (i.e., $g_{15}$ and $g_{16}$).

at a location that is in the domain of $\alpha_{l-1}$, which means a tile has already been placed at $\vec{x}$. So we say that such a tile placement step is correct if $(\alpha_i | 0 \leq i < l + 1)$ is a $T$-assembly sequence. If $|\vec{\alpha}| = 0$, then $\vec{\alpha} = \vec{\alpha} + (\vec{x} \mapsto t)$ results in the $T$-assembly sequence $(\alpha_0)$, where $\alpha_0$ is the assembly such that $\alpha_0(\vec{x}) = t$ and $\alpha_0(\vec{y})$ is undefined at all other locations $\vec{y} \neq \vec{x}$.

The proof of Lemma 2 relies on Algorithm 1 (shown on the next page) that uses $\vec{\alpha}$ to construct a new assembly sequence $\vec{\beta}$ such that the tile placement steps by $\vec{\beta}$ on the
Algorithm 1 The algorithm for $\beta$.

```
Initialize $j = 1, n = 0$ and $\beta = ()$
while $\text{Pos}(\tilde{a}[n]) \neq \tilde{v}_{4j-2}$ do
  $\beta = \beta + \tilde{a}[n]$
  $n = n + 1$
end while
while $\tilde{v}_{4j-2} \neq \tilde{v}_{2e} + \tilde{\Delta}$ do
  Let $i$ be such that $4i - 2$ is the index of $\tilde{v}_{4j-2} - \tilde{\Delta}$ in $M$
  Let $n$ be such that $\text{Pos}(\tilde{a}[n]) = \tilde{v}_{4i-2}$
  while $\text{Pos}(\tilde{a}[n]) \neq \tilde{v}_{4i}$ do
    $\beta = \beta + (\tilde{a}[n] + \tilde{\Delta})$
    $n = n + 1$
  end while
  Let $j'$ be such that $4j'$ is the index of $\tilde{v}_{4i} + \tilde{\Delta}$ in $M'$
  Let $n$ be such that $\text{Pos}(\tilde{a}[n]) = \tilde{v}_{4j'}$
  while $\text{Pos}(\tilde{a}[n]) \neq \tilde{v}_{j'+2}$ do
    $\beta = \beta + \tilde{a}[n]$
    $n = n + 1$
  end while
  $j = j' + 1$
end while
Let $n$ be such that $\text{Pos}(\tilde{a}[n]) = \tilde{v}_{2e}$
while $\text{Pos}(\tilde{a}[n]) \neq \tilde{v}_{j'+2}$ do
  $n = n + 1$
end while
return $\beta$
```

far side of $w'$ from the seed mimic a (possibly strict) subset of the tile placements by $\tilde{\alpha}$ on the far side of $w$ from the seed.

When $\beta$ is on the near side of $w'$ to the seed, it mimics $\tilde{\alpha}$, although $\beta$ does not necessarily mimic every tile placement by $\tilde{\alpha}$ on the near side of $w'$ to the seed. When $\beta$ crosses $w'$, going away from the seed, by placing tiles at $\tilde{v}_{4j-3}$ and $\tilde{v}_{4j-2}$ in this order, then the tile it places at $\tilde{v}_{4j-2}$ is of the same type as the tile that $\tilde{\alpha}$ places at $\tilde{v}_{4i-2} = \tilde{v}_{4j-2} - \tilde{\Delta}$. After $\beta$ crosses $w'$ by placing a tile at $\tilde{v}_{4j-2}$, $\beta$ places tiles that $\tilde{\alpha}$ places along $s$ from $\tilde{v}_{4i-2}$ to $\tilde{v}_{4i-1}$, but the tiles $\tilde{\beta}$ places are translated to the far side of $w'$ from the seed. When $\beta$ is about to cross $w'$ going toward the seed, by placing a tile at $\tilde{v}_{4j-1}$, then, since $\mathcal{T}$ is directed, the type of tile that it places at this location is equal to the type of tile that $\tilde{\alpha}$ places at $\tilde{v}_{4j-1}$. This means that $\tilde{\beta}$ may continue to follow $s$ but starting from $\tilde{v}_{4j}$. Eventually, $\tilde{\beta}$ will finish crossing $w'$ going away from the seed for the last time by placing a tile at $\tilde{v}_{2\varepsilon} + \tilde{\Delta}$. Then, $\tilde{\beta}$ places tiles that $\tilde{\alpha}$ places along $s$, starting from $\tilde{v}_{2\varepsilon}$, but the tiles that $\tilde{\beta}$ places are translated to the far side of $w'$ from the seed. Since $\Delta \neq 0$, $\tilde{\beta}$ will ultimately place a tile that is not in $X_k$, which means $\mathcal{T}$ does not self-assemble $X_k$.

See Fig. 22 for an example of the result of the assembly sequence $\tilde{\beta}$ returned by the algorithm.
We illustrate the behavior of this algorithm in Fig. 23, where we apply it to the assembly sequence $\vec{\alpha}$ shown in Fig. 21 and ultimately end up at the example shown in Fig. 22.

We must show that all of the tile placement steps executed by the algorithm for $\vec{\beta}$ are correct. In addition, we must prove that the tile placement steps executed by the algorithm for $\vec{\beta}$ place tiles along a simple path. Let $\vec{\alpha} = \vec{\alpha} + (\vec{x} \mapsto t')$ and $\vec{\alpha} = \vec{\alpha} + (\vec{x} \mapsto t)$ be consecutive tile placement steps executed by the algorithm for $\vec{\beta}$ and assume that the former is either the first tile placement step executed or it is correct. To show that the latter is correct, we will show that:

1. the tile configuration that consists of $t$ placed at $\vec{x}$ and $t'$ placed at $\vec{x}'$ is a 1-stable assembly (not necessarily $T$-producible) whose domain consists of two locations, and
2. the location $\vec{x}$ is unoccupied before $\vec{\alpha} = \vec{\alpha} + (\vec{x} \mapsto t)$ is executed.

The previous two conditions constitute a slightly stronger notion of “correctness” for a tile placement step, which we will call *adjacently correct*. After all, the two previous conditions imply that $\vec{\alpha} = \vec{\alpha} + (\vec{x} \mapsto t)$ is correct, but if $\vec{\alpha} = \vec{\alpha} + (\vec{x} \mapsto t)$ is correct, then condition 2 must hold but condition 1 need not because $\vec{x}$ does not have
Fig. 23 The trace of the algorithm shown in Fig. 1 when applied to the assembly sequence $\vec{\alpha}$ shown in Fig. 21. In each sub-figure, the new sub-path is bolded and is a continuation of the sub-path in the previous one. The last sub-figure above shows the same assembly sequence $\vec{\beta}$ depicted in Fig. 22.
to be adjacent to $\vec{x}'$. It suffices to prove that every tile placement step executed by the algorithm for $\vec{\beta}$ is adjacently correct. As a result, $\vec{\beta}$, like $\vec{\alpha}$, will place tiles along a simple path.

### 4.2 Proof of the New Window Movie Lemma

We now give the full proof of Lemma 2.

**Proof (of Lemma 2)** Sub-proof #1:

Since $\vec{\alpha}$ is a $T$-assembly sequence that follows a simple path, the tile placement steps in Loop 1 are adjacently correct and only place tiles that are on the near side of $w$ to the seed. Note that Loop 1 terminates with $\text{Pos}(\vec{\alpha}[n]) = \vec{v}'_2$. By the definition of $M'$, $\vec{v}'_2$ is the first location at which $\vec{\alpha}$ places a tile on the far side of $w$ from the seed. Upon entering Loop 2, let $m$ range over the integers $\{1, \ldots, e+1\}$. The variable $m$ will count the iterations of Loop 2. For $1 \leq l \leq m$, define $j_l$ to be the value of $j$ prior to iteration $l$ of Loop 2. Likewise, for $1 \leq l \leq m$, define $j'_l$ to be the value of $j'$ after Line 11 executes during iteration $l$ of Loop 2. We say that $j_m$ is the value of $j$ in the algorithm for $\vec{\beta}$ prior to the current iteration of Loop 2. When it is clear from the context, we will simply use “$j$” in place of “$j_m$”. Likewise, we will simply use “$j'$” in place of “$j'_m$”.

If $j$ is such that $\vec{v}'_{4j-2} = \vec{v}_{2e} + \Delta$, then Loop 2 terminates.

So, let $j$ be such that $\vec{v}'_{4j-2} \neq \vec{v}_{2e} + \Delta$ and assume the following Loop 2 invariant holds.

Prior to each iteration $m$ of Loop 2:

1. all previous tile placement steps executed by the algorithm for $\vec{\beta}$ are adjacently correct,
2. all tiles placed by $\vec{\beta}$ on locations on the far side of $w$ from the seed are placed by tile placement steps executed by Loop 2a,
3. all tiles placed by $\vec{\beta}$ on locations on the near side of $w$ to the seed are placed by tile placement steps executed by Loop 1 or Loop 2b,
4. if $m > 1$, then for all $1 \leq l < m$, $j_l \neq j_m$,
5. the location at which $\vec{\beta}$ last placed a tile (say $t$) is $\vec{v}'_{4j_m-3}$, and
6. the glue of $t$ that touches $w$ is $g'_{4j_m-3}$.

Sub-proof #2 - Loop 2 invariant initialization

Just before the first iteration of Loop 2, $m = 1$ and all prior tile placements have been completed within Loop 1. Parts 1 and 3 of the invariant follow directly from Sub-proof #1. Part 2 of the invariant is true since no tiles have been placed yet on the far side of $w$. Part 4 of the invariant holds since $m = 1$. Part 5 of the invariant holds because $j_m = j_1 = 1$, $\vec{v}'_{4j_m-3} = \vec{v}'_1$, and the location at which $\vec{\beta}$ last placed a tile $t$ is $\vec{v}'_1$, that is, the location that precedes $\vec{v}'_2$ in $\vec{\alpha}$. Finally, part 6 of the invariant holds because $t$ is the tile that $\vec{\alpha}$ placed at $\vec{v}'_1$ and, by the definition of $M'$, the glue of $t$ that touches $w$ is $g'_{4j_m-3} = g'_1$.
Sub-proof #2 - Loop 2 invariant maintenance

On Line 5, if \( j \) is such that \( \vec{v}_{4i-2}' = \vec{v}_{4i-2} + \vec{\Delta} \), then Loop 2 terminates. So, let \( j \) be such that \( \vec{v}_{4i-2}' \neq \vec{v}_{4i-2} + \vec{\Delta} \) and assume that the Loop 2 invariant holds. We will first prove (by induction) that parts 1 through 4 of the invariant still hold when Loop 2a terminates.

Within the current iteration of Loop 2, Line 7 sets \( n \) to a value such that \( Pos(\vec{\alpha}[n]) = \vec{v}_{4i-2} \), where \( i \) is such that \( 4i - 2 \) is the index of \( \vec{v}_{4i-2}' - \vec{\Delta} \) in \( M \).

For the base step of the induction, consider the tile placement step executed in the first iteration of Loop 2a. To establish the first part of the invariant, we now prove that this tile placement step is adjacently correct. First, we prove that it places a tile that is executed.

Having shown that \( t \) binds to \( \vec{v}_{4i-2}' \), we now prove that \( \vec{\beta} \) has not already placed a tile at \( Pos(\vec{\alpha}[n]) + \vec{\Delta} = \vec{v}_{4i-2}' \) before the tile placement step in the first iteration of Loop 2a is executed.

According to part 2 of the invariant, all locations on the far side of \( w' \) from the seed at which tiles are placed by \( \vec{\beta} \) are filled by tile placement steps executed by Loop 2a. Since \( \vec{v}_{4i-2}' \) is on the far side of \( w' \) from the seed, we only need to consider tile placement steps that place tiles at locations that are on the far side of \( w' \) from the seed.
Since we are assuming that $\vec{\beta} = \vec{\beta} + (\vec{\alpha} [n] + \vec{\Delta})$ is the tile placement step executed in the first iteration of Loop 2a, we know that any already completed tile placement step $\vec{\beta} = \vec{\beta} + (\vec{\alpha} [n'] + \vec{\Delta})$, for $0 \leq n' < |\vec{\alpha}|$, is executed in some iteration of Loop 2a but in a past iteration of Loop 2.

Define $\text{index}_{\vec{\alpha}}(\vec{x})$ to be the value of $n$ such that $\text{Pos} (\vec{\alpha}[n]) = \vec{x}$. Define the rule $f(j) = i$ such that $4i - 2$ is the index of $\vec{v}_{4j-2} - \vec{\Delta}$ in $M$. Note that $f$ is a valid function because, by part 3 of sufficiently similar, we have $\{\vec{v}_{4i-2} + \vec{\Delta} \mid 1 \leq i \leq \frac{e+1}{2}\} = \{\vec{v}_{4i-2} \mid 1 \leq j \leq \frac{e+1}{2}\}$. Moreover, $f$ is injective, because, intuitively, two different locations in $M'$ cannot translate with the same $\vec{\Delta}$ to the same location in $M$. Formally, assume that $f(a) = f(b)$ and let $c$ be such that $4c - 2$ is the index of $\vec{v}_{4a-2} - \vec{\Delta}$ in $M$ and let $d$ be such that $4d - 2$ is the index of $\vec{v}_{4b-2} - \vec{\Delta}$ in $M$. Since we are assuming $f(a) = f(b)$, then we have $c = d$. This means that $4c - 2 = 4d - 2$ is the index of $\vec{v}_{4a-2} - \vec{\Delta}$ in $M$. Likewise, $4c - 2 = 4d - 2$ is the index of $\vec{v}_{4b-2} - \vec{\Delta}$ in $M$. Then we have $\vec{v}_{4a-2} = \vec{v}_{4b-2} + \vec{\Delta}$, and $\vec{v}_{4a-2} + \vec{\Delta} = \vec{v}_{4b-2}$. In other words, we have $\vec{v}_{4a-2} = \vec{v}_{4b-2}$, which implies that $a = b$ and it follows that $f$ is injective.

For all $1 \leq l \leq m$, define $i_l$ to be the value of $i$ computed in Line 6. In other words, $i_l = f(j_l)$ and $i_m$ is the value of $i$ computed in Line 6 during the current iteration of Loop 2. Observe that $n$ (on Line 7) satisfies

$$\text{index}_{\vec{\alpha}}(\vec{v}_{4i_m-2}) \leq n < \text{index}_{\vec{\alpha}}(\vec{v}_{4i_m})$$

because $\vec{\beta} = \vec{\beta} + (\vec{\alpha} [n] + \vec{\Delta})$ is the tile placement step executed in the first iteration of Loop 2a, and, for some $1 \leq l < m, n'$ satisfies

$$\text{index}_{\vec{\alpha}}(\vec{v}_{4i_l-2}) \leq n' < \text{index}_{\vec{\alpha}}(\vec{v}_{4i_l})$$

because $\vec{\beta} = \vec{\beta} + (\vec{\alpha} [n'] + \vec{\Delta})$ is some tile placement step executed in Loop 2a but in a past iteration of Loop 2. In fact, in the first iteration of Loop 2a, $n = \text{index}_{\vec{\alpha}}(\vec{v}_{4i_m-2}) < \text{index}_{\vec{\alpha}}(\vec{v}_{4i_m})$. By part 4 of the invariant, for all $1 \leq l < m, i_m \neq j_l$. Since $f$ is injective, it follows that, for all $1 \leq l < m, i_m = f(j_m) \neq f(j_l) = i_l$. Then we have three cases to consider. Case 1, where $\vec{v}_{4i_m} = \vec{v}_{4i_l-2}$, is impossible, since these two locations are on opposite sides of $w$. In case 2, where $\text{index}_{\vec{\alpha}}(\vec{v}_{4i_m}) < \text{index}_{\vec{\alpha}}(\vec{v}_{4i_l-2})$, we have:

$$\text{index}_{\vec{\alpha}}(\vec{v}_{4i_m-2}) \leq n < \text{index}_{\vec{\alpha}}(\vec{v}_{4i_m}) < \text{index}_{\vec{\alpha}}(\vec{v}_{4i_l-2}) \leq n' < \text{index}_{\vec{\alpha}}(\vec{v}_{4i_l}).$$

Finally, in case 3, where $\text{index}_{\vec{\alpha}}(\vec{v}_{4i_m}) > \text{index}_{\vec{\alpha}}(\vec{v}_{4i_l-2})$, we have:

$$\text{index}_{\vec{\alpha}}(\vec{v}_{4i_m-2}) \leq n' < \text{index}_{\vec{\alpha}}(\vec{v}_{4i_l}) < \text{index}_{\vec{\alpha}}(\vec{v}_{4i_m-2}) \leq n < \text{index}_{\vec{\alpha}}(\vec{v}_{4i_m}).$$

In all possible cases, $n \neq n'$. Thus, since $\vec{\alpha}$ follows a simple path, $\text{Pos} (\vec{\alpha}[n]) \neq \text{Pos} (\vec{\alpha}[n'])$, which implies $\text{Pos} (\vec{\alpha}[n]) + \vec{\Delta} \neq \text{Pos} (\vec{\alpha}[n']) + \vec{\Delta}$. Therefore, $\text{Pos} (\vec{\alpha}[n]) + \vec{\Delta}$ is empty prior to the execution of $\vec{\beta} = \vec{\beta} + (\vec{\alpha} [n] + \vec{\Delta})$, i.e., no previous tile placement step placed a tile at that location before the first iteration of Loop 2a. This means that the tile placement step $\vec{\beta} = \vec{\beta} + (\vec{\alpha} [n] + \vec{\Delta})$ is adjacent
correct. This concludes the proof of correctness for the first iteration of Loop 2a (base step).

We now show (inductive step) that the rest of the tile placement steps executed in Loop 2a within the current iteration of Loop 2 are adjacently correct. Let \( \vec{\beta} = \vec{\bar{\beta}} + (\vec{\alpha}[n] + \vec{\Delta}) \) be a tile placement step executed in some (but not the first) iteration of Loop 2a and assume that all tile placement steps executed in past iterations of Loop 2a are adjacently correct and place tiles at locations that are on the far side of \( w' \) from the seed (inductive hypothesis). In particular, assume that the tile placement step \( \vec{\beta} = \vec{\bar{\beta}} + (\vec{\alpha}[n-1] + \vec{\Delta}) \) executed in Loop 2a, for the current iteration of Loop 2, is adjacently correct and places a tile at a location that is on the far side of \( w' \) from the seed. Since \( \vec{\alpha} \) follows a simple path, \( \text{Pos}(\vec{\alpha}[n]) \) is adjacent to \( \text{Pos}(\vec{\alpha}[n-1]) \) and the configuration consisting of a tile of type \( \text{Tile}(\vec{\alpha}[n]) \) placed at \( \text{Pos}(\vec{\alpha}[n]) \) and a tile of type \( \text{Tile}(\vec{\alpha}[n-1]) \) placed at \( \text{Pos}(\vec{\alpha}[n-1]) \) is stable. This means that \( \text{Pos}(\vec{\alpha}[n]) + \vec{\Delta} \) is adjacent to \( \text{Pos}(\vec{\alpha}[n-1]) + \vec{\Delta} \) and the configuration consisting of a tile of type \( \text{Tile}(\vec{\alpha}[n]) \) placed at \( \text{Pos}(\vec{\alpha}[n]) + \vec{\Delta} \) and a tile of type \( \text{Tile}(\vec{\alpha}[n-1]) \) placed at \( \text{Pos}(\vec{\alpha}[n-1]) + \vec{\Delta} \) is stable, thus proving part 1 of adjacently correct.

Now, when proving part 2, two cases arise. The first case is where \( \vec{\beta} = \vec{\bar{\beta}} + (\vec{\alpha}[n'] + \vec{\Delta}) \) is executed in a past iteration of Loop 2a in the current iteration of Loop 2. Here, we have \( n \neq n' \) because, within Loop 2a, we are merely translating a segment of \( \vec{\alpha} \), which follows a simple path.

The second case is where \( \vec{\beta} = \vec{\bar{\beta}} + (\vec{\alpha}[n'] + \vec{\Delta}) \) is executed in a past iteration of Loop 2. Here, using reasoning that is similar to the one we used to establish the correctness of the first iteration of Loop 2a based on inequalities (1) and (2) above, we have \( n \neq n' \). In both cases, \( n \neq n' \) implies \( \text{Pos}(\vec{\alpha}[n]) \neq \text{Pos}(\vec{\alpha}[n']) \), which means that the location of the current tile placement step in Loop 2a is different from the location of any previous tile placement step that was executed in Loop 2a. It follows that \( \text{Pos}(\vec{\alpha}[n]) + \vec{\Delta} \neq \text{Pos}(\vec{\alpha}[n']) + \vec{\Delta} \). This proves part 2, and therefore, the tile placement step \( \vec{\beta} = \vec{\bar{\beta}} + (\vec{\alpha}[n] + \vec{\Delta}) \) is adjacently correct. This concludes our proof that part 1 of the invariant holds at the end of Loop 2a.

Since Loop 2a mimics the portion of \( \vec{\alpha} \) between (and including) the points \( \vec{v}_4i_m - 2 \) and \( \vec{v}_{4i_m - 1} \), which, by definition of \( M \), is on the far side of \( w \) from the seed, it follows that \( \text{Pos}(\vec{\alpha}[n]) \) is on the far side of \( w \) from the seed during every iteration of Loop 2a. This means that \( \text{Pos}(\vec{\alpha}[n]) + \vec{\Delta} \) is on the far side of \( w' \) from the seed during every iteration of Loop 2a and thus part 2 of the invariant holds at the end of Loop 2a. For the same reason, part 3 of the invariant also holds at that point. Finally, part 4 of the invariant trivially holds since Loop 2a does not update \( j \). This concludes our proof that the first four parts of the invariant hold when Loop 2a terminates. We will now prove that these four parts still hold when Loop 2b terminates.

Loop 2b “picks up” where Loop 2a “left off”. Note that Loop 2a terminates with \( \text{Pos}(\vec{\alpha}[n]) = \vec{v}_{4i_m} \), with the last tile being placed at \( \vec{v}_{4i_m - 1} + \vec{\Delta} \). Define the rule \( g(i) = j \) such that \( 4j \) is the index of \( \vec{v}_{4i} + \vec{\Delta} \) in \( M' \). Note that \( g \), like \( f \), is a valid function because, by parts 2 & 3 of sufficiently similar, we have \( \{ \vec{v}_{4i} + \vec{\Delta} \mid 1 \leq i \leq \frac{\varepsilon - 1}{2} \} = \).
\[ \{ \vec{v}_{4j}^\prime \mid 1 \leq j \leq \frac{e-1}{2} \} \]. Similarly, \( g \), like \( f \), is injective. Line 11 sets the value of \( j_m^\prime \) to be such that \( 4j_m^\prime \) is the index of \( \vec{v}_{4im} \) in \( M' \). In other words, Line 11 computes \( j_m^\prime = g (i_m) \) and Line 12 sets the value of \( k \) such that \( Pos (\vec{a} [n]) = \vec{v}_{4j_m^\prime} \). Intuitively, \( \vec{v}_{4im} + \Delta = \vec{v}_{4j_m^\prime} \) is the location at which \( \vec{\beta} \) finishes crossing from the far side of \( w' \) from the seed back to the near side. Recall that Loop 2a “left off” by placing a tile (in its last iteration) at the location \( \vec{v}_{4im-1} + \Delta = \vec{v}_{4j_m^\prime-1} \).

Now, for the base step of the induction we use to prove that part 1 of the invariant holds after Loop 2b, consider the tile placement step executed in the first iteration of Loop 2b. Formally, we have:

\[
\vec{\beta} = \vec{\beta} + \vec{\alpha} [n]
\]

\[
= \vec{\beta} + (Pos (\vec{\alpha} [n]) \mapsto Tile (\vec{\alpha} [n]))
\]

\[
= \vec{\beta} + (\vec{v}_{4j_m^\prime} \mapsto Tile (\vec{\alpha} [n]))
\]

Thus, the tile placement step executed in the first iteration of Loop 2b will place a tile at \( \vec{v}_{4j_m^\prime} \), which, by the definition of \( M' \), is adjacent to but on the opposite side of \( w' \) from \( \vec{v}_{4j_m^\prime-1} \). Since \( \mathcal{T} \) is directed, the type of tile that \( \vec{\beta} \) places at \( \vec{v}_{4j_m^\prime-1} \) during the final iteration of Loop 2a must be the same as the type of the tile that \( \vec{\alpha} \) places at \( \vec{v}_{4j_m^\prime-1} \). This is the only place in the proof where we use the fact that \( \mathcal{T} \) is directed. By the definition of the tile placement step executed in the first iteration of Loop 2b, the type of tile that \( \vec{\beta} \) places at \( \vec{v}_{4j_m^\prime} \) is the same as the type of tile that \( \vec{\alpha} \) places at \( \vec{v}_{4j_m^\prime} \).

This means that the glue of the tile that \( \vec{\beta} \) places at \( \vec{v}_{4j_m^\prime} \) and that touches \( w' \) is equal to the glue of the tile that \( \vec{\beta} \) places at \( \vec{v}_{4j_m^\prime-1} \) and that touches \( w' \). This proves part 1 of adjacently correct for \( \vec{\beta} = \vec{\beta} + \vec{\alpha} [n] \). So, in order to show that \( \vec{\beta} = \vec{\beta} + \vec{\alpha} [n] \) is adjacently correct, it suffices to show that \( \vec{\beta} \) has not already placed a tile at \( Pos (\vec{\alpha} [n]) \), i.e., part 2 of adjacently correct.

By part 3 of the invariant, all tiles placed by \( \vec{\beta} \) on the near side of \( w' \) to the seed result from tile placement steps belonging to either Loop 1 or Loop 2b. Since \( \vec{v}_{4j_m^\prime} \) is on the near side of \( w' \) to the seed, we only need to consider tile placement steps in the algorithm for \( \vec{\beta} \) that place tiles at locations that are on the near side of \( w' \) to the seed. Since we are assuming that \( \vec{\beta} = \vec{\beta} + \vec{\alpha} [n] \) is the tile placement step executed in the first iteration of Loop 2b, we must consider two cases for any already completed tile placement step \( \vec{\beta} = \vec{\beta} + \vec{\alpha} [n'] \) with \( 0 \leq n' < |\vec{\alpha}| \). In the case where \( \vec{\beta} = \vec{\beta} + \vec{\alpha} [n'] \) is executed in some iteration of Loop 1 (before the first iteration of Loop 2), we have \( n' < index_\vec{\alpha} (\vec{v}_2) \) and \( index_\vec{\alpha} (\vec{v}_4) \leq n \). In this case, \( index_\vec{\alpha} (\vec{v}_2) < index_\vec{\alpha} (\vec{v}_4) \) implies \( n' \neq n \). In the second case, namely when \( \vec{\beta} = \vec{\beta} + \vec{\alpha} [n'] \) is executed in some iteration of Loop 2b but in a past iteration of Loop 2, \( n \) satisfies

\[
index_\vec{\alpha} (\vec{v}_{4j_m^\prime}) \leq n < index_\vec{\alpha} (\vec{v}_{4j_m^\prime+2})
\]
because \( \vec{\beta} = \vec{\beta} + \vec{a} [n] \) is the tile placement step executed in the first iteration of Loop 2b and, for some \( 1 \leq l < m, n' \) satisfies

\[
\text{index}_\vec{a} \left( \vec{v}_{4j}^l \right) \leq n' < \text{index}_\vec{a} \left( \vec{v}_{4j}^{l+2} \right). \tag{4}
\]

In order to show that \( n \neq n' \), since \( \vec{a} \) follows a simple path, it suffices to show that, for all \( 1 \leq l < m, j'_m \neq j_i \). By part 4 of the invariant, for all \( 1 \leq l < m, j_m \neq j_i \). By definition, for all \( 1 \leq l \leq m, i_l = f (j_i) \). Since \( f \) is injective, we have, for all \( 1 \leq l < m, i_m \neq i_l \). Since \( g \) is injective, we have, for all \( 1 \leq l < m, g (i_m) \neq g (i_l) \).

By definition, for all \( 1 \leq l \leq m, j'_m = g (i_m) \neq g (i_l) = j'_l \). So, in all cases, we have \( n \neq n' \), which implies that \( Pos (\vec{a} [n]) \neq Pos (\vec{a} [n']) \). This means that part 2 is satisfied and therefore the tile placement step \( \vec{\beta} = \vec{\beta} + \vec{a} [n] \) is adjacently correct. This concludes the proof of correctness for the first iteration of Loop 2b (base step).

We now show (inductive step) that the rest of the tile placement steps executed in Loop 2b and within the current iteration of Loop 2 are adjacently correct. So, let \( \vec{\beta} = \vec{\beta} + \vec{a} [n] \) be a tile placement step executed in some (but not the first) iteration of Loop 2b and assume all tile placement steps executed in past iterations of Loop 2b are adjacent correct and place tiles at locations that are on the near side of \( w' \) to the seed. In particular, the tile placement step \( \vec{\beta} = \vec{\beta} + \vec{a} [n - 1] \) executed in Loop 2b, for the current iteration of Loop 2, is adjacently correct and places a tile at a location that is on the near side of \( w' \) to the seed. Since \( \vec{a} \) follows a simple path, \( Pos (\vec{a} [n]) \) is adjacent to \( Pos (\vec{a} [n - 1]) \) and the configuration consisting of a tile of type \( \text{Tile} (\vec{a} [n]) \) placed at \( Pos (\vec{a} [n]) \) and a tile of type \( \text{Tile} (\vec{a} [n - 1]) \) placed at \( Pos (\vec{a} [n - 1]) \) is stable, thus proving part 1 of adjacently correct. Here, using reasoning that is similar to the one we used to establish the correctness of the first iteration of Loop 2b based on inequalities (3) and (4) above, we have \( n \neq n' \). This means that \( Pos (\vec{a} [n]) \neq Pos (\vec{a} [n']) \), thereby satisfying part 2. It follows that the tile placement step \( \vec{\beta} = \vec{\beta} + \vec{a} [n] \) is adjacently correct. This concludes our proof that part 1 of the invariant holds when Loop 2b terminates.

Since \( \vec{a} \) follows a simple path and the portion of \( \vec{a} \) between (and including) \( \vec{v}_{4j}^l \) and \( \vec{v}_{4j}^{l+1} \) is, by definition of \( M' \), on the near side of \( w' \) to the seed, \( Pos (\vec{a} [n]) \) is on the near side of \( w' \) to the seed during every iteration of Loop 2b. This means that all tile placement steps executed by Loop 2b only place tiles at locations on the near side of \( w' \) to the seed. This concludes our proof that parts 2 and 3 of the invariant hold when Loop 2b terminates.

We will now show that, for all \( 1 \leq l \leq m, j_{m+1} \neq j_i \). We already showed above that, for all \( 1 \leq l < m, j'_m \neq j'_l \). Then we have, for all \( 1 \leq l < m, j'_m + 1 \neq j'_l + 1 \). Since Line 16 computes the value of \( j \) for the next iteration of Loop 2 to be the value of \( j' + 1 \), we infer, for all \( 1 \leq l < m, j_{m+1} \neq j_{l+1} \), or, equivalently, for all \( 2 \leq l \leq m, j_{m+1} \neq j_i \).

Since \( j_1 = 1 \) and \( j' \) (computed on Line 11) cannot be equal to 0, we have \( j_{m+1} \neq j_1 \). It follows that, for all \( 1 \leq l \leq m, j_{m+1} \neq j_i \). This concludes our proof that part 4 of the invariant holds when Loop 2b terminates.
In this section, we prove our main lower bound, namely Theorem 2, based on a combinatorial analysis of the number of sufficiently similar distinct restricted glue window submovies. First, we give the following result, which basically says that we must examine only a “small” number of distinct restricted glue window submovies in order to find two different ones that are sufficiently similar.

**Lemma 3** Assume: \( \mathcal{T} = (T, \sigma, 1) \) is a 3D TAS, \( G \) is the set of all glues in \( T \), \( k \in \mathbb{Z}^+ \), \( X_k \) is a just-barely 3D shape with vertical extent \( k \), \( s \) is a simple path starting from the location of \( \sigma \) such that \( s \subseteq X_k \), \( \tilde{a} \) is a \( \mathcal{T} \)-assembly sequence that follows \( s \), \( m \in \mathbb{Z}^+ \), for all \( 1 \leq l \leq m \), \( w_l \) is a window, for all \( 1 \leq l < l' \leq m \), \( \Delta_{1,1'} \neq \emptyset \)
satisfies \( w_\ell = w_l + \tilde{\Delta}_{l,\ell} \), and for all \( 1 \leq l \leq m \), there is an odd \( 1 \leq e_l < 2k \) such that \( M_{\bar{a},w_l} \upharpoonright s \) is a non-empty restricted glue window submovie of length \( 2e_l \).

If \( m > G^k \cdot k \cdot 16^k \), then there exist \( 1 \leq l < l' \leq m \) such that \( e_l = e_{l'} = e \) and \( M_{\bar{a},w_l} \upharpoonright s = (\bar{v}_1, g_1), \ldots, (\bar{v}_{2e}, g_{2e}) \) and \( M_{\bar{a},w_{l'}} \upharpoonright s = (\bar{v}_1', g_1'), \ldots, (\bar{v}_{2e}', g_{2e}') \) are sufficiently similar non-empty restricted glue window submovies.

Intuitively, to prove Lemma 3, we first count the number of ways to choose the set of locations at which \( M \) crosses \( w = w_l \) at 2e “crossing locations”. Then, we count the number of ways to choose the set of locations at which \( M \) crosses \( w \) going away from the seed at \( e+\frac{1}{2} \) “away crossing locations”. Finally, we count the number of ways to assign glues to the selected away crossing locations. After summing over all odd values of \( e \), we get the indicated lower bound on \( m \) that notably neither contains a “factorial” term nor a coefficient on the “k” in the exponent of “G”.

**Proof** Let \( e \) be a fixed odd number such that \( 1 \leq e < 2k \). Let \( w \) be any window such that \( M = M_{\bar{a},w} \upharpoonright s = (\bar{v}_1, g_1), \ldots, (\bar{v}_{2e}, g_{2e}) \) is a non-empty restricted glue window submovie.

We will assume that \( e \) represents the number of times that \( \bar{a} \) crosses \( w \) (going either away from or toward the seed) as it follows \( s \). Here, \( e \) can be at most \( 2k - 1 \) because \( w \) is a translation of the \( yz \)-plane and \( s \subseteq X_k \).

1. First, we count the number of ways to choose the set \( \{\bar{v}_1, \ldots, \bar{v}_{2e}\} \), or the set of locations of \( M \). Clearly, there are \( \binom{4k}{2e} \) ways to choose a subset of \( 2e \) locations from a set comprised of \( 4k \) locations. However, for \( M = M_{\bar{a},w} \), since \( \bar{a} \) follows a simple path, it suffices to count the number of ways to choose the set \( \{\bar{v}_{2j-1} \mid i = 1, \ldots, e\} \). This is because, once we choose a location of \( M \), the location that is adjacent to the chosen location but on the opposite side of \( w \) is determined. There are \( \binom{2k}{e} \) ways to choose a subset of \( e \) elements from a set comprised of \( 2k \) elements.

2. Next, we count the number of ways to choose the set \( \{\bar{v}_{4i-2} \mid 1 \leq i \leq e+\frac{1}{2}\} \). Intuitively, this is the set of locations at which \( \bar{a} \) finishes crossing \( w \) going away from the seed. Observe that each chosen location \( \bar{v} \) of \( M \) is either on the far side of \( w \) from the seed or on the near side of \( w \) to the seed. Furthermore, \( \bar{v} \) is paired up with a different chosen location \( \bar{v}' \) of \( M \) in the sense that \( \bar{v} \) is adjacent to but on the opposite side of \( w \) from \( \bar{v}' \). Thus, choose \( e+\frac{1}{2} \) locations from the set comprised of the \( e \) chosen locations that are on the far side of \( w \) from the seed. For each \( 1 \leq i \leq e+\frac{1}{2} \), there is a unique element in this set that will be assigned to \( \bar{v}_{4i-2} \). There are \( \binom{e}{(e+1)/2} \) ways to choose \( e+\frac{1}{2} \) elements from a set comprised of \( e \) elements.

3. Having chosen the set of locations at which \( \bar{a} \) crosses \( w \) going away from the seed, observe that for each such location \( \bar{v}_{4i-2} \), where \( 1 \leq i \leq e+\frac{1}{2} \), there is a pair \((\bar{v}_{4i-2}, g_{4i-2})\) in \( M \), where \( g_{4i-2} \in G \). Since there are \( e+\frac{1}{2} \) pairs in this sequence, and the glue within each pair can be assigned in one of \( G \) possible ways, there are \( G^{e+\frac{1}{2}} \) ways to assign glues to each pair in the sequence.

By the above counting procedure, for all odd \( e \) in \( \{1, \ldots, 2k - 1\} \), if \( M_{\bar{a},w} \upharpoonright s = (\bar{v}_1, g_1), \ldots, (\bar{v}_{2e}, g_{2e}) \), then the number of ways to choose the sets \( \{\bar{v}_1, \ldots, \bar{v}_{2e}\} \) and \( \{\bar{v}_{4i-2} \mid 1 \leq i \leq e+\frac{1}{2}\} \) and assign the glue within each pair \((\bar{v}_{4i-2}, g_{4i-2})\) for
I \leq i \leq \frac{e + 1}{2}, \text{ is less than or equal to } \sum_{1 \leq e < 2k \atop e \text{ odd}} \left( \begin{pmatrix} 2k \\ e \end{pmatrix} \left( \frac{e}{e + 1} \right) G^{\frac{e + 1}{2}} \right). \text{ Then, we have}

\sum_{1 \leq e < 2k \atop e \text{ odd}} \left( \begin{pmatrix} 2k \\ e \end{pmatrix} \left( \frac{e}{e + 1} \right) G^{\frac{e + 1}{2}} \right) \leq \sum_{1 \leq e < 2k \atop e \text{ odd}} \left( \begin{pmatrix} 2k \\ k \end{pmatrix} \left( \frac{2k - 1}{2} \right) G^{\frac{2k - 1}{2}} \right)

\leq G^k \sum_{1 \leq e < 2k \atop e \text{ odd}} \left( 2^{2k} \right)^2

= G^k \sum_{1 \leq e < 2k \atop e \text{ odd}} 2^{4k}

= G^k \cdot k \cdot 16^k.

Thus, if \( m > G^k \cdot k \cdot 16^k \), then there are two numbers \( 1 \leq l < l' \leq m \), such that, for \( e = e_l = e_{l'} \), \( M = M_{\tilde{g}, w_l} \mid s = (\tilde{v}_1, g_l), \ldots, (\tilde{v}_{2e}, g_{2e}) \) and \( M' = M_{\tilde{g}', w_{l'}} \mid s = (\tilde{v}_1, g'_l), \ldots, (\tilde{v}_{2e}, g'_{2e}) \) are non-empty restricted glue window submovies satisfying the following conditions:

1. \( \left\{ \tilde{v}_i + \tilde{\Delta}_{l, l'} \mid 1 \leq i \leq 2e \right\} = \left\{ \tilde{v}_j' \mid 1 \leq j \leq 2e \right\}, \) and
2. \( \left\{ \tilde{v}_{4i-2} + \tilde{\Delta}_{l, l'} \mid 1 \leq i \leq \frac{e + 1}{2} \right\} = \left\{ \tilde{v}_{4j-2} \mid 1 \leq j \leq \frac{e + 1}{2} \right\}, \) and
3. for all \( 1 \leq i, j \leq \frac{e + 1}{2} \), if \( \tilde{v}_{4j-2} = \tilde{v}_{4i-2} + \tilde{\Delta}_{l, l'} \), then \( g_{4j-2} = g_{4i-2}. \)

Note that, since \( M \) and \( M' \) are both restricted to \( s \), we have, for all \( 1 \leq i \leq \frac{e + 1}{2} \), \( g_{4i-2} = g_{4i-3}. \) This means that, for all \( 1 \leq i, j \leq \frac{e + 1}{2} \), if \( \tilde{v}_{4j-2} = \tilde{v}_{4i-2} + \tilde{\Delta}_{l, l'} \), then \( g_{4j-2} = g_{4i-3} \), and it follows that \( M \) and \( M' \) are sufficiently similar. \( \square \)

Lemma 3 overcounts the number of “sufficiently dissimilar” restricted glue window submovies but will be enough to prove our main lower bound.

The following result chains Lemmas 3 and 2 in this order. We will use the contrapositive of the following lemma to prove our main lower bound.

**Lemma 4** Assume: \( T = (T, \sigma, 1) \) is a 3D TAS, \( G \) is the set of all glues in \( T \), \( k \in \mathbb{Z}^+ \), \( X_k \) is a just-barely 3D shape with vertical extent \( k \) and finite horizontal extent, \( s \subseteq X_k \) is a simple path in the full grid graph of \( X_k \) from the location of \( \sigma \) to some location in the furthest extreme column of \( X_k \) in either \( z \) plane such that there exists a \( T \)-assembly sequence \( \tilde{\alpha} \) that follows \( s, m \in \mathbb{Z}^+ \), for all \( 1 \leq l \leq m \), \( w_l \) is a window, for all \( 1 \leq l < l' \leq m \), \( \tilde{\Delta}_{l, l'} \neq 0 \) satisfies \( w_{l'} = w_l + \tilde{\Delta}_{l, l'} \), and for all \( 1 \leq l \leq m \), there is an odd \( 1 \leq e_l < 2k \) such that \( M_{\tilde{g}, w_l} \mid s \) is a non-empty restricted glue window submovie of length \( 2e_l \). If \( m > G^k \cdot k \cdot 16^k \), then \( T \) does not self-assemble \( X_k \).

**Proof** The hypothesis of Lemma 3 is satisfied. So there exist \( 1 \leq l < l' \leq m \) such that \( e = e_l = e_{l'} \) and \( M_{\tilde{g}, w_l} \mid s = (\tilde{v}_1, g_1), \ldots, (\tilde{v}_{2e}, g_{2e}) \) and \( M_{\tilde{g}', w_{l'}} \mid s = (\tilde{v}_1, g'_l), \ldots, (\tilde{v}_{2e}, g'_{2e}) \) are sufficiently similar non-empty restricted glue window
submovies. Thus, the hypothesis of Lemma 2 is satisfied. It follows that $T$ does not self-assemble $X_k$.

Here is our main lower bound:

**Theorem 3** Let $k, N \in \mathbb{Z}^+$. If $X_{k,N}$ is any finite just-barely 3D shape with vertical and horizontal extents $k$ and $N$, respectively, then $K_{USA}^1(X_{k,N}) = \Omega\left(N^{\frac{1}{\tau}}\right)$.

**Proof** Assume $T = (T, \sigma, \tau = 1)$ is a directed, 3D TAS that self-assembles $X_{k,N}$. Assume $\alpha \in A_{\square}[T]$ with $\text{dom} \alpha = X_{k,N}$. Let $s = (\vec{x}_0, \vec{x}_1, \ldots, \vec{x}_n)$ be a simple path in $G_{\alpha}^b$, such that, $\{\vec{x}_0\} = \text{dom} \alpha$ and $\vec{x}_n$ is in the furthest extreme column of $X_{k,N}$ from $\vec{x}_0$. Since $\tau = 1$, there is a $T$-assembly sequence $\vec{\alpha}$ that follows $s$. Assume $N \geq 3$.

Since $s$ is a simple path from $\vec{x}_0$ to some location in the furthest extreme column of $X_{k,N}$, there is some positive integer $m \geq \left\lceil \frac{N}{2} \right\rceil \geq \frac{N}{3}$ such that, for all $1 \leq l \leq m$, $w_l$ is a window that cuts $X_{k,N}$, for all $1 \leq l < l' \leq m$, there exists $\vec{\Delta}_{l,l'} \neq 0$ satisfying $w_{l'} = w_l + \vec{\Delta}_{l,l'}$, and for each $1 \leq l \leq m$, there exists a corresponding odd number $1 \leq e_l < 2k$ such that $M_{\tilde{\alpha},w_l} \upharpoonright s$ is a non-empty restricted glue window submovie of length $2e_l$. Moreover, $w_l$ cuts $X_{k,N}$ between some column that lies between the column in which $\vec{x}_0$ is located and the column in which $\vec{x}_n$ is located. The requirement that $s$ is a simple path from $\vec{x}_0$ to some location in the furthest extreme column of $X_{k,N}$ is crucial. For example, the rest of our proof would break if we were to assume that $s$ was merely the longest path from the location of the seed in $G_{\alpha}^b$ (see Fig. 24).

Since $T$ self-assembles $X_{k,N}$, (the contrapositive of) Lemma 4 says that $m \leq G^k \cdot k \cdot 16^k$. We also know that $\frac{N}{3} \leq m$, which means that $\frac{N}{3} \leq G^k \cdot k \cdot 16^k$. Thus, we have $N \leq 3 \cdot G^k \cdot k \cdot 16^k$ and it follows that $|T| \geq \frac{G}{6} \geq \frac{1}{6} \frac{N^\frac{1}{k}}{(3^k \cdot 16^k)^{\frac{1}{k}}} \geq \frac{1}{6} \frac{N^\frac{1}{k}}{(3^k \cdot 2^k \cdot 16^k)^{\frac{1}{k}}} = \frac{1}{6} \frac{N^\frac{1}{k}}{\frac{96}{16}} = \Omega\left(N^{\frac{1}{\tau}}\right)$. \hfill $\square$

**Corollary 5** $K_{USA}^1(R_{3,k,N}^3) = \Omega\left(N^{\frac{1}{\tau}}\right)$.

Although our proof of Theorem 2 crucially depends on Lemma 2 (via Lemma 4), it is possible to prove an “undirected” version of Theorem 2 by employing a version of Lemma 2 that does not assume $T$ is directed. However, in doing so, one would

![Fig. 24](image-url)
ultimately get $K_{SA}^1(X_{k,N}) = \Omega\left(\frac{N^\frac{1}{k}}{k^2}\right)$. This is because a version of Lemma 2 that does not assume $T$ is directed must assume the restricted glue window submovies are equal (up to translation), rather than sufficiently similar, as is the case in the version of Lemma 2 that we use to prove Theorem 2. Note, however, that this bound is still better than a bound of $\Omega\left(\frac{N^\frac{1}{k}}{k}\right)$ based on a standard Window Movie Lemma argument using Corollary 3.4 in [6]. If $T$ is not assumed to be directed in Lemma 2, then it is possible to construct an undirected 3D TAS $T$ that self-assembles a rectangle, even though it exhibits two sufficiently similar restricted glue window submovies.

4.4 Lemma 2 Requires the Directedness Assumption

If $T$ is not assumed to be directed in Lemma 2, then it is possible to construct an undirected 3D TAS $T$ that self-assembles $R_{3,4,17}$ even though it exhibits two sufficiently similar restricted glue window submovies. Depending on the non-deterministic tile placements made by $T$, at most three bit bumps are encoded within $R_{3,4,17}$. The value of the first (westernmost) bit bump is always 0, whereas the value of the second (going east) bit bump is non-deterministically chosen to be either 0 or 1. If the value of the second is 1, then a third (easternmost) bit bump self-assembles, the value of which is non-deterministically chosen to be either 0 or 1. However, if the value of the second bit bump happens to be 0, then a third bit bump does not self-assemble. We sketch the construction of such an undirected $T$ in Figs. 25 through 35. Note that $T$ produces three terminal assemblies, each one characterized by the values of the encoded bit bumps therein, and all having domain $R_{3,4,17}$, testifying to $T$ self-assembling $R_{3,4,17}$. Figures 25 through 31 show the assembly sequence that results in a terminal assembly that encodes three bit bumps with the values 0, 1, and 0, respectively. This assembly sequence induces two sufficiently similar restricted glue window submovies. Figures 32 and 33 show the assembly sequence that results in a terminal assembly that encodes three bit bumps with the values 0, 1, and 1, respectively. Finally, Figs. 34 and 35 show the assembly sequence that results in a terminal assembly that encodes only the first and second bit bumps with the values 0, and 0, respectively.

Self-assembly of the first bump (Fig. 25): The ‘$\sigma$’ tile denotes the seed tile and $w$ is a window. The seed initiates the self-assembly of the first bump in the $z = 0$ plane. The ‘?’ denotes the first location at which one of two different types of tiles may attach. The other such location is shown in Fig. 28. For the sake of example, assume
the glue touching $w$ has the label ‘a’, to which one of two different types of tiles may attach. All other glues in this construction induce deterministic tile attachments.

**Self-assembly of the second bump (Fig. 26):** The ‘1’ tile type is one of two different types of tiles that can attach to the exposed glue from Fig. 25. It self-assembles the second bump with the value 1. After the bump self-assembles, the self-assembly of a path of tiles is initiated, where the path self-assembles to the east, then north, then west across $w$, then up (in the $z = 1$ plane), then west and finally south to the location of the first bump, which, in this example, always has the value 0. The tiles along this path encode their relative positions within the path and the fact that the path originated from a bump with the value 1, indicated by the glues that abut $w$ having the label ‘1’. The last two tiles to attach in this path are located in the westernmost column, in the $z = 1$ plane and the $z = 0$ plane, respectively, with one east output glue each. These tiles effectively start reading the value of (the first) bit bump. Due to the presence of the first bump, only one of these east output glues is exposed, to which the next tile attaches and encodes the value of the bump. See Fig. 34 for the result if a ‘0’ tile type attaches instead of the ‘1’ tile type as depicted here.

**Self-assembly to the second bump (Fig. 27):** If the value of the second bump is 1, which is the case in this example, and if the value of the first bump is 0, which is also the case in this example, then the self-assembly of a path of tiles is initiated, where the path self-assembles to the east, then south and finally east across $w$. The tiles in this path encode their relative positions within the path and the values of the last two bumps that were read, namely the first and second bumps, which is indicated by the tile labeled ‘01’. For the sake of example, assume that the glue of the tiles along this path that abut $w$ are both ‘b’. The last two tiles to attach in this path after crossing $w$ start reading the second bump. Note that tiles in the westernmost third of $R_{4,17}^3$ are
Read the second bump (Fig. 28): The value of the next (second) bump is read. If the value of the bump that was just read is 1, which is the case in this example, then the self-assembly of a hard-coded path of tiles is initiated, where the path self-assembles to the east, then up (in the \( z = 1 \) plane), then east and over an existing tile and finally back down (to the \( z = 0 \) plane), ending before crossing \( w + \Delta \), immediately west of the location of the ‘?’.

The glue that touches \( w + \Delta \) is ‘a’ and the ‘?’ denotes the second location at which one of two different types of tiles may attach, similar to the situation depicted in Fig. 25.

Self-assembly of the third bump (Fig. 29): The ‘0’ tile type attaches (non-deterministically), instead of the ‘1’ tile type, and initiates the self-assembly of the third bump with the value 0. Then, the self-assembly of a path of tiles is initiated, where the path self-assembles to the east, then north, then west across \( w + \Delta \), then up (in the \( z = 1 \) plane), then west and finally south to the location of the second bit bump, which, in this example, has the value 1. The tiles along this path encode their relative positions within the path and the fact that the path originated from a bump with the value 0, indicated by the glues that abut \( w + \Delta \) having the label ‘0’. See Fig. 32 for the result if a ‘1’ tile type attached instead of a ‘0’ tile type as depicted here.

Self-assembly to the third bump (Fig. 30): The value of the second bump is read again (for the last time). If the value of the bump is 1, which is the case in this example, then the self-assembly of a path of tiles is initiated, where the path self-assembles to the east, then up (in the \( z = 1 \) plane), then east over an existing tile, then back down (to the \( z = 0 \) plane), then south and finally across \( w + \Delta \) to the location immediately west of the third bump. The tiles in this path encode their relative positions within the
path and the values of the last two bumps that were read, namely the second and third bumps, and is indicated by the tile labeled ‘10’. Moreover, the glues of this path that abut $w + \Delta$ are ‘b’.

**Read the third bump (Fig. 31):** The value of the third bump is read. If its value is 0, which is the case in this example, then the self-assembly of a hard-coded path of black tiles is initiated. This path of tiles is configured to initiate the self-assembly of subsequent tiles to fill in locations at which a tile has yet to be placed within the easternmost third of $R^3_{4,17}$. At this point, one can infer a $T$-assembly sequence $\tilde{\alpha}$, for some $T$ and a simple path $s$ followed by $\tilde{\alpha}$, such that, $M_{\tilde{\alpha},w} \upharpoonright s$ and $M_{\tilde{\alpha},w+\Delta} \upharpoonright s$ are sufficiently similar restricted glue window submovies but $T$ self-assembles $R^3_{4,17}$. Specifically, the values of the two southernmost glues that touch $w$ and $w + \Delta$, which $\tilde{\alpha}$ crosses going away from the seed, are, from the south, ‘a’ and ‘b’, respectively. Of course, since $T$ is not directed, it can produce other terminal assemblies, which we show in the remainder of this subsection.

**Case when the second and third bumps are both 1 (Fig. 32):** This example is a continuation of Fig. 28, where the ‘1’ tile type attaches, instead of the ‘0’ tile. In this case, the third bump with the value 1 self-assembles. Then, the self-assembly of a path of tiles is initiated, where the path self-assembles to the east, then north, then west across $w + \Delta$, then up (in the $z = 1$ plane), then west and finally south to the location of the second bit bump, which, in this example, has the value 1. The tiles along this path encode their relative positions within the path and the fact that the path originated from a bump with the value 1, indicated by the glues that abut $w + \Delta$ having the label ‘10’.
'1’. This path of tiles is comprised of the same tile types that comprise the path of tiles that was discussed in Fig. 26.

Read the second and third bumps (Fig. 33): The value of the second bump is read. If it has the value 1, which is the case in this example, then the self-assembly of
a hard-coded path of black tiles is initiated. This path of black tiles can be configured to initiate the self-assembly of subsequent tiles to fill in any locations at which a tile has yet to be placed within $R_4^{3,17}$.

**Case when the second bump is 0 (Fig. 34):** This is a continuation of Fig. 25. If the ‘0’ tile type attaches, instead of the ‘1’ tile type as in Fig. 26, then a bump with the value 0 self-assembles as the second bump. After the bump self-assembles, the self-assembly of a path of tiles is initiated, where the path self-assembles to the east, then north, then west across $w$, then up (in the $z = 1$ plane), then west and finally south to the location of the first bump. The tiles along this path encode their relative positions within the path and the fact that the path originated from a bump with the value 0, indicated by the glues that abut $w$ having the label ‘0’.

**Read the first bump (Fig. 35):** The value of the first bump is read and if the second bump has the value 0, which is the case in this example, and if the first bump has the value 0, which is also the case in this example, then the self-assembly of a hard-coded path of black tiles is initiated. This path of tiles is configured to initiate the self-assembly of subsequent tiles to fill in any locations at which a tile has yet to be placed within $R_4^{3,17}$.

5 Conclusion

In this paper, we gave improved lower and upper bounds on $K_{USA}^1(R_k^{3,N})$, namely $\Omega(N^{1/k})$ and $O(N^{1/k^2} + k)$. We leave open the question of determining tight bounds for $K_{USA}^1(R_k^{3,N})$ as well as for $K_{SA}^1(R_k^{3,N})$.

References

1. Winfree, E.: Algorithmic self-assembly of DNA. PhD thesis, California Institute of Technology (1998)
2. Seeman, N.C.: Nucleic-acid junctions and lattices. J. Theor. Biol. 99, 237–247 (1982)
3. Wang, H.: Proving theorems by pattern recognition – II. Bell Syst. Tech. J. XL(1), 1–41 (1961)
4. Rothemund, P.W.K., Winfree, E.: The program-size complexity of selfassembled squares (extended abstract). In: The Thirty-Second Annual ACM Symposium on Theory of Computing (STOC), pp. 459–468 (2000)
5. Manuch, J., Stacho, L., Stoll, C.: Two lower bounds for self-assemblies at temperature 1. J. Comput. Biol. 17(6), 841–852 (2010)
6. Meunier, P.-E., Patitz, M.J., Summers, S.M., Theyssier, G., Winslow, A., Woods, D.: Intrinsic universality in tile self-assembly requires cooperation. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 752–771 (2014)
7. Meunier, P., Woods, D.: The non-cooperative tile assembly model is not intrinsically universal or capable of bounded turing machine simulation. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19–23, 2017, pp. 328–341 (2017)
8. Meunier, P., Regnault, D., Woods, D.: The program-size complexity of self-assembled paths. In: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020, pp. 727–737 (2020)
9. Doty, D., Patitz, M.J., Summers, S.M.: Limitations of self-assembly at temperature 1. Theor. Comput. Sci. 412, 145–158 (2011)
10. Demaine, E.D., Demaine, M.L., Fekete, S.P., Ishaque, M., Rafalin, E., Schweller, R.T., Souvaine, D.L.: Staged self-assembly: nanomanufacture of arbitrary shapes with O(1) glues. Nat Comput 7(3), 347–370 (2008)
11. Doty, D., Patitz, M.J., Reishus, D., Schweller, R.T., Summers, S.M.: Strong fault-tolerance for self-assembly with fuzzy temperature. In: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2010), pp. 417–426 (2010)
12. Fekete, S.P., Hendricks, J., Patitz, M.J., Rogers, T.A., Schweller, R.T.: Universal computation with arbitrary polyomino tiles in non-cooperative self-assembly. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015, pp. 148–167 (2015)
13. Fu, B., Patitz, M.J., Schweller, R.T., Sheline, R.: Self-assembly with geometric tiles. In: Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I, pp. 714–725 (2012)
14. Gilbert, O., Hendricks, J., Patitz, M.J., Rogers, T.A.: Computing in continuous space with self-assembling polygonal tiles (extended abstract). In: Proceedings of the Twenty-Seven Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016, pp. 937–956 (2016)
15. Patitz, M.J., Schweller, R.T., Summers, S.M.: Exact shapes and Turing universality at temperature 1 with a single negative glue. In: Proceedings of the 17th International Conference on DNA Computing and Molecular Programming. DNA’11, pp. 175–189. Springer, Berlin, Heidelberg (2011). http://dl.acm.org/citation.cfm?id=2042033.2042050
16. Hendricks, J., Patitz, M.J., Rogers, T.A., Summers, S.M.: The power of duples (in self-assembly): It’s not so hip to be square. Theor. Comput. Sci. 743, 148–166 (2018)
17. Cook, M., Fu, Y., Schweller, R.T.: Temperature 1 self-assembly: Deterministic assembly in 3D and probabilistic assembly in 2D. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 570–589 (2011)
18. Fucy, D., Micka, S., Summers, S.M.: Optimal program-size encoding for self-assembled squares at temperature 1 in 3D. Algorithmica 77(4), 1240–1282 (2017)
19. Fucy, D., Summers, S.M., Wendlandt, C.: Self-assembly of and optimal encoding within thin rectangles at temperature-1 in 3D. Theor. Comput. Sci. 872, 55–78 (2021)
20. Fucy, D., Summers, S.M.: Optimal self-assembly of finite shapes at temperature 1 in 3D. Algorithmica 80(6), 1909–1963 (2018)
21. Adleman, L.M., Cheng, Q., Goel, A., Huang, M.-D.A.: Running time and program size for self-assembled squares. In: Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing (STOC), pp. 740–748 (2001)
22. Soloveichik, D., Winfree, E.: Complexity of self-assembled shapes. SIAM J. Comput. (SICOMP) 36(6), 1544–1569 (2007)
23. Aggarwal, G., Cheng, Q., Goldwasser, M.H., Kao, M.-Y., de Espanés, P.M., Schweller, R.T.: Complexities for generalized models of selfassembly. SIAM J. Comput. (SICOMP) 34, 1493–1515 (2005)
24. Lathrop, J.I., Lutz, J.H., Summers, S.M.: Strict self-assembly of discrete Sierpinski triangles. Theor. Comput. Sci. 410, 384–405 (2009)
25. Rothemund, P.W.K.: Theory and experiments in algorithmic selfassembly. PhD thesis, University of Southern California (2001)
26. Lutz, J.H., Shutters, B.: Approximate self-assembly of the sierpinski triangle. Theor. Comput. Syst. 51(3), 372–400 (2012)
Authors and Affiliations

David Furcy · Scott M. Summers · Logan Withers

David Furcy
furcyd@uwosh.edu

Logan Withers
loganswithers97@gmail.com

1 Computer Science Department, University of Wisconsin Oshkos, 800 Algoma Blvd, Oshkos, 54901, WI, USA