An Axiomatic Study of Leveraging Blockers to Self-Assemble Arbitrary Shapes via Temperature Programming

Matthew J. Patitz
University of Arkansas, Fayetteville, AR, USA

Trent A. Rogers
University of Arkansas, Fayetteville, AR, USA

Abstract

In this paper we present a theoretical design for tile-based self-assembling systems where individual tile sets that are universal for various tasks (e.g. building shapes or encoding arbitrary data for algorithmic systems) can be provided their input solely using sequences of variations in temperatures. Although there is prior theoretical work in so-called "temperature programming," the results presented here are based upon recent experimental work with "blocked tiles" that provides plausible evidence for their practical implementation. We develop and present an abstract mathematical model, the BlockTAM, for such systems that is based upon the experimental work, and provide constructions within it for universal number encoding systems and a universal shape building construction. These results mirror previous results in temperature programming, covering the two ends of the spectrum that seek to balance the scale factors of assemblies with the number of temperature phases required, while leveraging the features of the BlockTAM that we hope provide a pathway for future experimental implementations.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases DNA tiles, self-assembly, temperature programming

Digital Object Identifier 10.4230/LIPIcs.DNA.31.8

Supplementary Material *Software*: `http://self-assembly.net/software/BlockTAM_examples/BlockTAM_examples-source-2025-04-23-14-42-34.zip`
Software: `http://self-assembly.net/software/WebTAS_beta/WebTAS_beta-latest/` [8]

Funding *Matthew J. Patitz*: This author's work was supported in part by NSF grant 2329908.

1 Introduction

One of the central themes of technological advancement is mastering the manipulation of matter. In particular, humans have been fascinated with the construction of arbitrary structures since the beginning of human history. At times, this fascination has been due to the fact that survival has depended on a sufficient mastery of crafting structures while at other times this fascination has had more recreational roots. Over the millennia, humans have mastered building macroscale structures, but the idea of constructing nanoscale structures was unfathomable until relatively recently. Even though the manufacture of arbitrary nanoscale shapes may be a new endeavor for humanity, we are already seeing it begin to take a central role in the development of our future. Nanoscale structures are already proving to be useful both in terms of aiding in humanity's survival and recreational pursuits as nanoscale structures have seen a wide range of uses including molecular computers [24], drug delivery [22], and lithographically-patterned electronic/photonic devices [6].

Observing and influencing matter at the nanoscale presents a seemingly insurmountable challenge since the structures at this scale are smaller than the wavelengths of visible light. How can we build a structure that we cannot see using components whose location we cannot control? As opposed to the traditional macroscopic approach, self-assembly uses a bottom up approach: instead of attempting to mechanically manipulate components to place them in certain relative locations, chemically manipulate components so that they are prone to stick together in specific locations relative to one another. Unfortunately, this general approach is still quite abstract and difficult to leverage, with successful physical realizations remaining elusive. In 1981, Ned Seeman proposed leveraging the programmability and specificity of deoxyribonucleic acid (DNA) to implement such self-assembling systems, and he constructed the first intentionally designed DNA nanostructure [3, 19].

Building on the work of Ned Seeman, Erik Winfree introduced and physically realized a framework to make the design of DNA nanostructures more tractable (and hence, less error prone) [23]. This framework uses designed DNA strands to assemble molecules that behave like square "tiles" with programmable "glues" that allow a tile to bind to other particular tiles. This approach is known as *tile-based DNA self-assembly.* Peng Yin et al. [9] leveraged this approach to develop a technology capable of assembling arbitrary nanoscale structures (where the resolution is limited by the size of the molecule that acts as a tile). William Shih et al. [12] further refined this approach (albeit with non-planar, non-square tiles) to eliminate technical hurdles which resulted in less than desirable error rates.

In 2006, Paul Rothemund introduced a new paradigm for the assembly of arbitrary nanostructures from DNA [17]. Like Winfree's framework, Rothemund's framework constrains the problem of designing DNA nanostructures by placing it into a logical framework to make the problem more tractable. This framework is known as DNA origami. The idea is to create the target structure by folding a very long DNA strand, called the scaffold strand, into the target shape through the use of small staple strands which bind to multiple positions along the scaffold. These staple strands pinch, or collocate, certain parts of the scaffold strand together and the net effect of this is to fold the scaffold strand into the target structure.

Regardless of which ubiquitous method listed above is used, each significantly different target structure requires a unique set of DNA strands to assemble it. Designing, synthesizing, ordering, and preparing these DNA strands for experiments is costly in terms of time, money and labor. This process also introduces the possibility of human errors and unintended sequence interactions. One way to reduce this burden of requiring new DNA strands is to reduce the number of new strands needed by as much as possible. Soleveichik and Winfree showed that in an axiomatic model of tile-based self-assembly, a target structure can be assembled using (nearly) information theoretic optimal number of strands [20]. While the result of Soleveichik and Winfree greatly reduces (theoretically) the number of strands that needed to assemble a new target structure, it does not completely eliminate it.

Is it possible to design a fixed set of DNA strands that can assemble any arbitrary shape without the need for pipetting or any other manual intervention steps? In the standard axiomatic model of tile-based self-assembly, it is not possible to eliminate the need for new DNA strands for assembling new target structures (since this is how input is passed into the system specifying what structure it should assemble). But, in [21], Summers showed (theoretically) that a fixed set of tiles could assemble any arbitrary 2D shape by controlling the growth of the assembly using controlled temperature fluctuations. This was shown in the multiple temperature model originally introduced in [1]. Unfortunately, this model makes a couple of assumptions which have yet to be demonstrated in the laboratory setting, namely that temperature fluctuations do not lead to tiles aggregating off of the seed assembly and that the binding of one tile to an assembly can prevent the binding of another tile to the assembly. Fortunately, recent experimental work has shown temperature programming may be feasible in the laboratory setting through the use of blockers. This is discussed in the next section.

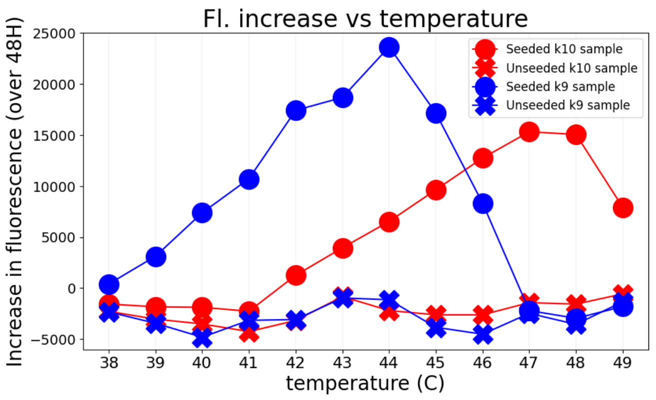
Experimental Motivation

As mentioned above, previous theoretical results on temperature programming relied on two yet to be demonstrated mechanisms: 1) suppressing interactions of sticky glues off-seed at lower temperatures and 2) using the existence of a tile in the assembly to prevent the binding of another tile.

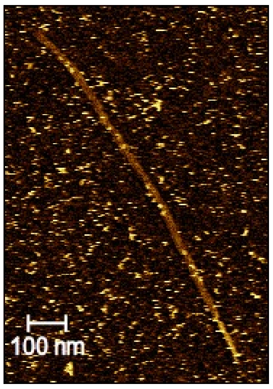
Recent experimental work [16] has shown that by "poisoning" systems with single glue domain length strands, called *blockers*, which are complementary to "input" glues, that one can not only "turn off" high strength (that is, very sticky) glues at lower temperatures, but also suppress spurious nucleation (that is, off-lattice interactions that lead to structures not grown from a seed) across all temperatures. This is thought to be due to the binding penalty induced by blockers being multiplicative for pathways that lead to spurious nucleation (which leads to an exponential impact on the spurious nucleation rate) while having a less significant impact on the growth rate of seeded assemblies since the penalty will result in linear slowdown [5]. Figure 1a, constructed from data in [15], shows this effect in practice. In the presence of a seed, growth may be slowed, but still proceeds at a reasonable rate (as indicated by the increase in fluorescence in samples with seeds), while spurious nucleation is suppressed to undetectable levels across all temperatures (as indicated by the relatively flat fluorescence trace of unseeded samples). Also, shown in this plot is the deactivation effect blockers have on stickier (that is, high-strength) glues at lower temperatures. It shows the growth rate of tiles consisting of length 10 glues becomes suppressed to undetectable levels at lower temperatures (evidenced by the insignificant increase in the seeded sample containing tiles with length 10 glues at temperatures $41C$ and below) where the growth rate of tiles consisting of length 9 glues is still significant (evidenced by the significant increase in the seeded samples at these temperatures).

The Blocked Tile Assembly Model is inspired by the glue deactivation and spurious nucleation suppression afforded by blockers. But, it still remains in the realm of theory since attempting to implement such system in the laboratory could face technical hurdles such as spurious nucleation due to the oscillation of temperature or strain at the interfaces of tiles with different glue lengths preventing further growth. In [15], temperature programming was physically realized. The authors showed that the key mechanisms required to physically realize temperature programming are attainable, and they used these features to program nanotube length using temperature oscillations. When interpreted through the lens of the Blocked Tile Assembly Model, the construction in [15] is simple. Two sets of tiles which assemble a cross-section of a nanotube were designed: one having length 10 glues (which are the high strength glues) and the other having length 9 glues (which are the low strength glues). These tile sets were designed to form alternating cross-sections of the nanotube. Blockers were added to the input side of the length 10 glues preventing those tiles from attaching at a lower temperature where the length 9 glues are active. On the other hand, the tiles with length 9 glues are unable to attach at warmer temperatures were the length 10 glues are active. The authors leverage the exclusivity of growth temperatures to design a system which requires the temperature to be oscillated in order to grow the next segment of the tube. In this manner, the number of oscillations dictates the length of the nanotube. (See Figure 2 for an example system which illustrates this idea.)

Figure 1b presents atomic force microscopy (AFM) data from [15]) which shows a nanotube grown using the temperature programming technique described above. The segments composed of tiles with length 9 glues were marked with a special molecule so that they appear with bright dots on the image. The temperature was oscillated 4 times (that is, it was held a high temperature and then a low temperature and this process was repeated 4



(a) Bulk fluorescence data taken from [15] providing evidence that blockers suppress spurious nucleation and deactivate stickier glues at lower temperatures. Each seeded sample is intended to grow a 12 helix nanotube and the unseeded samples consist of the tiles used in the seeded sample but do not contain the seed. A pair of molecules were added to two distinct tiles in each sample, so that the fluorescence signal increases when they are collocated indicating a nanotube is growing. Four samples were held at 12 temperatures for 48 hours and their fluorescence increase over that time is recorded on the plot. The samples labeled k10 have tiles with length 10 glues and the samples labeled k9 have tiles with length 9 glues. The relatively flat lines with x markers (which are the unseeded samples) indicate the fluorescence increase in unseeded samples is undetectable which we interpret to mean that the spurious nucleation rate is very low. Note that the negative fluorescence values are thought to be due to photo-bleaching. The seeded samples (the data points with circle markers) see significant fluorescence increase over the hold indicating that growth is occurring.



(b) A visualization of atomic force microscopy (AFM) data taken from [15]. The system used to grow this nanotube via a temperature programming technique was designed to grow the nanotube using alternating segments of tiles with length 10 glues and tiles with length 9 glues. The segment grown from tiles with length 9 glues was marked with a special molecule so that bright dots appear on the segment. This image shows a nanotube resulting from a system that underwent four temperature oscillations, so we would expect there to be four segments consisting of tiles with length 10 glues (unmarked) interleaved with four segments consisting of tiles with length 9 glues (marked).

Figure 1 Evidence that the key mechanisms assumed by the Blocked Tile Assembly Model are experimentally feasible.

times) so that we would expect to see 8 cross-section segments alternating between being unmarked and marked. Note that this is indeed what we see in the image. We see the seed, the bright tube in the bottom right-hand corner of the image, followed by eight alternating unmarked and marked segments. This provides evidence that the system is working as intended.

Our contributions

We first formally define the Blocked Tile Assembly Model that presents a formalization of the intuitive model used to design the system presented in [15], which is capable of using temperature to program tube length. A natural extension of the work presented in [15] is to examine whether we can leverage the mechanisms used in that paper to use temperature to program not just length, but the shape of the structure. This is akin to the idea of a 3D printer (although we work in 2 dimensions) which uses a universal ink to construct arbitrary structures. That is, can we assemble a target nanostructure purely through temperature fluctuations without the need to design, order, or add new strands?

We answer this question affirmatively through our two main results. The constructions we present rely on the same mechanisms already demonstrated in [15] with two exceptions. Our constructions rely on having three or more tile sets which have disjoint growth temperatures.

In other words, our constructions would require something like length 8 glues in addition to length 9 and length 10 glues. Another yet to be experimentally demonstrated mechanism is having mismatches in the glues between neighboring tiles. However, this mechanism has been demonstrated experimentally, albeit unintentionally, with other DNA tile motifs through the occurrence of errors in algorithmic self-assembly [24]. Neither of these technical challenges appear insurmountable which provides us hope that these constructions, or at least somewhat simplified versions of them, may be experimentally feasible.

Our first main result provides a way for a single BlockTAM tile and blocker set, always beginning from the same seed tile, to receive, and then encode, an arbitrary input value into the glues of a row of tiles solely via a sequence of temperature changes. We call this a *universal number encoder*, and such a system could be combined with an aTAM system designed to perform any algorithmic self-assembly task by first reading that information as its input and then carrying out its algorithmic growth. Dozens of powerful examples of algorithmic self-assembling systems have been theoretically demonstrated, e.g. [2, 10, 11, 14, 18, 20], and one in particular worth mentioning is that of [20] in which such an input sequence could be the information-theoretically optimal encoding of the definition of an arbitrary shape so that the assembly then proceeds to grow into a scaled version of that shape.

This leads naturally to our second main result, which we call a *universal shape builder*, and is a fixed tile and blocker set such that, given any arbitrary shape S, a system that always begins from the same seed tile can be controlled by a temperature sequence taken from the set of temperatures $\{2, 3, 4\}$ that is specific to S so that the resulting terminal assembly is in the shape of S at a constant scale factor of 6.

As an input module to the construction of [20], our universal number encoder is also a version of a universal shape builder, and although it is optimal in terms of the number of temperature phases required to build an arbitrary shape, the scale factor of the shape is extremely large.[1] However, while our second construction requires a number of temperature phases that is linear in the number of points in the target shape, its scale factor is a mere 6. In combination, these results provide evidence of the broad powers and potential of the BlockTAM and inspiration for both further theoretical and experimental implementations.

Both results have been fully designed, implemented, and simulated. Python code that can be used to generate example systems for both results (along with pre-existing examples) [13] and a web-based simulator [8] can be accessed from: `http://self-assembly.net/wiki/index.php/Blocked_Tile_Assembly_Model_(BlockTAM)`.

2 Preliminary Definitions and Models

2.1 The Blocked Tile Assembly Model Intuition

We begin by providing a high-level overview of the aTAM and BlockTAM so that readers familiar with the aTAM may skip the formal definitions section.

The abstract Tile Assembly Model is a tile-based model of self-assembly where growth proceeds from an initial assembly called the seed and proceeds asynchronously and nondeterministically via single-tile attachments. The systems which make up the abstract Tile Assembly Model (aTAM) are called tile assembly systems (TAS) and they are triples (T, σ, τ) where T is the set of tile types, σ is the seed assembly, and τ is the temperature, or minimum

[1] The scale factor would depend on the particular shape, but would likely be on the order of at least tens of thousands.

binding threshold, of the system. Growth of the system begins from the seed σ and grows via single tile attachments (where the tiles come from the tile set T) provided that each tile binds to the assembly with at least a combined strength of τ.

The Blocked Tile Assembly Model (BlockTAM) is an adaptation of the Multiple Temperature Model introduced in [4]. To formally define this model, we first define a 1 Phase Blocked Tile Assembly Model (1BlockTAM). The BlockTAM can be thought of as consisting of a sequence of 1BlockTAM systems where the resulting terminal assembly of the i^{th} system is fed into the $(i+1)^{\text{th}}$ system as a seed.

A system in the 1 Phase Blocked Tile Assembly Model (1BlockTAM) is a 4-tuple (T, C, σ, τ) where T, σ and τ are defined as in the aTAM, and C is defined to be a set of "blockers". A blocker is defined as a triple (t, d, s) where t is a tile, d is a direction on the tile, and s is a strength. Growth of the system proceeds as in the aTAM with the exception that any glue which has a blocker whose strength is $\geq \tau$ is treated as a strength 0 glue (to reflect the fact that the glue is "blocked"). We formalize this by constructing an aTAM system from the 1BlockTAM system where any glues that are "blocked" in the system have their strength set to 0 and defining the behavior of the 1BlockTAM system to be the same as the derived aTAM system.

2.2 Abstract Tile Assembly Model

The definitions below are an adaptation of the abstract Tile Assembly Model (aTAM) definition presented in [7]. We note that [18] and [11] are good introductions to the model for unfamiliar readers.

Let $\mathbb{N}$ be the set of nonnegative integers, and for $n \in \mathbb{N}$, let $[n] = \{0, 1, ..., n-2, n-1\}$. Similarly, for $l, u \in \mathbb{Z}$, let $[\![l, u]\!] = \{k \in \mathbb{Z} \mid l \leq k < u\}$.

Fix Σ to be some alphabet with Σ^* its finite strings. A *glue* $g \in \Sigma^* \times \mathbb{N}$ consists of a finite string *label* and non-negative integer *strength*. A *tile type* is a tuple $t \in (\Sigma^* \times \mathbb{N})^4$, thought of as a unit square with a glue on each side. A *tile set* is a finite set of tile types. We always assume a finite set of tile types, but allow an infinite number of copies of each tile type to occupy locations in the $\mathbb{Z}^2$ lattice, each called a *tile*.

Given a tile set T, a *configuration* is an arrangement (possibly empty) of tiles in the lattice $\mathbb{Z}^2$, i.e. a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$. Two adjacent tiles in a configuration *interact*, or are *bound* or *attached*, if the glues on their abutting sides are equal (in both label and strength) and have positive strength. Each configuration α induces a *binding graph* B_α whose vertices are those points occupied by tiles, with an edge of weight s between two vertices if the corresponding tiles interact with strength s. An *assembly* is a configuration whose domain (as a graph) is connected and non-empty. The *shape* $S_\alpha \subseteq \mathbb{Z}^2$ of assembly α is the domain of α. For some $\tau \in \mathbb{Z}^+$, an assembly α is *τ-stable* if every cut of B_α has weight at least τ, i.e. a τ-stable assembly cannot be split into two pieces without separating bound tiles whose shared glues have cumulative strength τ. Given two assemblies α, β, we say α is a *subassembly* of β (denoted $\alpha \sqsubseteq \beta$) if $S_\alpha \subseteq S_\beta$ and for all $p \in S_\alpha$, $\alpha(p) = \beta(p)$ (i.e., they have tiles of the same types in all locations of α).

A *tile-assembly system* (TAS) is a triple $\mathcal{T} = (T, \sigma, \tau)$, where T is a tile set, σ is a finite τ-stable assembly called the *seed assembly*, and $\tau \in \mathbb{Z}^+$ is called the *binding threshold* (a.k.a. *temperature*). If the seed σ consists of a single tile, i.e. $|\sigma| = 1$, we say $\mathcal{T}$ is *singly-seeded.* Given a TAS $\mathcal{T} = (T, \sigma, \tau)$ and two τ-stable assemblies α and β, we say that α *$\mathcal{T}$-produces β in one step* (written $\alpha \to_1^{\mathcal{T}} \beta$) if $\alpha \sqsubseteq \beta$ and $|S_\beta \setminus S_\alpha| = 1$. That is, $\alpha \to_1^{\mathcal{T}} \beta$ if β differs from α by the addition of a single tile. The *$\mathcal{T}$-frontier* is the set $\partial^{\mathcal{T}} \alpha = \bigcup_{\alpha \to_1^{\mathcal{T}} \beta} S_\beta \setminus S_\alpha$ of locations in which a tile could τ-stably attach to α. When $\mathcal{T}$ is clear from context we simply refer to these as the *frontier* locations.

We use $\mathcal{A}^T$ to denote the set of all assemblies of tiles in tile set T. Given a TAS $\mathcal{T} = (T, \sigma, \tau)$, a sequence of $k \in \mathbb{Z}^+ \cup \{\infty\}$ assemblies $\alpha_0, \alpha_1, \ldots$ over $\mathcal{A}^T$ is called a $\mathcal{T}$-*assembly sequence* if, for all $1 \leq i < k$, $\alpha_{i-1} \to_1^{\mathcal{T}} \alpha_i$. The *result* $\lim \alpha$ of an assembly sequence α is the unique limiting assembly of the sequence. For finite assembly sequences, this is the final assembly; whereas for infinite assembly sequences, this is the assembly consisting of all tiles from any assembly in the sequence. We say that α $\mathcal{T}$*-produces* β (denoted $\alpha \to^{\mathcal{T}} \beta$) if there is a $\mathcal{T}$-assembly sequence starting with α whose result is β. We say α is $\mathcal{T}$*-producible* if $\sigma \to^{\mathcal{T}} \alpha$ and write $\mathcal{A}[\mathcal{T}]$ to denote the set of $\mathcal{T}$-producible assemblies. We say α is $\mathcal{T}$*-terminal* if α is τ-stable and there exists no assembly that is $\mathcal{T}$-producible from α. We denote the set of $\mathcal{T}$-producible and $\mathcal{T}$-terminal assemblies by $\mathcal{A}_\Box[\mathcal{T}]$. If $|\mathcal{A}_\Box[\mathcal{T}]| = 1$, i.e., there is exactly one terminal assembly, we say that $\mathcal{T}$ is *directed*. When $\mathcal{T}$ is clear from context, we may omit $\mathcal{T}$ from notation.

2.3 The 1-Phase Blocked Tile Assembly Model

Assembly in the BlockTAM proceeds through a series of temperature *phases*, each of which consists of a hold at a specified temperature during which growth proceeds until all assemblies are terminal. Once all assemblies are terminal, the temperature of the system is set to the next temperature in a specified series and remains there until all growth is again terminal. This process continues until all growth is terminal during the last temperature phase. Note that a temperature sequence must be finite and that infinite growth (in the limit) during a phase is only allowed during the final phase.

Although the BlockTAM shares similarities with other tile-assembly models that allow for temperature variations (e.g. [1,21]), additional complexity arises from the fact that, at different temperatures, different subsets of the blockers in a system may be actively blocking glues. Therefore, to ease explanation of the BlockTAM, here we first define its dynamics during a single phase, then in Section 2.4 define the full model in terms of phases.

In order to easily translate between a tile and its blocked version, we assume all distinct glues have distinct labels. That is, there are not two glues with the same labels, but different strengths. This is a simple transformation that can be obtained by relabeling each glue so that the new label is a concatenation of the glue's original label along with its strength.

A blocker is defined as a triple (t, d, s) where t is a tile, d is a direction on the tile, and s is a strength. We say that a blocker $c = (t, d, s)$ binds to tile t on its d edge with strength s. Let t be a tile, C be a set of blockers, and τ be a temperature, then we define the tile t blocked at temperature τ by blocker set C, denoted $\text{blocked}(t, C, \tau)$ to be the tile with the same glues as t unless there is a blocker which binds at the location of the glue on the tile with strength greater than or equal to τ. Note that when C and τ are obvious from context, we simply write $\text{blocked}(t)$. Also, since we define an assembly to map points to tiles we can compose this function with an assembly to obtain a function which maps each point to a blocked version of the tile. We define the function unblocked to be the inverse of the blocked function. That is, $\text{unblocked}(\text{blocked}(t, C, \tau)) = t$. The function blocked is invertible due to our assumption distinct glues have distinct labels.

A 1-Phase BlockTAM system (1BlockTAS) is a 4-tuple $\mathcal{B} = (T, C, \sigma, \tau)$ where T is a tile set, C is a blocker set, σ is the seed assembly, and τ is the temperature. We define the equivalent TAS of $\mathcal{B}$, denoted $\text{TEQ}(\mathcal{B})$, to be the aTAM system $\mathcal{T} = (T', \text{blocked}(\sigma), \tau)$ where T' consists of a modified version of T such that any glue whose blocker binds with strength $\geq \tau$ is set to have strength 0. That is $T' = \{\text{blocked}(t, C, \tau) | t \in T\}$. Given a 1BlockTAS $\mathcal{B} = (T, C, \sigma, \tau)$ and two τ-stable assemblies α and β formed from tiles in T, we say that α $\mathcal{B}$-produces β in one step, written $\alpha \to_1^{\mathcal{B}} \beta$, if $\text{blocked}(\alpha) \to_1^{\text{TEQ}(\mathcal{B})} \text{blocked}(\beta)$. The $\mathcal{B}$-frontier is the set $\partial^{\text{TEQ}(\mathcal{B})}(\text{blocked}(\alpha))$.

Given a 1BlockTAS $\mathcal{B} = (T, C, \sigma, \tau)$, a sequence of $k \in \mathbb{Z}^+ \cup \{\infty\}$ assemblies $\alpha_0, \alpha_1, \dots$ over $\mathcal{A}^T$ is called a $\mathcal{B}$ assembly sequence if blocked(α_0), blocked$(\alpha_1), \dots$ is a TEQ$(\mathcal{B})$-assembly sequence. We say that α $\mathcal{B}$-produces β denoted $\alpha \to^{\mathcal{B}} \beta$ if blocked$(\alpha) \to^{\text{TEQ}(\mathcal{B})}$ blocked(β). We say that α is $\mathcal{B}$-producible if $\sigma \to^{\mathcal{B}} \alpha$ and write $\mathcal{A}[\mathcal{B}]$ to denote the set of $\mathcal{B}$-producible assemblies. We α is $\mathcal{B}$-terminal if blocked(α) is TEQ$(\mathcal{B})$-terminal. We denote the set of $\mathcal{B}$-producible and $\mathcal{B}$-terminal assemblies by $\mathcal{A}_{\Box}[\mathcal{B}]$. We say that $\mathcal{B}$ is directed provided that TEQ$(\mathcal{B})$ is directed.

2.4 The Blocked Tile Assembly Model

The following formal definitions are an adaptation of those used to define the Multiple Temperature Model in [21].

A BlockTAM system (BlockTAS) is a 4-tuple $\mathcal{B}^* = (T, C, \sigma, \langle\tau\rangle_{i=0}^{k-1})$ where T is a tile set, C is a blocker set, σ is the seed assembly, and $\langle\tau\rangle_{i=0}^{k-1}$ is a sequence of non-negative binding thresholds. The number k is the temperature complexity.

We define an *assembly sequence for the i^{th} temperature phase*, for $0 \leq i < k$, of $\mathcal{B}^*$ as follows. For $i = 0$, an assembly sequence is an assembly sequence $\vec{\alpha}$ of the 1BlockTAS $\mathcal{B}_0 = (T, C, \sigma, \tau_0)$. For $i > 0$, an assembly sequence for temperature phase i of $\mathcal{B}^*$ is a sequence of assemblies $\vec{\alpha} = (\alpha_0, \alpha_1, \dots)$ satisfying the following conditions.

1. There exists l such that α_l is τ_{i-1}-stable $\partial^{\mathcal{B}_{i-1}} \alpha_l = \varnothing$, and $(\alpha_0, \alpha_1, \dots, \alpha_l)$ is an assembly sequence for temperature phase $i-1$ of $\mathcal{B}^*$; and
2. for all $j > l$, either $\alpha_{j-1} \to_1^{\mathcal{B}_i} \alpha_j$ or α_j is obtained from α_{j-1} by deleting the portion of α_{j-1} which does not contain the seed across a cut having strength less than τ_i.

Intuitively, an assembly sequence for the i^{th} temperature phase is defined recursively. That is, we say that $(\alpha_0, \alpha_1, ...)$ is an assembly sequence for the i^{th} temperature phase provided that (1) if $i = 0$, then $(\alpha_0, \alpha_1, ...)$ is an assembly sequence of the 1BlockTAS (T, C, σ, τ_0) and (2) if $i > 0$, then two conditions must be met. The first condition says that there exists l such that the sequence can be broken up at l so that the sequence up to index l is an assembly sequence for the $(i-1)^{th}$ temperature phase. Condition 2 says that the sequence after index l is a sequence of assemblies such that α_j can be obtained from α_{j-1} by either (1) a single tile addition (with combined binding strength of at least τ_i) or (2) by "melting off" a subassembly from α_{j-1} with a cut of strength $< \tau_i$.

An *assembly sequence* in $\mathcal{B}^*$ is an assembly sequence for the i^{th} temperature phase of $\mathcal{B}^*$ for some $i \in \mathbb{N}$. We say that an assembly sequence $\vec{\alpha}$ in $\mathcal{B}^*$ *finishes every temperature phase of* $\mathcal{B}^*$ if $\vec{\alpha}$ is a finite assembly sequence for temperature stage $k-1$ of $\mathcal{B}^*$ and its final assembly, denoted α, is τ_{k-1}-stable and $\partial^{\beta_{k-1}} \alpha = \varnothing$. In this case, we call α a *terminal assembly* and write $\alpha \in \mathcal{A}_{\Box}[\mathcal{B}^*]$. If $|\mathcal{A}_{\Box}[\mathcal{B}^*]| = 1$, then we say that $\mathcal{B}^*$ is *directed.* Given system $\mathcal{B}^*$, let $\tau_{max} = \mathtt{max}(\tau_0, \dots, \tau_{k-1})$ (i.e. the maximum τ value of any stage). We say that $\mathcal{B}^*$ is *τ_{max}-robust* if every terminal assembly of every temperature phase is τ_{max}-stable. In a τ_{max}-robust system, no temperature change causes any portions of any assemblies to dissociate and thus growth is monotone. All constructions in this paper are τ_{max}-robust.

If for every $\alpha \in \mathcal{A}_{\Box}[\mathcal{B}^*]$, dom $\alpha = S$, then we say that $\mathcal{B}^*$ self-assembles the shape S. Given S, a connected set of points in $\mathbb{Z}^2$, we define a version of S *scaled by factor* c, and denote it by $c \cdot S$, as $c \cdot S = \{(x, y) \in \mathbb{Z}^2 \mid (\lfloor \frac{x}{c} \rfloor, \lfloor \frac{y}{c} \rfloor) \in X\}$. If for every $\alpha \in \mathcal{A}_{\Box}[\mathcal{B}^*]$, there exists $c \in \mathbb{N}$ such that dom $\alpha = c\dot{S}$, then we say that $\mathcal{B}^*$ self-assembles the shape S at scale factor c. Note that the Blocked Tile Assembly Model is defined so that a "blocked glue" on a tile which binds into the assembly immediately binds to any neighboring glue.

2.5 BlockTAM System Example

We now provide a simple example of a BlockTAM system to familiarize the reader with key ideas and notation regarding the model. This example is inspired by the system designed in [15]. Part (a) of Figure 2 shows a BlockTAS $\mathcal{B}$. In this example, we assume that all blockers bind with the same strength as the glue. If a temperature 4 TAS $\mathcal{T}$ was defined to have the same tile set and seed as $\mathcal{B}$, the assembly shown in part (b) of Figure 2 would be an element of $\mathcal{A}[\mathcal{T}]$ (that is, the assembly is producible by $\mathcal{T}$). Parts (c) and (d) Figure 2 shows the equivalent blocked tile assembly systems for each temperature phase of $\mathcal{B}$ and the terminal assembly for each. Note that in part (c), the shown producible assembly of $\text{TEQ}(\mathcal{B}_0)$ is terminal since the blue tile lacks the strength required to attach. In part (d), the shown assembly is terminal in the context of $\text{TEQ}(\mathcal{B}_1)$ since the red tile is blocked (meaning it has strength 0) and is consequently unable to attach.

3 Encoding Input Values via Temperature Sequences

In this section we present a construction that works in two parts. In the first, the input is a set of temperatures (i.e. positive integers), which we'll denote as Γ. Using this set of temperatures, the first part of the construction returns a tile set, set of blockers, and seed tile that compose a *universal number encoder* that is specific for that set of temperatures.

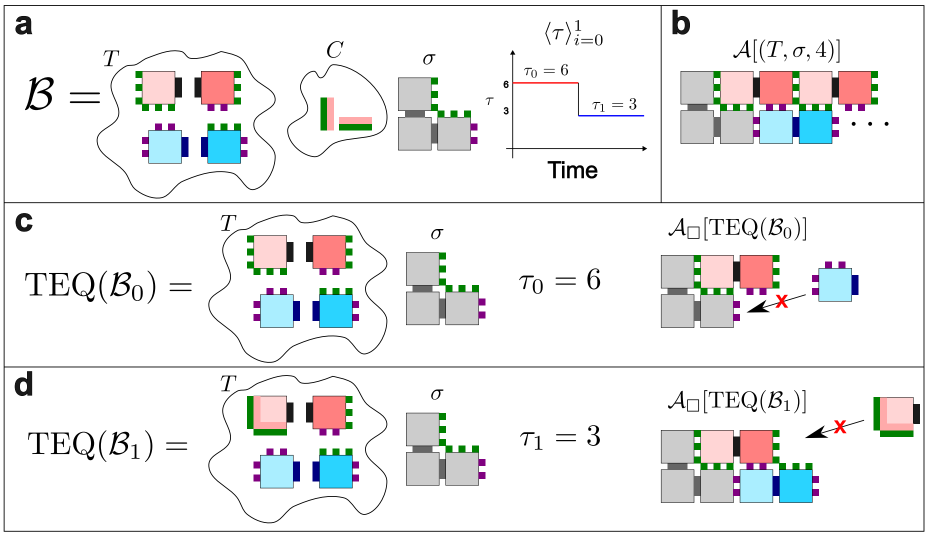

a
$\mathcal{B} =$
T
C
σ
$\langle \tau \rangle_{i=0}^{1}$
$\tau_0 = 6$
$\tau_1 = 3$
6
3
τ
Time
b
$\mathcal{A}[(T, \sigma, 4)]$
. . .
c
$\text{TEQ}(\mathcal{B}_0) =$
$\tau_0 = 6$
$\mathcal{A}_\Box[\text{TEQ}(\mathcal{B}_0)]$
d
$\text{TEQ}(\mathcal{B}_1) =$
$\tau_1 = 3$
$\mathcal{A}_\Box[\text{TEQ}(\mathcal{B}_1)]$

Figure 2 An illustration of an example BlockTAM system. Part **a** shows its definition. Here, a glue label is represented as the color of the protruding squares on a tile and it's strength is the number of protruding squares. For example, the red tile has a green strength 3 glue on its east side. We represent a strength 6 glue by a long rectangle protruding from the edge of the tile. We represent a blocker as two rectangular slices where one rectangle is the same color as the glue which it blocks and the other rectangle is the color of the tile to which the blocker binds. In this example, we assume that all blockers bind with the same strength as the glue. Part **b** shows the assembly formed by the tile set in a traditional TAS without blockers. The tile set grows a width 2 ribbon with alternating red and blue tile segments. Part **c** shows the equivalent blocked tile assembly system for temperature phase 0 of $\mathcal{B}$ along with its terminal assembly. Note the blue tile is unable to bind to the terminal assembly because the two purple glues are strength 2, summing to 4 which is less than the binding threshold. Part **d** shows the equivalent blocked tile assembly system for temperature phase 1 of $\mathcal{B}$ along with its terminal assembly. Note the red tile is unable to bind to the terminal assembly because it is blocked (making its strength 0 which is less than the binding threshold).

The input to the second part is an arbitrary value $n \in \mathbb{N}$, and the output is a sequence of temperatures over Γ such that a BlockTAM system made using that sequence and the universal number encoder previously produced results in a terminal assembly that is a rectangle with an encoding of the value n (in a base to be discussed) represented by its northern glues. The design is such that, as previously mentioned, a standard temperature-2 aTAM tile set could then attach to read that encoding as its input and carry out arbitrary algorithmic self-assembly.

A notable feature of our construction it is that is generalized so that it works for any set Γ consisting of at least 3 temperatures, creating a universal number encoder specific to that Γ. This allows for maximal design flexibility in terms of optimizing between tile complexity (i.e. the number of unique tile types required), glue diversity (in terms of numbers of unique sequences and unique strengths), and the temperature complexity (i.e. the number of distinct temperature phases required). In general, the greater the variety of temperatures allowed (i.e. larger Γ) the greater the tile and glue complexity but the lower the temperature complexity.

3.1 Generation of a universal number encoder

As mentioned, the input is a set of temperatures, Γ. The construction requires at least three distinct allowable temperature values since having only two possible temperatures in a BlockTAM system does not allow any way to create series of temperature sequences that are distinct from each other for different values to encode without using unary encoding, which is infeasible for all but the very smallest values.

Let $k = |\Gamma|$ be the number of temperatures, and let $(\tau_1, \tau_2, \dots, \tau_k)$ be Γ sorted from lowest to highest. For clarity, we will often use τ_{max} to refer to τ_k. The tile set that we are creating will be specific to the values of Γ, but will be able to build assemblies representing arbitrary values of $n \in \mathbb{N}$. Each such n will be encoded in some base b, which is determined as follows: if $k = 3$ then $b = 2$, otherwise $b = k - 2$. Let $n \in \mathbb{N}$ be an arbitrary number to eventually be encoded. In general, encoding n in base b requires $\lceil \log_b n \rceil$ values. However, if $k == 3$ it will be necessary to represent each bit of the base 2 representation of n using 2 bits (for reasons to be explained later). Therefore, let v represent the number of values needed to represent the encoding of a given value n in base b.

The target assembly will consist of $c = v + 1$ columns. The glue exposed on the north of the top tile of the first column will be a special "start" glue, and the northern glue of each of the following $c - 1$ columns will encode subsequent values of the encoding of n. The north-south glues binding the tiles of a column will all be of strength equal to τ_{max} so that each column is τ-stable across all temperatures of the system. However, the glues that bind each column to another will consist of varying strengths from $\langle\tau\rangle$. Therefore, when glues of any strength $< \tau_{max}$ are used, it will be necessary for multiple glues of that strength to bind the columns together to ensure stability if/when the system temperature is set to τ_{max}.

Let h be the height of a column (with the determination of the value h to be explained later). We refer to the bottom location of a column as r_0 and the one above that as r_1, continuing up to the topmost at r_{h-1}. Across all columns, the row at each location r_i for $0 \leq i < h$ is dedicated to tiles whose east/west glues are of the same strength, which we'll refer to as s_i. Furthermore, the binding of any pair of columns to each other will be via one or more glues all of the same strength. To compute h we first determine how many instances of each temperature in Γ are required to sum to the smallest value $\geq \tau_{max}$. We then take the sum of that sum to be h. For example, if the set of possible temperatures is $\{2, 3, 4, 5\}$, $h = 3 + 2 + 2 + 1 = 8$ since it takes 3 glues of strength 2 to sum to $\geq \tau_{max} = 5$, 2 each of strength 3 and 4, and of course only 1 of strength 5. Thus, every column in a system using those temperature values would be of height $h = 8$. (See Figures 3 and 4 for examples.)

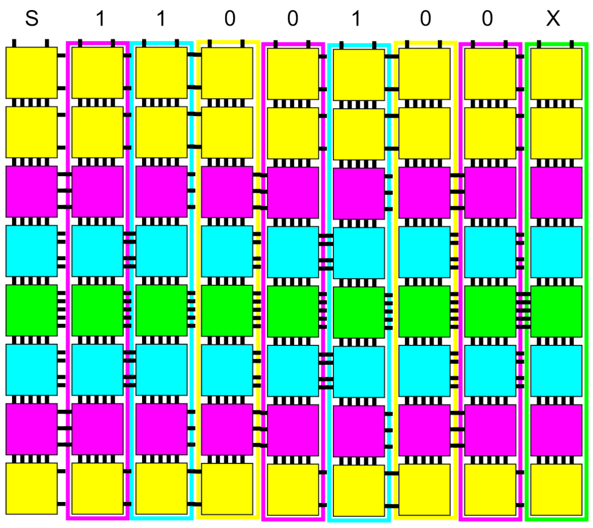


Figure 3 An example of the terminal assembly of the universal number encoder for temperatures $\{2, 3, 4, 5\}$ encoding the number 100_{10} in binary.

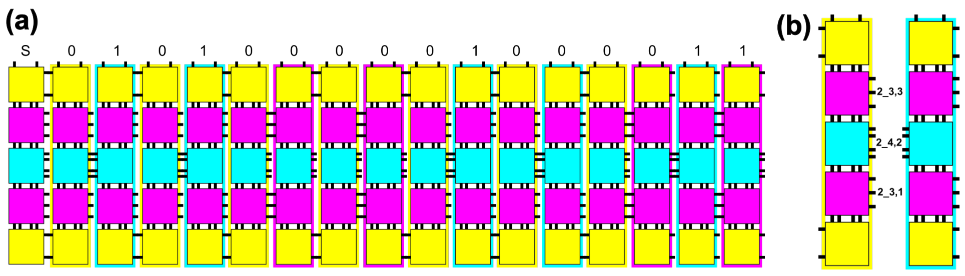


Figure 4 **(a)** An example of the terminal assembly of the number encoding system with temperatures $\{2, 3, 4\}$ encoding the number 100_{10} in binary. Note that the fact that there are only 3 permissible temperatures requires that each bit of the binary representation of 100_{10} (i.e. 1100100_2), as well as the end marker, is represented by 2 bits so that the final column can be accurately identified by any tiles that may later grow other the northern-facing encoding. Each 0 is replaced by 00, each 1 by 01, and the end marker is represented by 11. Thus, the full encoded value is "S0101000001000011". **(b)** An example pair of columns for a universal number encoder for temperatures $\{2, 3, 4\}$. The glue strengths for each position, from bottom to top, are $2, 3, 4, 3, 2$. Growth of the column on the left would be initiated by a temperature 2 phase, and the column on the right by a temperature 4 phase (which is clear since each column only has input, i.e. west-facing, glues in the locations specific to those strengths). Note that the left column does not have output (east-facing) glues in the strength 2 position since its own growth is initiated at temperature 2, and similarly the right column does not have a glue in the strength 4 position. Labels of east/west glues between the columns consist of $t_i\_t_j, r_k$ where t_i the growth temperature of the left column, t_j is the growth temperature of the right column, and r_k is the row index. The label "2_4, 2" is common to the east/west glues of the blue tiles, allowing the right tile to attach and the right column to then form, assuming the next temperature phase after completion of the left column was at temperature 4. If, for instance, it was temperature 3, tiles unique to the $(2, 3)$ column would instead be unblocked and attach to the pink tiles of the left column, allowing the $(2, 3)$ to grow instead.

With the value h as the necessary height of each column, we then compute the assignment of each s_i, that is, which east/west strengths are assigned to each row r_i. Since these are the glues that bind neighboring columns to each other, we try to assign them to maximize stability. Therefore, we assign the center location to the strongest glue, s_{max}. Then, since each weaker glue requires two or more locations (so that the multiple copies of each can sum to $\geq \tau_{max}$), we assign them moving outward from the center, alternating up and down for each subsequently weaker glue until all positions are assigned. Examples can be seen in Figures 3 and 4. For instance, Figure 3 shows how the strengths $2, 3, 4$, and 5 are assigned, with the ordering from bottom to top being $2, 3, 4, 5, 4, 3, 2, 2$.

Since the north/south glues between the tiles of each column are of strength s_{max}, and now we know the strengths of the east/west glues for the tiles at each height, we can determine how to create the columns necessary to represent the values of the numbers to be encoded. Note that a seemingly straightforward method would be to create one unique column for each value that needs to be represented to the north. However, then it is not possible for the encoding to contain the same symbol in two consecutive positions (e.g. the binary representation of $14_{10} = 1110_2$, which has two pairs of consecutive 1s) since then the same column would have to be able to connect to a copy of itself, which would result in "pumping" into an infinite sequence of 1s. A relatively simple fix to that problem would be to have a column that doesn't represent any value that could then be located between each value-encoding column. However, this would double the number of temperature phases needed, and that is the primary metric that we are trying to minimize. Therefore, we use the following scheme which compresses the temperature sequence at the expense of requiring slightly more unique tile types (approximately double).

Our method of creating columns for encoded values works by permuting over all pairs of (non-identical) temperatures from Γ. Let $t = |\Gamma|$ be the number of unique temperatures. For each $\tau_i \in \Gamma$, there are $t - 1$ temperatures $\tau_j \in \Gamma$ such that $t_i \neq t_j$. For each such pair we create exactly one unique column and we assign a value for that column to encode from the set $\{0, 1, \ldots b - 1\} \cup \{\text{“}X\text{”}\}$. (Note that, in the case where $t = 3$, no "X" character is explicitly used but instead the ending condition is encoded using a special sequence of bits, to be discussed in Section 3.2.) This mapping from each pair of temperatures to an encoded value is referred to as the `column_transition_map` (and the code used to generate it can be seen in Listing 1 in the Appendix). The resulting map ensures that any encoded symbol can follow any other, except for the "X" symbol which is the final symbol to be encoded by the last temperature phase. (Again, when $t == 3$, there is no "X" symbol, just 0s and 1s, which will each be able to follow each symbol.) Note that the column that contains the seed tile is unique, composed of tiles that only appear in that column, and its northernmost glue encodes the start value "S". The glue encoding "S" is of strength 2, and all other value encoding glues are of strength 1. Once the number encoding assembly is terminal, the system temperature can be set to 2, allowing for a standard aTAM tile set to attach and grow across from left to right, reading the encoded value then proceeding with arbitrary algorithmic behavior.

To generate the tile type for each location of each column, we use the previously computed glue strengths for each side of each location r_i and the information in the `column_transition_map`. For each temperature pair (t_i, t_j), again where $t_i \neq t_j$, we generate a set of h distinct tile types to create a unique column. The purpose of the (t_i, t_j) column is to (1) encode the value `column_transition_map`(t_i, t_j) in the north glue of its northernmost tile, (2) to have its growth initiated by a temperature phase of temperature t_2, and (3) to grow immediately to the right of a column whose growth was initiated by a temperature phase of temperature t_1. To effect this, each north/south glue label between two tiles of a column are specific to the column type, i.e. (t_i, t_j), and location r_i, e.g. $t_i\_t_j, r_k$ for the glue pair between the kth and $k+1$th tile in the column for (t_i, t_j). On the east, output side of the tile in location r_k is a glue of strength s_k (which is the strength value assigned to that row) for all locations except that for which $s_k == t_j$, which has no east-facing glue. As previously mentioned, this is to prevent the column from being able to initiate growth of another column to its right at the same temperature at which it grew. The label of each such glue will be $t_j\_s_k, k$, so that it can bind to a tile in the kth location of the column for temperature sequence (t_j, s_k) (assuming that the next phase is at temperature s_k). Figure 4 shows an example of these glues.

In this way, the full tile set is constructed. Next, the blocker set is created. This simply consists of a blocker for every west-facing glue that is strength $s-1$ if the glue's strength is s. This ensures that those "input" glues, which initiate the growth of each new column, are only active during phases of the temperatures for which their columns were designed, and growth proceeds with a single column forming for each phase.

With the designated seed tile being a tile of the type designed for the middle of the leftmost column, the resulting tile set, blocker set, and seed is a universal number encoder for Γ. We next describe how to, given a specific value of n to encode, compute the temperature sequence necessary for a BlockTAS to assemble an assembly encoding n.

3.2 Temperature sequence generation

The input to this portion of the construction is a number $n \in \mathbb{N}$ and a universal number encoder from the previous portion. From that encoder, we can determine the base b in which to encode n. Additionally, we can know which of two paths to take to compute the encoding e of n: If the number of temperatures, t, is 3, then e will consist of a "doubled" base 2 encoding, otherwise e will simply be the encoding of n in base b (which is set to $t-2$), with the character "X" appended to the end. Note that in either case, the encoding will begin with the "S" symbol which is encoded by the column with the seed tile.

To compute the doubled encoding for the case of $t == 3$, we note that with only three temperatures to select from, once a given phase of assembly completes, which is necessarily at some $t_i \in \Gamma$, then there are only two choices for some other $t_j \in \Gamma$ such that $t_i \neq t_j$. Because of this, we can't directly encode a symbol to mark the end of sequence, which makes it difficult or impossible for an algorithmic (aTAM-style) tile set to then read the encoded value and effectively use it as input that controls its algorithmic behavior. (It would either have to pause at the end of the assembly "waiting" for a stop symbol, or instead be able to grow on its own past the last symbol without cooperating with the assembly which means that at other points of growth it could "ignore" the assembly and grow without cooperation, yielding an invalid input. This is a well-known problem in the domain of tile-based self-assembly.) Therefore, we utilize the method of representing each bit of the binary encoding of n as follows. We represent each 0 by 00, and each 1 by a 01. We build our encoding in this way, and to indicate the end of the sequence, after all bits have been encoded this way, we append two consecutive 1s. In this way, a tile set growing over the north of the terminal assembly and reading the encoding will be able to detect when it has read the final position.

Now with the encoded string $e_0, e_1, ..., e_v$, we convert that into a series of temperatures over Γ. Because there are no blockers for tiles in the column containing the seed, that column grows at any temperature in Γ, as all north/south glues are of strength τ_{max}. This means that no separate temperature phase is necessary to account for the placement of the "S" symbol and the first temperature phase is used to grow the column of the first encoded value of n.

Additionally, the east-facing, output glues of the seed column use the same labels as a column that grew at the second highest temperature so that new glues do not need to be designed for tiles to attach to that column. Thus, for the first, and then each subsequent, encoded value we simply iterate over the values $e_0, e_1, \ldots, e_v$ and examine the `column_transition_map` keeping track of which temperature t_i was used to grow the previous column (pretending it was the second largest temperature if that was the seed column). For each e_m, We find the unique t_j such that `column_transition_map`$(t_i, t_j) = e_m$. We then append that t_J as the next temperature in the sequence. Iterating through the full encoded string, the resulting temperature sequence is the exact sequence that will cause the universal number encoder to build an assembly of columns expressing the encoded value of n in its northern glues. (For details of a few examples of systems, please see Section A.1 of the Appendix.)

4 Using Temperature Changes to Steer Assembly Through Shapes

The most common goal in the design of a self-assembling system is for the resulting system to deterministically form into assemblies of a single, target shape. However, the goal of accurate shape-building must usually be balanced with competing factors such as minimizing the number of unique tile types required and the scale factor at which the shape is formed. In this section, we present a *universal shape builder* consisting of a fixed tile set, blocker set, seed tile, and temperature set that is capable of being utilized to build any arbitrary, finite (and necessarily connected) shape. That is, given an arbitrary shape S, a temperature sequence exists that causes the BlockTAM system composed of these components to self-assemble a scaled version of S, i.e. S^c. If S has a Hamiltonian path, our construction assembles S at scale factor 3. If not, our construction assembles it at scale factor 6. It has valid temperature set $\{2, 3, 4\}$, consists of 145 tile types with glues of strengths 1, 2, 3, and 4, and has 16 strength-3 and 16 strength-4 blockers. There is a single designated tile type for the seed.

We first explain the algorithm that takes as input an arbitrary shape S and outputs a path through it. We then explain the tile set and how it is capable of following such a path. Finally, we demonstrate how that path can be converted into a series of temperatures to drive the universal shape constructor along it.

4.1 Finding a "turtle graphics" path

In [21] it was shown how, beginning with an arbitrary input shape S, S can be scaled by a factor of two to S^2, and then a Hamiltonian path through S^2 can be created that visits each point in S^2 exactly once. (Note that any shape S that already has a Hamiltonian path through it does not require the scaling by two, but we'll assume that, in the worst case for an arbitrary S, the scaling is required.) Using the technique of [21] we can convert that path into a series of "turtle graphics" instructions. That is, starting on the first point of the path, we simply follow a series of instructions taken from the set $\{$`Straight`, `Right`, `Left`$\}$ (which we'll abbreviate as $\{S, R, L\}$) to visit every point on the path (and thus in S^2) exactly once. Let $p = |S|$ be the number of locations, or *pixels*, in the original shape S. The output of this step is an ordered list $L \in \{S, R, L\}^{4p-1}$, that is, a list of $4p - 1$ directions that allow one to begin at one point in S^2 and visit every other (since each pixel in S is composed of 4 in S^2, and we don't need an instruction to get to the pixel from which the path begins).

4.2 The universal tile set

We now describe the tile set and blockers of the universal shape builder by explaining how they are able to follow such a path. The assembly that follows L is composed of *macrotiles* that are 3×3 squares of tiles. That is, each pixel of S^2 is represented by a 3×3 square of tiles (yielding an overall scale factor of 6 from S). Starting from the seed tile and with a temperature phase of $\tau = 4$, the first macrotile grows the *base* portion of a macrotile, which is composed of 3 tiles (that can be seen as green in the macrotiles in Figure 5). The macrotile then becomes terminal, until the next temperature phase begins. A phase of 3 causes the blue tiles to become unblocked and attach, finishing the macrotile and orienting the output glue toward the left. A phase of 2 causes the blue and pink tiles to be blocked but allows the yellow tiles to attach, after which the macrotile becomes terminal. Then, a next temperature phase of $\tau = 4$ allows the orange tiles to become unblocked and attach, exposing the macrotile's output glue straight ahead. A phase of $\tau = 3$ would instead cause the orange tiles to be blocked and the pink tiles to be unblocked and thus attach. They expose a right-facing output glue for the macrotile. Refer to Figure 5 to see that all portions of macrotile growth are fully stabilized before the temperature can increase.

The result of the temperature phases described are that the macrotile completes, is fully stable at all temperatures (at the end of each intermediate temperature phase and the final), and exposes exactly one output glue of strength 4 that can initiate the growth of the next macrotile. While growth of the next macrotile begins with a temperature phase of $\tau = 4$, note that in the case of the direction being S, the final phase of growth of the current macrotile is also at $\tau = 4$. In this case, the base portion of the next macrotile forms during the same phase that the current macrotile completes. This is not a problem and is easily accounted for when designing our temperature sequence.

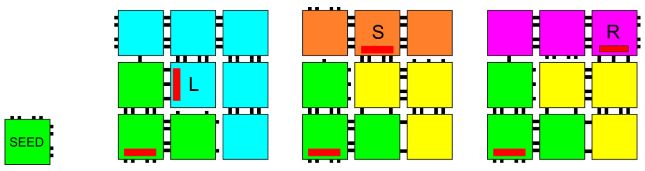


Figure 5 The seed tile (left) and 3 basic macrotiles of the universal shape builder. Glue that have blockers are indicated by red rectangles. Each macrotile first grows its base portion, composed of the same three tiles (green) in each. This growth occurs at temperature 4, and once the three green tiles have attached, the assembly is terminal. The first macrotile grows from the seed tile as its bottom-left tile. The others begin with the bottom left green tile attaching to a previously formed macrotile. After the base, which is common to each of the macrotiles, is terminal, a new temperature phase begins to allow subsequent growth. If the new temperature is 3, the blue tiles attach, effecting a left turn and completing the macrotile, and the assembly becomes terminal at that temperature. If it is 2, then the three yellow tiles attach, before growth halts. Then, a new temperature phase begins. If it is temperature 4, the orange tiles attach to cause the next macrotiles to grow straight ahead. If it is 3, the pink tiles attach for a right turn. In the cases where the yellow tiles attach, note that two of them attach to the bottom-right green tile, each with a strength-2 bond. This creates a minimum cut of total strength 4 in the binding graph meaning that the yellow tiles are stably attached even at the highest temperature, which is 4. When the blue and pink tiles attach with initial strength-3 bonds, they each also grow so that their leftmost tiles form a single-strength bond with the upper green tile to again guarantee a minimum cut of total strength 4. Therefore, in all three scenarios, S, R, and L, by the end of the temperature phases that complete the growth of the macrotile, all potions are 4-stable.

The final, and relatively straightforward, aspect of the construction is necessary to handle growth of macrotiles following one or more R or L turns. For each clockwise rotation of $90, 180$, and 270 degrees, we simply create a new copy of each tile making up any of the three versions of the macrotile. We give each copy new glue labels that are specific for that rotation, with the only glue labels that are shared between tiles intended for different rotations being the four glues on the exteriors. These glue labels are designed so that each macrotile output glue is specific to the input glue (i.e. the glue exposed by the green tile) of the correctly oriented version. Since the output locations for R and L turns are not symmetrically located on the macrotile exterior, it is also necessary to make a reflected version of the basic macrotile, and copies of that for each of the rotations. (An example of the full set of rotations and reflections can be seen in Figure 7 in the Appendix.) We use an additional glue prefix of "x" for all of the labels of glues on tiles of the reflected copies so that tiles intended for a macrotile of one rotation and/or reflection are not able to bind inside of macrotiles with different rotations and/or reflections, but again we make sure that each rotated and/or reflected macrotile's output glue labels correctly match those of the properly rotated and/or reflected macrotile input glues to match their output directions (relative to their orientations). (A complete simple example can be seen in Figure 8 of the Appendix.)

4.3 Generating the temperature sequence

From the above description, it is clear that the universal shape builder can correctly follow an arbitrary turtle graphics path. Thus, all that remains is to show how a shape's path can be turned into a proper temperature sequence. This is also quite trivial, as we have already seen which temperature sequences are necessary for each movement of S, R, or L. Since we require that every step is either S, R, or L, if the path through S^2 starts with a backward step we simply rotate the shape so that the first step is S, R, or L, which doesn't change the shape since shapes are invariant up to rotation. The first temperature is 4, allowing the base of the first macrotile to grow. Then, for L a 3 is appended; for S, a 2 and 4 are appended; and for R 2 and 3. To initiate the next macrotile, if the last direction was L or R, we append 4, but if it was S we don't need to do that. The final main consideration is for handling reflected macrotiles. We keep track of each turn that uses the L version of the macrotile, as that causes a flip between reflected and non-reflected versions, and note that, when in a reflected macrotile it is an R turn that uses the blue L tiles and flips the reflection status. Finally, to make sure that the final macrotile is complete, after a $\tau = 4$ phase to grow its base, we include a $\tau = 3$ phase to cause it to complete. (The code used to compute the temperature sequence can be seen in Listing 2 in the Appendix.) We've made software freely available that allows one to draw a path, save it as a BlockTAM system [13], then simulate it in a web-based simulator [8]. (Figure 9 of the Appendix shows a screenshot.)

Thus, we have demonstrated a universal shape builder. While only requiring 145 tile types and 32 blockers, it requires an average of just over 2 temperature phases per pixel of S^2 (or S if scaling isn't required). While infeasible for experimental implementations of large shapes, the design techniques may provide tools useful for other and improved constructions.

References

1 Gagan Aggarwal, Michael H. Goldwasser, Ming-Yang Kao, and Robert T. Schweller. Complexities for generalized models of self-assembly. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 2004.

2 Florent Becker, Daniel Hader, and Matthew J Patitz. Strict self-assembly of discrete self-similar fractals in the abstract tile assembly model. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'25), New Orleans, USA*, pages 2387–2466. SIAM, 2025. doi:10.1137/1.9781611978322.80.

3 Junghuei Chen and Nadrian C Seeman. Synthesis from dna of a molecule with the connectivity of a cube. *Nature*, 350(6319):631–633, 1991.

4 Qi Cheng, Gagan Aggarwal, Michael H. Goldwasser, Ming-Yang Kao, Robert T. Schweller, and Pablo Moisset de Espanés. Complexities for generalized models of self-assembly. *SIAM Journal on Computing*, 34:1493–1515, 2005. doi:10.1137/S0097539704445202.

5 Constantine Evans, Angel Cervera Roldan, Trent Rogers, and Damien Woods. Tile blockers as a simple motif to control self-assembly: kinetics and thermodynamics. Unpublished.

6 Ashwin Gopinath, Evan Miyazono, Andrei Faraon, and Paul WK Rothemund. Engineering and mapping nanocavity emission via precision placement of DNA origami. *Nature*, 2016.

7 Daniel Hader, Aaron Koch, Matthew J. Patitz, and Michael Sharp. The impacts of dimensionality, diffusion, and directedness on intrinsic universality in the abstract tile assembly model. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2607–2624. SIAM, 2020. doi:10.1137/1.9781611975994.159.

8 Daniel Hader and Matthew J. Patitz. WebTAS (beta): A browser-based simulator for tile-assembly models, 2025. URL: http://self-assembly.net/software/WebTAS_beta/WebTAS_beta-latest/.

9 Yonggang Ke, Luvena L Ong, William M Shih, and Peng Yin. Three-dimensional structures self-assembled from DNA bricks. *Science*, 338(6111):1177–1183, 2012.

10 James I. Lathrop, Jack H. Lutz, Matthew J. Patitz, and Scott M. Summers. Computability and complexity in self-assembly. *Theory Comput. Syst.*, 48(3):617–647, 2011. doi:10.1007/s00224-010-9252-0.

11 James I. Lathrop, Jack H. Lutz, and Scott M. Summers. Strict self-assembly of discrete Sierpinski triangles. *Theoretical Computer Science*, 410:384–405, 2009. doi:10.1016/J.TCS.2008.09.062.

12 Dionis Minev, Christopher M Wintersinger, Anastasia Ershova, and William M Shih. Robust nucleation control via crisscross polymerization of highly coordinated DNA slats. *Nature Communications*, 12(1):1–9, 2021.

13 Matthew J. Patitz and Trent Rogers. Blocked TAM wiki page and software downloads, 2025. URL: http://self-assembly.net/wiki/index.php/Blocked_Tile_Assembly_Model_(BlockTAM).

14 Matthew J. Patitz and Scott M. Summers. Self-assembly of decidable sets. *Natural Computing*, 10(2):853–877, 2011. doi:10.1007/s11047-010-9218-9.

15 Trent Rogers, Constantine Evans, and Damien Woods. Controlling self-assembly with blockers: programming nanostructure size with temperature. Unpublished.

16 Trent Rogers, Constantine Evans, and Damien Woods. Controlling self-assembly with blockers: tunable nucleation in growth conditions. Unpublished.

17 Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, March 2006. doi:10.1038/nature04586.

18 Paul W. K. Rothemund and Erik Winfree. The program-size complexity of self-assembled squares (extended abstract). In *STOC '00: Proceedings of the thirty-second annual ACM Symposium on Theory of Computing*, pages 459–468, Portland, Oregon, United States, 2000. ACM. doi:10.1145/335305.335358.

19 Nadrian C Seeman. Nucleic acid junctions and lattices. *Journal of theoretical biology*, 99(2):237–247, 1982.

20 David Soloveichik and Erik Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36(6):1544–1569, 2007. doi:10.1137/S0097539704446712.

21 Scott M. Summers. Reducing tile complexity for the self-assembly of scaled shapes through temperature programming. *Algorithmica*, 63(1-2):117–136, June 2012. doi:10.1007/s00453-011-9522-5.

22 Ning Wang, Chang Yu, Tingting Xu, Dan Yao, Lingye Zhu, Zhifa Shen, and Xiaoying Huang. Self-assembly of dna nanostructure containing cell-specific aptamer as a precise drug delivery system for cancer therapy in non-small cell lung cancer. *Journal of Nanobiotechnology*, 20(1):1–19, 2022.

23 Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, June 1998.

24 Damien Woods, David Doty, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, and Erik Winfree. Diverse and robust molecular algorithms using reprogrammable dna self-assembly. *Nature*, 567(7748):366–372, 2019. doi:10.1038/S41586-019-1014-9.

A Technical Appendix

This Technical Appendix contains additional technical details related to the results in the main body of the paper.

Listing 1 Code used to generate the transition mapping of temperature pairs to encoded symbols for the universal number encoder.

```
def get_column_transition_map(temp_list, base, final_temp):
    temp_list = sorted(temp_list)
    column_transition_map = {}

    for temp1 in temp_list:
        if temp1 != final_temp:
            count = 0
            for temp2 in temp_list:
                if temp1 != temp2:
                    if count < base:
                       column_transition_map[(temp1,temp2)] = count
                       count += 1
                    else:
                       column_transition_map[(temp1,temp2)] = 'X'

    return column_transition_map
```

A.1 Number encoder examples

We now provide examples with a few details about universal number encoders for two different sets of temperatures and how they each encode a pair of numbers.

Let $\Gamma_1 = \{2, 3, 4\}$ and $\Gamma_2 = \{2, 3, 4, 5\}$ be two sets of temperatures. Using the construction, the universal number encoders produced have the characteristics listed in the first two rows of Table 1. Additionally, Figure 6 demonstrates how the first three columns of the first encoder would grow for the encoded value of 100_{10}, and Figures 3 and 4 show their terminal assemblies.

Table 1 Comparisons of universal number encoding systems.

Temperatures	Number Tile Types	Column Height	Base	Temps(100)	Temps(10000)
2,3,4	35	5	2	16	30
2,3,4,5	80	8	2	8	15
2,3,4,5,6	170	10	3	6	10
2,3,4,5,6,7	364	14	4	5	8

A.2 Additional details of the universal shape builder

Here we provide a few supplementary figures for the universal shape builder construction.

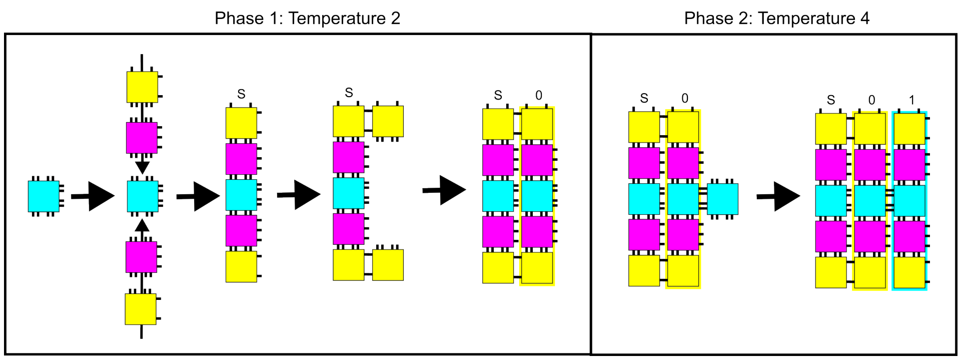


Figure 6 Depiction of the growth during the first 2 temperature phases of the universal number encoder for temperatures $\{2, 3, 4\}$ when encoding the number 100_{10}, whose full encoding is the string "S0101000001000011" which requires the temperature sequence $[2, 4, 2, 4, 2, 3, 2, 3, 2, 4, 2, 3, 2, 3, 4, 3]$. The growth of phase 1, at temperature 2, begins from the seed tile and terminates once the first column (which will ultimately be the leftmost of the assembly) and second column are complete and terminal (at that phase's temperature). (Note that phase 1 is a special case in which two columns grow, but during all other phases only a single additional column grows.) Phase 2 begins from those two completed columns, and in this example its temperature is 4, so only tiles with strength-4 west-facing (input) glues are unblocked during this phase and thus able to attach, allowing the third column to then complete the growth that begins from the blue tile. (Note that the only glues that have blockers are west-facing glues of columns.). The terminal assembly of phase 2 consists of the first three completed columns of which each pair is bound together by exactly the necessary number of glues of the respective strengths to be stable at the highest system temperature, in this case 4.

Listing 2 Code used to generate the temperature sequence for a turtle graphics path

```
def get_temperature_sequence_for_path(path):
    temperature_sequence = []
    B = [4]
    L = [3]
    R = [2, 3]
    S = [2, 4]

    temperature_sequence += B
    b_reflected = False
    for i in range(len(path)):
        move = path[i]
        if move == 'S':
            temperature_sequence += S
        elif move == 'R':
            if b_reflected:
                temperature_sequence += L
                b_reflected = False
            else:
                temperature_sequence += R
        elif move == 'L':
            if b_reflected:
                temperature_sequence += R
            else:
                temperature_sequence += L
                b_reflected = True
        if (move in ['L', 'R']):
            temperature_sequence += B
    temperature_sequence += L
    return temperature_sequence
```

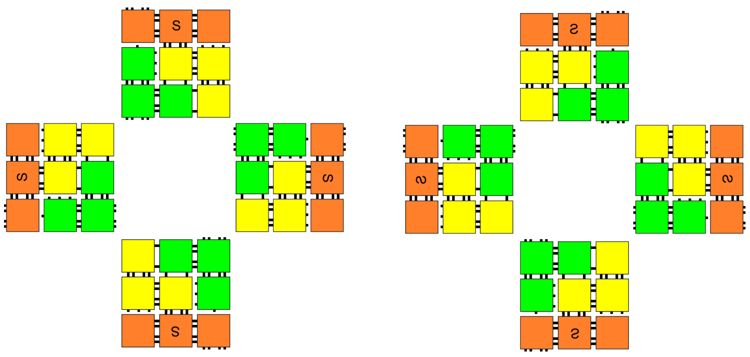


Figure 7 Macrotiles used in the universal shape builder. (Left) The four rotated versions of the *S* type of macrotile, (Right) the four rotated versions of the reflected type of macrotile.

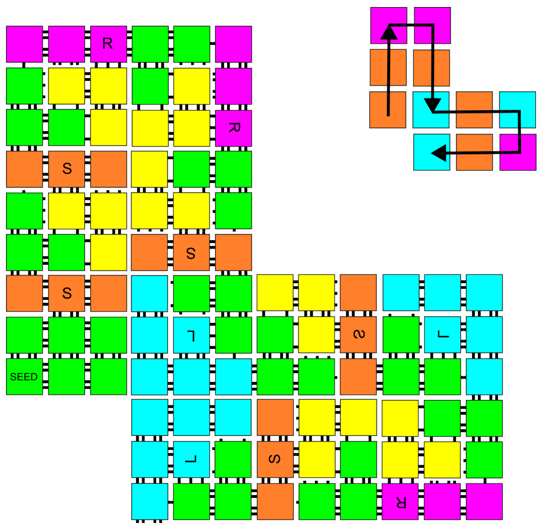
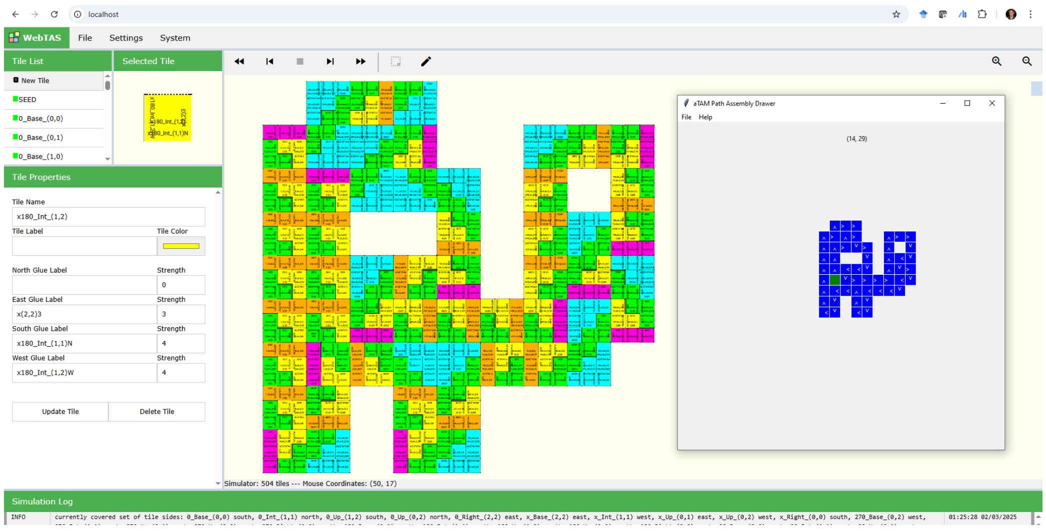


Figure 8 Example shape building example.



Tile Properties

Tile Name: x180_Int_(1,2)

Tile Label:

Tile Color:

North Glue Label		Strength	0
East Glue Label	x(2,2)3	Strength	3
South Glue Label	x180_Int_(1,1)N	Strength	4
West Glue Label	x180_Int_(1,2)W	Strength	4

Simulator: 504 tiles --- Mouse Coordinates: (50, 17)

Simulation Log

```
INFO    currently covered set of tile sides: 0_Base_(0,0) south, 0_Int_(1,1) north, 0_Up_(1,2) south, 0_Up_(0,2) north, 0_Right_(2,2) east, x_Base_(2,2) east, x_Int_(1,1) west, x_Up_(0,1) east, x_Up_(0,2) west, x_Right_(0,0) south, 270_Base_(0,2) west,    01:25:28 02/03/2025
```

Figure 9 An example of a (poorly drawn) turtle-shaped turtle graphics path drawn, saved, and then simulated in the BlockTAM.