Simulation of the Abstract Tile Assembly Model Using Crisscross Slats

Phillip Drake
University of Arkansas, USA

Daniel Hader
University of Arkansas, USA

Matthew J. Patitz
University of Arkansas, USA

Abstract

The abstract Tile Assembly Model (aTAM) provides an excellent foundation for the mathematical study of DNA-tile-based self-assembling systems, especially those wherein logic is embedded within the designs of the tiles so that they follow prescribed algorithms. While such algorithmic self-assembling systems are theoretically powerful, being computationally universal and capable of building complex shapes using information-theoretically optimal numbers of tiles, physical DNA-based implementations of these systems still encounter formidable error rates and undesired nucleation that hinder this theoretical potential. Slat-based self-assembly is a recent development wherein DNA forms long slats that combine together in 2 layers, rather than square tiles in a plane. In this approach, the length of the slats is key; while tiles typically only bind to 2 neighboring tiles at a time, slats may bind to dozens of other slats. This increased coordination between slats means that several mismatched slats must coincidentally meet in just the right way for errors to persist, unlike tiles where only a few are required. Consequently, while still a novel technology, large slat-based DNA constructions have been successfully implemented in the lab with resilience to many tile-based construction problems. These improved error characteristics come at a cost however, as slat-based systems are often more difficult to design and simulate than tile-based ones. Moreover, it has not been clear whether slats, with their larger sizes and different geometries, have the same theoretical capabilities as tiles. In this paper, we show that slats are capable of doing anything that tiles can, at least at scale. We demonstrate that any aTAM system may be converted to and simulated by an effectively equivalent system of slats. Furthermore, we show that these simulating slat systems can be made more efficiently, using shorter slats and a smaller scale factor, if the simulated tile system avoids certain uncommon growth patterns. Specifically, we consider 5 classes of aTAM systems with increasing complexity, from zig-zag systems which grow in a rigid pattern to the full class of all aTAM systems, and show how they may be converted to equivalent slat systems. We show that the simplest class may be simulated by slats at only a $2c \times 2c$ scale, where c is the freely chosen coordination number of the slats, and further show that the full class of aTAM systems can be simulated at only a $5c \times 5c$ scale. These results prove that slats have the full theoretical power of aTAM tiles while also providing constructions that are compact enough for potential DNA-based implementations of slat systems that are both capable of powerful algorithmic self-assembly and possessing of the strong error resilience of slats.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases DNA origami, tile-assembly, self-assembly, aTAM, kinetic modeling, computational modeling

Digital Object Identifier 10.4230/LIPIcs.DNA.30.3

Related Version *Full Version*: `https://arxiv.org/abs/2405.06205` [11]

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

1 Introduction

In self-assembly, simple, disorganized components combine to form structures more complex than themselves, driven primarily by local interactions and environmental conditions. From the crystallization of water molecules into the intricate 6-fold symmetry of snowflakes, to the clustering of space dust and gasses into robust solar systems with mechanisms to mitigate the deleterious effects of debris and radiation, self-assembly processes occur at all scales of nature. Such processes are central to many fields of science and engineering, including the relatively young field of DNA-nanotechnology. Here, synthetic strands of DNA are used, not as a means to store genetic information, but rather as building blocks for nano-scale structures, far too small to assemble using conventional human building techniques. Taking advantage of the base-pairing dynamics of DNA, synthetic strands can be mixed in solution under carefully tuned conditions so that they naturally combine to form incredibly precise shapes [10, 22, 25, 36, 42], and even follow embedded algorithms [13, 15, 26, 32, 38, 47, 48]. On an atomic scale, DNA is far too complex to completely and efficiently model so heuristics and simplifications are often used when designing DNA-based self-assembling systems. Tile-assembly models are one such simplification that have seen great success in facilitating the design of such systems. In tile-assembly, it is assumed that DNA strands are designed so that they tend to combine into small, generally rigid units called *tiles* resembling squares (or sometimes other shapes). These units are augmented with extra lengths of single-stranded DNA that dangle from their sides (often called “glues” or “handles”) to enable individual units to selectively combine with one another. The utility of tile-assembly comes from its simplicity and relationship with existing models in mathematics and computer science. While individual DNA strands are difficult to model, when designed to behave like tiles their self-assembly is relatively well understood and many important dynamics can be easily captured by simple mathematical rules.

Theoretically, tile-assembly models have been extensively studied, and models such as the abstract Tile-Assembly Model (aTAM) have been shown to be algorithmically universal in that they are capable of simulating arbitrary Turing machines [23, 34, 41, 43]. Practically, tile-assembly models have seen significant use as design tools for complicated DNA-based nano-structures [13, 47]. There are however a few key difficulties that arise when attempting to realize tile-based DNA constructions. One primary difficulty is nucleation. To ensure that the self-assembly process occurs as expected, it is generally important that assembly begins from a chosen starting *seed* structure; however it can be extremely difficult to guarantee that growth does not begin spuriously by the improbable combination of a small number of tiles away from the seed. Using conventional approaches to DNA-based tile-assembly, spurious nucleation is a major hurdle to building large structures. Another difficulty comes from so-called *growth errors*. While tiles may be designed so that the correct tile attachments are thermodynamically preferred, it is unlikely that erroneous attachments can be prevented altogether. Typically such errors are short lived due to the entropic penalties they incur, but if enough occur in quick succession and in just the right way, it’s possible for the errors to become locked-in. Techniques such as proofreading (in various forms) [3–5, 35, 39, 40, 44] have been developed to mitigate these problems, but they still act as a major obstacle to larger scale DNA-based tile-assembly.

One recent development, however, has seemingly overcome both of these problems through the use of *slat*-shaped DNA units [7, 12, 29, 45] rather than square tile-shaped ones. Unlike tiles which attach to at most 4 neighbors and combine in a plane, slats are long and designed to attach in multiple layers so that a single slat may span across and attach to dozens of others. For a square tile where 2 of its sides attach to an existing assembly, erroneous attachments often occur when just one of the sides correctly binds to the assembly. One of two attachments is still relatively strong and an erroneous tile may remain attached for a substantial amount of time. Even worse, it is only really necessary for 4 or so individual tiles to coincidentally co-locate for spurious nucleation to occur. While unlikely, this is almost guaranteed to happen frequently in a mass-action system on the scale of moles. Because each slat needs to attach to 8 or even 16 others to achieve a stable bond, erroneous attachments are generally much shorter lived and less likely to lock-in, and the likelihood of spurious nucleation drops precipitously (effectively to zero [29]).

In the lab, slats are generally implemented either using DNA-origami or as individual strands of DNA. In the DNA-origami motif, slats are often 6-helix bundles (a very common origami construction) with single-stranded DNA "handles" extending from one side. While both techniques are still novel, origami-based slats have been demonstrated to be incredibly robust to spurious nucleation and computer simulations have indicated that slats naturally exhibit error correcting behavior since individual erroneous attachments have little effect on correct growth later in the assembly process [8]. Theoretical models of slat-assembly have been introduced, naturally expanding on tile-assembly models, but little is currently known about their dynamics. In this paper, we consider the abstract Slat Assembly Model (aSAM) introduced in [8], and investigate its relationship to the aTAM. Specifically we consider the extent to which aTAM tiles may be simulated by aSAM slats. In this context, simulation refers to "intrinsic simulation" a notion borrowed from the study of cellular automata [30, 31] and which has been used extensively to compare tile-assembly models and develop a rich complexity theory for them [1, 9, 16, 18, 20, 21, 27, 46]. Unlike typical simulations between models of computation, where the dynamics of one model are captured symbolically by the dynamics of another, intrinsic simulation is inherently geometric. For a system S, be it of tiles or slats, to simulate another system S' requires that S "looks like" S' when zoomed-out and furthermore that any order of attachments in S' may be replicated by attachments in S.

Our Results

In this paper, we show that all systems in the aTAM may be intrinsically simulated by aSAM systems. Moreover, we show that if one is willing to forgo some less useful dynamics of the aTAM, then this simulation may be done quite efficiently, both in the scale factor required for the simulation and in the complexity of the necessary slats. Specifically, we consider 5 different classes of aTAM systems of increasing complexity. The first class, *zig-zag* systems, are still fully capable of Turing universal computation, but are restricted to growing solely in a back-and-forth pattern. The second class of systems, called *standard* systems, represents a simplified set of aTAM dynamics common to most theoretical constructions. These standard systems make simplifying assumptions such as requiring that no tiles mismatch with their neighbors, requiring each tile to attach with no more strength than necessary, and requiring that only 1 terminal assembly is possible. Despite this, all but the most convoluted theoretical aTAM constructions may generally be defined as standard systems. In standard systems, it is assumed that no tiles attach using "across-the-gap" cooperation, where a tile binds to 2 tiles of an existing assembly that are not adjacent to one another. Such attachments are generally more difficult to simulate and rarely, if ever, appear in physically implemented tile-based systems. Still, the 3rd class of aTAM systems considered in this paper are standard systems augmented with the ability to perform across-the-gap cooperation. In the 4th class we allow tiles to mismatch with their neighbors so long as they attach with sufficient strength and the system is directed (i.e. makes a unique assembly), and our 5th consists of all aTAM systems.

Table 1 An overview of our results. For each class of aTAM systems, corresponding to each of our theorems, we list: the simulation scale factor (the size of our macrotiles), the greatest number of slats that appear in any macrotile, and the largest slat length used during the simulation. Defined later, macrotiles represent blocks of slats which simulate individual tiles and c is the cooperativity.

aTAM class	Result	Macrotile size	Greatest num slats	Greatest slat length
Zig-zag	Thm.1	$2c \times 2c$	$4c$	$3c$
Standard	Thm.2	$3c \times 3c$	$8c$	$3c$
Standard plus across-the-gap	Thm.3	$3c \times 3c$	$8c$	$4c$
Directed temperature-2	Thm.4	$4c \times 4c$	$10c$	$4c$
Nondeterministic (full aTAM)	Thm.5	$5c \times 5c$	$13c$	$5c$

Table 1 details our results with respect to a parameter c of our aSAM systems called the "cooperativity" or sometimes "coordination number" of the slats. This number may be freely chosen, independent of our results, and its value effectively describes how many functionally redundant attachment domains appear along the length of each slat. In practice, increasing this number will result in a slat system more robust to spontaneous nucleation and growth errors at the cost of requiring longer and more numerous slats. In our results, we show that we can simulate arbitrary aTAM systems using slats of length no greater than $5c$ at a scale factor of $5c$, that is each aTAM tile is represented by a $5c \times 5c$ block of slats. We further show that this may be optimized to slats of maximum length $3c$ with a $2c$ scale factor for zig-zag systems. For classes in between the zig-zag systems and full aTAM, we show that the scale factor and maximum slat length grow accordingly. We present these results in increasing complexity (and note that software for converting classes of aTAM systems to slats, simulate their self-assembly, and visualize the results can be found online [17]. It should be noted that a difference between a scale factor of $2c$ and $5c$ is negligible in a purely theoretical context. The real motivation for exploring and simulating different families of aTAM systems is to try and find "practical" transformations from the logic of tile-based assembly into error-robust slats which may be implementable. These constructions therefore are not only intended to show that aTAM dynamics may be simulated by slat dynamics, but also serve to illustrate the difficulties that arise when one tries to do so and how these difficulties may affect the slats necessary for simulation. Furthermore, while slat-based self-assembly is still in its infancy, we are optimistic that these constructions, while presented here purely theoretically, may provide designs that help to physically realize them in the not too distant future.

Due to page constraints, many technical details of our results are omitted from this version of the paper. A version including a full technical appendix may be found on Arxiv [11].

2 Preliminary Definitions and Models

In this section, we provide definitions and overviews of the models and concepts used throughout the paper.

2.1 The abstract Tile Assembly Model

Our conversions begin from systems within the abstract Tile-Assembly Model [43] (aTAM). These definitions are borrowed from [16] and we note that [37] and [24] are good introductions to the model for unfamiliar readers.

Let $\mathbb{N}$ be the set of non-negative integers, and for $n \in \mathbb{N}$, let $[n] = \{0, 1, ..., n-2, n-1\}$. Let 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*. There is a single glue of strength 0, referred to as the *null* glue. 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[1] $\alpha : \mathbb{Z}^2 \dashrightarrow T$. 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.

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*). 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. 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* of an assembly sequence 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 $\mathcal{A}_\square[\mathcal{T}]$. If $|\mathcal{A}_\square[\mathcal{T}]| = 1$, i.e., there is exactly one terminal assembly, we say that $\mathcal{T}$ is *directed*.

2.2 Classes of aTAM systems

In [43], the aTAM was shown to be computationally universal when $\tau = 2$, but this is not the case when $\tau = 1$ [28]. Furthermore, any aTAM system with $\tau = 1$ can trivially be transformed into a $\tau = 2$ system by changing all of its strength-1 glues to be strength-2; and in any aTAM system where $\tau = 2$, any glue whose strength is greater than 2 may trivially be replaced by a glue of strength $= 2$ without changing any behaviors of the system. Additionally, any aTAM system with $\tau > 2$ may be simulated by a system with $\tau = 2$ [9]. Therefore, all results in this paper will only discuss aTAM systems with $\tau = 2$ and it will be assumed that, other than the *null* glue of strength 0, all glues in aTAM systems are of strength 1 or 2. When a tile initially binds to an assembly in a $\tau = 2$ system, it must immediately bind with at least (1) a single strength-2 glue, or (2) two strength-1 glues (we

[1] we use the dashed arrow $\dashrightarrow$ to indicate a partial function

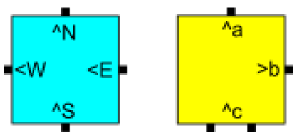


Figure 1 Examples of IO-marked tile type signatures: (Left) The light blue tile's signature is Input=(S,1),(E,1), Output=(N,1),(W,1). (Right) The yellow tile's signature is Input=(S,2), Output=(N,1),(E,1).

call this a *cooperative* attachment). This is because the sum of the bond strengths must be ≥ 2. If, when a tile initially binds to an assembly, it does so by immediately forming bonds of strength > 2, we call this *overbinding*. When a tile t attaches into a location (x, y) by cooperatively binding to tiles on opposite sides of each other (i.e in locations $(x-1, y)$ and $(x+1, y)$, or $(x, y-1)$ and $(x, y+1)$), we say that t has attached *across-the-gap*.

When two tiles in adjacent locations do not share matching non-null glues on their abutting faces, we say that their glues are *mismatched* and note that this is only possible when their other glues contribute a cumulative attachment strength of at least 2.

Let the symbols "$\vee$", "$<$", "$\wedge$", "$>$" be called the *input markings* for the directions N, E, S, W, respectively and the *output markings* for the directions S, W, N, E, respectively. (Visually, if the input markings were placed on the corresponding sides of a tile, they would be "pointing into" the tile, and vice-versa for output markings) We say that a tile is *IO-marked* if a subset of its glues whose sum is ≥ 2 have input markings as prefixes to their labels, and all other non-null glues have output markings as prefixes to their labels. Since each direction has a unique input marking, and the input marking of each direction is the same as the output marking of the opposite direction, it is clear that for glues to match and form a bond, an input-marked glue on any given tile side d can only bind with an output-marked glue on the opposite side of another tile, and vice versa. Note that it is possible to convert any aTAM system to an equivalent IO-marked aTAM system. Furthermore this conversion can be done so that each IO-marked tile has a minimal set of input glues, that is all input glues on a tile are necessary for the attachment[2]. This ensures that assemblies made of IO-marked tiles always only have output glues exposed. See Section A.1 of the Technical Appendix for examples conversions from unmarked to IO-marked tile types. Given an IO-marked tile type t, we denote its *signature* as the string "Input=" followed by a pair for every input side d of t, consisting of d and the integer strength of the glue on side d of t, plus the string "Output=" followed by a pair for every output side d of t, consisting of d and the integer strength of the glue on side d of t. See Figure 1 for examples. Additionally, with this notation, multiple strengths may be assigned to each direction, in which case the notation refers to a set of signatures, one for each combination of glue strengths assigned to each side.

Here we provide a quick overview of the different classes of aTAM systems considered in this paper. Formal characterizations may be found in Section A.2 of the Technical Appendix. It is assumed that all classes are IO-marked. In the first class, called *zig-zag* systems, tiles never present an input glue to the south, instead growth occurs northward in rows that alternate between eastward and westward growth as illustrated in Figure 2. Most tile attachments are cooperative except on the edges of the assembly and when a new row is started. Despite being the most restricted class of models considered in our results, this

[2] If a tile could attach in multiple ways using different combinations of input glues it is split into multiple IO-marked tiles representing the unmarked tile, each with a different subset of input glues

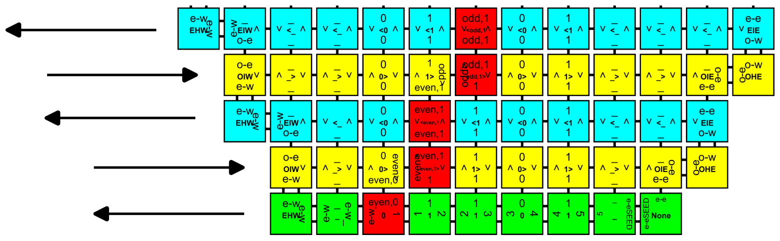


Figure 2 An example zig-zag aTAM system that simulates a Turing machine. The seed tile is the rightmost of the bottom row. The first row (green) grows right to left. After growing upward by one tile, the second row grows left to right and extends one extra tile beyond the row below. Subsequent rows continue to alternate direction and extend in length by 1. Each row represents a configuration of the Turing machine with each tile representing a tape cell, the north glues representing the contents of each cell, and the red tiles showing the location of the simulated tape head and current state of the machine. If a row is growing in the direction in which the tape head needs to move after the last transition, that occurs. If it is growing in the opposite direction, the tape head and state remain the same for that row, and then the next row (which will be of alternating direction) simulates the head movement and state change.

class of TASs is still capable of simulating the execution of arbitrary Turing machines. The next class consists of what we call *standard* systems. These are directed systems where all tiles attach with exactly enough glues to meet the temperature threshold (which is 2), no tiles attach across-the-gap, and no mismatches occur. We call such systems "standard" because, except for the most convoluted theoretical constructions, most systems defined in aTAM literature tend to satisfy these conditions or can easily be altered to satisfy these conditions. The third class considered consists of standard systems where across-the-gap attachments are allowed. The fourth class additionally allows mismatches but must still remain directed (i.e. only 1 final assembly is possible), in other words this class represents all directed temperature-2 systems. And finally, the fifth class consists of all aTAM systems.

2.3 The abstract Slat Assembly Model

The abstract Slat Assembly Model (aSAM), originally introduced in [8], is a generalization of the aTAM. Since most of its definitions are analogous to those of the aTAM, in this section we provide an informal overview. (Formal definitions can be found in Appendix A.3 of the Technical Appendix.) The primary difference between slats and tiles is that the former are defined as $n \times 1 \times 1$ polyominoes of cubes in 3D space. Therefore, with slats we expand our list of directions and sides to also include "Up" ($+z$ direction) and "Down" ($-z$ direction), resulting in the set of face directions $D = \{N, E, S, W, U, D\}$. Similar to tiles, slats can have *glues* (also referred to as *handles*) on each of their $4n + 2$ faces. Each glue is identified by a string *label*, and a non-negative integer *strength*. Each glue has a complementary glue which shares its strength. In this paper we will often denote complementary glues using the same labels but with one appended by an asterisk (e.g. "label" and "label*"). Furthermore, we make a distinction between *slats* and *slat types*, the latter being just a description of the glues and length of a slat with no defined position or orientation. The position and orientation of slats is restricted to the 3D integer lattice and two slats which sit incident to one another are said to be *attached* or *bound* with strength s if they share complementary glues of strength s on their abutting faces. An *assembly* is simply a set of slats such that no two occupy the same coordinates in $\mathbb{Z}^3$.

A *slat assembly system* (SAS) $\mathcal{S} = (S, \sigma, \tau)$ consists of a finite set of slat types S, an assembly σ called the *seed assembly* that acts as the starting point for growth, and a positive integer τ called the *binding threshold* (a.k.a. *temperature*). The binding threshold describes the minimum cumulative glue strength needed for a slat to stably attach to a growing assembly. Growth in the aSAM is described by a sequence of slat attachments. Any slat which could sit on the perimeter of an assembly so that it would be attached to other slats with a cumulative strength meeting the binding threshold is a candidate for attachment, and attachments are assumed to happen non-deterministically. Any assembly that could result from a sequence of slat attachments beginning from the seed assembly of a SAS $\mathcal{S}$ and using only those slat types in the slat set of $\mathcal{S}$ is said to be *producible* in $\mathcal{S}$. Any assembly that permits no additional slat attachments is called *terminal*.

For all results of this paper, we work within a restricted class of systems of the aSAM satisfying the following conventions. Slat types intended to be horizontally aligned always bind in the plane $z = 1$ and we only assign glues to their D faces, using only the "starred" versions of glue labels (i.e. those with the "$*$" symbol). Vertically aligned slat types always attach in the plane $z = 0$, and we only assign glues to their U faces, using the "un-starred" versions of glue labels. Additionally, we ensure that no two slats share more than one pair of complementary glues. Furthermore, all glues on slats are assumed to be strength-1 and each slat can only bind to any other single slat with a single glue. Therefore the temperature parameter τ effectively becomes the *cooperativity* of a system (a.k.a. the *coordination number*, as used in [29, 45]). That is, if $\tau = c$, then each slat must cooperatively bind with c other slats in order to attach to an assembly. We impose these restrictions on our designs so that their behavior is similar to the slat systems successfully experimentally demonstrated in [45]. Furthermore, systems with these restrictions allow for more efficient computer simulation.[3]

2.4 Definition of simulation of an aTAM system by an aSAM system

Here we describe what is meant by an aSAM system *intrinsically simulating* an aTAM system. From here on, the term "simulation" will refer to intrinsic simulation. This definition is analogous to the typical definition of intrinsic simulation between aTAM systems which may be found in [18]. Here we assume that $\mathcal{T} = (T, \sigma, \tau)$ is a TAS being simulated by the SAS $\mathcal{S} = (S, \sigma', \tau')$. For $\mathcal{S}$ to simulate $\mathcal{T}$, it must be the case that $\mathcal{S}$ "looks like" $\mathcal{T}$ at scale. To this end, we also require the definition of a *macrotile representation function* R and a *scale factor* m. The function R maps $m \times m$ blocks, called *macrotiles*, of slat locations (which may or may not contain slats) to individual tile types in T. To be precise, R is a partial function since it might be the case that a macrotile does not immediately map to a tile type in T. Once R does map a macrotile to a tile type however, it must continue to map the macrotile to the same tile type regardless of any additional slats that attach within the macrotile. This property reflects the fact that tiles in the aTAM may not change type or detach after they have attached. When a slat attachment causes a macrotile to map under R to tile type t for the first time, it is said that the macrotile *resolves* into t. Applying the macrotile representation function R to each macrotile of an $\mathcal{S}$-assembly yields a $\mathcal{T}$-assembly. This process defines the *assembly representation function* R^* from $\mathcal{S}$-assemblies to $\mathcal{T}$-assemblies. While it is allowed for macrotiles to contain slats even if it does not map to a tile type, we only allow slats to attach in macrotiles adjacent (not-diagonally) to ones which have already

[3] A Python-based graphical simulator for the aSAM$^-$ called SlatTAS can be downloaded from `self-assembly.net` via a link on the page here [19].

resolved. This prevents a "simulator" from growing slats to perform complex calculations in region that will never map to a tile in the simulated system and ensures that slats only grow in macrotile locations that could feasibly map to tiles. The macrotile blocks that admit slat attachments but have not yet resolved are called *fuzz* regions since at a scale they resemble small hairs growing along the side of a simulated assembly.

For $\mathcal{S}$ to simulate $\mathcal{T}$, 3 conditions must be satisfied. First, $\mathcal{S}$ and $\mathcal{T}$ must have *equivalent productions* meaning that R surjectively maps all $\mathcal{S}$-assemblies to $\mathcal{T}$-assemblies and all terminal $\mathcal{S}$-assemblies to terminal $\mathcal{T}$-assemblies. Second $\mathcal{T}$ must *follow* $\mathcal{S}$ meaning that all sequences of slat attachments in $\mathcal{S}$ map to corresponding slat attachments in $\mathcal{T}$ ($\mathcal{T}$ can only do what $\mathcal{S}$ does). And finally, $\mathcal{S}$ must *model* $\mathcal{T}$ meaning that all sequences of tile attachments in $\mathcal{T}$ have at least one corresponding sequence of slat attachments in $\mathcal{S}$ ($\mathcal{S}$ can only do what $\mathcal{T}$ does). The formal definition of *models* also has a provision that ensures all non-deterministic attachments in $\mathcal{T}$ are truly simulated by non-deterministic attachments in $\mathcal{S}$ rather than being predetermined in advance.

3 Results

In this section we present our results showing that classes of aTAM systems, with increasingly complex dynamics, can be simulated by aSAM systems. Each result is proven by construction and associated software for designing, converting, simulating, and visualizing these systems can be found online [17]. Note that the first four results are for classes of aTAM systems defined to have $\tau = 2$, but each construction trivially works for $\tau = 1$ as well, simply by treating all glues of the simulated aTAM systems as τ-strength. The final result is presented for $\tau = 2$, but explanation of a simple extension to handle arbitrary values of τ is presented in the Technical Appendix (as are the details of most proofs) due to space constraints.

3.1 Zig-zag systems

Zig-zag systems are particularly interesting because, despite their incredibly limited range of dynamics, they are computationally universal [6, 20, 23, 33, 34]. Our first result shows that any zig-zag aTAM system can be simulated by an aSAM system with macrotiles of size only $2c \times 2c$.

▶ **Theorem 1.** *Let $\mathcal{T} = (T, \sigma, 2)$ be an arbitrary zig-zag aTAM system. For any $c > 2$ such that $c \mod 2 = 0$, there exists an aSAM system $\mathcal{S} = (S, \sigma', c)$ and macrotile representation function R such that $\mathcal{S}$ simulates $\mathcal{T}$ under R using cooperativity c and macrotiles of size $2c \times 2c$. Furthermore, the longest slat in S is of length $3c$.*

Proof. We prove Theorem 1 by construction, and thus, starting with arbitrary zig-zag aTAM system $\mathcal{T} = (T, \sigma, 2)$ and given any $c > 2$ such that $c \mod 2 = 0$ we show how to create aSAM system $\mathcal{S} = (S, \sigma', c)$ and macrotile representation function R such that $\mathcal{S}$ simulates $\mathcal{T}$ under R. First, without loss of generality we assume that $\mathcal{T}$ grows its first row from its seed tile from the right to the left (i.e. "RtoL"), and then its second row grows immediately above that, from left to right (i.e. "LtoR"), and then all subsequent rows zig-zag from RtoL then LtoR. (The construction could simply be rotated appropriately to handle any direction of zig-zag growth.)

Each tile in $\mathcal{T}$ is simulated by a macrotile of size $2c \times 2c$ in $\mathcal{S}$. For $c = 4$, this means that the 8×8 square whose southwest coordinate is $(8i, 8j)$, for every $i, j \in \mathbb{Z}$, will map under R to either empty space or to a tile in $\mathcal{T}$. An example is shown in Figures 3 and 4.

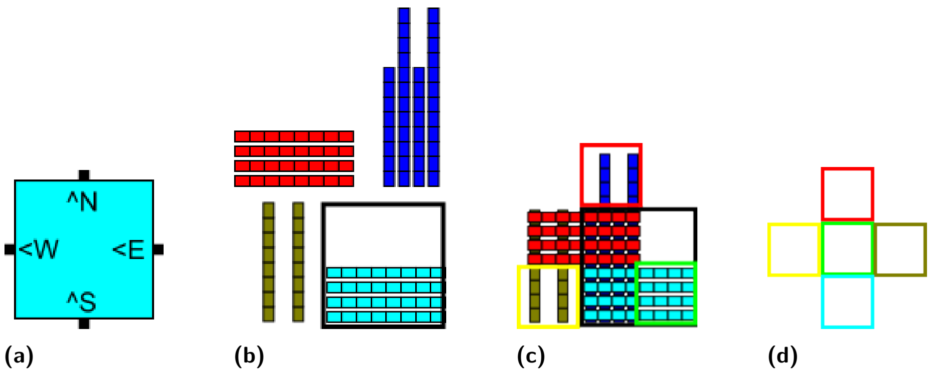


Figure 3 (a) An IO-marked tile type t from a zig-zag aTAM tile set, for a row that grows RtoL. It has strength-1 inputs on the south and east, and strength-1 outputs on the north and west. (b) A set of 4 slat groups for the macrotile simulating t at $c = 4$. The light blue group contains body slats that are entirely within the square of the macrotile (depicted by the black square) which they cause to resolve to t. The dark blue group contains two body slats and two output slats (i.e. the two extending into the north macrotile to serve as the north output of strength-1). The red and gold slat groups combined are the output slats that serve as the west output and extend into the western neighboring macrotile location. (c) An example of the assembled $2c \times 2c$ macrotile for t, with cells marked to show the portions of t that they represent, following the conventions of (d). (d) A cell enclosed in a green square represents the cell in which the initial body slats of a macrotile bind, causing it to resolve to t. The cells enclosed in red, gold, light blue, and yellow squares denote the cells in which the slats expose glues representing the output glues of the north, east, south, and west sides of t, respectively.

Each slat in a macrotile is of a unique type.[4] We use the term *slat group* to refer to each set of c (or sometimes $c/2$) slats that are oriented in the same direction and grouped together (both logically, and also in that each slat in a slat group can attach to a growing assembly at exactly the same time as the others in that group). (For example, in Figure 3b there are 4 slat groups.) For convenience, we will characterize all of the slat types of a macrotile in two categories: (1) *body slats*: slats that are completely contained within one of the $2c \times 2c$ macrotile regions and either (a) their binding causes that region to map to a tile in $\mathcal{T}$ under R, or (b) they bind after that macrotile has resolved, and (2) *output slats*: slats that either (a) cause a macrotile to map to a tile of $\mathcal{T}$ but also extend into a neighboring macrotile location, or (b) bind in a macrotile location this is unresolved both before and immediately after their binding. (In Figure 3c, the 4 light blue, and the shorter 2 of the dark blue slats are body slats. The longer two dark blue, and all of the red and gold slats are output slats. Furthermore, the longer dark blue slats are of length $3c$, which is the greatest length of slats in this construction.)

Let $t_n \in T$, for $0 \leq n < |T|$, be the nth tile in tile set T. We will refer to the string "t_n" as the unique name of t_n. We now discuss how the slats that form a macrotile simulating t_n are designed. Given the directions of growth, and the dynamics of a zig-zag aTAM system, the following list contains all possible valid signatures for t_n of any zig-zag system (with the trivial exception that some tile type could have one or more fewer outputs, and the binding of such a tile into an assembly would cause growth to terminate and the assembly to become terminal, as such tiles are trivially handled by macrotiles without corresponding output slats).

[4] Using techniques of [8], it is possible to reuse slat types within macrotiles to reduce the slat complexity, but for ease of explanation we present our constructions without that optimization.

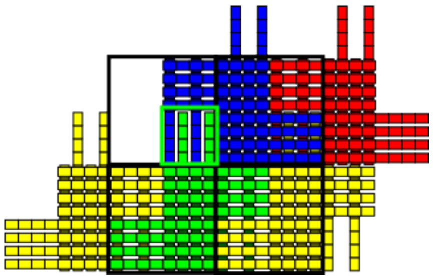


Figure 4 An example of a portion of an assembly composed of $2c \times 2c$ macrotiles (some partial) simulating a zig-zag aTAM system for $c = 4$. Four of the macrotile locations are outlined in black squares. The macrotile simulating t from Figure 3 would attach into the top left such macrotile location. Its (light blue) body slats would attach to the 2 west output slats from the macrotile to the east (dark blue) and the 2 north output slats from the macrotile to the south (green). These light blue slats can bind in any order, and as soon as one binds, the macrotile resolves to t. Due to the fact that $\tau = 4$, only once they have all 4 bound can the north output slats (dark blue) bind. Only once all 4 of those have bound can the red output slats bind, then finally the 2 gold output slats. Thus, the growth of a macrotile is well-ordered, and outputs are only presented after a macrotile resolves, enforcing the restrictions of simulation.

1. Initial (seed) row tiles (Figure 5):
   a. Seed tile: Input=∅, Output=(W,2),(N,1)
   b. Row interior tiles: Input=(E,2), Output=(W,2),(N,1)
   c. Leftmost tile: Input=(E,2), Output=(N,2)
2. LtoR row tiles (Figure 6):
   a. Leftmost tile: Input=(S,2), Output=(E,1),(N,1)
   b. Row interior tiles: Input=(W,1),(S,1), Output=(E,1),(N,1)
   c. Right row pre-extension tile: Input=(W,1),(S,1), Output=(E,2),(N,1)
   d. Right row extension tile: Input=(W,2), Output=(E,2),(N,1)
   e. Rightmost tile: Input=(W,2), Output=(N,2)
3. RtoL row tiles (Figure 7):
   a. Rightmost tile: Input=(S,2), Output=(W,1),(N,1)
   b. Row interior tiles: Input=(E,1),(S,1), Output=(W,1),(N,1)
   c. Left row pre-extension tile: Input=(E,1),(S,1), Output=(W,2),(N,1)
   d. Left row extension tile: Input=(E,2), Output=(W,2),(N,1)
   e. Leftmost tile: Input=(E,2), Output=(N,2)

Figures 5-7 show tiles with those signatures and their corresponding *macrotile templates*, which are sets of slat groups that correspond to the particular set of input and output glue directions and strengths that a simulated tile has. Note that extension and pre-extension tiles are those which grow in the zig-zag pattern, but using a strength-2 glue rather than cooperation from below. To build S, for each t_n, we instantiate the macrotile template associated with t_n's signature. Instantiating a macrotile consists of first making a unique copy of each slat type in the macrotile template whose name has the prefix "t_n" prepended to the unique name of that slat type in the macrotile template. We will refer to the set of slats for the macrotile template instantiated for t_n as S_n. For every location where a vertical slat of S_n is at the same (x, y) coordinates as a horizontal slat of S_n (but under it since it will be at $z = 0$ and the horizontal slat at $z = 1$), an un-starred glue unique to that location is placed there on the vertical slat, and the starred complement of that glue is placed on

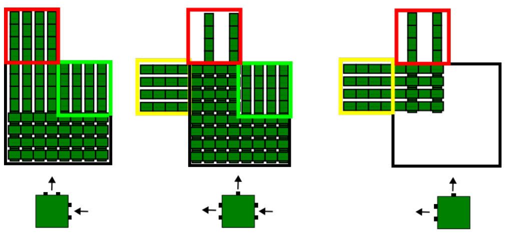


Figure 5 The tiles for the seed and initial row of a zig-zag aTAM system (that grows RtoL from the seed tile), and their corresponding slat-based macrotile templates for $c = 4$. The rightmost corresponds to the seed tile, the middle corresponds to the interior tiles of the initial row, and the leftmost corresponds to the leftmost tile of the initial row. Cells are bounded by squares to show their functionality, following the coloring convention from Figure 3d. Note that during simulation the seed macrotile begins fully assembled and additional slats attach to simulate additional macrotiles.

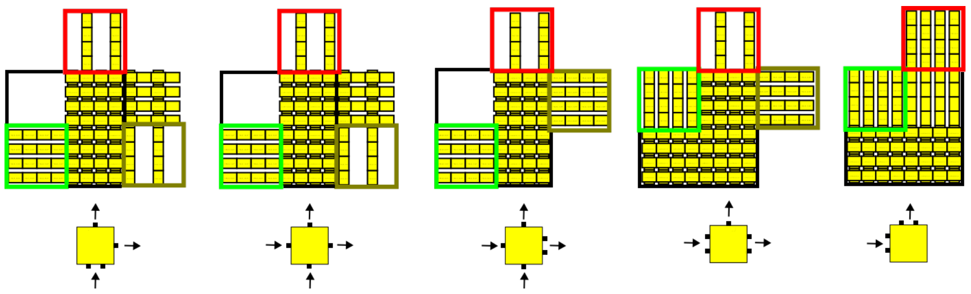


Figure 6 Tile types of all possible valid signatures for tiles of a row that grows LtoR in a zig-zag aTAM system, and their corresponding slat-based macrotile templates for $c = 4$. Cells are bounded by squares to show their functionality, following the coloring convention from Figure 3d.

the horizontal slat. All such glues are also given the prefix "t_n" to ensure that they will not match glues of macrotiles instantiated for any other tiles. These glues are called *interior glues*, since they bind slats of the same macrotile to each other.

The final step of building S_n for t_n is to account for the glues of t_n and to place glues on the slats of S_n to cause their behavior to be simulated. To do this, within the $c \times c$ cell representing a glue of t_n (whose location is determined by the particular macrotile template that matches t_n's signature), we label the glue in each location of each slat with the same glue label as the corresponding glue on t_n, followed by the cell coordinates of the location (i.e. "(i, j)" for $0 \leq i, j < c$, with $(0, 0)$ being the south-westernmost location), followed by a star for glues on horizontal slats. This guarantees that each glue label in each cell is unique. Output glues of strength-2 on t_n are represented by c slats filling a $c \times c$ cell, allowing slats of the opposite orientation to attach to the assembly by binding solely to them (analogous to the strength-2 glue of t_n being sufficient to allow a tile attachment). Output glues of strength-1 are represented by just $c/2$ slats extending across a $c \times c$ cell. Because of this, $c/2$ additional slats extending across that cell from the opposite direction, representing the strength-1 output glue of an adjacent macrotile, are required before the necessary glues are in place to allow body slats for the next macrotile to attach. In this way, cooperative behavior is enforced. An example can be see in Figure 4.

The design of the conventions and macrotile templates guarantee that slats can bind only in desired locations, and that they do so with total binding strength exactly c. For a vertical slat to attach, it must initially bind with c distinct horizontal slats, and vice versa. (Recall that only strength-1 glues are used, and also that all vertical slats have all glues on their U

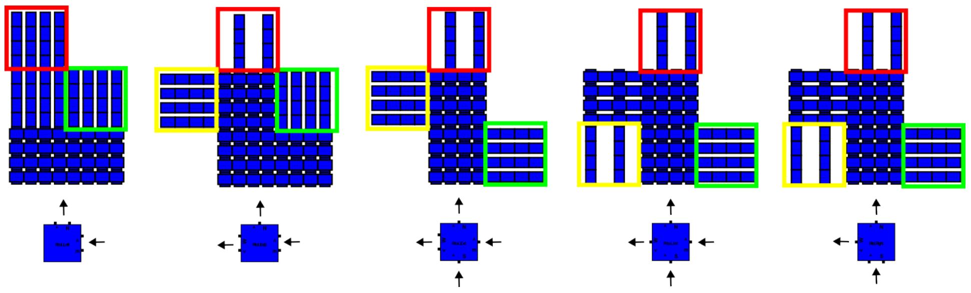


Figure 7 Tile types of all possible valid signatures for tiles of a row that grows RtoL in a zig-zag aTAM system, and their corresponding slat-based macrotile templates for $c = 4$. Cells are bounded by squares to show their functionality, following the coloring convention from Figure 3d.

sides, which are un-starred, and all horizontal slats have all glues on their D sides, which are starred.) From Figure 3 it is clear to see how an individual macrotile (representing a tile in a RtoL row) assembles in a well-ordered progression, causing it to first resolve and only then attach slats that provide the outputs. It is also clear how any tile from T can be converted to a set of slats that will simulate it in a similar way, simply noting the signature for any such tile and the macrotile template for the matching signature, selected from those shown in Figures 5-7.

By inspecting the macrotile templates associated with the valid zig-zag tile signatures, it can be verified that the outputs of any macrotile are always positioned correctly for the binding of body slats of a macrotile that needs to use those as inputs, while keeping that next macrotile in the correct relative position. The careful design of the glues ensures that only the slats designed to attach to any given location can do so. The macrotile representation function R can simply contain a mapping of body slats to the tile types from which they were derived and use that mapping for any macrotile location containing a body slat, while mapping any macrotile location without a body slat to an empty location. The seed σ' simply consists of the set of slats of the macrotile created for the seed tile of $\mathcal{T}$, which has exactly one $c \times c$ cell where there are c slats representing the strength-2 output glue of $\mathcal{T}$'s seed and to which a slat can bind. Starting from this assembly it is also clear to see that, as the macrotiles of $\mathcal{S}$ assemble, there will always be exactly one $c \times c$ cell in which slats can bind. In $\mathcal{T}$, the frontier is always of size 1, and any slats that can attach in $\mathcal{S}$ will either be (1) body slats that cause the macrotile location mapping to that frontier location to resolve under R into the next tile that could attach in $\mathcal{T}$, or (2) body or output slats of the macrotile that was most recently resolved in that frontier location. This provides an inductive argument where the inductive hypothesis is that, given an assembly β producible in $\mathcal{S}$ mapping under R to assembly α producible in $\mathcal{T}$, the exposed glues on β allow exactly one macrotile, mapping to the correct next tile of α under R, to assemble next. This is true of the seed macrotile (the base case), and also given any assembly producible via the macrotiles generated by the macrotile templates shown in Figures 5-7, so the induction holds and $\mathcal{S}$ correctly simulates $\mathcal{T}$ under R. Therefore, $\mathcal{S}$ simulates $\mathcal{T}$, an arbitrary zig-zag aTAM system, under R using cooperativity c and macrotiles of size $2c \times 2c$ with the longest slats being of length $3c$ and Theorem 1 is proven. ◀

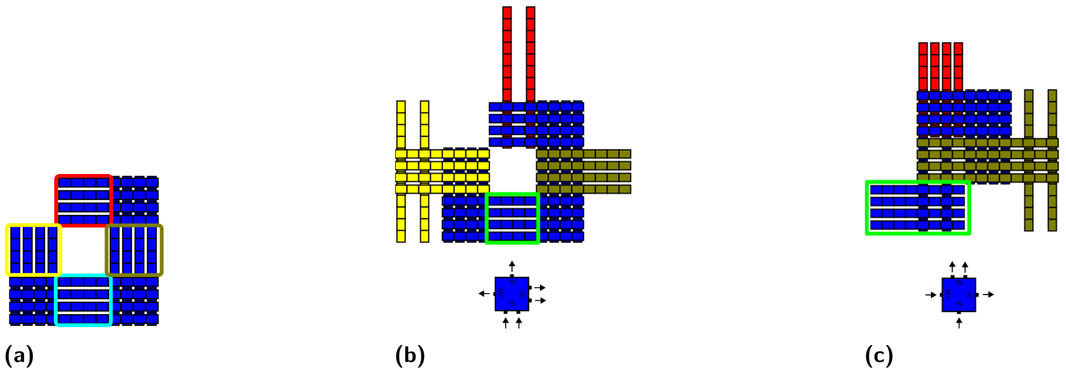


Figure 8 (a) Strength-2 macrotile template for a standard aTAM system. Cells are bounded by squares to show their functionality, and mark cell locations where output slat templates may be added to the macrotile. One of the marked cells may be designated as an input, and have its domains assigned such that they connect with those provided by the output slats of a neighboring macrotile whose output is of the same glue type. Cells are signified using the same color conventions as Figure 3d. (b) Macrotile experiencing south strength-2 input. On the east, output extends in multiple directions in order to allow for cooperation with both north, and south inputs. (c) Macrotile exhibiting south and west strength-1 inputs. Output slat templates are colored in accordance to Figure 3d, and input domain locations are marked with a green box.

3.2 Standard systems

Next, we prove that by only slightly increasing the scale factor of the simulation, i.e. the size of macrotiles, from $2c \times 2c$ for zig-zag systems to $3c \times 3c$, that any standard aTAM system can be simulated by an aSAM system. Since the majority of aTAM constructions in the literature are standard systems (e.g. [23, 34, 37, 41]), this shows that a very modest scale factor can be used to simulate a huge diversity of very complex aTAM systems.

▶ **Theorem 2.** *Let $\mathcal{T} = (T, \sigma, 2)$ be an arbitrary standard aTAM system. For any $c > 2$ such that $c \mod 2 = 0$, there exists an aSAM system $\mathcal{S} = (S, \sigma', c)$ such that $\mathcal{S}$ simulates $\mathcal{T}$ using cooperativity c and scale factor $3c$. Furthermore, the longest slat in S is of length $3c$.*

The simulation construction for standard aTAM systems is very similar to the construction for zig-zag systems, except that a larger set of input and output direction combinations need to be considered. To accommodate this change, the macrotiles during standard aTAM simulations are $3c \times 3c$ instead of $2c \times 2c$, but the argument is essentially unchanged. Strength-2 glues are still simulated by leaving all c slats available for binding in a corresponding macrotile cell while strength-1 glues are simulated by using half this many from each of two outputs. However, the geometries of the macrotiles are a bit different. To handle the more diverse sets of signatures, our construction makes use of macrotile templates of different geometries for tiles with strength-2 input glues (see Figures 8a and 8b) versus those with two strength-1 input glues (see Figure 8c), as well as *output slat templates* that differ for strength-2 versus strength-1 output glues as well as for those that are on vertical sides (north, south) versus horizontal sides (east, west) (see Figure 8b). Otherwise, all of the same techniques from the proof of Theorem 1 apply. Figure 8 shows a few example macortiles.

3.3 Standard with across-the-gap simulation

Next, we prove that by only slightly increasing the maximum slat length of the simulation, from $3c$ to $4c$, any standard aTAM system with across-the-gap cooperation can be simulated by an aSAM system supporting both types of cooperative binding (adjacent and across-the-gap).

▶ **Theorem 3.** *Let $\mathcal{T} = (T, \sigma, 2)$ be an arbitrary standard with across-the-gap aTAM system. For any $c > 2$ such that $c \mod 2 = 0$, there exists an aSAM system $\mathcal{S} = (S, \sigma', c)$ such that $\mathcal{S}$ simulates $\mathcal{T}$ using cooperativity c and macrotiles of size $3c \times 3c$. Furthermore, the longest slat in S is of length $4c$.*

The construction for Theorem 3 is similar in form to the previous two, so here we just describe a few important features, with the full construction being deferred to the Arxiv version of this paper [11] due to space constraints. As with the proof of Theorem 2, there are distinct macrotile templates for tiles with strength-2 inputs, but here there are also distinct macrotile templates for tiles with two strength-1 input glues that are adjacent and those that are across-the-gap. Again, there are output slat templates specific to strengths and orientations. Specifically, across-the-gap cooperation is handled in the center cell of each macrotile. Growth in each macrotile is otherwise very similar. All of the same techniques from the proof of Theorem 1 apply.

3.4 Directed temperature-2 simulation

▶ **Theorem 4.** *Let $\mathcal{T} = (T, \sigma, 2)$ be an arbitrary directed temperature-2 aTAM system. For any $c > 2$ such that $c \mod 2 = 0$, there exists an aSAM system $\mathcal{S} = (S, \sigma', c)$ such that $\mathcal{S}$ simulates $\mathcal{T}$ using cooperativity c and macrotiles of size $4c \times 4c$. Furthermore, the longest slat in S is of length $4c$.*

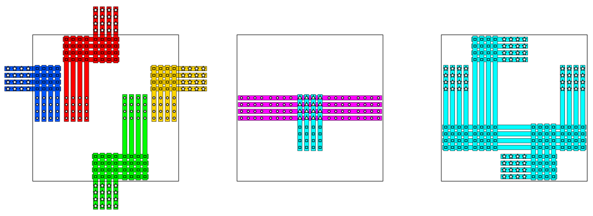
This construction follows the same general format of the previous 3, though we defer it to the Arxiv version of this paper [11] due to space constraints. Just as with the previous simulations, strength-2 aTAM glues are simulated using c slats in neighboring macrotiles, while strength-1 glues are simulated using half that many. The main difference between this simulation and the previous ones comes from the fact that mismatches are allowed to occur in the simulated aTAM system. In order to simulate this behavior, it must be guaranteed that the presence of any additional output slats in a macrotile for any tile type t both do not prevent the macrotile from resolving to t, and do not block any outputs from t apart from those which are already occupied. To accommodate this, the scale factor is increased to provide specific macrotile cells wherein body slats may attach for every combination of potential input glues. To this end, output slats are also generally longer and reach into multiple cells of the adjacent macrotiles in order to ensure that there are cells corresponding to each combination of glues that may contribute to simulate a tile attachment.

3.5 Full aTAM simulation

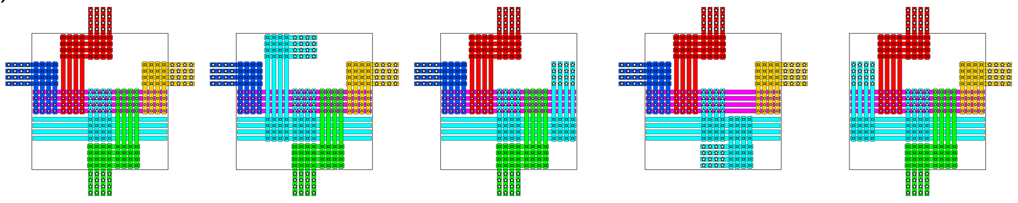
In this section, we present a theorem stating that all temperature-2 aTAM systems can be simulated by aSAM systems and give a brief overview of the proof's construction. Due to space constraints, we sketch our construction and just mention that arbitrary temperatures can be handled. Further details can be found in the Arxiv version [11].

▶ **Theorem 5.** *Let $\mathcal{T} = (T, \sigma, 2)$ be an arbitrary aTAM system. For any $c > 2$ such that $c \mod 2 = 0$, there exists an aSAM system $\mathcal{S} = (S, \sigma', c)$ such that $\mathcal{S}$ simulates $\mathcal{T}$ using cooperativity c with a scale factor of $5c$ and a maximum slat length of $5c$.*

In the construction of this proof, it is assumed that the aTAM system consists of IO-marked tiles (otherwise the method discussed in Section A.1 of the Technical Appendix can be used to make it so), simulation takes place using $5c \times 5c$ macrotiles, and slats are always defined in groups of c. The general layout of the macrotiles does not change significantly



(a)



(b)

Figure 9 (a) Left: Input slats for all directions. Center: Decision slats in the decision rows of a macrotile. Right: Output slats growing in all directions. (b) Left: a macrotile receiving inputs from all 4 of its neighbors. Red slats encode an incoming glue from the north, yellow from the east, green from the south, and blue from the west. Magenta slats attach non-deterministically to the glues presented by these slats and each encode a possible tile from T to which this macrotile may resolve. Cyan slats decide a winner among the magenta slats. The remaining illustrations are example macrotiles which only receive input from 3 sides so the remaining side may act as an output.

with the type of tile being simulated, though some slats may or may not appear depending on whether the simulated tile has glues on all sides. Slats in S may be divided logically into 3 families, *input*, *decision*, and *output* slats, each of which is responsible for a different function. The general form of these slats is illustrated in Figure 9a. When a neighboring macrotile has resolved, it will eventually present glues along its sides indicating which output glues are present on the simulated tile. Input slats attach to these glues and act to move the information about simulated glues to the center of the macrotile forming c horizontal rows. The left of Figure 9a illustrates input slats from the 4 cardinal directions using different colors. Note that the glues holding input slats together are unique to their specific location and the aTAM glue being represented. Consequently, input slats always attach as a group. Once enough input slats have attached, the central horizontal rows encode all the information about present input glues of the adjacent simulated tiles. In these horizontal rows, decision slats may attach (illustrated as magenta in Figure 9a. Decision slats are defined per tile type in the simulated system and the glues present on the decision slats ensure that each may only attach when the corresponding input glues are present, as encoded by the input slats. For instance, when simulating a tile attachment using a strength-2 north glue, the corresponding decision slats will be able to bind solely to the north input slats encoding the respective aTAM glue. On the other hand, when simulating a cooperative attachment, say from the north and west, the corresponding decision slats will only have half the necessary glues to attach to both the north and west input slats. In this way the decision slats for tile type $t \in T$ may only attach when the input slats encoding the input glues of t are present. Note that in the case of overbinding or mismatches, there might be multiple decision slats that may attach in the center rows, allowing for the simulation of an undirected attachment. The specific tile type to which the macrotile resolves is the one encoded by the decision slat that attaches in the northmost decision row. Vertical slats attaching to these decision tiles propagate the information from this northmost decision row into the row of c slats below, which will be where the output slats attach. Once the macrotile has resolved and these vertical slats attach, the corresponding output slats will grow to each side that represents an output glue (and isn't already occupied). Output slats going to all directions are illustrated on the right of Figure 9a, but in actuality, only those corresponding to output glues on the simulated tile will be present. Non-determinism is thus only present in two locations of a macrotile during this simulation. First, multiple decision slats may be able to attach in the decision rows, and second when simulating tiles with mismatched glues, it may be possible for two adjacent macrotiles to present output slats to abutting sides. This is however not a problem since for this to occur, both macrotiles must have already resolved and while it may lead to input glues growing where output glues should have, this only happens in locations dedicated to the corresponding direction and thus cannot affect other parts of the macrotile.

References

1 Andrew Alseth and Matthew J Patitz. The need for seed (in the abstract tile assembly model). In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4540–4589. SIAM, 2023.

2 Sarah Cannon, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Matthew J. Patitz, Robert T. Schweller, Scott M. Summers, and Andrew Winslow. Two hands are better than one (up to constant factors): Self-assembly in the 2HAM vs. aTAM. In Natacha Portier and Thomas Wilke, editors, *STACS*, volume 20 of *LIPIcs*, pages 172–184. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013.

3 Ho-Lin Chen and Ashish Goel. Error free self-assembly using error prone tiles. In C. Ferretti, G. Mauri, and C. Zandron, editors, *10th International Workshop on DNA Computing, DNA10*, volume 3384 of *LNCS*, pages 62–75. Springer Verlag, 2005.

4 Ho-Lin Chen, Rebecca Schulman, Ashish Goel, and Erik Winfree. Reducing facet nucleation during algorithmic self-assembly. *Nano Letters*, 7(9):2913–2919, September 2007. `doi:10.1021/nl070793o`.

5 Holin Chen, Ashish Goel, Erik Winfree, and Chris Luhrs. Self-assembling tile systems that heal from small fragments. In *in Preliminary Proceedings of DNA Computing 13*, pages 30–46, 2007.

6 Matthew Cook, Yunhui Fu, and Robert T. Schweller. Temperature 1 self-assembly: Deterministic assembly in 3D and probabilistic assembly in 2D. In *SODA 2011: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2011.

7 Jie Deng, Dionis Minev, Anastasia Ershova, and William M Shih. Branching crisscross polymerization of single-stranded dna slats. *Journal of the American Chemical Society*, 2024.

8 David Doty, Hunter Fleming, Daniel Hader, Matthew J Patitz, and Lukas A Vaughan. Accelerating self-assembly of crisscross slat systems. In *29th International Conference on DNA Computing and Molecular Programming (DNA 29)(2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023.

9 David Doty, Jack H. Lutz, Matthew J. Patitz, Robert T. Schweller, Scott M. Summers, and Damien Woods. The tile assembly model is intrinsically universal. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, FOCS 2012, pages 302–310, 2012.

10 Shawn M. Douglas, Hendrik Dietz, Tim Liedl, Björn Högberg, Franziska Graf, and William M. Shih. Self-assembly of DNA into nanoscale three-dimensional shapes. *Nature*, 459:414 EP–, May 2009. `doi:10.1038/nature08016`.

11 Phillip Drake, Daniel Hader, and Matthew J Patitz. Simulation of the abstract tile assembly model using crisscross slats. *arXiv preprint arXiv:2405.06205*, 2024.

12 Anastasia Ershova, Dionis Minev, F Eduardo Corea-Dilbert, Devon Yu, Jie Deng, Walter Fontana, and William M Shih. Enzyme-free exponential amplification via growth and scission of crisscross ribbons from single-stranded dna components. *Journal of the American Chemical Society*, 146(1):218–227, 2023.

13 Constantine Glen Evans. *Crystals that count! Physical principles and experimental investigations of DNA tile self-assembly.* PhD thesis, California Institute of Technology, 2014.

14 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. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015), San Diego, CA, USA, January 4-6, 2015*, pages 148–167, 2015.

15 Hongzhou Gu, Jie Chao, Shou-Jun Xiao, and Nadrian C. Seeman. A proximity-based programmable dna nanoscale assembly line. *Nature*, 465(7295):202–205, May 2010. doi:10.1038/nature09026.

16 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.

17 Daniel Hader and Matthew J. Patitz. Abstract slat assembly model (asam). http://self-assembly.net/wiki/index.php/Abstract_Slat_Assembly_Model_(aSAM), 2023. [Online; accessed 9-July-2024].

18 Daniel Hader and Matthew J. Patitz. The impacts of dimensionality, diffusion, and directedness on intrinsic cross-model simulation in tile-based self-assembly. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 71:1–71:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.71.

19 Daniel Hader and Matthew J. Patitz. SlatTAS: A Graphical Simulator for the aSAM^{-}. http://self-assembly.net/wiki/index.php?title=SlatTAS, 2023. [Online; accessed 28-April-2023].

20 Jacob Hendricks, Matthew J. Patitz, and Trent A. Rogers. Universal simulation of directed systems in the abstract tile assembly model requires undirectedness. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2016), New Brunswick, New Jersey, USA* October 9-11, 2016, pages 800–809, 2016.

21 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. *Theoretical Computer Science*, 743:148–166, 2018.

22 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.

23 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.

24 James I. Lathrop, Jack H. Lutz, and Scott M. Summers. Strict self-assembly of discrete Sierpinski triangles. *Theoretical Computer Science*, 410:384–405, 2009.

25 Wenyan Liu, Hong Zhong, Risheng Wang, and Nadrian C. Seeman. Crystalline two-dimensional dna-origami arrays. *Angewandte Chemie International Edition*, 50(1):264–267, 2011. doi:10.1002/anie.201005911.

26 Kyle Lund, Anthony J. Manzo, Nadine Dabby, Nicole Michelotti, Alexander Johnson-Buck, Jeanette Nangreave, Steven Taylor, Renjun Pei, Milan N. Stojanovic, Nils G. Walter, Erik Winfree, and Hao Yan. Molecular robots guided by prescriptive landscapes. *Nature*, 465(7295):206–210, May 2010. doi:10.1038/nature09012.

27 Pierre-Étienne Meunier, Matthew J. Patitz, Scott M. Summers, Guillaume Theyssier, Andrew Winslow, and Damien Woods. Intrinsic universality in tile self-assembly requires cooperation. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 2014), (Portland, OR, USA, January 5-7, 2014)*, pages 752–771, 2014.

28 Pierre-Étienne Meunier and Damien Woods. The non-cooperative tile assembly model is not intrinsically universal or capable of bounded turing machine simulation. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 328–341, New York, NY, USA, 2017. ACM. `doi:10.1145/3055399.3055446`.

29 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.

30 Nicolas Ollinger. The intrinsic universality problem of one-dimensional cellular automata. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 632–641. Springer, 2003.

31 Nicolas Ollinger. Intrinsically universal cellular automata. In *The Complexity of Simple Programs, in Electronic Proceedings in Theoretical Computer Science*, volume 1, pages 199–204, 2008. URL: `http://arxiv.org/abs/0906.3213`.

32 Jennifer E. Padilla, Wenyan Liu, and Nadrian C. Seeman. Hierarchical self assembly of patterns from the robinson tilings: Dna tile design in an enhanced tile assembly model. *Natural Computing*, 11(2):323–338, June 2012. `doi:10.1007/s11047-011-9268-7`.

33 Matthew J. Patitz, Robert T. Schweller, and Scott M. Summers. Exact shapes and Turing universality at temperature 1 with a single negative glue. In Luca Cardelli and William M. Shih, editors, *DNA Computing and Molecular Programming - 17th International Conference, DNA 17, Pasadena, CA, USA, September 19-23, 2011. Proceedings*, volume 6937 of *Lecture Notes in Computer Science*, pages 175–189. Springer, 2011.

34 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`.

35 John Reif, Sudheer Sahu, and Peng Yin. Compact error-resilient computational DNA tiling assemblies. In *DNA: International Workshop on DNA-Based Computers*. LNCS, 2004.

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

37 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.

38 Paul WK Rothemund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of dna sierpinski triangles. *PLoS biology*, 2(12):e424, 2004.

39 David Soloveichik, Matthew Cook, and Erik Winfree. Combining self-healing and proofreading in self-assembly. *Natural Computing*, 7(2):203–218, 2008. `doi:10.1007/s11047-007-9036-x`.

40 David Soloveichik and Erik Winfree. Complexity of compact proofreading for self-assembled patterns. In Alessandra Carbone and Niles A. Pierce, editors, *DNA Computing, 11th International Workshop on DNA Computing, DNA11, London, ON, Canada, June 6-9, 2005. Revised Selected Papers*, volume 3892 of *Lecture Notes in Computer Science*, pages 305–324. Springer, 2005.

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

42 Grigory Tikhomirov, Philip Petersen, and Lulu Qian. Fractal assembly of micrometre-scale DNA origami arrays with arbitrary patterns. *Nature*, 552(7683):67–71, 2017.

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

44 Erik Winfree and Renat Bekbolatov. Proofreading tile sets: Error correction for algorithmic self-assembly. In Junghuei Chen and John H. Reif, editors, *DNA Computing, 9th International Workshop on DNA Based Computers, DNA9, Madison, WI, USA, June 1-3, 2003, revised Papers*, volume 2943 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.

45 Christopher M Wintersinger, Dionis Minev, Anastasia Ershova, Hiroshi M Sasaki, Gokul Gowri, Jonathan F Berengut, F Eduardo Corea-Dilbert, Peng Yin, and William M Shih. Multi-micron crisscross structures grown from dna-origami slats. *Nature Nanotechnology*, pages 1–9, 2022.

46 Damien Woods. Intrinsic universality and the computational power of self-assembly. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 373(2046), 2015. doi:10.1098/rsta.2014.0214.

47 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.

48 Yin Zhang, Angus McMullen, Lea-Laetitia Pontani, Xiaojin He, Ruojie Sha, Nadrian C. Seeman, Jasna Brujic, and Paul M. Chaikin. Sequential self-assembly of dna functionalized droplets. *Nature Communications*, 8(1):21, 2017. doi:10.1038/s41467-017-00070-0.

A Technical Appendix

This Technical Appendix contains formal definitions from the main body due to space constraints. Full proofs and construction details may be found in the Arxiv version of this paper [11].

A.1 IO marking example

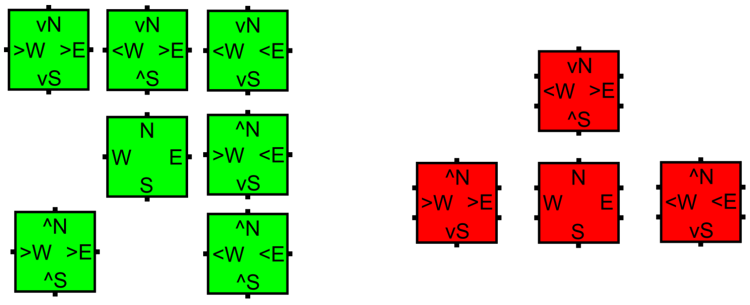


Figure 10 To make a set of IO-marked tile types whose collective behavior will be the same as an un-marked tile type, one IO-marked tile type is created for every minimal set of glues whose combined strength is $\geq \tau$ (minimal in that, if any individual glue was removed from the set, the combined strength would be $< \tau$), with those marked as input glues and the others as output glues. Two examples are shown. (Left) In the center, the un-marked tile type has a strength-1 glue on each side. Surrounding it are the six IO-marked tile types created from it, one for each possible pair of strength-1 glues marked as inputs. Note that this is the worst-case increase in tile complexity. (Right) In the center, the un-marked tile type has two strength-2 glues and two strength-1 glues. Surrounding it are the three IO-marked tile types created from it, one each with one of the strength-2 glues as the sole input, and the other with the pair of strength-1 glues as the inputs.

Here we give a simple demonstration of how a regular, unmarked aTAM tile set T can be used to generate an equivalent an IO-marked aTAM tile set T_{IO}. First we note that this example demonstrates how an arbitrary, already existing tile set T can be used to generate an equivalent T_{IO}, resulting in a constant-sized increase in tile complexity. Namely, in the worst case, each tile type of T requires the creation of 6 unique tile types in T_{IO}. Depending on the number and strengths of the glues on a tile type in T, the number of tile types generated for T_{IO} could range from 1 to 6. However, an increase in tile complexity is not necessarily required for the creation of an IO-marked tile set, since for systems such as zig-zag systems, it is known in advance which glues of any given tile type may ever serve as its input glues (and that set is fixed for every given tile type). The IO-marked tile set in that case doesn't require any more tile types than the unmarked set. See Figure 10 for two examples and explanation.

A.2 Formal characterization of classes of aTAM systems

▶ **Definition 6** (IO TAS). *An aTAM TAS $\mathcal{T} = (T, \sigma, 2)$ is an* IO TAS *iff all tile types in T are IO-marked and all non-null glues on the perimeter of σ have output markings.*

▶ **Observation 7.** *Given an IO TAS $\mathcal{T}$, all non-null glues exposed on the perimeter of any producible assembly of $\mathcal{T}$ have output markings.*

Observation 7 follows immediately from the a simple inductive argument. As the base case, the smallest producible assembly, the seed assembly, has only output-marked glues on its perimeter. The induction hypothesis is that, starting with a producible assembly with only output-marked glues on its perimeter, the addition of any tile to form a new producible assembly also results in an assembly with only output-marked glues on its perimeter. The induction hypothesis is proven by noting that all tiles are IO-marked and since input-marked glues can only bind to output-marked glues and no tile has input-marked glues whose strengths sum to > 2, any tile that τ-stably binds to an assembly must do so by binding all of its input-marked glues, leaving only output-marked glues to potentially be unbound and exposed on the perimeter. So called *zig-zag* aTAM systems were originally defined in [6] and are widely used in the literature (e.g. [20, 23, 33, 34]) due to their very simple dynamics and the fact that they are computationally universal (i.e. for every Turing machine there exists a zig-zag aTAM system that simulates it). Here we provide a definition that utilizes IO-marked systems, but note that regular aTAM systems can easily be converted to IO systems with only a constant increase in the number of tile types (as done in [2] and depicted in Section A.1), and that zig-zag system can be designed to be IO-marked without any additional tile types being required. Intuitively, a zig-zag tile assembly system is a system that grows to the left or right, grows upward by one tile, and then grows in the opposite direction. (Note that our definition restricts zig-zag systems to add new rows only to the north, but we can trivially rotate such systems so that growth is in any one of the cardinal directions. Therefore, for ease of presentation and w.l.o.g. we simply consider northward growing zig-zag systems, and our results still hold for the more general definition.) For an example of a zig-zag system, see Figure 2.

▶ **Definition 8** (Zig-zag TAS). *An aTAM system $\mathcal{T} = (T, \sigma, 2)$ is a* zig-zag *system if the following conditions hold:*

1. *$\mathcal{T}$ is an IO TAS.*
2. $|\sigma| = 1$
3. *$\mathcal{T}$ is directed.*
4. *The frontier of any producible assembly in $\mathcal{T}$ is never larger than 1, and thus $\mathcal{T}$ has a single valid assembly sequence.*
5. *There is never an exposed glue on the south of any tile of any producible assembly.*
6. *Rows grow in alternating directions, i.e. if the a row grows from right to left, then the next row grows left to right and vice versa.*

▶ **Definition 9** (Standard TAS). *Let $\mathcal{T} = (T, \sigma, 2)$ be a TAS in the aTAM. We say that $\mathcal{T}$ is* standard *if and only if:*

1. *$\mathcal{T}$ is an IO TAS.*
2. *$\mathcal{T}$ is directed.*
3. *For every $t \in T$, the sides that have input markings are either exactly (1) a single glue of strength-2, or (2) two diagonally adjacent strength-1 glues (i.e. not on opposite sides).*
4. *There are no mismatches in the terminal assembly (i.e. all adjacent pairs of tile sides in $\alpha \in \mathcal{A}_\Box[\mathcal{T}]$ have the same glue label and strength on both sides)*

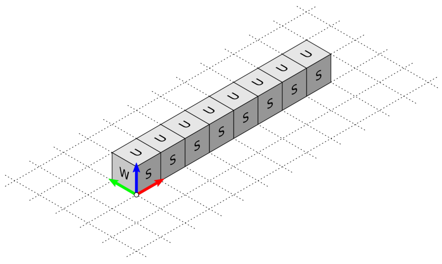
▶ **Definition 10** (Standard TAS with across-the-gap). *Let $\mathcal{T} = (T, \sigma, 2)$ be a TAS in the aTAM. We say that $\mathcal{T}$ is* standard with across-the-gap *if and only if*

1. *$\mathcal{T}$ is an IO TAS.*
2. *$\mathcal{T}$ is directed.*
3. *For every $t \in T$, the sides that have input markings are either exactly (1) a single glue of strength-2, or (2) two strength-1 glues (i.e. on any combination of two sides).*
4. *There are no mismatches in the terminal assembly (i.e. all adjacent pairs of tile sides in $\alpha \in \mathcal{A}_\Box[\mathcal{T}]$ have the same glue label and strength on both sides.*

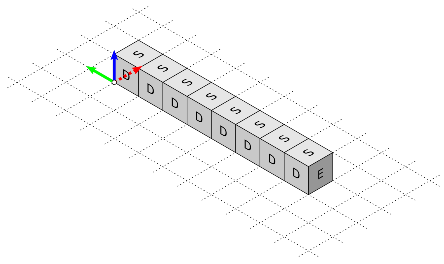
A.3 Formal definitions for the abstract Slat Assembly Model

The abstract Slat Assembly Model (aSAM) is essentially a restricted version of the Polyomino Tile Assembly Model (polyTAM) [14]. The polyTAM itself is a generalization of the aTAM [43] in which, rather than square tiles, the basic components are polyominoes (which are shapes composed of unit squares attached along their edges). (Note that we take much of our notation, slightly adapted, from aTAM definitions such as those in [24].) The polyTAM was defined for two-dimensional polyominoes, but the aSAM utilizes three-dimensional polyominoes whose shapes are restricted to be linear arrangements of unit cubes. The basic components of the aSAM are called *slats* and each is a $1 \times 1 \times n$ polyomino composed of n unit cubes, for some $n \in \mathbb{N}$. Each unit cube of a slat has 4 or 5 exposed faces: 5 if it is on one of the two ends of the slat (in the dimension of length n), and 4 otherwise (i.e. it is an interior cube). On each exposed face of a unit cube may be a *glue*, and each glue is an ordered pair (l, s) where l is a string and serves as the glue's *label* and $s \in \mathbb{Z}^+$ is its *strength*. An exposed face may also have no glue (which may also be referred to as the *null glue*). The character "$*$" is considered a special character in glue labels and any label may have at most a single "$*$" character, which must appear as its rightmost. Given a glue $g = (l, s)$, if the label l does not end with the character "$*$", then we say the label $l' = l^*$ (i.e. the string l concatenated with "$*$") represents the *complement* of l. If l does end with "$*$", then its complement is represented by the string l truncated by one character to remove the "$*$". Thus a pair of labels are complementary to each other if they consist of the same string up to exactly one of them terminating in "$*$", e.g. "foo" and "foo^*" are complementary labels. Any two glues that have complementary labels must have the same strength value. If two slats are placed so that faces containing complementary glues are adjacent to each other, those glues *bind* with strength equal to the common strength value of those two glues.

A *slat type* is defined by its length n and the set of glues on its constituent cubes. For convenience, each slat type is assigned a canonical placement and orientation in $\mathbb{Z}^3$, with the default being that it has one cube at $(0, 0, 0)$ and it extends along the x-axis to $(n-1, 0, 0)$, and the cubes have designated N, E, S, W, U, D (i.e. north, east, south, west, up, down) sides which face in the $+y, +x, -y, -x, +z, -z$ directions, respectively (in the canonical placement). Additionally for convenience, the cubes are numbered from 0 to $n-1$ starting from the cube at $(0,0,0)$ and proceeding in order along the x-axis (of the slat type in its canonical placement), and for a slat type t the ith cube is denoted by $t[i]$. A slat is an instance of a slat type, and may be in any rotation or orientation in the $\mathbb{Z}^3$ lattice. A slat is defined by (1) its type t, (2) its translation, which is identified by the coordinates of its $t[0]$, (3) its direction, taken from the set $\{-x,+x,-y,+y,-z,+z\}$ where the letter denotes which axis the length-n dimension is parallel to and $+$ or $-$ denotes whether the coordinates of block $t[n-1]$ are more positive or more negative in that dimension than $t[0]$, respectively, and (4) its "up" direction which is the side of the cubes pointing in the $+z$ direction in the slat's current orientation (unless the slat is oriented along the z axis, in which the "up" direction is the side of the cubes pointing in the $+x$ direction). See Figure 11 for an example of a slat type in canonical placement and a rotated version.



(a) An example slat type t of length 8 in its canonical placement.



(b) Slat t rotated so that $t[0]$ is still at $(0,0,0)$ but its direction is $+y$ and its up direction is $-y$ (i.e. the "south" label appears on the top faces, in the $+z$ direction).

Figure 11 Depiction of slats in the aSAM.

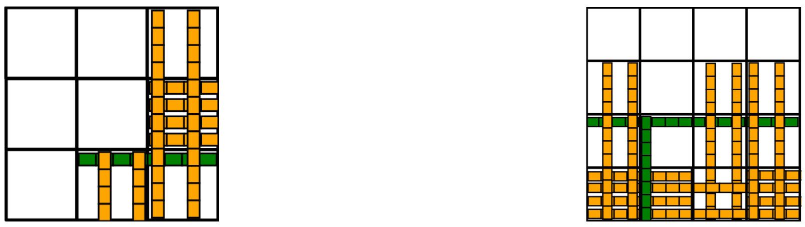
An *assembly* over a set of slat types, S, consists of placements of slats of types from S in $\mathbb{Z}^3$ such that no blocks of any two slats share the same space. Given an assembly α, if two slats t_1 and t_2 are placed in α such that for some block of t_1, say $t_1[i]$, and some block of t_2, say $t_2[j]$, a face of $t_1[i]$ is adjacent to a face of $t_2[j]$ (irrespective of the directions of t_i and t_j) and the glues of those faces are complementary, then they form a bond with the common strength value of those glues. If there is no cut in $\mathbb{Z}^3$ separating the slats of an assembly into two separate components without cutting bonds whose strengths sum to at least some value τ, then we say the assembly is τ-*stable.* Given an assembly α, a value $\tau \in \mathbb{Z}^+$, and a set of slat types S, any set F of i surfaces on blocks of the slats composing α, where $0 < i \leq \tau$, such that a slat of some type $t \in S$ can be positioned in $\mathbb{Z}^3$ (1) without any of its blocks overlapping any blocks of slats in α and (2) its glues form bonds of strength summing to $\geq \tau$ with the glues on the surfaces of F, we call F a S_τ-*frontier* location of α. Essentially, a S_τ-frontier location of assembly α is a location to which a slat of some type in the set S can validly attach with at least strength τ. When S and τ are clear from context we will simply refer to such locations as *frontier* locations.

A *slat assembly system*, or SAS, is an ordered triple (S,σ,τ) where S is a set of slat types, σ is an assembly of slats from S referred to as the *seed assembly*, and τ is the minimum binding threshold which is often referred to as the temperature in the aTAM and we call the *cooperativity* in the aSAM. (Note that in [29] they used the term *coordination* for the same concept.) Given a SAS $\mathcal{S} = (S,\sigma,\tau)$, it is assumed that there are an infinite number of slats of each type from S available for attachment, and assembly begins from the seed assembly σ. Assembly proceeds in discrete steps, with each step consisting of the nondeterministic selection of a frontier location F for the current assembly α, then the nondeterministic selection of a slat type $t \in S$ that can bind in F (in case there are more than one), and then the attachment of a slat of type t to α, translated and rotated appropriately to bind to α using the glues of F. Note that the aSAM does not require that there be a path through which the slat of type t must be able to move in $\mathbb{Z}^3$ from arbitrarily far from α into that location without encountering overlaps along the way (i.e. it can be considered to just "appear" in the correct location) Given an assembly α in $\mathcal{S}$, if β can result from α in a single such step, we say that α *produces* β *in one step* and denote it as $\alpha \to_1^{\mathcal{S}} \beta$.

We use $\mathcal{A}^S$ to denote the set of all assemblies of slats over slat type set S. Given a SAS $\mathcal{S} = (S, \sigma, \tau)$, a sequence of $k \in \mathbb{Z}^+ \cup \{\infty\}$ assemblies $\alpha_0, \alpha_1, \ldots, \alpha_k$ over $\mathcal{A}^S$ is called a $\mathcal{S}$*-assembly sequence* if $\alpha_0 = \sigma$ and, for all $1 \leq i < k$, $\alpha_{i-1} \to_1^{\mathcal{S}} \alpha_i$. The *result* of an assembly sequence 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 slats from any assembly in the sequence. We say that α $\mathcal{S}$*-produces* β (denoted $\alpha \to^{\mathcal{S}} \beta$) if there is a $\mathcal{S}$-assembly sequence starting with α whose result is β. We say α is $\mathcal{S}$*-producible* if $\sigma \to^{\mathcal{S}} \alpha$ and write $\mathcal{A}[\mathcal{S}]$ to denote the set of $\mathcal{S}$-producible assemblies. We say α is $\mathcal{S}$*-terminal* if α is τ-stable and there exists no assembly which is $\mathcal{S}$-producible from α. We denote the set of $\mathcal{S}$-producible and $\mathcal{S}$-terminal assemblies by $\mathcal{A}_\Box[\mathcal{S}]$. When $\mathcal{S}$ is clear from context, we may omit $\mathcal{S}$ from this notation. If there is only a single terminal assembly of $\mathcal{S}$, i.e. $|\mathcal{A}_\Box[\mathcal{S}]| = 1$, then we say that $\mathcal{S}$ is *directed.* Otherwise, it is *undirected.* Note that a SAS $\mathcal{S}$ may have multiple (even infinitely many) distinct valid assembly sequences but yet $\mathcal{S}$ may be directed.

A.4 Notes on Directedness

For the constructions described in proofs of Theorems 1, 2, and 3, it may be noted that the constructions may be adjusted as to allow for non-directed behavior; However, this same statement cannot be made for the construction described in Theorem 4. The adjustment of Theorems 1 - 3 to allow for the simulation of non-directed behavior rely on the addition of the *decision slat* group in S, as described in the proof of Theorem 5. In doing so, the resolution of a macrotile must occur within a single slat attachment, whose attachment into the system would prevent any future decision slats from resolving the macrotile. This requires that all potential macrotile attachment locations must occur within adjacent cells such that for n adjacent cells wherein a macrotile may resolve, a slat of length nc must be able to extend across each cell, resolving to the desired macrotile while blocking any other macrotiles from being resolved in the same space. An example of this behavior is shown in Figure 12. Because all adjacent pairs of tile sides in $\alpha \in \mathcal{A}_\Box[\mathcal{T}]$ have the same glue label and strength on both sides (i.e. there are no mismatches in the terminal assembly) within the classes of simulation for Theorems 1 - 3, there only exists a minimal subset of input glues at any location within the $\mathcal{T}$-frontier. Therefore the resolution of a macrotile may only occur from all available input glues at a single macrotile location, meaning that any slat which encodes the information of a macrotile must only cover those cells which are occupied by slats which provide said input. However, this use of the *decision slats* is not so readily applicable to the construction described in Theorem 4. This is because a single location in the $\mathcal{T}$-frontier may be faced by $> \tau$ input glues, constituting a 'mismatch'. In such a situation, if non-directed behavior is to be allowed, then there may be multiple cell locations within a macrotile in which different macrotiles may attempt, and independently succeed in resolving which may not be reached by a single slat of length nc. Such a situation would invalidate the simulation.



(a) (b)

Figure 12 (a) A Standard macrotile, whose characteristics are described by the proof of theorem 2, placing the northernmost input slat, colored green, which encodes the information of the macrotile. Only one slat needs to be placed in order to prevent any opposing input slats from entering the macrotile, and thus no issue occurs. (b) A Directed Temperature-2 macrotile, whose characteristics are described by the proof of theorem 4, placing the northmost / westmost input slats required to encode the information of the macrotile. Because multiple slats must be placed in order to fill all potential input slat locations (i.e. account for E+W, E+S, S+W tile input signatures), an issue will occur where multiple tiles may successfully resolve in the same macrotile location.