Universal Computation with Arbitrary Polyomino Tiles in Non-Cooperative Self-Assembly*

Sándor P. Fekete † Jacob Hendricks ‡ Matthew J. Patitz § Trent A. Rogers ¶
Robert T. Schweller ‖

Abstract

In this paper we explore the power of geometry to overcome the limitations of non-cooperative self-assembly. We define a generalization of the abstract Tile Assembly Model (aTAM), such that a tile system consists of a collection of polyomino tiles, the Polyomino Tile Assembly Model (polyTAM), and investigate the computational powers of polyTAM systems at *temperature 1*, where attachment among tiles occurs without glue cooperation (i.e., without the enforcement that more than one tile already existing in an assembly must contribute to the binding of a new tile). Systems composed of the unit-square tiles of the aTAM at temperature 1 are believed to be incapable of Turing universal computation (while cooperative systems, with temperature > 1, are able). As our main result, we prove that for any polyomino P of size 3 or greater, there exists a temperature-1 polyTAM system containing only shape-P tiles that is computationally universal. Our proof leverages the geometric properties of these larger (relative to the aTAM) tiles and their abilities to effectively utilize geometric blocking of particular growth paths of assemblies, while allowing others to complete. In order to prove the computational powers of polyTAM systems, we also prove a number of geometric properties held by all polyominoes of size ≥ 3.

To round out our main result, we provide strong evidence that size-1 (i.e. aTAM tiles) and size-2 polyomino systems are unlikely to be computationally universal by showing that such systems are incapable of *geometric bit-reading*, which is a technique common to all currently known temperature-1 computationally universal systems. We further show that larger polyominoes with a limited number of binding positions are unlikely to be computationally universal, as they are only as powerful as temperature-1 aTAM systems. Finally, we connect our work with other work on domino self-assembly to show that temperature-1 assembly with at least 2 distinct shapes, regardless of the shapes or their sizes, allows for universal computation.

1 Introduction

Theoretical studies of algorithmic self-assembly have produced a wide variety of results that establish the computational power of tile-based self-assembling systems. From the introduction of the first and perhaps simplest model, the abstract Tile Assembly Model (aTAM) [26], it was shown that self-assembling systems, which are based on relatively simple components autonomously coalescing to form more complex structures, are capable of Turing-universal computation. This computational power exists within the aTAM, and has been harnessed to algorithmically guide extremely complex theoretical constructions (e.g. [5, 14, 20, 21, 23, 25]) and has even been exploited within laboratories to build nanoscale self-assembling systems from DNA-based tiles which self-assemble following algorithmic behavior (e.g. [1, 8, 13, 15, 22, 24]).

While physical implementations of these systems are constantly increasing in scale, complexity, and robustness, they are orders of magnitude shy of achieving results similar to those of many naturally occurring self-assembling systems, especially those found in biology (e.g. the formation of many cellular structures or viruses). This disparity motivates theoretical studies that can focus efforts on first discovering the "tricks" used so successfully by nature, and then on incorporating them into our own models and systems. One

*A full version of this paper can be found at [9].

†Department of Computer Science, TU Braunschweig, 38106 Braunschweig, Germany. s.fekete@tu-bs.de.

‡Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR, USA. jhendric@uark.edu. This author's research was supported in part by NSF grants CCF-1117672 and CCF-1422152.

§Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR, USA. mpatitz@self-assembly.net. This author's research was supported in part by NSF grants CCF-1117672 and CCF-1422152.

¶Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR, USA. tar003@uark.edu. This author's research was supported by the NSF Graduate Research Fellowship Program under grant No. DGE-1450079, and NSF grants CCF-1117672 and CCF-1422152.

‖University of Texas–Pan American, Edinburg, TX 78539, USA. rtschweller@utpa.edu. This author's research was supported in part by NSF grant CCF-1117672.

of the fundamental properties so successfully leveraged by many natural systems, but absent from models such as the aTAM, is geometric complexity of components. For instance, self-assembly in biological systems relies heavily upon the complex 3-dimensional structures of proteins, while tile-assembly systems are typically restricted to basic square (or cubic) tiles. In this paper, we greatly extend previous work that has begun to incorporate geometric aspects of self-assembling components [3, 6, 10, 12] with the development of a model allowing for more geometrically complex tiles, called *polyominoes*, and an examination of the surprising computational powers of systems composed of polyominoes.

The process of tile assembly begins from a *seed* structure, typically a single tile, and proceeds with tiles attaching one at a time to the growing assembly. Tiles have *glues*, taken from a set of glue types, around their perimeters which allow them to attach to each other if their glues match. Algorithmic self-assembling systems developed by researchers, both theoretical and experimental, tend to fundamentally employ an important aspect of tile assembly known as *cooperation*. In theoretical models, cooperation is available when a particular parameter, known as the *temperature*, is set to a value > 1 which can then enforce that the binding of a tile to a growing assembly can only occur if that tile matches more than one glue on the perimeter of the assembly. Using cooperation, it is simple to show that systems in the aTAM are capable of universal computation (by simulating particular cellular automata [26], or arbitrary Turing machines [14, 21, 25], for instance). However, it has long been conjectured that in the aTAM without cooperation, i.e. in systems where the temperature is equal to 1, universal computation is impossible [7,16,17]. Interestingly, though, a collection of "workarounds" have been devised in the form of new models with a variety of properties and parameters which make computation possible at temperature 1 (e.g. [2, 10, 12, 18, 19]).

In this paper, we introduce the Polyomino Tile Assembly Model (polyTAM), in which each tile is composed of a collection of unit squares joined along their edges. This allows for tiles with arbitrary geometric complexity and a much larger variety of shapes than in earlier work involving systems composed of both square and 2×1 rectangular tiles [11, 12], or those with tiles composed of square bodies and edges with bumps and dents [10]. Our results prove that geometry, in the polyTAM as in natural self-assembling systems, affords great power. Namely, *any* polyomino shape which is composed of only 3 or more unit squares has enough geometric complexity to allow a polyTAM system at temperature 1, composed only of tiles of that shape, to perform Turing universal computation. This impressive potency is perhaps even more surprising when it is realized that while a single unit-square polyomino (a.k.a. a *monomino*, or a standard aTAM tile) is conjectured not to provide this power, the same shape expanded in scale to a 2×2 square polyomino does. The key to this power is the ability of arbitrary polyominoes of size 3 or greater to both combine with each other to form regular grids, as well as to combine in a variety of relative offsets that allow some tiles to be shifted relatively to those grids and then perform geometric blocking of the growth of specific configurations of paths of tiles, while allowing other paths to complete their growth. With just this seemingly simple property, it is possible to design temperature-1 systems of polyominoes that can simulate arbitrary Turing machines.

In addition to this main positive result about the computational abilities of all polyominoes of size ≥ 3, we also provide negative results that further help to refine understanding of exactly what geometric properties are needed for Turing universal computation in temperature-1 self-assembly. We prove that a fundamental gadget (which we call the *bit-reading gadget*) used within all known systems that can compute at temperature 1 in any tile-assembly model, is impossible to construct with either the square tiles of the aTAM or with dominoes (a.k.a. duples). This provides further evidence that systems composed solely of those shapes are incapable of universal computation. Furthermore, we prove that regardless of the size and shape of a polyomino, systems composed of polyominoes with only (1) ≤ 3 positions on its perimeter at which to place glues, or (2) 4 positions for glues that are restricted to binding with each other as complementary pairs of sides, are no more powerful than aTAM temperature-1 systems, again providing evidence that they are incapable of performing Turing universal computation.

This paper is organized as follows. In Section 2 we define the Polyomino Tile Assembly Model and related terminology. In Section 3 we formally define a bit-reading gadget and then present an overview of how they can be used in a temperature-1 tile assembly system to simulate arbitrary Turing machines. Then, in Section 4 we prove some fundamental lemmas about the geometric properties of polyominoes and the ways in which they can combine in the plane to form grids. Section 5 contains the proof of our main result, while Section 6 contains our results that hint at the computational weakness of some systems. Finally, Section 7 describes how the positive results of this paper along with that of [12] proves that any polyomino system composed of any two polyomino shapes is capable of universal computation.

2 Polyomino Tile Assembly Model

In this section we define the Polyomino Tile Assembly Model (polyTAM) and relevant terminology.

Polyomino Tiles A *polyomino* is a plane geometric figure formed by joining one or more equal unit squares edge to edge; it can also be considered a finite subset of the regular square tiling with a connected interior. For convenience, we will assume that each unit square is centered on a point in Z^2. We define the set of *edges* of a polyomino to be the set of faces from the constituent unit squares of the polyomino that lie on the boundary of the polyomino shape. A *polyomino tile* is a polyomino with a subset of its edges labeled from some *glue* alphabet Σ, with each glue having an integer *strength* value. Two tiles are said to *bind* when they are placed so that they have non-overlapping interiors and adjacent edges with matching glues; each matching glue binds with force equal to its strength value. An *assembly* is any connected set of polyominoes whose interiors do not overlap. Given a positive integer $\tau \in \mathbb{N}$, an assembly is said to be τ*-stable* or (just *stable* if τ is clear from context), if any partition of the assembly into two non-empty groups (without cutting individual polyominoes) must separate bound glues whose strengths sum to $\geq \tau$.

The *bounding rectangle* B around a polyomino P is the rectangle with minimal area (and corners lying in $\mathbb{Z}^2$) that contains P. For each polyomino shape, we designate one pixel (i.e. one of the squares making up P) p as a distinguished pixel that we use as a reference point. More formally, a *pixel* p in a polyomino P (or polyomino tile) is defined in the following manner. Place P in the plane so that the southwest corner of the bounding rectangle of P lies at the origin. Then a pixel $p = (p_1, p_2) \in P$ is a point in $\mathbb{Z}^2$ which is occupied by a unit square composing the polyomino P. We say that a pixel $p' \in P$ lies on the perimeter of the bounding rectangle B if an edge of the pixel p' lies on an edge of B.

Tile System A *tile assembly system* (TAS) is an ordered triple $\mathcal{T} = (T, \sigma, \tau)$ (where T is a set of polyomino tiles, and σ is a τ-stable assembly called the *seed* consisting of integer translations of elements of T). τ is the *temperature* of the system, specifying the minimum binding strength necessary for a tile to attach to an assembly. Throughout this paper, the temperature of all systems is assumed to be 1, and we therefore frequently omit the temperature from the definition of a system (i.e. $\mathcal{T} = (T, \sigma)$).

If the tiles in T all have the same polyomino shape, $\mathcal{T}$ is said to be a *single-shape* system; more generally $\mathcal{T}$ is said to be a c-shape system if there are c distinct shapes in T. If not stated otherwise, systems described in this paper should by default be assumed to be single-shape systems. If T consists of unit-square tiles, $\mathcal{T}$ is said to be a *monomino* system.

Assembly Process Given a tile-assembly system $\mathcal{T} = (T, \sigma, \tau)$, we now define the set of *producible* assemblies $\mathcal{A}[\mathcal{T}]$ that can be derived from $\mathcal{T}$, as well as the *terminal* assemblies, $\mathcal{A}_\square[\mathcal{T}]$, which are the producible assemblies to which no additional tiles can attach. The assembly process begins from σ and proceeds by single steps in which any single copy of some tile $t \in T$ may be attached to the current assembly A, provided that it can be translated so that its placement does not overlap any previously placed tiles and it binds with strength $\geq \tau$. For a system $\mathcal{T}$ and assembly A, if such a $t \in T$ exists, we say $A \rightarrow_1^{\mathcal{T}} A'$ (i.e. A grows to A' via a single tile attachment). We use the notation $A \rightarrow^{\mathcal{T}} A''$, when A grows into A'' via 0 or more steps. Assembly proceeds asynchronously and nondeterministically, attaching one tile at a time, until no further tiles can attach. An assembly sequence in a TAS $\mathcal{T}$ is a (finite or infinite) sequence $\vec{\alpha} = (\alpha_0 = \sigma, \alpha_1, \alpha_2, \ldots)$ of assemblies in which each α_{i+1} is obtained from α_i by the addition of a single tile. The set of producible assemblies $\mathcal{A}[\mathcal{T}]$ is defined to be the set of all assemblies A such that there exists an assembly sequence for $\mathcal{T}$ ending with A (possibly in the limit). The set of *terminal* assemblies $\mathcal{A}_\square[\mathcal{T}] \subseteq \mathcal{A}[\mathcal{T}]$ is the set of producible assemblies such that for all $A \in \mathcal{A}_\square[\mathcal{T}]$ there exists no assembly $B \in \mathcal{A}[\mathcal{T}]$ in which $A \rightarrow_1^{\mathcal{T}} B$. A system $\mathcal{T}$ is said to be directed if $|\mathcal{A}_\square[\mathcal{T}]| = 1$, i.e., if it has exactly one terminal assembly.

Note that the aTAM is simply a specific case of the polyTAM in which all tiles are monominoes, i.e., single unit squares.

3 Universal Computation with Geometric Bit-Reading

In this section we provide an overview of how universal computation can be performed in a temperature-1 system with appropriate use of geometric aspects of tiles and assemblies. Refer to Figure 1 for an illustration.

3.1 Bit-Reading Gadgets

First, we discuss a primitive tile-assembly component that enables computation by self-assembling systems. This component is called the *bit-reading gadget*, and essentially consists of pre-existing assemblies that appropriately encode bit values (i.e., 0 or 1) and paths that grow past them and are able to "read" the values of the encoded bits; this results in those bits being encoded in the tile types of the paths beyond the encoding assemblies. In tile-assembly systems in which the temperature is ≥ 2, a bit-reader gadget is trivial: the assembly encoding the bit value

can be a single tile with an exposed glue that encodes the bit value, and the path that grows past to read the value simply ensures that a tile must be placed cooperatively with, and adjacent to, that encoding the bit (i.e., the path forces a tile to be placed that can only bind if one of its glues matches that exposed by the last tile of the path, and the other matches the glue encoding the bit value). However, in a temperature-1 system, cooperative binding of tiles cannot be enforced, and therefore the encoding of bits must be done using geometry. Figure 1 provides an intuitive overview of a temperature-1 system with a bit-reading gadget. Essentially, depending on which bit is encoded by the assembly to be read, exactly one of two types of paths can complete growth past it, implicitly specifying the bit that was *read*. It is important that the bit reading must be unambiguous, i.e., depending on the bit *written* by the pre-existing assembly, exactly one type of path (i.e., the one that denotes the bit that was written) can possibly complete growth, with all paths not representing that bit being prevented. Furthermore, the correct type of path must always be able to grow. Therefore, it cannot be the case that either all paths can be blocked from growth, or that any path not of the correct type can complete, regardless of whether a path of the correct type also completes, and these conditions must hold for any valid assembly sequence to guarantee correct computation.

DEFINITION 3.1. *We say that a* bit-reading gadget *exists for a tile assembly system $\mathcal{T} = (T, \sigma, \tau)$, if the following hold. Let $T_0 \subset T$ and $T_1 \subset T$, with $T_0 \cap T_1 = \varnothing$, be subsets of tile types which represent the bits* 0 *and* 1*, respectively. For some producible assembly $\alpha \in \mathcal{A}[\mathcal{T}]$, there exist two connected subassemblies, $\alpha_0, \alpha_1 \sqsubseteq \alpha$ (with w equal to the maximal width of α_0 and α_1, i.e., the largest extent in x-direction spanned by either subassembly), such that if:*

1. *α is translated so that α_0 has its minimal y-coordinate ≤ 0 and its minimal x-coordinate $= 1$,*
2. *a tile of some type $t \in T$ is placed at $(w + n, h)$, where $n, h \geq 1$, and*
3. *the tiles of α_0 are the only tiles of α in the first quadrant to the left of t,*

then at least one path must grow from t (staying strictly above the x-axis) and place a tile of some type $t_0 \in T_0$ as the first tile with x-coordinate $= 0$, while no such path can place a tile of type $t' \in (T \setminus T_0)$ as the first tile to with x-coordinate $= 0$. (This constitutes the reading of a 0 *bit.)*

Additionally, if α_1 is used in place of α_0 with the same constraints on all tile placements, t is placed in the same location as before, and no other tiles of α are in the first quadrant to the left of t, then at least one path must grow from t and stay strictly above the x-axis and strictly to the left of t, eventually placing a tile of some type $t_1 \in T_1$ as the first tile with x-coordinate $= 0$, while no such path can place a tile of type $t' \in (T \setminus T_1)$ as the first tile with x-coordinate $= 0$. (Thus constituting the reading of a 1 *bit.)*

We refer to α_0 and α_1 as the *bit writers*, and the paths which grow from t as the *bit readers*. Also, note that while this definition is specific to a bit-reader gadget in which the bit readers grow from right to left, any rotation of a bit reader is valid by suitably rotating the positions and directions of Definition 3.1. As mentioned in Figure 1, depending on the actual geometries of the polyominoes used and their careful placement, it may be possible to enforce the necessary blocking of all paths of the wrong type, while still allowing at least one path of the correct type to complete growth in any valid assembly sequence. The necessary requirements on these geometries and placements are the subject of the novel results of this paper.

3.2 Turing-Machine Simulation In order to show that a polyomino shape (i.e., a system composed of tiles of only that shape) is computationally universal at $\tau = 1$, we show how it is possible to simulate an arbitrary Turing machine using such a polyomino system. In order to simulate an arbitrary Turing machine, we show how to self-assemble a zig-zag Turing machine [2, 19]. A zig-zag Turing machine at $\tau = 1$ works by starting with its input row as the seed assembly, then growing rows one by one, alternating growth from left to right with growth from right to left. As a row grows across the top of the row immediately beneath it, it does so by forming a path of single tile width, with tiles connected by glues, which pass information horizontally through their glues, while the geometry of the row below causes only one of two choices of paths to grow at regular intervals, effectively passing information vertically via the geometry, using bit-reading gadgets.

Each cell of the Turing machine's tape is encoded by a series of bit-reader gadgets that encode in binary the symbol in that cell and, if the read/write head is located there, what state the machine is in. Additionally, as each cell is read by the row above, the necessary information must be geometrically written above it so that the next row can read it. See Figure 2 for an example depicting a high-level schematic without showing details of the individual polyominoes. Figure 3 shows the same system after two rows have completed growth.

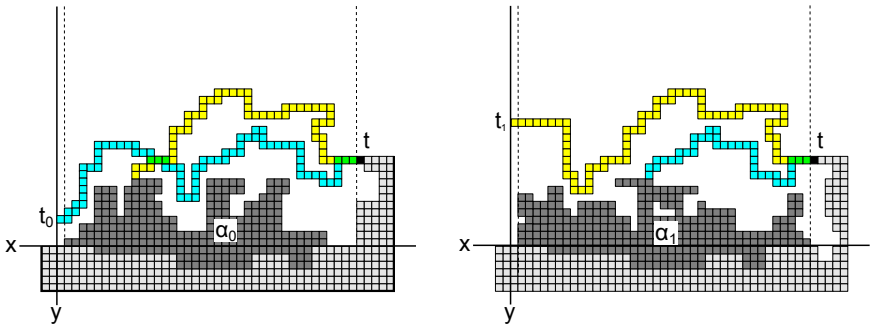


Figure 1: Abstract schematic of a bit-reading gadget. (Left) The blue path grown from t "reads" the bit 1 from α_0 (by being allowed to grow to $x = 0$ and placing a tile $t_0 \in T_0$), while the yellow path (which could read a 0 bit) is blocked by α_0. (Right) The yellow path grown from t reads the bit 0 from α_1, while the blue path that could potentially read a 1 is blocked by α_1. Clearly, the specific geometry of the used polyomino tiles and assemblies is important in allowing the yellow path in the left figure to be blocked without also blocking the blue path.

For a more specific example that shows the placement of individual, actual polyomino tiles as well as the order of their growth, see Figure 4. Note that the simulation of a zig-zag Turing machine can be performed by horizontal or vertical growth, and in any orientation.

4 Technical Lemmas: Grids of Polyominoes

As mentioned above, in order to show that all polyominoes of size greater than 2 are universal, we show that a bit-reading gadget can be constructed with these polyominoes. In this section we show two lemmas about single-shaped polyTAM systems that will aid in the construction of bit-writers used to show that any polyomino of size greater than 2 can be used to define a single-shape polyTAM system capable of universal computation. Throughout this section, any mention of a polyTAM system refers to a single-shape system.

The following lemma says that if a polyomino P can be translated by a vector $\vec{v}$ so that no pixel positions of the translated polyomino overlap the pixel positions of P, then for any integer $c \neq 0$, no pixel positions of P translated by $c \cdot \vec{v}$ overlap the pixel positions of P. The proof of Lemma 4.1 can be found in [3]; the statement of the lemma has been included for the sake of completeness.

LEMMA 4.1. *Consider a two-dimensional, bounded, connected, regular closed set S, i.e., S is equal to the topological closure of its interior points. Suppose S is translated by a vector v to obtain shape S_v, such that S and S_v have disjoint interiors. Then the shape S_{c*v} obtained by translating S by $c * v$ for any integer $c \neq 0$ and S have disjoint interiors.*

Informally, the following lemma says that any polyomino gives rise to a polyTAM system that can produce an infinite "grid" of polyominoes, as shown in Figure 5.

LEMMA 4.2. *Given a polyomino P. There exists a directed, singly seeded, single-shape tile system $\mathcal{T} = (T, \sigma)$ (where the seed is placed so that pixel $p \in P$ is at location $(0, 0)$ and the shape of tiles in T is P) and vectors $\vec{v}, \vec{w} \in \mathbb{Z}^2$, such that $\mathcal{T}$ produces the terminal assembly α, which we refer to as a* grid*, with the following properties. (1) Every position in α of the form $c_1\vec{v} + c_2\vec{w}$, where $c_1, c_2 \in \mathbb{Z}$, is occupied by the pixel p, and (2) for every $c_1, c_2 \in \mathbb{Z}$, the position in $\mathbb{Z}^2$ of the form $c_1\vec{v} + c_2\vec{w}$ is occupied by the pixel p for some tile in α.*

Details of the proof are omitted for space reasons and can be found in the full paper [9].

Let $p, p' \in P$ be distinct pixels in the polyomino P at positions (x, y) and (x', y') respectively, let $\vec{r} = (x - x', y - y')$, and let $\vec{v}, \vec{w}$ be as defined in Lemma 4.2. Then, if there exists $c_1, c_2 \in \mathbb{Z}$ such that $(x, y) + c_1\vec{v} + c_2\vec{w} = (x', y')$, we say that the polyomino which occupies (x', y') is $\vec{r}$-shifted with respect to (or relative to) the polyomino at (x, y). If a polyomino at position (x', y') is 0-shifted with respect to a polyomino at (x, y), we say that the polyomino at position (x', y') is *on grid* with the polyomino at (x, y). If a polyomino is not on grid with a polyomino at (x, y), we say that the polyomino is *off grid*. Henceforth, if we do not mention the tile which another tile is shifted in respect to, assume that the tile is shifted with respect to the seed.

For the remainder of this section, for a polyomino P, we let $V \subset \mathbb{Z}^2$ denote the set of vectors such that $\vec{r} \in V$ provided that there exists some directed, singly seeded system, single shape $\mathcal{T}' = (T', \sigma')$ with shape given by P whose terminal assembly α' contains an $\vec{r}$-shifted polyomino tile, and we let B denote the subset of vectors $B \subset V$ such that $\vec{b} \in B$ provided that there exists a directed, singly seeded system, single shape $\mathcal{T} = (T, \sigma)$ with shape given by P such that the terminal assembly

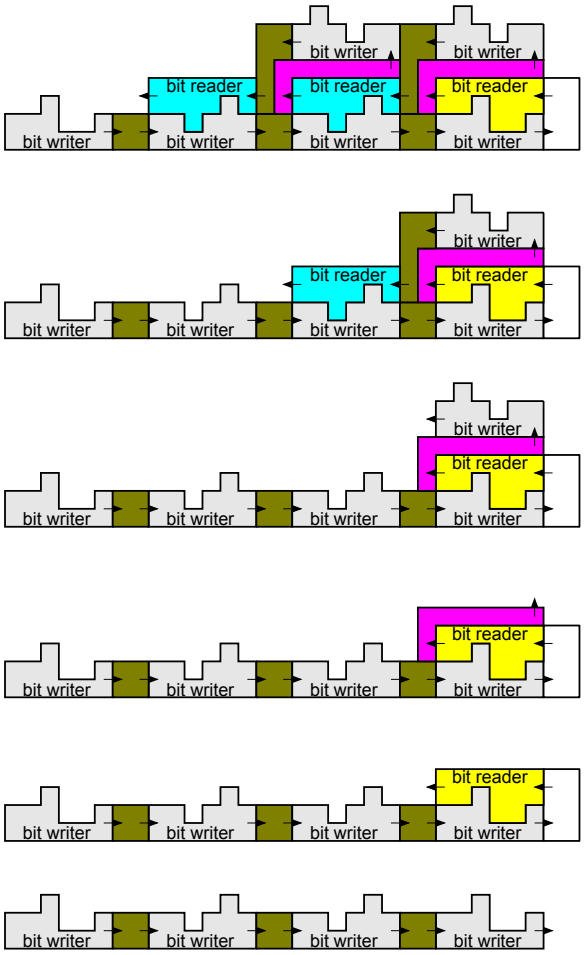


Figure 2: High-level schematic view of a zig-zag Turing machine and the bit-reading/writing gadgets that make up each row of the simulation. The bottom shows the seed row, consisting of bit-writer gadgets separated by spacers. Then, depicted as consecutive upward figures, the second row begins its growth. Yellow/blue portions depict locations of bit-reader gadgets (for 0 and 1, respectively), which grow pink paths upward after completing in order to grow bit writer gadgets (grey), and then gold spacers back down to the point where the next bit reader can grow.

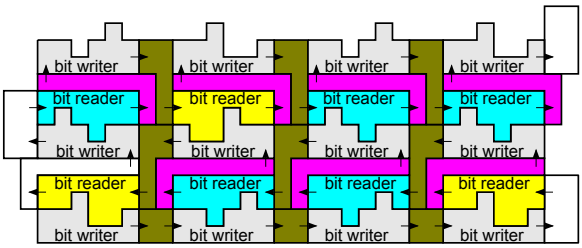


Figure 3: High-level schematic view of a zig-zag Turing machine and the bit-reading/writing gadgets that make up the first two rows of simulation.

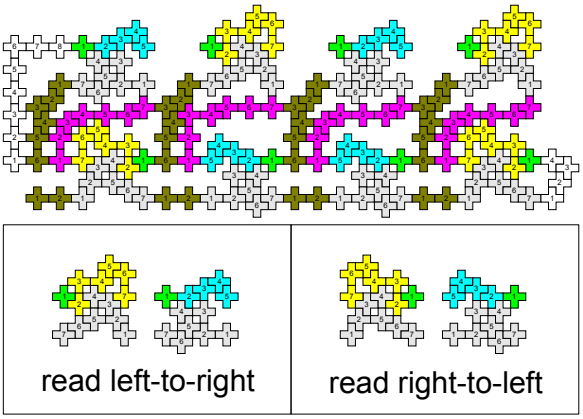


Figure 4: The system of Figure 2 after two rows of the zig-zag simulation have been completed (omitting the output bit writer gadgets of the second row), implemented with "plus-sign" polyominoes. The bottom left shows 0 and 1 bit-writer and reader combinations, with the writer having grown from right to left and the reader from left to right. The bottom right shows the same, but with growth directions reversed. Grey tiles represent bit-writer gadgets. Green tiles represent the beginning of bit-reader gadgets that are common to either bit; yellow represents the path that can grow to signify a 0 bit being read, and blue a 1 bit. Other colors correspond to those for the gadgets used in Figure 2, with numbers corresponding to the growth order of the tiles in each gadget.

α of $\mathcal{T}$ consists of exactly two tiles: the seed tile σ and a $\vec{b}$-shifted tile. B can be thought of as the set of vectors such that the polyomino P and a copy of P shifted by a vector in B are non-overlapping and contain pixels that share a common edge. We can think of B as a set of basis vectors for V in the following sense. If $\vec{r} \in V$, then $\vec{r}$ can be written as a linear combination of shifts in B. The following lemma is a more formal statement of this fact.

LEMMA 4.3. *For any vector $\vec{r} \in V$, $\vec{r} = \Sigma c_i \vec{b_i}$ for some $c_i \in \mathbb{Z}$ and $\vec{b_i} \in B$.*

Proof. This follows from the fact that if $\vec{r} \in V$, then there exists some directed, singly seeded system $\mathcal{T} = (T, \sigma)$ which contains an $\vec{r}$-shifted polyomino tile A. Then, there must be a path of neighboring polyomino tiles from the seed tile S to A. Starting from S, each consecutive tile along this path to A must be a $\vec{b}$-shifted tile for some $\vec{b}$ in B, and the sum of these vectors is $\vec{r}$.□

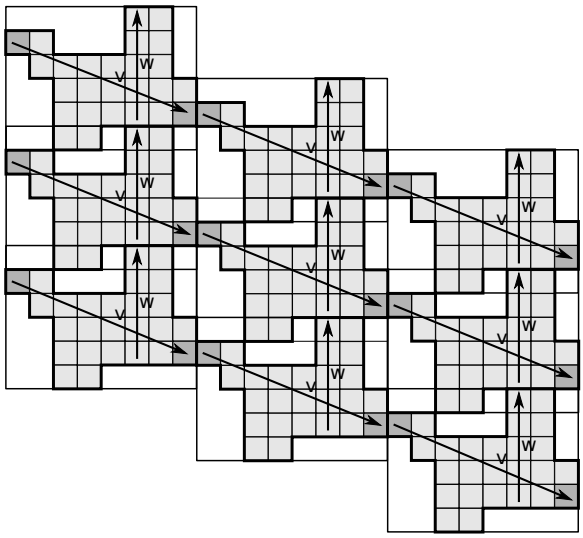


Figure 5: A lattice formed from an example polyomino, using the vectors $\vec{v}$ and $\vec{w}$.

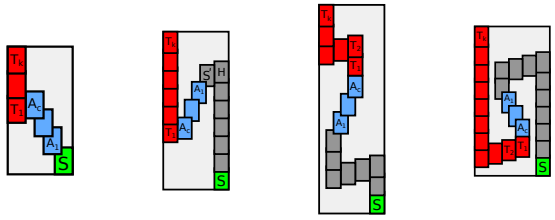


Figure 6: A depiction of the properties given in Lemma 4.4 for four different cases. Each of the small rectangles (red, green, blue or gray in color) serves as the bounding rectangle of some polyomino. Hence, α is contained in the union of all of the regions bounded by these rectangles. The seed tile S of α is contained in the green rectangle. As assembly proceeds from the seed the first r-shifted polyomino tile is A_c. From this point on in the assembly process, the rectangles are on grid with A_c.

For a rectangle R and a tile T, we say that T *lies in the southeast* (respectively northwest) *corner* of R iff the south and east edges of the bounding rectangle of the tile T lie on the south and east edges of R. Let V contain every linear combination $\Sigma c_i \vec{b_i}$ for $c_i \in \mathbb{Z}$ and $\vec{b_i} \in B$. The next two lemmas formalize the following notion. For $c \in \mathbb{Z}$ and $\vec{b} \in B$, the following lemma shows how if $\vec{r} = c \cdot \vec{b}$, we can give a system $\mathcal{T}_{\vec{r}}$ that contains an $\vec{r}$-shifted tile. Furthermore, the properties given in the lemma statement ensure that if we have two such system, $\mathcal{T}_{\vec{r}_1}$ and $\mathcal{T}_{\vec{r}_2}$, corresponding to two shift vectors $\vec{r}_1$ and $\vec{r}_2$, $\mathcal{T}_{\vec{r}_1}$ and $\mathcal{T}_{\vec{r}_2}$ can be "concatenated" to give a system that contains an $(\vec{r}_1+\vec{r}_2)$-shifted polyomino tile. See Figure 6 for schematic depictions of the properties given in the following lemma.

LEMMA 4.4. *Let $\vec{r} = c \cdot \vec{b}$ for any $c \in \mathbb{Z}$ and $\vec{b} \in B$. Then there exists some directed, singly seeded system $\mathcal{T} = (T, \sigma)$ with all tiles shaped P such that the terminal assembly α of $\mathcal{T}$ contains an $\vec{r}$-shifted tile. Moreover, the system $\mathcal{T}$ and assembly α have the following properties.*

1. *There is a unique assembly sequence that yields α,*
2. *For some $m, n \in \mathbb{N}$, α is contained in an $m \times n$ rectangle R and the seed tile S lies in the southeast corner of R, and*
3. *the last tile, A, to attach to α lies in the northwest corner of R.*

Please see the full paper [9] for the proof of Lemma 4.4.

As previously mentioned, the properties given in Lemma 4.4 ensure that if we have two such systems, $\mathcal{T}_{\vec{r}_1}$ and $\mathcal{T}_{\vec{r}_2}$ corresponding to two shift vectors $\vec{r}_1$ and $\vec{r}_2$, $\mathcal{T}_{\vec{r}_1}$ and $\mathcal{T}_{\vec{r}_2}$ can be "concatenated" to give a system that contains an $(\vec{r}_1 + \vec{r}_2)$-shifted polyomino tile. The following lemma formalizes this notion of "concatenation".

LEMMA 4.5. *Let $\vec{r} = \Sigma_{i=0}^{n} c_i \vec{b_i}$ for any $c_i \in \mathbb{Z}$ and $\vec{b_i} \in B$. Then there exists some directed, singly seeded system $\mathcal{T} = (T, \sigma)$ with all tiles shaped P such that the terminal assembly α of $\mathcal{T}$ contains an $\vec{r}$-shifted polyomino. Moreover, the system $\mathcal{T}$ and assembly α have the following properties.*

1. *There is a unique assembly sequence that yields α,*
2. *For some $m, n \in \mathbb{N}$, α is contained in an $m \times n$ rectangle R and the seed tile S lies in the southeast corner of R, and*
3. *the last polyomino tile, A, to attach in the system lies in the northwest corner of R.*

Proof. This follows by applying Lemma 4.4 to each of the summands of $\vec{r} = \Sigma c_i \vec{b_i}$. □

In Lemma 4.5, we start with a seed tile in the southeast corner of a rectangular region R and proceed to place an r-shifted tile in the northwest corner of the rectangle. Note that by using the techniques used to prove Lemma 4.4 and Lemma 4.5, we can show analogous lemmas where the seed tile lies in any corner of R and an r-shifted tile lies in the opposite corner.

Now we state the main lemma for this section. This lemma will allow us to construct bit-writing gadgets used in the construction given in Section 5. Intuitively, the lemma states that given a polyomino P and vector $\vec{r} \in V$, we can define a polyTAM system that starts growth from a seed tile, S, in the southeast corner of a rectangle R, and without growing outside of R, places $\vec{r}$-shifted tiles on the west edge of R in such a way that it is possible to then continue growth to the west of R. The possibility of continuing growth to the west of R is formally stated as Property 4 in Lemma 4.6. This lemma will allow us to assemble a series of bit-writers while also resting assured that once these bit-writers have assembled, bit-reader assemblies can continue growth. It is helpful to see Figure 7 for an overview of the properties given in the following lemma.

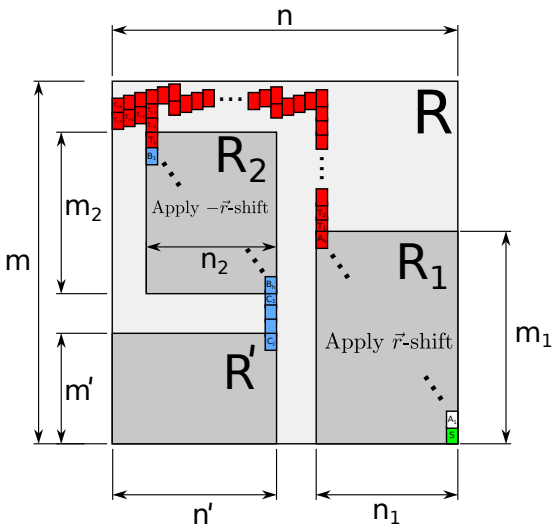


Figure 7: The rectangular region R_1 contains tiles of α defined using Lemma 4.5 with vector $\vec{r}$. The red rectangular regions each serve as the bounding rectangle of an $\vec{r}$-shifted tile of α. This path of tiles, $\{T_i\}_{i=1}^{l}$, places a final tile in the northwest corner of the region R_2. R_2 contains tiles of α defined using Lemma 4.5 with vector $-\vec{r}$-shifted. The blue rectangular regions each contain a single tile that is on grid with S. This path of tiles places a final tile in the northeast corner of the region R'. Note that by modifying the path of tiles $\{T_i\}_{i=1}^{l}$, we can make the dimension m' and n' of R' arbitrarily large.

LEMMA 4.6. *Let P be a polyomino. Let $\vec{r}$ be a vector in V. Then there exists a directed, singly seeded system $\mathcal{T} = (T, \sigma)$ with all tiles of shape P which produces α such that $\mathcal{T}$ and α have the following properties.*

1. *There is a unique assembly sequence that yields α,*
2. *for some $m, n \in \mathbb{N}$, α is contained in an $m \times n$ rectangle R and the seed tile S lies in the southeast corner of R,*
3. *for any tile A of α such that the west edge of the bounding rectangle of A lies on the west edges of R, A is $\vec{r}$-shifted, and*
4. *for any $m'', n'' \in \mathbb{N}$, we can choose $\mathcal{T}$ such that the last tile L to attach to α lies on grid in the northeast corner of a rectangle R' with dimensions $m' \times n'$ where $m' > m''$ and $n' > n''$. Moreover, the south and west edges of R' lie on the south and west edges of R, and no portion of any tile of α lies inside of R' and outside of the bounding rectangle of L.*

Please see the full paper [9] for the proof of Lemma 4.6.

As in Lemma 4.5, we start with a seed tile in the southeast corner of a rectangular region R and proceed to place r-shifted tiles on the west edge of the rectangle. Note that we can show analogous lemmas where the seed tile starts in any corner of a rectangle R, and r-shifted tiles are placed on a chosen opposite edge.

5 All Polyominoes of Size at Least 3 Can Perform Universal Computation at $\tau = 1$

We can now proceed to state our main result: any polyomino P of size at least three can be used for polyomino tile-assembly systems that are computationally universal at temperature 1. Formally stated:

THEOREM 5.1. *Let P be a polyomino such that $|P| \geq 3$. Then for every standard Turing Machine M and input w, there exists a TAS with $\tau = 1$ consisting only of tiles of shape P that simulates M on w.*

It follows from the procedure outlined in Section 3.2 and the Lemmas of Section 4 that in order to simulate an arbitrary Turing Machine by a TAS consisting only of tiles of some polyomino shape P, it is sufficient to construct a system consisting only of tiles of shape P for which there exists a bit-reading gadget, because the additional paths required for a zig-zag Turing machine simulation are guaranteed to be producible by the lemmas of Section 4.

To simplify our proof, we consider different categories of shapes of P as separate cases, which first requires an additional definition.

DEFINITION 5.1. (BASIC POLYOMINO) *A polyomino P is said to be a* basic *polyomino if and only if for every vector $\vec{x}$ modulo the polyomino grid for P, there exists a system $\mathcal{T}$ containing only tiles with shape P such that*

$\mathcal{T}$ produces α and α contains a $\vec{x}$-shifted polyomino. Otherwise we call P non-basic.

Essentially, basic polyominoes are those which have the potential to grow paths that place tiles at any and all shift vectors relative to the grid.

Our proof consists of showing how to build bit-reader gadgets for each of the following cases based on the shape P:

(1) P has thickness 1 in one direction, i.e., it is an $m \times 1$ polyomino.

(2) P has thickness 2 in two directions, i.e., $d_x = d_y = 2$.

(3) P is basic and has thickness at least 3 in one and at least 2 in the other direction.

(4) P is non-basic.

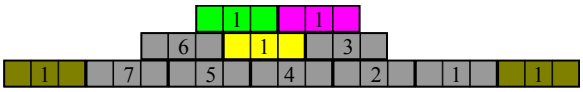
The Lemmas of Section 4 provide us with the basic facilities to build paths of tiles which occupy particular points while avoiding others. By carefully designing the grids and offsets for the tiles of each polyomino P, we are able to construct the constituent paths of the bit-reading gadgets.

Let P be an arbitrary polyomino, with $|P| \geq 3$. Without loss of generality (as the following arguments all hold up to rotation), let the bounding box of P be of dimensions $m \times n$, with $m \geq n$, and d_y be the largest distance between two pixels of P in the same column, and d_x be the largest distance between two pixels of P in the same row. For ease of notation, we refer to the southernmost of all westernmost pixels of P as $p_0 = (0, 0)$, and to all other pixels by their integer coordinates.

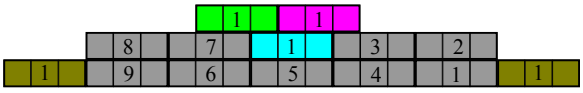
For any polyomino P, we know that tiles of shape P can produce a grid by Lemma 4.2; throughout this section, we simply refer to this as the grid (for P). We also note that the grid for a given P may be slanted as in Figure 5, and that the construction of the zig-zag Turing machine is simply slanted accordingly. If we say that a tile is $\vec{v}$-shifted for some vector $\vec{v}$, we mean that it is off grid by the vector $\vec{v}$.

For all figures in this section, we use the same color conventions as in Figure 1. Thus, the green tiles in this section represent the t tile; as discussed in the caption of the figure, the yellow and blue tiles represent the two potential paths grown from t, while the dark grey tiles represent tiles that prevent the growth of paths from t. We refer to these grey tiles as *blockers*. We use the convention that if a path of yellow polyominoes grows, then a 0 is being read. Similarly, if a path of blue polyominoes grows, then a 1 is being read. Consequently, we call polyominoes that prevent the growth of the blue path 1-*blockers* and polyominoes that prevent the growth of the yellow path 0-*blockers*. In addition, tiles of the same color are numbered in order to indicate the order of their placement where the higher numbered tiles are placed later in the assembly sequence.

5.1 Case (1): P Is an $m \times 1$ Polyomino If P is a straight line, and therefore $n = 1$, we can simply use a scheme as illustrated in Figure 8. In Figure 8a, a 0 bit is read as indicated by the placement of the yellow polyomino. Notice that the tile labeled 3 in Figure 8a prevents the attachment of the blue-colored tile. After the yellow tile attaches, a fuchsia tile attaches as shown in the figure which allows for growth to continue. Figure 8b shows a similar scenario in the case that a 1 is read. The key property of this bit reader is that the yellow and blue tiles have different offsets relative to the green tile, which is always possible if P is a line of length ≥ 3. Note that the bit reader shown in Figure 8 is a left-to-right bit-reading gadget. A right-to-left bit-reading gadget can be constructed in a similar fashion.



(a) A left-to-right bit-reading gadget reading a 0 bit.



(b) A left-to-right bit-reading gadget reading a 1 bit.

Figure 8: The two different bit-reading schemes for an $m \times 1$ polyomino. Note that the bit reader in this figure proceeds from left to right.

The case for any P which is an $n \times 1$ straight line can be handled in the same way. Thus, we now only need to consider the cases $m \geq n \geq 2$. Because P is connected, this implies both $d_x \geq 2$ and $d_y \geq 2$. Furthermore, we assume the that the grid constructed from Lemma 4.2 using P is created by attaching the southernmost pixel on the eastern edge of P to the northernmost pixel on the western side of the P, as suggested by Figure 5.

5.2 Case (2): P Is Such That $d_x = d_y = 2$ Before describing bit-reading constructions, we analyze the possible cases for the shape of P; refer to Figure 9. Also, we note that $d_x = d_y = 2$ implies that P is basic.

First consider the situation in which $|P|$ is even. If both $(1, 0)$ and $(0, 1)$ belong to P, the assumption

$d_x = d_y = 2$ implies that $(1,1)$ must also belong to P, but no further pixels. Thus, P is a 2×2-square, which will be treated as **Case (2a)**. Now, without loss of generality consider the case that $(0,1)$ belongs to P, but $(1,0)$ does not. It follows from $d_x = d_y = 2$ that $(1,1)$ belongs to P, as well as $(1,2)$. This conclusion can be repeated until all pixels of P are allocated. It follows that P is an even zig-zagging shape. This is shown as **Case (2b)** in Figure 9.

Now consider the case in which $|P|$ is odd. If both $(1,0)$ and $(0,1)$ belong to P, $(1,1)$ cannot be part of P, and P is an L-shape consisting of three pixels. If without loss of generality $(0,1)$ belongs to P, but $(1,0)$ does not, we can conclude analogous to (2a) that P is an odd zig-zagging shape, shown as **Case (2c)** in Figure 9; this also comprises the case of an L-shape with three pixels.

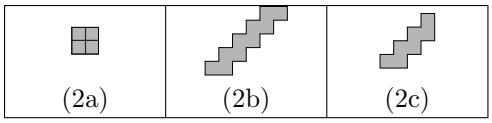
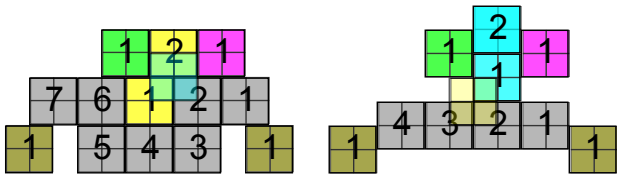


Figure 9: The possible shapes in Case (2), when $d_x = d_y = 2$.

Now we sketch the bit-reading schemes. As these cases are relatively straightforward, we simply refer to the corresponding figures. Note that the logic of the arrangement is color coded: the first polyomino we add to our tile set is the green polyomino along with the blue and yellow polyominoes that allow for an blue tile to attach to the east of it in an on grid position and a yellow tile to attach to the east of it shifted $(-1,-1)$ relative to the polyomino grid.

5.2.1 Case (2a) If P is a 2×2 square we use the scheme shown in Figure 10.



(a) A left-to-right bit reader reading a "0" bit.

(b) A left-to-right bit reader reading a "1" bit.

Figure 10: A general bit-reading scheme for a left-to-right bit reader in case (2a), in which P is a 2×2 square.

5.2.2 Case (2b) If P is an even zig-zagging polyomino, as shown in part (2b) of Figure 9, we use the bit-reading schemes shown in Figure 11.

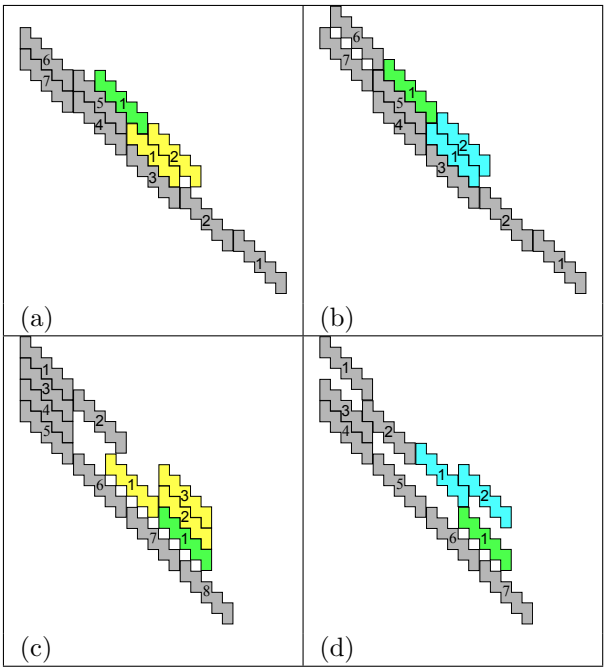


Figure 11: The bit-reading schemes for Case (2b) of Figure 9.

5.2.3 Case (2c) If P is an odd zig-zagging polyomino, as shown in part (2c) of Figure 9, we use the bit-reading schemes shown in Figure 12.

This concludes Case (2).

5.3 Case (3): P Is Basic And Not In Case (1) or (2) We now describe how to construct a system that contains a bit-reading gadget in the case that $\max\{d_x, d_y\} \geq 3$, $\min\{d_x, d_y\} \geq 2$, and P is a basic polyomino. This means that it is possible to construct a path using tiles of shape P which place a tile at any possible offset in relation to the grid. We will use this ability to place blockers and bit-reader paths exactly where we need them, with those locations specified throughout the description of this case. Without loss of generality, assume that $\max\{d_x, d_y\} = d_y$.

Case (3) Overview A schematic diagram showing the growth of the bit-reading gadget system we construct is shown in Figure 13. Note that the figure depicts what the bit-reader would look like if the grid formed by P was a square grid. In cases of a slanted grid (such as that shown in Figure 5), the bit-gadgets would be

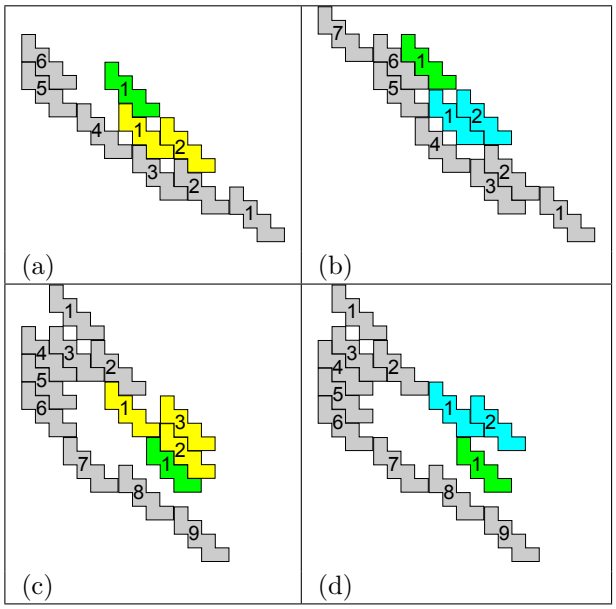


Figure 12: The bit-reading schemes for Case (2c) of Figure 9.

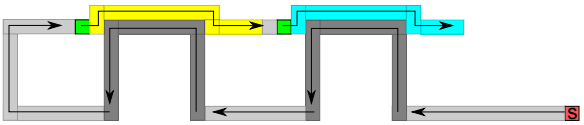


Figure 13: A schematic diagram showing the growth of the bit-reading gadget system for Case (3). Note that in the actual construction, a larger gap would exist between the two bit-readers to allow the path between them to first extend upward and create the necessary bit-writer, then come back down and continue growth of the next bit-reader.

correspondingly slanted. Growth of the system begins with the seed as shown in Figure 13. From the seed, the system grows a path of tiles west (shown as a light grey path in the figure) to which one of the two bit writers attach (shown as dark grey in the figure). Once one of the bit writers assembles, growth proceeds as shown in the schematic view until the other bit writers assemble. Then growth continues upward to the next level (i.e. the seed row can be considered a "zig" row and the next row a "zag" row of the zig-zag Turing machine simulation) as shown in the figure until a green tile is placed. Depending on the bit writer gadget to the east of the green tile either a yellow path of tiles grows, indicating that a 0 has been read (as shown in the schematic view with the westernmost bit writer), or a blue path of tiles grows, indicating that a 1 has been read (as shown in the schematic view with the easternmost bit writer). Henceforth, we refer to the system described by the schematic view in Figure 13 as the bit-reading gadget.

The light grey tiles that compose the bit-reading gadget are easily constructed by placing glues on the polyomino P so that they grow the paths shown in Figure 13 (again, modulo the slant of the particular grid formed by P), which are on grid with the seed, where the grid is formed following the technique used in the proof of Lemma 4.2. The construction of the other tiles is now described. The green tile is constructed by placing a glue on its western side so that it attaches to the grey tiles on grid as shown in the schematic view. Furthermore, glues are placed on the green tile and the first blue tile so that the blue tile attaches to the green tile in an on-grid manner. Glues are placed on the green tile and the first yellow tile so that the southern edge of the southernmost pixel on the east perimeter of the green tile attaches to the northern edge of the northernmost pixel on the western perimeter of the yellow tile (thus putting the yellow tile off grid).

Case (3) Bit-Writer Construction First we describe the construction of the bit-writer subassemblies of the bit-reading gadget by describing the placement of the blockers in relation to the position of the green tile. We will discuss how to create tile sets which can create the necessary sets of paths for the gadgets, and then the final tile set will simply consist of a union of those tile sets. Suppose that the 0-blocker is a $\vec{x_1}$-shifted polyomino and the 1-blocker is a $\vec{x_2}$-shifted polyomino. (Recall that P is a basic polyomino, and thus it is possible to build a path such that a blocker can be at any shift relative to the grid.) We construct two separate systems, say $\mathcal{T}_{0B}$ and $\mathcal{T}_{1B}$ as described in Lemma 4.6 so that the 0-blocker and 1-blocker, respectively, are the northernmost tiles on the western edges of the assemblies (shown in part (a) and (b) of Figure 14). We denote the assemblies produced by these systems as α_0 and α_1, respectively. Next, extend the tile sets of the systems if needed so that the last tiles placed lie on grid in the same grid row as the seed as shown in part (c) and (d) of Figure 14. In addition, extend the tile sets of the two systems (if needed) so that the last tile placed has pixels that lie in the same column as the westernmost pixel in the blocker or to the west of that column. This is shown schematically in part (e) and (f) of the figure. Now, place the green polyomino so that its bounding rectangle's southwest corner lies at the origin, and place the assemblies α_0 and α_1 constructed above so that the 1-blocker and 0-blocker lie relative to the green tile as described above (shown in part (g) of Figure 14). With-

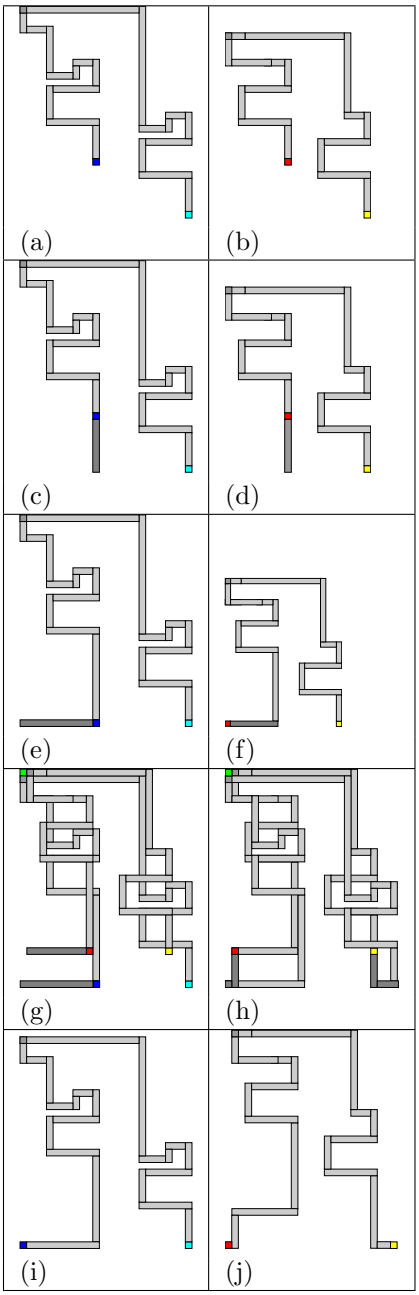
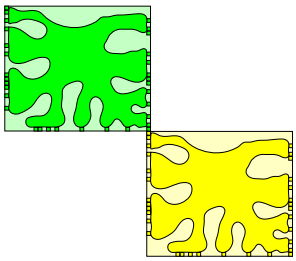
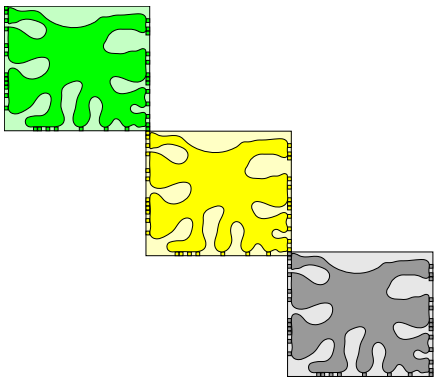


Figure 14: A schematic representation of the construction of the bit writers. The blockers are represented by dark grey squares in the northwest of the assemblies. The assembly which places the 0-blocker begins growth from the blue tile and the assembly which places the 1-blocker begins growth from the yellow tile. The last tile placed in the assembly that contains the 0-blocker is shown in blue and the last tile placed by the assembly containing the 1-blocker is shown in red. Paths of new tiles that are added at each step are dark grey.

out loss of generality suppose that the seed of α_1 lies to the southeast of the seed of α_0 (as is the case in the figure). Then we can extend the tile set of $\mathcal{T}_{0B}$ so that whenever α_1 is placed as described above, the seeds of α_0 and α_1 lie at the same position, since both paths are on grid in those locations. In addition, without loss of generality suppose that the tile placed last in α_1 is further west than the last tile placed in α_0. Then we extend the tile set of $\mathcal{T}_{0B}$ so that the last tile placed in α_0 is at the same position as the last tile placed in α_1. These two steps are shown in part (h) of the figure. The construction of the bit writer gadgets is now complete and the schematic diagram of the completed bit writers is shown in parts (i) and (j) of Figure 14.



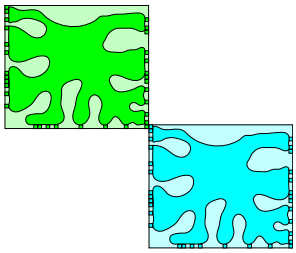
(a) The green tile is on grid and the yellow tile is $(-1,-1)$-shifted.



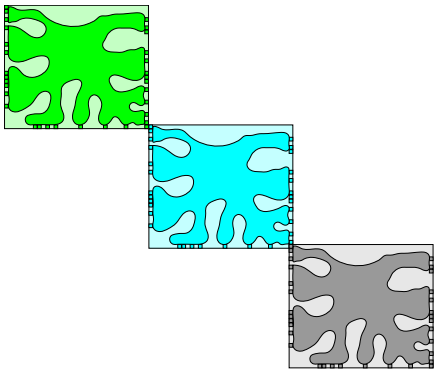
(b) The 0-blocker is placed so that the easternmost pixel on the north perimeter of the 0-blocker overlaps the westernmost pixel on the south perimeter of the yellow tile.

Figure 15: Placement of the 0-blocker, which blocks the yellow (i.e. 0-reader) path.

Case (3) Bit-Reader Construction Figures 15 and 16 show the placement of the 0-blocker and 1-blocker, respectively. Figure 17 shows how the glues are placed on the first and second tiles in the yellow path (in the figure the second yellow tile is shown as an orange tile for clarity) so that the second yellow tile binds to the first yellow tile in the system. In part (a) of Figure 17,



(a) The green tile and blue tile are both on grid.



(b) The 1-blocker is placed so that the northernmost pixel on the western perimeter of the 1-blocker overlaps the southernmost pixel on the east perimeter of the blue tile.

Figure 16: Placement of the 1-blocker, which blocks the blue (i.e. 1-reader) path.

an orange tile (representing the second tile to attach in the yellow path) is placed so that it now lays directly on top of the yellow tile. The d_y pixels which lie in the column with the most pixels are shown as a red column in part (b) of the figure. Notice that when the orange tile is translated by the vector $(1, 0)$ the m red pixels on the yellow tile now lay adjacent to the m red pixels on the orange tile (see part (c)). Now, we shift the orange tile by the vector $(0, 2)$ and make two observations: (1) the bounding rectangle of the orange tile now no longer overlaps the bounding rectangle of the grey tile, (2) the orange tile has a pixel which lies adjacent to a pixel in the yellow tile and/or a pixel which overlaps a pixel in the yellow tile as shown in part (d) of the figure. In the case that the orange tile contains pixels which overlap pixels in the yellow tile, we translate the orange tile to the north until no pixels overlap, but pixels lie adjacent to each in the two tile (shown in part (e) of the figure).

We now have a configuration as shown in part (f) of Figure 17 in which there are not any overlapping pixels and the yellow and orange tiles have pixels which lie adjacent to each other. We can now place glues on the green, yellow and orange tiles so that they assemble as shown with the yellow tile attaching to the green tile and the orange tile attaching to the yellow tile.

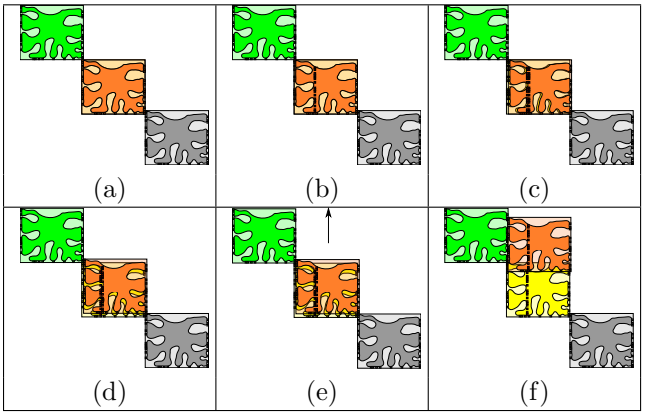
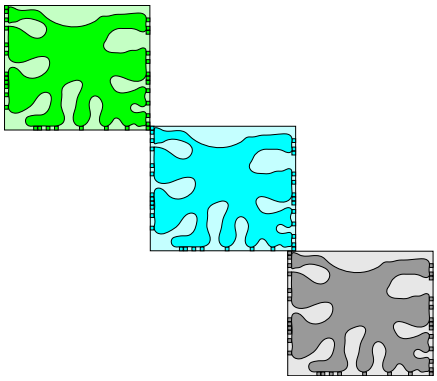


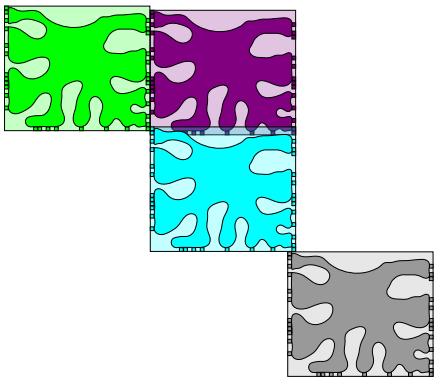
Figure 17: The steps involved in placing the grey blocker and yellow tiles so that an blue tile is prevented from binding to the green tile, but allows for a yellow tile to bind to the green and then continue growth of the yellow path on the grid. The orange tile represents the second tile of the yellow path (to make it easier to distinguish), and figures (a) through (e) show how it can be initially placed immediately on top of the first yellow tile, and then moved into a position which allows for correct binding.

We now describe how glues are placed on the first and second tiles to assemble in the path of blue-colored tiles. Figure 18a shows how the 0-blocker lies in relation to the blue and green tiles. Notice that a tile can attach to the north of the blue tile without overlapping any pixels on other tiles. Thus, the second tile to attach in the blue path is placed to the north of the first blue tile in the path such that it is on the grid. This is shown in Figure 18b where we use a purple tile to represent the second tile in the blue path for clarity. Consequently, we place glues on the first and second tiles to attach in the blue path in a manner such that the second tile in the path binds on grid with respect to the first tile.

Case (3) Right-to-Left Bit-Reading Gadget Construction As the above sections describe how to build the left-to-right bit-reading gadget, we now construct the right-to-left bit-reading gadget by using mirrored versions of the arguments given above with a few small changes. In the left-to-right bit-reading gadget we can always place the yellow tile so that it is a $(-1, -1)$-shifted polyomino. Notice that this is not the case when the bit reader is growing to the west. Thus we make the following changes to the argument above when constructing the right-to-left bit-reading gadget. For convenience, we call the first tile to attach in the blue path



(a) The green tile is on grid and the yellow tile is $(-1,-1)$-shifted.



(b) The 0-blocker is placed so that the westernmost pixel on the north perimeter of the 0-blocker overlaps the easternmost pixel on the south perimeter of the yellow tile.

Figure 18: Placement of the second tile in the blue path (shown as purple to make it easier to distinguish).

t_1 and the first tile to attach in the yellow path t_0. To begin, we attach t_1 to the green tile so that the northernmost pixel on the east perimeter of t_1 attaches to the southernmost pixel on the western perimeter of the green tile via their east/west glues. Say that this places the blue tile so that it is an (x_1, x_2)-shifted polyomino. Note that this means t_1 is not necessarily on grid since as noted above the grid we are using is formed by attaching the southernmost pixel on the east perimeter of P to the northernmost pixel on the western perimeter of P. Now, observe that this implies that we can also attach a (x_1+1, x_2-1)-shifted tile to the green tile (by the points that we used for their attachment at (x_1, x_2)). We thus construct glues so that t_0 attaches to the green tile such that it is a (x_1+1, x_2-1)-shifted polyomino. Now, we can construct the bit-writers as in Section 5.3 with the blockers shifted in the following ways: (1) the 1-blocker is shifted so that when it is placed its northernmost pixel on the east perimeter overlaps the southernmost pixel on the western perimeter of t_1, and (2) the 0-blocker is placed so that its easternmost pixel on its north perimeter overlaps the westernmost pixel on the south perimeter of t_0. We can then use the mirrored version of the construction in section 5.3 to grow the rest of the path of tiles composing the yellow and blue paths.

Case (3) Correctness of the Bit-Reading Gadget

Let us now examine what our constructed system will assemble. Growth will start with the seed and then grow two bit-writer subassemblies consecutively. For concreteness, suppose that α_1 is grown first and then α_0. After α_0 is assembled, a path of tiles will grow upward and over to place a green tile such that the green tile will be placed with its position relative to the grey tile as shown in part (a) of Figure 17. It then follows by the way we placed the green and yellow tiles and the location of α_0 that the yellow path will be able to assemble. This will eventually lead to the placement of the second green tile, which is placed to the west of the second bit writer. The relative placements of that green tile and the blocker of α_1 ensure that a blue path, and only a blue path, will assemble. This concludes the necessary demonstration of the correct growth of a bit-reader gadget. (The full Turing machine simulation also includes bit-writers, designed as previously described, to output between bit-readers and the necessary zig-zag paths.) □

5.4 Case (4): P is non-basic For this case, suppose that P is a non-basic polyomino. Note that we are not making the claim that non-basic polyominoes exist; in fact we conjecture that they do not. However, a proof seems at least as complicated as handling this case with an explicit construction. Details are omitted because of limited space; for details, see the full version of this paper [9].

6 Computationally Limited Systems

In this section, we provide a set of results which suggest that some systems of polyominoes are incapable of universal computation by showing that they are either unable to utilize bit-reading gadgets (which are fundamental features of all known computational tile assembly systems), or that they can be simulated by standard aTAM temperature-1 systems (which are conjectured to be incapable of universal computation), and are thus no more powerful than them.

6.1 Monomino and Domino Systems Cannot Read Bits

THEOREM 6.1. *There exists no temperature 1 monomino system (a.k.a. aTAM temperature-1 system) $\mathcal{T}$ such that a bit-reading gadget exists for $\mathcal{T}$.*

Proof. We prove Theorem 6.1 by contradiction. Therefore, assume that there exists an aTAM system $\mathcal{T} = (T, \sigma, 1)$ such that $\mathcal{T}$ has a bit-reading gadget. (Without loss of generality, assume that the bit-reading gadget reads from right to left and has the same orientation as in Definition 3.1.) Let (t_x, t_y) be the coordinate of the tile t from which the bit-reading paths originate (recall that it is the same coordinate regardless of whether or not a 0 or a 1 is to be read from α_0 or α_1, respectively). By Definition 3.1, it must be the case that if α_0 is the only portion of α in the first quadrant to the left of t, then at least one path can grow from t to eventually place a tile from T_0 at $x = 0$ (without placing a tile below $y = 0$ or to the right of $(t_x - 1)$. We will define the set P_0 as the set of all such paths which can possibly grow. Analogously, we will define the set of paths, P_1, as those which can grow in the presence of α_1 and place a tile of a type in T_1 at $x = 0$. Note that by Definition 3.1, neither P_0 nor P_1 can be empty.

Since all paths in P_0 and P_1 begin growth from t at (t_x, t_y) and must always be to the left of t, at least the first tile of each must be placed in location $(t_x - 1, y)$. We now consider a system where t is placed at (t_x, t_y) and is the only tile in the plane (i.e. neither α_0 nor α_1 exist to potentially block paths), and will inspect all paths in P_0 and P_1 in parallel. If all paths follow exactly the same sequence of locations (i.e. they overlap completely) all the way to the first location where they place a tile at $x = 0$, we will select one that places a tile from T_0 as its first at $x = 0$ and call this path p_0, and one which places a tile from T_1 as its first at $x = 0$ and call it p_1. This situation will then be handled in Case (1) below. In the case where all paths do not occupy the exact same locations, then there must be one or more locations where paths branch. Since all paths begin from the same location, we move along them from t in parallel, one tile at a time, until the first location where some path, or subset of paths, diverge. At this point, we continue following only the path(s) which take the clockwise-most branch. We continue in this manner, taking only clockwise-most branches and discarding other paths, until reaching the location of the first tile at $x = 0$. (Figure 19 shows an example of this process.) We now check to see which type(s) of tiles can be placed there, based on the path(s) which we are still following. We again note that by Definition 3.1, some path must make it this far, and must place a tile of a type either in T_0 or T_1 there. If there is more than one path remaining, since they have all followed exactly the same sequence of locations, we randomly select one and call it p'. If there is only one, call it p'. Without loss of generality, assume that p' can place a tile from T_0 at that location. This puts us in Case (2) below.

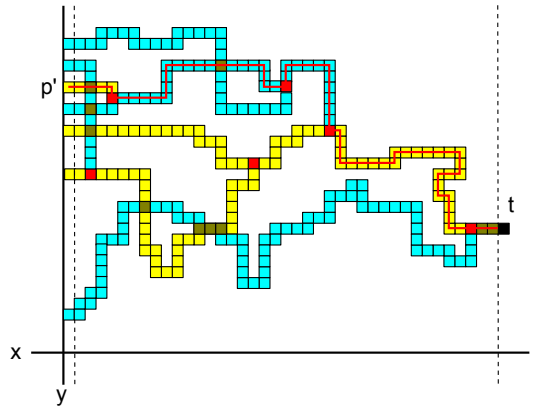


Figure 19: Example sets P_0 and P_1, with p' traced with a red line. Red squares represent branching points of paths, gold squares represent overlapping points of different branches.

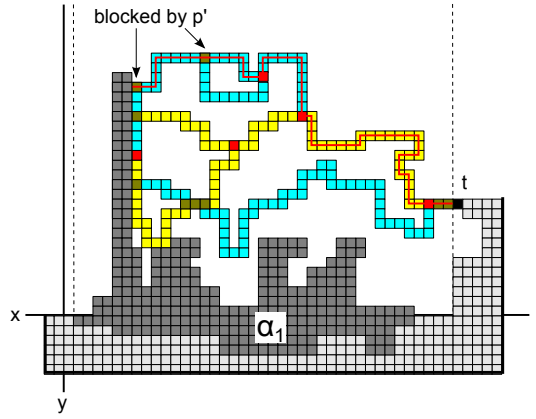


Figure 20: An example of the growth of p' (traced with a red line) blocked by α_1. By first letting as much of p' grow as possible, it is guaranteed that all other paths must be blocked from reaching $x = 0$.

Case (1) Paths p_0 and p_1 occupy the exact same locations through all tile positions and their placement of their first tiles at $x = 0$. Also, there are no other paths which can grow from t, so, since by Definition 3.1 some path must be able to complete growth in the presence of α_0, either must be able to. Therefore, we place α_0 appropriately and select an assembly sequence in which p_1 grows, placing a tile from T_1 as its first at $x = 0$. This is a contradiction, so Case (1) cannot be true.

Case (2) We now consider the scenario where α_1 has been placed as the bit-writer according to Definition 3.1, and with t at (t_x, t_y). Note that path p' must now

always, in any valid assembly sequence, be prevented from growing to $x = 0$ since it places a tile from T_0 at $x = 0$, while some path from T_1 must always succeed. We use the geometry of the paths of T_1 and path p' to analyze possible assembly sequences.

We create a (valid) assembly sequence which attempts to first grow only p' from t (i.e. it places no tiles from any other branch). If p' reaches $x = 0$, then this is not a valid bit-reader and thus a contradiction. Therefore, p' must not be able to reach $x = 0$, and since the only way to stop it is for some location along p' to be already occupied by a tile, then some tile of α_1 must occupy such a location. This means that we can extend our assembly sequence to include the placement of every tile along p' up to the first tile of p' occupied by α_1, and note that by the definition of a connected path of unit square tiles in the grid graph, that means that some tile of p' has a side adjacent to some tile of α_1. At this point, we can allow any paths from P_1 to attempt to grow. However, by our choice of p', as the "outermost" path due to always taking the clockwise-most branches, any path in P_1 (and also any other path in P_0 for that matter) must be surrounded in the plane by p', α_1, and the lines $y = 0$ and $x = t_x$ (which they are not allowed to grow beyond). (An example can be seen in Figure 20.) Therefore, no path from P_1 can grow to a location where $x = 0$ without colliding with a previously placed tile or violating the constraints of Definition 3.1. (This situation is analogous to a prematurely aborted computation which terminates in the middle of computational step.) This is a contradiction that this is a bit-reader, and thus none must exist.

THEOREM 6.2. *There exists no single shape polyomino tile system $\Gamma = (T, \sigma, 1)$ where all tiles of T consist of either two unit squares arranged in a vertical bar, or of two unit squares arranged in a horizontal bar (i.e. vertical or horizontal duples), such that a bit-reading gadget exists for Γ.*

Proof. The proof of Theorem 6.2 is nearly identical to that of Theorem 6.1. Without loss of generality, we prove the impossibility of a bit-reader with horizontally oriented duples. The only differing point to consider between the proof for squares vs. duples is in the analysis of Case (2) where the claim is made that if p' is blocked by α_1, some tile of p' must have a side adjacent to some tile of α_1. In the case of duples, as can be seen in Figure 21, there are also the possibilities that tiles of p' and α_1 are diagonally adjacent or separated by a gap of a single unit square. However, neither of these possibilities can allow for duples of any path in P_1 to pass through, and they therefore remain blocked, thus again proving that no bit-reader must exist.

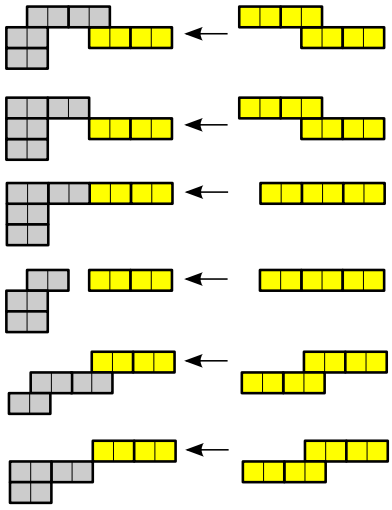


Figure 21: (Right) Partial paths of duples growing from right to left, (Left) portions of those paths being blocked by grey tiles. Any possible way of blocking a path of duples results in either (1) no gap between the path and the blocking assembly, or (2) a single unit square gap, which no path of only horizontally oriented duples can pass through.

It is interesting to note that by the addition of a single extra square to a duple, creating a 3×1 polyomino, it is possible to create gaps between blocking assemblies and blocked paths which allow another path to pass through. This is because the gap can be diagonally displaced from the last tile of the blocked path. An example can be seen in Figure 8.

6.2 Scale-Free Simulation We now provide a definition which captures what it means for one polyomino system to simulate another. This definition is meant to capture a very simple notion of simulation in which the simulating system follows the assembly sequences of the simulated system via a simple mapping of tile types and with no scale factor (as opposed to more complex notions of simulation which allow for scaled simulations such as in [4, 5, 12, 17], for instance).

DEFINITION 6.1. (SCALE-FREE SIMULATION) *A tile system $\mathcal{T} = (T, \sigma, \tau)$ is said to* scale-free simulate *a tile system $\mathcal{T}' = (T', \sigma', \tau')$ if there exists a surjective function $f : T \to T'$ and a bijective function $M : \mathcal{A}[\mathcal{T}] \to \mathcal{A}[\mathcal{T}']$ such that the following properties hold.*

1. *For $A, A' \in \mathcal{A}[\mathcal{T}]$, $A \to_1^{\mathcal{T}} A'$ via the addition of tile $t \in T$ if and only if $M(A) \to_1^{\mathcal{T}'} M(A')$ via the addition of a tile in the preimage $f^{-1}[t]$.*

2. *Any sequence* $(A_1 = \sigma, \ldots, A_k)$ *is an assembly sequence for* $\mathcal{T}$ *if and only if* $(M(\sigma) = \sigma', \ldots, M(A_k))$ *is an assembly sequence for* $\mathcal{T}'$.

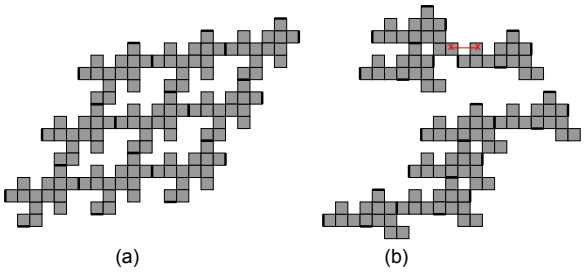


Figure 22: (a) Case (1) considers the scenario in which a 4-position limited polyomino with uniquely paired glues does not have overlapping neighboring positions, or an overlapping diagonal neighbor, and can therefore form a 2D lattice. (b) In case (2), neighbor positions are mutually exclusive. In this case the assembly is linear and is scale-free simulated by a linear monomino tile system with a quadratic increase in tile types to account for the non-determinism introduced by the choice of glue positions.

6.3 Polyominoes with Limited Glue Positions

In this section we analyze the potential of polyomino systems to compute if the number of distinct positions on the polyominoes at which glues may be placed is bounded. We show that any polyomino system which utilizes 3 or fewer distinct glue locations, or a system that uses 4 glue locations but adheres to a "unique pairing" constraint, is scale-free simulated by a temperature 1 aTAM system (Theorems 6.3, 6.4), and is thus very likely to be incapable of universal computation. On the other hand, we show that with only 4 glue positions and no unique pairing restriction, universal computation is possible (Theorem 6.5).

Definition 6.2. (*c*-position limited) *Consider a set* T *of polyomino tiles all of some shape polyomino* P. *Consider the subset* S *of all edges of* P *such that some* $t \in T$ *places a glue label on a side in* S. *We say that* T *has* glue locations S. *If* $c \geq |S|$, *we say that* T *is* c-position limited. *Further, any single shape polyomino system* $\mathcal{T} = (T, \sigma, \tau)$ *is said to be* c*-position limited if* T *is* c*-position limited.*

Definition 6.3. (uniquely paired) *A polyomino system with glue locations* S *is said to be uniquely paired if for each* $s \in S$, *there is a unique* $s' \in S$ *such that glues in position* s *can only bind with glues in position* s'.

Monomino systems, for example, are uniquely paired as the north face glue position only binds with the south face position, and the east position only binds with the west position.

Theorem 6.3. *Any 3-position limited polyomino system* $\Gamma = (T, \sigma, 1)$ *is scale-free simulated by a monomino tile system (a.k.a. a temperature 1 aTAM system).*

Proof. If the system is 2-position limited, a monomino system that replaces each $t \in T$ with a linear east/west glue monomino tile (i.e. a tile which only has glues on its west and east sides) will do the trick. (Note that Lemma 3 of [3] implies that if the glue positions on a polyomino are sufficient to allow it to bind, without overlap, in some position to another copy of itself, then an infinite sequence of copies can bind in a line at the same relative positions to their neighbors.) In the case of a 3-position limited system, the construction described for 4-position limited uniquely paired systems with a quadratic tile complexity increase (see the proof of Theorem 6.4) may be applied.

Theorem 6.4. *Any 4-position limited, uniquely paired polyomino system* $\Gamma = (T, \sigma, 1)$ *is scale-free simulated by a monomino tile system at temperature 1 (a.k.a. a temperature 1 aTAM system).*

Proof. Consider some 4-position limited, uniquely paired system $\Gamma = (T, \sigma, 1)$, and denote the shape of the tiles in Γ as polyomino P. Let $\vec{v}$ denote the translation difference between two bonded polyominoes from T that are bonded with the first pair of glue positions, and let $\vec{u}$ denote the translation for the second pair of bonding positions. As an example, consider the polyomino of Figure 22(a). The north-south glue positions are separated by vector $\vec{u} = (3, 5)$, and the east-west glue positions are separated by vector $\vec{v} = (6, 1)$. To show that Γ is simulated by some monomino system, we consider two cases. For polyomino P, let $P_{\vec{x}}$ denote the polyomino obtained by translating P by some vector $\vec{x}$. For case 1, we assume P, $P_{\vec{v}}$, $P_{\vec{u}}$, and $P_{\vec{v}+\vec{u}}$ are mutually non-overlapping. For case 2, we assume that either $P_{\vec{v}}$ and $P_{\vec{u}}$ overlap, or that $P_{\vec{v}}$ and $P_{-\vec{u}}$ overlap (note that P overlaps $P_{\vec{u}+\vec{v}}$ if and only if $P_{\vec{v}}$ overlaps $P_{-\vec{u}}$, and thus is covered by case 2.) The two cases are depicted in Figure 22.

Case 1: In this scenario, the tiles of Γ grow in a 2D lattice with basis vectors $\vec{u}$ and $\vec{v}$ and can be simulated by a monomino system that simply creates a square monomino tile for each element of $t \in T$, placing the glue types of the first pair of uniquely paired glues of t on the north and south edges of the representing monomino, and the other pair on the east and west edges. The bijective mapping that satisfies the

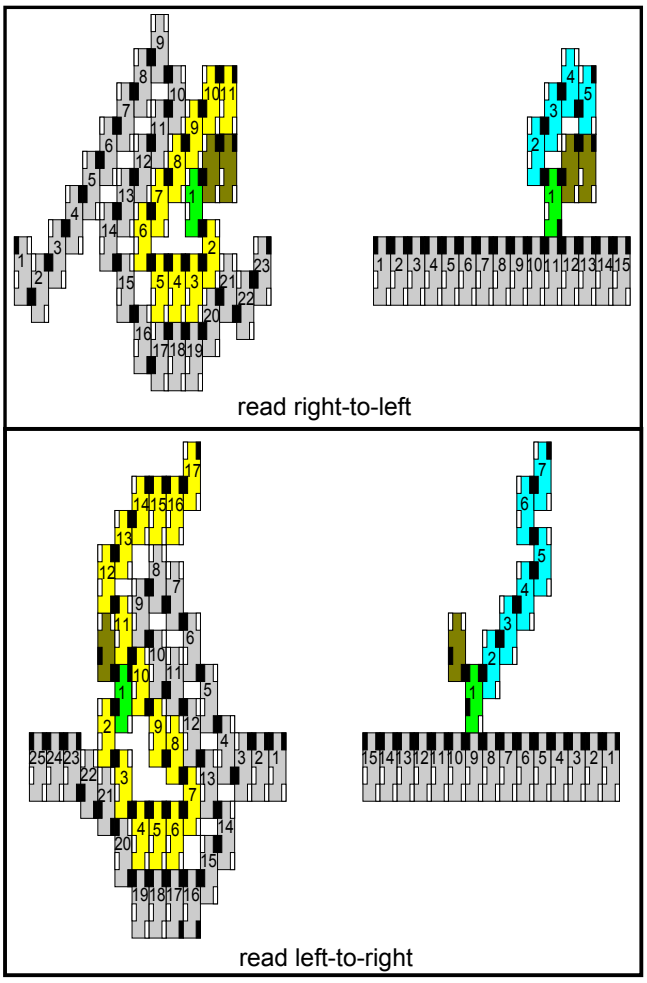


Figure 23: The following bit-reading gadget demonstrates how a 1×4 polyomino that uses only 4 distinct glue positions can perform universal computation. Due to the asymmetry of the glue locations on the east and west edges of the polyomino, the right-to-left and left-to-right bit reading gadgets are also asymmetric. Geometric bit reading is not possible with fewer than 4 glue positions, and is not possible with only 4 positions and unique pairing of positions.

scale-free simulation requirement simply replaces each t in a producible assembly of Γ with the corresponding monomino for that t, thereby yielding an appropriate assembly over the unit square tiles.

Case 2: Without loss of generality, assume it is the case that $P_{\vec{v}}$ and $P_{\vec{u}}$ overlap. We first observe that growth in this scenario is linear (see Figure 22). We will simulate Γ with a linear east/west monomino tile system (i.e. a system where tiles have glues only on their west and east sides). However, unlike in case 1, a simple tile for tile replacement is not sufficient as this does not allow for the simulation of the potential non-deterministic placement of one of two mutually overlapping tiles. To deal with this, we increase the tile set size by a quadratic factor. Let $\vec{u}$ and $\vec{v}$ be defined as before for case 1. As a first subcase, assume that the system we are interested in simulating is such that no polyomino is attachable to the seed at translation $-\vec{u}$ or $-\vec{v}$ from the seed, i.e., tiles only attach at positive linear combinations of $\vec{u}$ and $\vec{v}$. With this restriction, the 4 glue positions of the tiles can be thought of as 2 *input* positions and 2 *output* positions, where a tile always attaches based on the binding of a glue on an input side. Let g_{in1} and g_{in2} represent the two input positions, and g_{out1} and g_{out2} the two output positions, where g_{in1} is uniquely paired with g_{out1}, and g_{in2} with g_{out2}. Then, for a tile $t \in T$ and $d \in \{in1, in2, out1, out2\}$, let $g_d(t)$ be the glue label at location d on tile t. To simulate Γ, we generate a set T' of at most $2|G|$ east/west monomino tiles (where G is the set of all glue labels of the polyomino tile set T), and we will specify the east glue position as g_e and the west as g_w for tiles in T' (and we'll treat g_w as the input and g_e as the output sides). For $0 \leq i < |T|$, we generate a set T'_i of tiles from tile $t_i \in T$ as follows. Let $a = g_{in1}(t_i)$ and $b = g_{in2}(t_i)$. For every tile $t' \in T$ such that $g_{out1}(t') = a$, we generate tile $t'' \in T'_i$ such that $g_w(t'') = a \cdot g_{out2}(t')$ (where $a \cdot g_{out2}(t')$ is just the concatenation of the labels of glues a and $g_{out2}(t')$), and $g_e(t'') = g_{out1}(t_i) \cdot g_{out2}(t_i)$. Similarly, for every every tile $t' \in T$ such that $g_{out2}(t') = b$, we generate tile $t'' \in T'_i$ such that $g_w(t'') = g_{out1}(t') \cdot b$, and $g_e(t'') = g_{out1}(t_i) \cdot g_{out2}(t_i)$. Essentially, whenever a tile from T' is placed, it presents on its output side both of the output glues of the tile from T that it was designed to simulate. (This is because in Γ, either of those output glues could be used as an input glue to bind the next tile, but only one of them.) Therefore, any tile wishing to use one of those output glues as an input glue must now have a glue label which matches the concatenation of that glue and any other which may have been paired with it as an output. In such a way, we generate each set T'_i to simulate $t_i \in T$, and $T' = \bigcup_{i=0}^{<|T|} T'_i$. Therefore, for our scale-free simulator $\Gamma' = (T', \sigma', 1)$ the function

f defined for the scale-free simulation simply maps each tile in $T'_i \subseteq T'$ to its corresponding $t_i \in T$ and σ' is simply formed by corresponding tiles between T and T'.

The bijection from producible assemblies $\mathcal{A}[\Gamma]$ and those in $\mathcal{A}[\Gamma']$ is defined as follows. Consider some assembly $A \in \mathcal{A}[\Gamma]$. Starting from the seed, for each tile attaching in sequence, attach a corresponding tile in T' to the seed of Γ'. In particular, if tile t with input glues a and b attaches via glue a in Γ, then attach a tile from $t' \in T'$ with glue label "$a \cdot i$" where i matches the unused output glue label of the tile that t attached to, and the output glue of t' is the concatenation of the output glues of t. Similarly define the simulator tile attachment in the case that glue b is the bonding glue. By repeating this process, an element of $\mathcal{A}[\Gamma']$ is generated for any element of $\mathcal{A}[\Gamma]$, and this mapping is bijective and thus provides a scale-free simulation of the input system Γ with at most quadratic increase in tile complexity.

Finally, it is easy to see that the restriction that the seed only grows in direction $\vec{u}$ or $\vec{v}$ (and not $-\vec{u}$ or $-\vec{v}$) can easily be removed by taking a more general system and first doubling it's tile set so that each tile type only every attaches in a $\vec{u}/\vec{v}$ or $-\vec{u}/-\vec{v}$ direction. The simulation system is then constructed with two symmetric applications of the previous construction.

THEOREM 6.5. *There exist 4-position limited polyTAM systems that are computationally universal at temperature 1.*

Proof. We prove this by providing constructions for right-to-left and left-to-right bit reading gadgets for the 1×4 polyomino that uses only 4 unique glues positions. The details of the bit reading gadgets are presented in Figure 23. It is straightforward to connect the bit reading gadgets in a fashion similar to previous results to construct a zig-zag Turing machine simulation.

7 Multiple Polyomino Systems

In previous sections we have focussed on the computational power of systems consisting of singly shaped polyominoes and showed that single polyomino systems are universal for polyominoes of size ≥ 3, while monomino and domino systems are likely not capable of such computation. We now show that any multiple shape polyomino system (i.e. one that utilizes at least 2 distinct polyomino shapes, regardless of their size) is capable of universal computation.

LEMMA 7.1. *For every standard Turing Machine M and input w, there exists a TAS with $\tau = 1$ consisting only of tiles shaped as dominoes, 2×1 and 1×2 polyominoes, that simulates M on w.*

Proof. To see this we provide a sketch of a right-to-left bit reading gadget in Figure 24. A left-to-right gadget can be derived similarly, and together these gadgets can be used to construct a zig-zag Turing machine simulation in the same manner as with previous constructions in this paper.

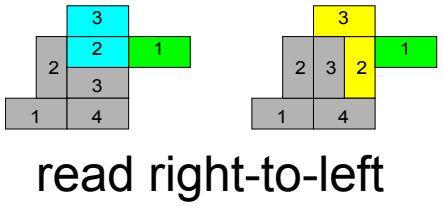


Figure 24: A 2-shape system consisting of the two distinct domino polyominoes can be designed to read bits for the simulation of a zig-zag Turing machine.

THEOREM 7.1. *For every standard Turing Machine M and input w, and any 2 distinct polyominoes P and Q, there exists a TAS with $\tau = 1$ consisting only of tiles shaped as P or Q that simulates M on w.*

Proof. If either P or Q are size 3 or larger, we get the result from theorem 5.1. If one polyomino is a monomino and the other a domino, we get the result from the paper [12]. Finally, if P and Q are the two distinct domino shapes, then we get the result by Lemma 7.1.

Acknowledgements

This work was initiated at the 29th Bellairs Winter Workshop on Computational Geometry on March 21-28, 2014 in Holetown, Barbados. We thank the other participants of that workshop for a fruitful and collaborative environment. We especially thank Erik Demaine, Scott Summers, Damien Woods, Andrew Winslow, and Dave Doty for helpful discussions and many of the initial ideas.

References

[1] Robert D. Barish, Rebecca Schulman, Paul W. Rothemund, and Erik Winfree, *An information-bearing seed for nucleating algorithmic self-assembly*, Proceedings of the National Academy of Sciences **106** (2009), no. 15, 6054–6059.

[2] Matthew Cook, Yunhui Fu, and Robert T. Schweller, *Temperature 1 self-assembly: Deterministic assembly in 3D and probabilistic assembly in 2D*, Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2011, pp. 570–589.

[3] Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Matthew J. Patitz, Robert T. Schweller, Andrew Winslow, and Damien Woods, *One tile to rule them all: Simulating any tile assembly system with a single universal tile*, Proc 41st International Colloquium on Automata, Languages, and Programming (ICALP), LNCS, vol. 8572, Springer, 2014, pp. 368–379.

[4] Erik D. Demaine, Matthew J. Patitz, Trent A. Rogers, Robert T. Schweller, Scott M. Summers, and Damien Woods, *The two-handed assembly model is not intrinsically universal*, Proc. 40th International Colloquium on Automata, Languages and Programming (ICALP), LNCS, vol. 7965, Springer, 2013, pp. 400–412.

[5] David Doty, Jack H. Lutz, Matthew J. Patitz, Robert T. Schweller, Scott M. Summers, and Damien Woods, *The tile assembly model is intrinsically universal*, Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2012, pp. 302–310.

[6] David Doty, Matthew J. Patitz, Dustin Reishus, Robert T. Schweller, and Scott M. Summers, *Strong fault-tolerance for self-assembly with fuzzy temperature*, Proc. 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2010, pp. 417–426.

[7] David Doty, Matthew J. Patitz, and Scott M. Summers, *Limitations of self-assembly at temperature 1*, Theoretical Computer Science **412** (2011), 145–158.

[8] Constantine Glen Evans, *Crystals that count! Physical principles and experimental investigations of dna tile self-assembly*, Ph.D. thesis, California Institute of Technology, 2014.

[9] Sándor P. Fekete, Jacob Hendricks, Matthew J. Patitz, Trent A. Rogers, and Robert T. Schweller, *Universal computation with arbitrary polyomino tiles in non-cooperative self-assembly*, CoRR **abs/1408.3351** (2014), http://arxiv.org/abs/1408.3351.

[10] Bin Fu, Matthew J. Patitz, Robert T. Schweller, and Robert Sheline, *Self-assembly with geometric tiles*, Proc. 39th International Colloquium on Automata, Languages and Programming (ICALP), LNCS, Springer, 2012, pp. 714–725.

[11] Jacob Hendricks, Matthew J. Patitz, and Trent A. Rogers, *Doubles and negatives are positive (in self-assembly)*, Proc. Unconventional Computation and Natural Computation (UCNC), LNCS, vol. 8553, Springer, 2014, pp. 190–202.

[12] Jacob Hendricks, Matthew J. Patitz, Trent A. Rogers, and Scott M. Summers, *The power of duples (in self-assembly): It's not so hip to be square*, Proc. 20th Int. Comp. and Combinatorics Conference (COCOON), LNCS, vol. 8591, Springer, 2014, pp. 215–226.

[13] T.H. LaBean, E. Winfree, and J.H. Reif, *Experimental progress in computation by self-assembly of DNA tilings*, DNA Based Computers **5** (1999), 123–140.

[14] James I. Lathrop, Jack H. Lutz, Matthew J. Patitz, and Scott M. Summers, *Computability and complexity in self-assembly*, Theory Comput. Syst. **48** (2011), no. 3, 617–647.

[15] Chengde Mao, Thomas H. LaBean, John H. Relf, and Nadrian C. Seeman, *Logical computation using algorithmic self-assembly of DNA triple-crossover molecules.*, Nature **407** (2000), no. 6803, 493–6.

[16] Ján Maňuch, Ladislav Stacho, and Christine Stoll, *Two lower bounds for self-assemblies at temperature 1*, Journal of Computational Biology **17** (2010), no. 6, 841–852.

[17] Pierre-Etienne Meunier, Matthew J. Patitz, Scott M. Summers, Guillaume Theyssier, Andrew Winslow, and Damien Woods, *Intrinsic universality in tile self-assembly requires cooperation*, Proc. Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014), 2014, pp. 752–771.

[18] Jennifer E. Padilla, Matthew J. Patitz, Raul Pena, Robert T. Schweller, Nadrian C. Seeman, Robert Sheline, Scott M. Summers, and Xingsi Zhong, *Asynchronous signal passing for tile self-assembly: Fuel efficient computation and efficient assembly of shapes*, Proc. Unconventional Computation and Natural Computation (UCNC), 2013, pp. 174–185.

[19] Matthew J. Patitz, Robert T. Schweller, and Scott M. Summers, *Exact shapes and turing universality at temperature 1 with a single negative glue*, Proc. 17th Int. Conference on DNA Computing and Molecular Programming (DNA), LNCS, vol. 6937, Springer, 2011, pp. 175–189.

[20] Matthew J. Patitz and Scott M. Summers, *Self-assembly of discrete self-similar fractals*, Natural Computing **1** (2010), 135–172.

[21] Matthew J. Patitz and Scott M. Summers, *Self-assembly of decidable sets*, Natural Computing **10** (2011), no. 2, 853–877.

[22] Paul W. K Rothemund, Nick Papadakis, and Erik Winfree, *Algorithmic self-assembly of dna sierpinski triangles*, PLoS Biol **2** (2004), no. 12, e424.

[23] Paul W. K. Rothemund and Erik Winfree, *The program-size complexity of self-assembled squares (extended abstract)*, Proc. Thirty-Second Annual ACM Symposium on Theory of Computing (STOC), 2000, pp. 459–468.

[24] Rebecca Schulman and Erik Winfree, *Synthesis of crystals with a programmable kinetic barrier to nucleation*, Proc. National Academy of Sciences **104** (2007), no. 39, 15236–15241.

[25] David Soloveichik and Erik Winfree, *Complexity of self-assembled shapes*, SIAM Journal on Computing **36** (2007), no. 6, 1544–1569.

[26] Erik Winfree, *Algorithmic self-assembly of DNA*, Ph.D. thesis, California Institute of Technology, June 1998.