5-2015

Simulation in Algorithmic Self-assembly

Jacob Hendricks
University of Arkansas, Fayetteville

Simulation in Algorithmic Self-assembly

Simulation in Algorithmic Self-assembly

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Computer Science

by

Jacob Hendricks
University of Arkansas
Bachelor of Science in Mathematics, 2004
University of Texas
Master of Arts in Mathematics, 2008

May 2015
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

Abstract

Winfree introduced a model of self-assembling systems called the abstract Tile Assembly Model [64] (aTAM) where square tiles with glues on their edges attach spontaneously via matching glues to form complex structures. A generalization of the aTAM called the 2HAM (*two-handed* aTAM) not only allows for single tiles to bind, but also for "supertile" assemblies consisting of any number of tiles to attach. We consider a variety of models based on either the aTAM or the 2HAM.

The underlying commonality of the work presented here is *simulation*. We introduce the polyTAM, where a tile system consists of a collection of polyomino tiles, and show that for any polyomino P of size ≥ 3 and any Turing machine M, there exists a temperature-1 polyTAM system containing only shape-P tiles that simulates M. We introduce the RTAM (Reflexive Tile Assembly Model) that works like the aTAM except that tiles can nondeterministically flip prior to binding. We show that the temperature-1 RTAM cannot simulate a Turing machine by showing the much stronger result that the RTAM can only self-assemble "periodic" patterns.

We consider a variety of models and define notions of simulation which serve as relations between two tile assembly systems (possibly belonging to different models). Using simulation as a basis of comparison, we first show that cellular automata and the class of all tile assembly systems in the aTAM are equivalent. Next, we introduce the Dupled aTAM (DaTAM) and show that the temperature-2 aTAM and the temperature-1 DaTAM are "mutually exclusive" by showing that there is an aTAM system that cannot be simulated by any DaTAM system, and vice versa. Third, we consider the restricted glues Tile Assembly Model [52] (rgTAM) and show that there is an aTAM system that cannot be simulated by any rgTAM system. We introduce the Dupled restricted glues Tile Assembly Model (DrgTAM), and show that the DrgTAM is intrinsically universal for the aTAM. Finally, we consider a variation of the Signal-passing Tile Assembly Model (STAM) [50] called the STAM$^+$ and show that the STAM$^+$ is intrinsically universal and that the 3-D 2HAM is intrinsically universal for the STAM$^+$.

Acknowledgements

First and foremost, I would like to thank my advisor, friend and co-author Matt Patitz. As a non-traditional student, I entered the PhD program with a good deal of self-doubt, and throughout my work as a PhD student, Matt has believed in me even when I did not. What I know about being a productive researcher I have learned by his example, and for this reason, this thesis owes its existence to him. I cannot say enough about what Matt has done for me; I just hope that he knows this and that someday I have a chance to repay him.

When I joined the self-assembly research group at the University of Arkansas, the group consisted of Matt and Trent Rogers. As we worked on projects together, Trent and I were both grappling with the same concepts, and in doing so, Trent has been indispensable in my understanding of theoretical self-assembly. When it looks as though Trent and I are trading mumblings coupled with a few hand-gestures, we are, in fact, working on a new paper. He's a remarkable student with a very bright future who is always trying to push research further as well as trying to push my buttons. This thesis would not exist without him. I would also like to thank my friend and fellow graduate student Tyler Fochtman. The self-assembly group wouldn't be complete with the "wild card".

I would also like to thank Scott Summers, a friend and co-author, who has been an excellent example for me. Scott's guidance in seeking employment was especially helpful. I look forward to working with Matt, Trent and Scott on future projects.

Dedication

I would like to thank my loving wife for her unconditional support and patience while I was working on the projects that eventually led to this thesis.

Contents

Chapter 1

Introduction

1.1 Theoretical models of self-assembly

Self-assembly is the process by which disorganized components autonomously combine to form organized structures. In DNA-based self-assembly, the combining ability of the components is implemented using complementary strands of DNA as the "glue". In [64], Winfree introduced a useful mathematical model of self-assembling systems called the abstract Tile Assembly Model (aTAM) where the autonomous components are described as square tiles with specifiable glues on their edges and the attachment of these components occurs spontaneously when glues match. The aTAM provides a convenient way of describing self-assembling systems and their resulting assemblies, and serves as the underpinning of many studies of the properties of self-assembling systems. The results of such studies include showing the power of these systems to perform computations [64, 41, 54, 62], the ability to build shapes efficiently (in terms of the number of unique types of components, i.e. tile types, needed) [1, 59, 62, 9], and limitations to what can be built and computed [41, 42, 53, 9, 46].

From the broad collection of results in the aTAM, one property of systems that has been shown to yield enormous power is *cooperation.* The notion of cooperation captures the phenomenon where the attachment of a new tile to a growing assembly requires it to bind to more than one tile (usually 2) already in the assembly. The requirement for cooperation is determined by a system parameter known as the *temperature*, and when the temperature is equal to 1 (a.k.a. temperature-1 systems), there is no requirement for cooperation. A long-standing conjecture is that temperature-1 aTAM systems are in fact not capable of universal computation or efficient shape building, although it is well-known that temperature $\geq$ 2 systems are. However, in actual laboratory implementations of DNA-based tiles [58, 6, 60, 43, 66], the self-assembly performed by temperature-2 systems does not match the error-free behavior dictated by the aTAM, but instead, a frequent source of errors is the binding of tiles using only a single bond. Thus, temperature-1 behavior erroneously occurs and cannot be completely prevented. This has led to the development of a number of error-correction and error-prevention techniques [61, 10, 65, 60, 55] for use in temperature-2 systems.

We also consider a generalization of the aTAM called the *two-handed* abstract Tile Assembly Model (2HAM) which not only allows for single tiles to bind, but also allows for "supertile" assemblies consisting of possibly any number of tiles to bind. Two supertiles may bind when glues of two supertile assemblies match in such a way that the combined strengths of these glues are greater than or equal to a temperature parameter and when the edges of tiles containing these glues abut, no two tile share the same location. This latter requirement models the physical notion of "steric hinderance". In this thesis, we consider a variety of models each of which sets out to model some physical phenomenon. Each of these models is based on the aTAM or the 2HAM. For a comprehensive survey of tile-based self-assembly including many results about the aTAM and the 2HAM, see [51, 20].

1.2 Simulation in self-assembly

In computational theory, a powerful and widely used tool for determining the relative powers of systems is simulation. For instance, in order to prove the equivalence, in terms of computational power, of Turing machines and various abstract models such as tag systems, counter machines, cellular automata, and tile-based self-assembly, systems have been developed in each which demonstrate their abilities to simulate arbitrary Turing machines, and vice versa. This has been used to prove that whatever can be computed by a system within one model can also be computed by a system in another. Additionally, the notion of a universal Turing machine is based upon the fact that there exist Turing machines which can simulate others.

The methods of simulation which are typically employed involve mappings of behaviors and states in one model or system to those in another, often following some "natural" mapping function, and also often in such a way that the simulation is guaranteed to generate the same final result as the simulated system, and maybe even some or all of its intermediate states. Nonetheless, there is usually no requirement that the simulator "do it the same way," i.e. the dynamical behavior of the simulator need not mirror that of the simulated. For instance, as one Turing machine A simulates another, B, its head movements may be in a significantly different pattern than B's since, for instance, it may frequently move to a special portion of the tape which encodes B's transition table, then back to the "data" section.

While such types of simulation can be informative when asking questions about the equivalence of computational powers of systems, oftentimes it is the behavior of a system which is of interest, not just its "output." Self-assembling systems, which are those composed of large numbers of relatively simple components which autonomously combine to form structures using only local interactions, often fall into this category since the actual ways in which they evolve and build structures are of key importance. Because the dynamical behaviors of these systems are of such importance, work in these models (e.g. [22, 19, 46, 35, 34, 68]) has turned to a notion of simulation developed within the domain of cellular automata, whose dynamical behaviors are also often of central importance. This notion of simulation, called *intrinsic universality* (see [45, 27, 13, 15, 33, 47, 48, 4, 3] for some examples related to various models such as cellular automata), is defined in such a way that the simulations performed are essentially "in place" simulations which mirror the dynamics of the simulated systems, modulo a scale factor allowed the simulator. Intrinsic universality has been used to show the existence of "universal" systems, somewhat analogous to universal Turing machines, which can simulate all other systems within a given model or class of systems, but in a dynamics-preserving way. Previous work [22] has shown that there exists a single aTAM tile set which is capable of simulating any arbitrary aTAM system, and thus that tile set is intrinsically universal (IU) for the aTAM (and we also say that the aTAM is IU). Further work in [19] showed that the 2HAM is much more complicated in terms of IU, with the existence of hierarchies (separated by temperature) of 2HAM systems with strictly-increasing power of simulation. These simulations are performed by scaled blocks of tiles known as macrotiles in the simulator used to simulate individual tiles in the simulated systems. It was proven in [19] that for every temperature $\tau \geq 2$, there exists a system at temperature τ such that no system at temperature $\tau' < \tau$ can simulate it. However, they also showed that for each $\tau \geq 2$, the class of 2HAM systems at τ is IU.

Not only are there notions of simulation that pertain to systems with a single model, but there are also notions of simulation that relate two systems of two distinct models. This is analogous to the idea that a CA can simulate a Turing machine. When simulation is used in this way, it provides a rigorous means of comparing two models. For example, in [9] it is shown that for any system in the aTAM, there is a system in the 2HAM which simulates it. This implies that the 2HAM is "more powerful" that the aTAM. Then, as the 2HAM is IU, there exists a single 2HAM system that when appropriately seeded can simulate any given aTAM system. In this way, the notion of intrinsic universality extends to apply not just to single models but to two models. For example, we say that the 2HAM is intrinsically universal for the class of aTAM systems.

1.3 Structure of this thesis

1.3.1 Computational universality in non-cooperative self-assembly

Despite the conjectured weakness of temperature-1 (non-cooperative) systems, an alternative approach has been to try to find ways of modifying them in the hope of developing systems which can operate at temperature-1 while exhibiting powers of temperature-2 systems but without the associated errors. Research along this path has resulted in an impressive variety of alternatives in which temperature-1 systems are capable of Turing universal computation: using 3-D tiles [12], allowing probabilistic computations with potential for error [12], including glues with repulsive forces [52], and using a model of staged assembly [7]. While these are theoretically very interesting results, the promise for use in the laboratory of each is limited by current technologies.

Using the geometry of tiles to simulate Turing machines

We introduce the Polyomino Tile Assembly Model (polyTAM), in which each tile is composed of a collection of unit squares joined along their edges. This allows for tiles with arbitrary geometric complexity. We prove that geometry in the polyTAM, as in natural self-assembling systems, affords great power. Namely, *any* polyomino shape which is composed of only 3 or more unit squares has enough geometric complexity to allow a polyTAM system at temperature 1, composed only of tiles of that shape, to perform Turing universal computation. We also provide negative results that further help to refine understanding of exactly what geometric properties are needed for Turing universal computation in temperature-1 self-assembly. We prove that a fundamental gadget (which we call the *bit-reading gadget*) used within all known systems that can compute at temperature 1 in any tile-assembly model, is impossible to construct with either the square tiles of the aTAM or with dominoes (a.k.a. duples). This provides further evidence that systems composed solely of those shapes are incapable of universal computation. Moreover, we show that for any two distinct polyominoes, there is a polyTAM system that can simulate a Turing-machine such that the shape of a tile in this system is on of the two given polyominoes. This work was done with co-authors Sándor P. Fekete, Matthew Patitz, Trent Rogers, and Robbie Schweller [29].

A model that is not computationally universal at temperature-1

We introduce the *Reflexive Tile Assembly Model* (RTAM), which can be thought of as the aTAM with the relaxed constraint that tiles in the RTAM *are* allowed to flip horizontally and/or vertically. We show that at temperature 1, the class of directed RTAM systems – systems which yield a single pattern up to reflection and ignoring tile orientation – are only capable of assembling patterns that are essentially periodic. Then, following the thesis set forth in [26], we conclude that the temperature-1 RTAM is not computationally universal. We also show that like the aTAM at temperature 2, the class of directed temperature-2 RTAM systems is computationally universal. This shows a fundamental dividing line between the powers of RTAM temperature-1 and temperature-2 directed systems. In addition, we prove a variety of results about the self-assembly of finite shapes in the RTAM that start from a single tile seed. These results show that unlike the self-assembly of shapes in the aTAM, where any finite shape can be "hardcoded" to self-assemble from a single seed tile, self-assembly of shapes in the RTAM is much more complicated. This work was done with co-authors Matthew Patitz and Trent Rogers [36].

1.3.2 Using simulation to compare models of self-assembly

As new models of self-assembly are introduced, we can compare these models using in a variety of ways which fall into two categories. First, we can measure the capacity of a new model using benchmarks that have become standard in the self-assembly community. These benchmarks include proving bounds on how efficient a system of a model can build squares and proving that a model is Turing-complete - among others. While these benchmarks have proven useful in gauging the strengths and weakness of a model, often we desire means of measuring the relative strengths of two models. Given two models, a common way of comparing them is to consider shapes which can be self-assembled by some system of the first model but cannot be self-assembled by any system in the second model. While gives a basis of comparison between two models in terms of *what* the models can and cannot produce, a priori, this does not give a comparison of *how* the models go about assembly. An interesting way of measuring the relative strengths of two models which takes dynamics into account stems from the notion simulation. However, even between systems within the same model, defining a satisfactory notion of simulation, namely one which captures the essence of one system "behaving" like the other while also generating analogous results, or output, can be difficult. In this thesis, we define such notions of simulation that serves as a relation between systems of a variety of different models, and even define a notion of simulation between a cellular automaton and a system of the aTAM.

Comparing models to the aTAM

As early as the aTAM's initial introduction, its power to simulate cellular automata (CA) was explored. Winfree et al. showed that the 2-D aTAM can be used to simulate 1-D CA [67], and Winfree [64] showed that the 3-D aTAM can simulate 2-D CA. Furthermore, the aTAM is commonly colloquially referred to as an asynchronous, nondeterministic CA in which quiescent states that change to "tile" states never change again (analogous to write-once memory). These comparisons led naturally to our exploration of simulations between the two models using the same dimensions for each, namely 2-D. We give a notion of simulation of a CA by an aTAM system as well as a notion of simulation of an aTAM system by a CA, and show that with regards to these notions of simulation, CA and aTAM are equivalent. This work was done with Matthew Patitz [35].

We next introduce the *Dupled abstract Tile Assembly Model* (DaTAM), which is essentially the aTAM extended to allow both square tiles and rectangular tiles called duples which are simply pre-formed pairs of 2 tiles joined along one edge before assembly begins. Note that the DaTAM can be thought of as a special class of polyTAM systems consisting of 1×1 tiles and 1×2 tiles (called duples). We then provide a series of results which show that neither the aTAM at temperature-2 nor the DaTAM at temperature-1 is strictly more powerful than the other, namely that in each there are shapes which can be self-assembled which are impossible to self-assemble in the other, and that there are also systems in each which cannot be simulated by the other. Essentially, it shown that the DaTAM is not capable of simulating glue cooperation while the aTAM is not capable of exhibiting the simultaneous placement of two tile that may occur in a DaTAM system. This work was done with co-authors Matthew Patitz, Trent Rogers and Scott Summers [39].

Next, we show that the restricted glues TAM (rgTAM) which allow glues with repulsive (rather than just attractive) forces [52] is also not capable of simulating glue cooperation. Second, we introduce the Dupled restricted glue TAM (DrgTAM) which allows for both square tiles and duples, and it allows for glues with negative strength (i.e. those which exert repulsive force). However, it is restricted similar to the rgTAM in that the magnitude of glue strengths cannot exceed 1 (i.e. only strengths 1 and -1 are allowed). Third, we show that the DrgTAM in some measures is greater than the sum of its parts. That is, the resulting DrgTAM is capable of both universal computation *and* the simulation of glue cooperation. This is the first such result for passive (i.e. non-active) tile assembly systems. In fact, we show the stronger result that there is a single tile set in the DrgTAM which can be configured to, in a temperature-1 system, simulate any arbitrary aTAM system, making it intrinsically universal for the aTAM. Coupled with the result in [39] which proves that there are temperature-1 systems in the DTAM, which are thus also in the DrgTAM, that cannot be simulated by the aTAM at temperature-2, this actually implies that the DrgTAM is more powerful than the temperature-2 aTAM. This work was done with Matthew Patitz and Trent Rogers [37].

Comparing models to the 2HAM

A newly developed model, the Signal-passing Tile Assembly Model (STAM) [50], is based upon the 2HAM but with a powerful and important difference. Tiles in the aTAM and 2HAM are static, unchanging building blocks which can be thought of as analogous to write-once memory, where a location can change from empty to a particular value once and then never change again. Instead, the tiles of the STAM each have the ability to undergo some bounded number of transformations as they bind to an assembly and while they are connected. Each transformation is initiated by the binding event of a tile's glue, and consists of some other glue on that tile being turned either "on" or "off". Experimental work shows that capabilities in this area are improving, and now include the linear transmission of signals, where one glue binding event can activate one other glue on a DNA tile [49]. Note that while in the general STAM it is possible for signals to turn glues both "on" and "off", our results pertain only to systems which turn glues "on" (which we call $STAM^+$ systems).

We show that there is a 3-D 2HAM tile set U which is intrinsically universal (IU) for the class $\mathfrak{C}$ of all STAM$^+$ systems at temperature 1 and 2. For every $\mathcal{T} \in \mathfrak{C}$, a single input supertile can be created, and using just copies of that input supertile and the tiles from U, at temperature 2 the resulting system will faithfully simulate $\mathcal{T}$. Furthermore, the simulating system will use only 2 planes of the third dimension. (The signal tile set simplification results are integral in the construction for this result, especially in allowing it to use only 2 planes.) This result is noteworthy especially because it shows that the dynamic behavior of signal tiles (excluding glue deactivation) can be *fully duplicated* by static tile systems which are allowed to "barely" use three dimensions. Furthermore, for every temperature $\tau > 1$ there exists a 3D 2HAM tile set which can simulate the class of all STAM$^+$ systems at temperature τ. Additionally, we show that for every $\tau \geq 1$ there exists a tile set U_τ which is IU for the STAM$^+$ at temperature τ. To simplify the proof of these results we provide methods of reducing the number of signals on a tile which may be of experimental importance as physical implementations of signal tiles become simpler (and perhaps more feasible) as the number of signals decreases. This work was done with co-authors Jennifer Padilla, Matthew Patitz and Trent Rogers [34].

Chapter 2

Preliminaries

Mathematical models of physical components capable of self-assembly provide a means of rigorously studying self-assembly. These models ground theoretical studies, allowing for the construction of precise statements and proofs. Insofar as a model correctly captures a physical system, theoretical study of a model can help guide experimental research, and in turn, experimental research helps to verify choices that may have been made in the definition of a model. In later chapters, we will define many different models based on two previously defined models. These two models are the *abstract Tile Assembly Model* (or aTAM for short) and the *Two-handed Tile Assembly Model* (abbreviated as 2HAM). In this chapter, we first give definitions of the abstract Tile Assembly Model (aTAM), which is a tile-based theoretical model which does not model the kinetics of physical self-assembling systems, but instead assumes that self-assembly occurs in an error-free way as single unit components which model individual physical components which self-assemble called *tiles* "attach" one at a time, progressing the process of self-assembly. The reader is encouraged to see [59, 64, 42] for a formal development of the model. Second, we define the 2HAM, which is also a tile-based model of self-assembly that is similar to the aTAM with the exception that self-assembling components (composed of possibly many connected tile) can "attach" to other self-assembling components. For a formal definition of the 2HAM, the reader is encouraged to see [9].

2.1 Informal description of the abstract Tile Assembly Model

A *tile type* is a unit square with four sides, each consisting of a *glue label*, often represented as a finite string, and a nonnegative integer *strength*. A glue g that appears on multiple tiles (or sides) always has the same strength s_g. There are a finite set T of tile types, but an infinite number of copies of each tile type, with each copy being referred to as a *tile*. An *assembly* is a positioning of tiles on the integer lattice $\mathbb{Z}^2$, described formally as a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$. Let $\mathcal{A}^T$ denote the set of all assemblies of tiles from T, and let $\mathcal{A}^T_{<\infty}$ denote the set of finite assemblies of tiles from T. We write $\alpha \sqsubseteq \beta$ to denote that α is a *subassembly* of β, which means that dom $\alpha \subseteq$ dom β and $\alpha(p) = \beta(p)$ for all points $p \in$ dom α. Two adjacent tiles in an assembly *interact*, or are *attached*, if the glue labels on their abutting sides are equal and have positive strength. Each assembly induces a *binding graph*, a grid graph whose vertices are tiles, with an edge between two tiles if they interact. The assembly is τ-*stable* if every cut of its binding graph has strength at least τ, where the strength of a cut is the sum of all of the individual glue strengths in the cut. When τ is clear from context, we simply say that a τ-stable assembly is stable.

A *tile assembly system* (TAS) is a triple $\mathcal{T} = (T, \sigma, \tau)$, where T is a finite set of tile types, $\sigma : \mathbb{Z}^2 \dashrightarrow T$ is a finite, τ-stable *seed assembly*, and τ is the *temperature*. An assembly α is *producible* if either $\alpha = \sigma$ or if β is a producible assembly and α can be obtained from β by the stable binding of a single tile. In this case we write $\beta \to_1^{\mathcal{T}} \alpha$ (to mean α is producible from β by the attachment of one tile), and we write $\beta \to^{\mathcal{T}} \alpha$ if $\beta \to_1^{\mathcal{T}*} \alpha$ (to mean α is producible from β by the attachment of zero or more tiles). When $\mathcal{T}$ is clear from context, we may write $\to_1$ and $\to$ instead. We let $\mathcal{A}[\mathcal{T}]$ denote the set of producible assemblies of $\mathcal{T}$. An assembly is *terminal* if no tile can be τ-stably attached to it. We let $\mathcal{A}_\Box[\mathcal{T}] \subseteq \mathcal{A}[\mathcal{T}]$ denote the set of producible, terminal assemblies of $\mathcal{T}$. A TAS $\mathcal{T}$ is *directed* if $|\mathcal{A}_\Box[\mathcal{T}]| = 1$. Hence, although a directed system may be nondeterministic in terms of the order of tile placements, it is deterministic in the sense that exactly one terminal assembly is producible (this is analogous to the notion of *confluence* in rewriting systems).

Since the behavior of a TAS $\mathcal{T} = (T, \sigma, \tau)$ is unchanged if every glue with strength greater than τ is changed to have strength exactly τ, we assume that all glue strengths are in the set $\{0, 1, \ldots, \tau\}$.

2.2 Formal description of the abstract Tile Assembly Model

In this section we provide a set of definitions and conventions that are used throughout this thesis.

We work in the 2-dimensional discrete space $\mathbb{Z}^2$. Define the set

$$U_2 = \{(0,1), (1,0), (0,-1), (-1,0)\}$$

to be the set of all *unit vectors* in $\mathbb{Z}^2$. We also sometimes refer to these vectors by their cardinal directions N, E, S, W, respectively. All *graphs* in this thesis are undirected. A *grid graph* is a graph $G = (V, E)$ in which $V \subseteq \mathbb{Z}^2$ and every edge $\{\vec{a}, \vec{b}\} \in E$ has the property that $\vec{a} - \vec{b} \in U_2$.

Intuitively, a tile type t is a unit square that can be translated, but not rotated, having a well-defined "side $\vec{u}$" for each $\vec{u} \in U_2$. Each side $\vec{u}$ of t has a "glue" with "label" $\text{label}_t(\vec{u})$–a string over some fixed alphabet–and "strength" $\text{str}_t(\vec{u})$–a nonnegative integer–specified by its type t. Two tiles t and t' that are placed at the points $\vec{a}$ and $\vec{a} + \vec{u}$ respectively, *bind* with *strength* $\text{str}_t(\vec{u})$ if and only if $(\text{label}_t(\vec{u}), \text{str}_t(\vec{u})) = (\text{label}_{t'}(-\vec{u}), \text{str}_{t'}(-\vec{u}))$.

In the subsequent definitions, given two partial functions f, g, we write $f(x) = g(x)$ if f and g are both defined and equal on x, or if f and g are both undefined on x.

Fix a finite set T of tile types. A T-*assembly*, sometimes denoted simply as an *assembly* when T is clear from the context, is a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$ defined on at least one input, with points $\vec{x} \in \mathbb{Z}^2$ at which $\alpha(\vec{x})$ is undefined interpreted to be empty space, so that $\text{dom}\ \alpha$ is the set of points with tiles. We write $|\alpha|$ to denote $|\text{dom}\ \alpha|$, and we say α is *finite* if $|\alpha|$ is finite. For assemblies α and α', we say that α is a *subassembly* of α', and write $\alpha \sqsubseteq \alpha'$, if $\text{dom}\ \alpha \subseteq \text{dom}\ \alpha'$ and $\alpha(\vec{x}) = \alpha'(\vec{x})$ for all $x \in \text{dom}\ \alpha$.

We now give a brief formal definition of the aTAM. See [64, 59, 57, 42] for other developments of the model. Our notation is that of [42], which also contains a more complete definition.

Given a set T of tile types, an *assembly* is a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$. An assembly is *$\tau$-stable* if it cannot be broken up into smaller assemblies without breaking bonds of total strength at least τ, for some $\tau \in \mathbb{N}$.

Self-assembly begins with a *seed assembly* σ and proceeds asynchronously and nondeterministically, with tiles adsorbing one at a time to the existing assembly in any manner that preserves τ-stability at all times. A *tile assembly system* (*TAS*) is an ordered triple $\mathcal{T} = (T, \sigma, \tau)$, where T is a finite set of tile types, σ is a seed assembly with finite domain, and $\tau \in \mathbb{N}$. A *generalized tile assembly system* (*GTAS*) is defined similarly, but without the finiteness requirements. We write $\mathcal{A}[\mathcal{T}]$ for the set of all assemblies that can arise (in finitely many steps or in the limit) from $\mathcal{T}$. An assembly $\alpha \in \mathcal{A}[\mathcal{T}]$ is *terminal*, and we write $\alpha \in \mathcal{A}_\Box[\mathcal{T}]$, if no tile can be τ-stably added to it. It is clear that $\mathcal{A}_\Box[\mathcal{T}] \subseteq \mathcal{A}[\mathcal{T}]$.

An assembly sequence in a TAS $\mathcal{T}$ is a (finite or infinite) sequence $\vec{\alpha} = (\alpha_0, \alpha_1, \ldots)$ of assemblies in which each α_{i+1} is obtained from α_i by the addition of a single tile. The *result* $\text{res}(\vec{\alpha})$ of such an assembly sequence is its unique limiting assembly. (This is the last assembly in the sequence if the sequence is finite.) The set $\mathcal{A}[\mathcal{T}]$ is partially ordered by the relation $\longrightarrow$ defined by

$$\begin{aligned} \alpha \longrightarrow \alpha' \quad \text{iff} \quad & \text{there is an assembly sequence } \vec{\alpha} = (\alpha_0, \alpha_1, \ldots) \\ & \text{such that } \alpha_0 = \alpha \text{ and } \alpha' = \text{res}(\vec{\alpha}). \end{aligned}$$

If $\vec{\alpha} = (\alpha_0, \alpha_1, \ldots)$ is an assembly sequence in $\mathcal{T}$ and $\vec{m} \in \mathbb{Z}^2$, then the *$\vec{\alpha}$-index* of $\vec{m}$ is $i_{\vec{\alpha}}(\vec{m}) = \min\{i \in \mathbb{N} | \vec{m} \in \text{dom } \alpha_i\}$. That is, the $\vec{\alpha}$-index of $\vec{m}$ is the time at which any tile is first placed at location $\vec{m}$ by $\vec{\alpha}$. For each location $\vec{m} \in \bigcup_{0 \le i \le l} \text{dom } \alpha_i$, define the set of its input sides $\text{IN}^{\vec{\alpha}}(\vec{m}) = \{\vec{u} \in U_2 | \text{str}_{\alpha_{i_\alpha}(\vec{m})}(\vec{u}) > 0\}$.

We say that $\mathcal{T}$ is *directed* (a.k.a. *deterministic*, *confluent*, *produces a unique assembly*) if the relation $\longrightarrow$ is directed, i.e., if for all $\alpha, \alpha' \in \mathcal{A}[\mathcal{T}]$, there exists $\alpha'' \in \mathcal{A}[\mathcal{T}]$ such that $\alpha \longrightarrow \alpha''$ and $\alpha' \longrightarrow \alpha''$. It is easy to show that $\mathcal{T}$ is directed if and only if there is a unique terminal assembly $\alpha \in \mathcal{A}[\mathcal{T}]$ such that $\sigma \longrightarrow \alpha$.

A set $X \subseteq \mathbb{Z}^2$ *weakly self-assembles* if there exists a TAS $\mathcal{T} = (T, \sigma, \tau)$ and a set $B \subseteq T$ such that $\alpha^{-1}(B) = X$ holds for every terminal assembly $\alpha \in \mathcal{A}_{\Box}[\mathcal{T}]$. Essentially, weak self-assembly can be thought of as the creation (or "painting") of a pattern of tiles from B (usually taken to be a unique "color") on a possibly larger "canvas" of un-colored tiles.

A set X *strictly self-assembles* if there is a TAS $\mathcal{T}$ for which every assembly $\alpha \in \mathcal{A}_{\Box}[\mathcal{T}]$ satisfies dom $\alpha = X$. Essentially, strict self-assembly means that tiles are only placed in positions defined by the shape. Note that if X strictly self-assembles, then X weakly self-assembles. (Let all tiles be in B.)

2.3 Informal definition of the 2HAM

The 2HAM [11, 18] is a generalization of the abstract Tile Assembly Model (aTAM) [64] in that it allows for two assemblies, both possibly consisting of more than one tile, to attach to each other. Since we must allow that the assemblies might require translation before they can bind, we define a *supertile* to be the set of all translations of a τ-stable assembly, and speak of the attachment of supertiles to each other, modeling that the assemblies attach, if possible, after appropriate translation. We now give a brief, informal, sketch of the d-dimensional 2HAM, for $d \in \{2, 3\}$, which is normally defined as a 2D model but which we extend to 3D as well, in the natural and intuitive way.

A *tile type* is a unit square if $d = 2$, and cube if $d = 3$, with each side having a *glue* consisting of a *label* (a finite string) and *strength* (a non-negative integer). We assume a finite set T of tile types, but an infinite number of copies of each tile type, each copy referred to as a *tile*. A *supertile* is (the set of all translations of) a positioning of tiles on the integer lattice $\mathbb{Z}^d$. Two adjacent tiles in a supertile *interact* if the glues on their abutting sides are equal and have positive strength. Each supertile induces a *binding graph*, a grid graph whose vertices are tiles, with an edge between two tiles if they interact. The supertile is *τ-stable* if every cut of its binding graph has strength at least τ, where the weight of an edge is the strength of the glue it represents. That is, the supertile is stable if at least energy τ is required to separate the supertile into two parts. A 2HAM *tile assembly system* (TAS) is a pair $\mathcal{T} = (T, \tau)$, where T is a finite tile set and τ is the *temperature*, usually 1 or 2. (Note that this is considered the "default" type of 2HAM system, while a system can also be defined as a triple (T, S, τ), where S is the *initial configuration* which in the default case is just infinite copies of all tiles from T, but in other cases can additionally or instead consist of copies of pre-formed supertiles.) Given a TAS $\mathcal{T} = (T, \tau)$, a supertile is *producible*, written as $\alpha \in \mathcal{A}[\mathcal{T}]$, if either it is a single tile from T, or it is the τ-stable result of translating two producible assemblies without overlap. Note that if $d = 3$, or if $d = 2$ but it is explicitly mentioned that *planarity* is to be preserved, it must be possible for one of the assemblies to start infinitely far from the other and by merely translating in d dimensions arrive into a position such that the combination of the two is τ-stable, without ever requiring overlap. This prevents, for example, binding on the interior of a region completely enclosed by a supertile. A supertile α is *terminal*, written as $\alpha \in \mathcal{A}_{\Box}[\mathcal{T}]$, if for every producible supertile β, α and β cannot be τ-stably attached. A TAS is *directed* if it has only one terminal, producible supertile.

2.4 Formal Definition of the d-Dimensional 2HAM

We now formally define the 2HAM in $d \in \{2, 3\}$ dimensions.

We work in the d-dimensional discrete space $\mathbb{Z}^d$. Define the set U_d to be the set of all *unit vectors* in $\mathbb{Z}^d$ (i.e. vectors of length 1 in $\mathbb{Z}^d$). We also sometimes refer to these vectors by the directions north, east, south, west, up, and down (N, E, S, W, U, D). All *graphs* in this thesis are undirected. A *grid graph* is a graph $G = (V, E)$ in which $V \subseteq \mathbb{Z}^d$ and every edge $\{\vec{a}, \vec{b}\} \in E$ has the property that $\vec{a} - \vec{b} \in U_d$.

Intuitively, a tile type t is a unit square if $d = 2$ and a unit cube if $d = 3$ that can be translated, but not rotated, having a well-defined "side $\vec{u}$" for each $\vec{u} \in U_d$. Each side $\vec{u}$ of t has a "glue" with "label" $\text{label}_t(\vec{u})$–a string over some fixed alphabet–and "strength" $\text{str}_t(\vec{u})$–a nonnegative integer–specified by its type t. Two tiles t and t' that are placed at the points $\vec{a}$ and $\vec{a}+\vec{u}$ respectively, *bind* with *strength* $\mathrm{str}_t(\vec{u})$ if and only if

$$(\mathrm{label}_t(\vec{u}), \mathrm{str}_t(\vec{u})) = (\mathrm{label}_{t'}(-\vec{u}), \mathrm{str}_{t'}(-\vec{u})).$$

In the subsequent definitions, given two partial functions f, g, we write $f(x) = g(x)$ if f and g are both defined and equal on x, or if f and g are both undefined on x.

Fix a finite set T of tile types. A T-*assembly*, sometimes denoted simply as an *assembly* when T is clear from the context, is a partial function $\alpha : \mathbb{Z}^d \dashrightarrow T$ defined on at least one input, with points $\vec{x} \in \mathbb{Z}^d$ at which $\alpha(\vec{x})$ is undefined interpreted to be empty space, so that dom α is the set of points with tiles. We write $|\alpha|$ to denote $|\text{dom } \alpha|$, and we say α is *finite* if $|\alpha|$ is finite. For assemblies α and α', we say that α is a *subassembly* of α', and write $\alpha \sqsubseteq \alpha'$, if dom $\alpha \subseteq$ dom α' and $\alpha(\vec{x}) = \alpha'(\vec{x})$ for all $x \in$ dom α.

Two assemblies α and β are *disjoint* if dom $\alpha \cap$ dom $\beta = \varnothing$. For two assemblies α and β, define the *union* $\alpha \cup \beta$ to be the assembly defined for all $\vec{x} \in \mathbb{Z}^d$ by $(\alpha \cup \beta)(\vec{x}) = \alpha(\vec{x})$ if $\alpha(\vec{x})$ is defined, and $(\alpha \cup \beta)(\vec{x}) = \beta(\vec{x})$ otherwise. Say that this union is *disjoint* if α and β are disjoint.

The *binding graph of* an assembly α is the grid graph $G_\alpha = (V, E)$, where $V =$ dom α, and $\{\vec{m}, \vec{n}\} \in E$ if and only if (1) $\vec{m} - \vec{n} \in U_d$, (2) $\mathrm{label}_{\alpha(\vec{m})}(\vec{n} - \vec{m}) = \mathrm{label}_{\alpha(\vec{n})}(\vec{m} - \vec{n})$, and (3) $\mathrm{str}_{\alpha(\vec{m})}(\vec{n} - \vec{m}) > 0$. Given $\tau \in \mathbb{N}$, an assembly is τ-*stable* (or simply *stable* if τ is understood from context), if it cannot be broken up into smaller assemblies without breaking bonds of total strength at least τ; i.e., if every cut of G_α has weight at least τ, where the weight of an edge is the strength of the glue it represents. In contrast to the model of Wang tiling, the nonnegativity of the strength function implies that glue mismatches between adjacent assemblies do not prevent them from binding, so long as sufficient binding strength is received from the (other) adjacent sides of the tiles at which the glues match.

For assemblies $\alpha, \beta : \mathbb{Z}^d \dashrightarrow T$ and $\vec{u} \in \mathbb{Z}^d$, we write $\alpha + \vec{u}$ to denote the assembly defined for all $\vec{x} \in \mathbb{Z}^d$ by $(\alpha + \vec{u})(\vec{x}) = \alpha(\vec{x} - \vec{u})$, and write $\alpha \simeq \beta$ if there exists $\vec{u}$ such that $\alpha + \vec{u} = \beta$; i.e., if α is a translation of β. Given two assemblies $\alpha, \beta : \mathbb{Z}^d \dashrightarrow T$, we say α is a *subassembly* of β, and we write $\alpha \sqsubseteq \beta$, if $S_\alpha \subseteq S_\beta$ and, for all points $p \in S_\alpha$, $\alpha(p) = \beta(p)$. Define the *supertile* of α to be the set $\tilde{\alpha} = \{ \beta \mid \alpha \simeq \beta \}$. A supertile $\tilde{\alpha}$ is *τ-stable* (or simply *stable*) if all of the assemblies it contains are τ-stable; equivalently, $\tilde{\alpha}$ is stable if it contains a stable assembly, since translation preserves the property of stability. Note also that the notation $|\tilde{\alpha}| \equiv |\alpha|$ is the size of the supertile (i.e., number of tiles in the supertile) is well-defined, since translation preserves cardinality (and note in particular that even though we define $\tilde{\alpha}$ as a set, $|\tilde{\alpha}|$ does not denote the cardinality of this set, which is always $\aleph_0$).

For two supertiles $\tilde{\alpha}$ and $\tilde{\beta}$, and temperature $\tau \in \mathbb{N}$, define the *combination* set $C^\tau_{\tilde{\alpha},\tilde{\beta}}$ to be the set of all supertiles $\tilde{\gamma}$ such that there exist $\alpha \in \tilde{\alpha}$ and $\beta \in \tilde{\beta}$ such that (1) α and β are disjoint (steric protection), (2) if $d = 3$, or if $d = 2$ and *planarity* is explicitly being required, it must be possible to form $\gamma \equiv \alpha \cup \beta$ by the translation in d-dimensions of α beginning infinitely far from β such that, at all times, α and β are disjoint (3) $\gamma \equiv \alpha \cup \beta$ is τ-stable, and (4) $\gamma \in \tilde{\gamma}$. That is, $C^\tau_{\tilde{\alpha},\tilde{\beta}}$ is the set of all τ-stable supertiles that can be obtained by "attaching" $\tilde{\alpha}$ to $\tilde{\beta}$ stably, with $|C^\tau_{\tilde{\alpha},\tilde{\beta}}| > 1$ if there is more than one position at which β could attach stably to α.

It is common with seeded assembly to stipulate an infinite number of copies of each tile, but our definition allows for a finite number of tiles as well. Our definition also allows for the growth of infinite assemblies and finite assemblies to be captured by a single definition, similar to the definitions of [42] for seeded assembly.

Given a set of tiles T, define a *state* S of T to be a multiset of supertiles, or equivalently, S is a function mapping supertiles of T to $\mathbb{N} \cup \{\infty\}$, indicating the multiplicity of each supertile in the state. We therefore write $\tilde{\alpha} \in S$ if and only if $S(\tilde{\alpha}) > 0$.

A *(two-handed) tile assembly system* (*TAS*) is an ordered triple $\mathcal{T} = (T, S, \tau)$, where T is a finite set of tile types, S is the *initial state*, and $\tau \in \mathbb{N}$ is the temperature. If not stated otherwise, assume that the initial state S is defined $S(\tilde{\alpha}) = \infty$ for all supertiles $\tilde{\alpha}$ such that $|\tilde{\alpha}| = 1$, and $S(\tilde{\beta}) = 0$ for all other supertiles $\tilde{\beta}$. That is, S is the state consisting of a countably infinite number of copies of each individual tile type from T, and no other supertiles. In such a case we write $\mathcal{T} = (T, \tau)$ to indicate that $\mathcal{T}$ uses the default initial state. For notational convenience we sometimes describe S as a set of supertiles, in which case we actually mean that S is a multiset of supertiles with infinite count of each supertile. We also assume that, in general, unless stated otherwise, the count for any supertile in the initial state is infinite.

Given a TAS $\mathcal{T} = (T, S, \tau)$, define an *assembly sequence* of $\mathcal{T}$ to be a sequence of states $\vec{S} = (S_i \mid 0 \leq i < k)$ (where $k = \infty$ if $\vec{S}$ is an infinite assembly sequence), and S_{i+1} is constrained based on S_i in the following way: There exist supertiles $\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}$ such that (1) $\tilde{\gamma} \in C^{\tau}_{\tilde{\alpha},\tilde{\beta}}$, (2) $S_{i+1}(\tilde{\gamma}) = S_i(\tilde{\gamma}) + 1$,[1] (3) if $\tilde{\alpha} \neq \tilde{\beta}$, then $S_{i+1}(\tilde{\alpha}) = S_i(\tilde{\alpha}) - 1$, $S_{i+1}(\tilde{\beta}) = S_i(\tilde{\beta}) - 1$, otherwise if $\tilde{\alpha} = \tilde{\beta}$, then $S_{i+1}(\tilde{\alpha}) = S_i(\tilde{\alpha}) - 2$, and (4) $S_{i+1}(\tilde{\omega}) = S_i(\tilde{\omega})$ for all $\tilde{\omega} \notin \{\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}\}$. That is, S_{i+1} is obtained from S_i by picking two supertiles from S_i that can attach to each other, and attaching them, thereby decreasing the count of the two reactant supertiles and increasing the count of the product supertile. If $S_0 = S$, we say that $\vec{S}$ is *nascent.*

Given an assembly sequence $\vec{S} = (S_i \mid 0 \leq i < k)$ of $\mathcal{T} = (T, S, \tau)$ and a supertile $\tilde{\gamma} \in S_i$ for some i, define the *predecessors* of $\tilde{\gamma}$ in $\vec{S}$ to be the multiset $\text{pred}_{\vec{S}}(\tilde{\gamma}) = \{\tilde{\alpha}, \tilde{\beta}\}$ if $\tilde{\alpha}, \tilde{\beta} \in S_{i-1}$ and $\tilde{\alpha}$ and $\tilde{\beta}$ attached to create $\tilde{\gamma}$ at step i of the assembly sequence, and define $\text{pred}_{\vec{S}}(\tilde{\gamma}) = \{\tilde{\gamma}\}$ otherwise. Define the *successor* of $\tilde{\gamma}$ in $\vec{S}$ to be $\succ_{\vec{S}} (\tilde{\gamma}) = \tilde{\alpha}$ if $\tilde{\gamma}$ is one of the predecessors of $\tilde{\alpha}$ in $\vec{S}$, and define $\succ_{\vec{S}} (\tilde{\gamma}) = \tilde{\gamma}$ otherwise. A sequence of supertiles $\vec{\tilde{\alpha}} = (\tilde{\alpha}_i \mid 0 \leq i < k)$ is a *supertile assembly sequence* of $\mathcal{T}$ if there is an assembly sequence $\vec{S} = (S_i \mid 0 \leq i < k)$ of $\mathcal{T}$ such that, for all $1 \leq i < k$, $\succ_{\vec{S}} (\tilde{\alpha}_{i-1}) = \tilde{\alpha}_i$, and $\vec{\tilde{\alpha}}$ is *nascent* if $\vec{S}$ is nascent.

The *result* of a supertile assembly sequence $\vec{\tilde{\alpha}}$ is the unique supertile $\text{res}(\vec{\tilde{\alpha}})$ such that there exist an assembly $\alpha \in \text{res}(\vec{\tilde{\alpha}})$ and, for each $0 \leq i < k$, assemblies $\alpha_i \in \tilde{\alpha}_i$ such that $\text{dom}\ \alpha = \bigcup_{0 \leq i < k} \text{dom}\ \alpha_i$ and, for each $0 \leq i < k$, $\alpha_i \sqsubseteq \alpha$. For all supertiles $\tilde{\alpha}, \tilde{\beta}$, we write $\tilde{\alpha} \to_{\mathcal{T}} \tilde{\beta}$ (or $\tilde{\alpha} \to \tilde{\beta}$ when $\mathcal{T}$ is clear from context) to denote that there is a supertile assem-

[1]with the convention that $\infty = \infty + 1 = \infty - 1$

bly sequence $\vec{\tilde{\alpha}} = (\tilde{\alpha}_i \mid 0 \leq i < k)$ such that $\tilde{\alpha}_0 = \tilde{\alpha}$ and $\text{res}(\vec{\tilde{\alpha}}) = \tilde{\beta}$. It can be shown using the techniques of [57] for seeded systems that for all two-handed tile assembly systems $\mathcal{T}$ supplying an infinite number of each tile type, $\to_{\mathcal{T}}$ is a transitive, reflexive relation on supertiles of $\mathcal{T}$. We write $\tilde{\alpha} \to^1_{\mathcal{T}} \tilde{\beta}$ ($\tilde{\alpha} \to^1 \tilde{\beta}$) to denote an assembly sequence of length 1 from $\tilde{\alpha}$ to $\tilde{\beta}$ and $\tilde{\alpha} \to^{\leq 1}_{\mathcal{T}} \tilde{\beta}$ ($\tilde{\alpha} \to^{\leq 1} \tilde{\beta}$) to denote an assembly sequence of length 1 from $\tilde{\alpha}$ to $\tilde{\beta}$ if $\tilde{\alpha} \neq \tilde{\beta}$ and an assembly sequence of length 0 otherwise.

A supertile $\tilde{\alpha}$ is *producible*, and we write $\tilde{\alpha} \in \mathcal{A}[\mathcal{T}]$, if it is the result of a nascent supertile assembly sequence. A supertile $\tilde{\alpha}$ is *terminal* if, for all producible supertiles $\tilde{\beta}$, $C^{\tau}_{\tilde{\alpha},\tilde{\beta}} = \varnothing$.[2] Define $\mathcal{A}_{\Box}[\mathcal{T}] \subseteq \mathcal{A}[\mathcal{T}]$ to be the set of terminal and producible supertiles of $\mathcal{T}$. $\mathcal{T}$ is *directed* (a.k.a., *deterministic*, *confluent*) if $|\mathcal{A}_{\Box}[\mathcal{T}]| = 1$.

[2]Note that a supertile $\tilde{\alpha}$ could be non-terminal in the sense that there is a producible supertile $\tilde{\beta}$ such that $C^{\tau}_{\tilde{\alpha},\tilde{\beta}} \neq \varnothing$, yet it may not be possible to produce $\tilde{\alpha}$ and $\tilde{\beta}$ simultaneously if some tile types are given finite initial counts, implying that $\tilde{\alpha}$ cannot be "grown" despite being non-terminal. If the count of each tile type in the initial state is ∞, then all producible supertiles are producible from any state, and the concept of terminal becomes synonymous with "not able to grow", since it would always be possible to use the abundant supply of tiles to assemble $\tilde{\beta}$ alongside $\tilde{\alpha}$ and then attach them.

Chapter 3

Computational Universality in Non-cooperative Tile Assembly Systems

3.1 Using Geometry to Simulate a Turing Machine

3.1.1 Introduction

Theoretical studies of algorithmic self-assembly have produced a wide variety of results that establish the computational power of tile-based self-assembling systems. From the introduction of the first and perhaps simplest model, the abstract Tile Assembly Model (aTAM) [64], it was shown that self-assembling systems, which are based on relatively simple components autonomously coalescing to form more complex structures, are capable of Turing-universal computation. This computational power exists within the aTAM, and has been harnessed to algorithmically guide extremely complex theoretical constructions (e.g. [62, 59, 41, 54, 22, 53]) and has even been exploited within laboratories to build nanoscale self-assembling systems from DNA-based tiles which self-assemble following algorithmic behavior (e.g. [58, 60, 43, 5, 40, 28]).

While physical implementations of these systems are constantly increasing in scale, complexity, and robustness, they are orders of magnitude shy of achieving results similar to those of many naturally occurring self-assembling systems, especially those found in biology (e.g. the formation of many cellular structures or viruses). This disparity motivates theoretical studies that can focus efforts on first discovering the "tricks" used so successfully by nature, and then on incorporating them into our own models and systems. One of the fundamental properties so successfully leveraged by many natural systems, but absent from models such as the aTAM, is geometric complexity of components. For instance, self-assembly in biological systems relies heavily upon the complex 3-dimensional structures of proteins, while tile-assembly systems are typically restricted to basic square (or cubic) tiles. We greatly extend previous work that has begun to incorporate geometric aspects of self-assembling components [31, 25, 39, 17] with the development of a model allowing for more geometrically complex tiles, called *polyominoes*, and an examination of the surprising computational powers of systems composed of polyominoes.

The process of tile assembly begins from a *seed* structure, typically a single tile, and proceeds with tiles attaching one at a time to the growing assembly. Tiles have *glues*, taken from a set of glue types, around their perimeters which allow them to attach to each other if their glues match. Algorithmic self-assembling systems developed by researchers, both theoretical and experimental, tend to fundamentally employ an important aspect of tile assembly known as *cooperation*. In theoretical models, cooperation is available when a particular parameter, known as the *temperature*, is set to a value > 1 which can then enforce that the binding of a tile to a growing assembly can only occur if that tile matches more than one glue on the perimeter of the assembly. Using cooperation, it is simple to show that systems in the aTAM are capable of universal computation (by simulating particular cellular automata [64], or arbitrary Turing machines [54, 41, 62], for instance). However, it has long been conjectured that in the aTAM without cooperation, i.e. in systems where the temperature is equal to 1, universal computation is impossible [26, 46, 44]. Interestingly, though, a collection of "workarounds" have been devised in the form of new models with a variety of properties and parameters which make computation possible at temperature 1 (e.g. [52, 12, 31, 50, 39]).

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

In addition to this main positive result about the computational abilities of all polyominoes of size ≥ 3, we also provide negative results that further help to refine understanding of exactly what geometric properties are needed for Turing universal computation in temperature-1 self-assembly. We prove that a fundamental gadget (which we call the *bit-reading gadget*) used within all known systems that can compute at temperature 1 in any tile-assembly model, is impossible to construct with either the square tiles of the aTAM or with dominoes (a.k.a. duples). This provides further evidence that systems composed solely of those shapes are incapable of universal computation.

3.1.2 Polyomino Tile Assembly Model

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

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

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

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

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

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

Note that the aTAM is simply a specific case of the polyTAM in which all tiles are monominoes, i.e., single unit squares. Moreover, the class of systems $\mathcal{T} = (T, \sigma, \tau)$ where each tile in T has the polyomino shape of either a single unit square or a domino is called the *Dupled Tile Assembly Model*, and is abbreviated as DaTAM.

3.2 Universal Computation with Geometric Bit-Reading

In this section we provide an overview of how universal computation can be performed in a temperature-1 system with appropriate use of geometric aspects of tiles and assemblies. Refer to Figure 3.1 for an illustration.

3.2.1 Bit-Reading Gadgets

First, we discuss a primitive tile-assembly component that enables computation by self-assembling systems. This component is called the *bit-reading gadget*, and essentially consists of pre-existing assemblies that appropriately encode bit values (i.e., 0 or 1) and paths that grow past them and are able to "read" the values of the encoded bits; this results in those bits being encoded in the tile types of the paths beyond the encoding assemblies. In tile-assembly systems in which the temperature is ≥ 2, a bit-reader gadget is trivial: the assembly encoding the bit value can be a single tile with an exposed glue that encodes the bit value, and the path that grows past to read the value simply ensures that a tile must be placed cooperatively with, and adjacent to, that encoding the bit (i.e., the path forces a tile to be placed that can only bind if one of its glues matches that exposed by the last tile of the path, and the other matches the glue encoding the bit value). However, in a temperature-1 system, cooperative binding of tiles cannot be enforced, and therefore the encoding of bits must be done using geometry. Figure 3.1 provides an intuitive overview of a temperature-1 system with a bit-reading gadget. Essentially, depending on which bit is encoded by the assembly to be read, exactly one of two types of paths can complete growth past it, implicitly specifying the bit that was *read.* It is important that the bit reading must be unambiguous, i.e., depending on the bit *written* by the pre-existing assembly, exactly one type of path (i.e., the one that denotes the bit that was written) can possibly complete growth, with all paths not representing that bit being prevented. Furthermore, the correct type of path must always be able to grow. Therefore, it cannot be the case that either all paths can be blocked from growth, or that any path not of the correct type can complete, regardless of whether a path of the correct type also completes, and these conditions must hold for any valid assembly sequence to guarantee correct computation.

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

1. α is translated so that α_0 has its minimal y-coordinate ≤ 0 and its minimal x-coordinate

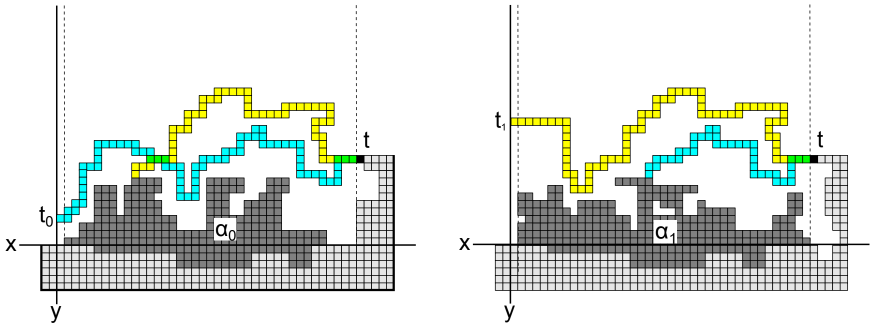


Figure 3.1: Abstract schematic of a bit-reading gadget. (Left) The blue path grown from t "reads" the bit 1 from α_0 (by being allowed to grow to $x = 0$ and placing a tile $t_0 \in T_0$), while the yellow path (which could read a 0 bit) is blocked by α_0. (Right) The yellow path grown from t reads the bit 0 from α_1, while the blue path that could potentially read a 1 is blocked by α_1. Clearly, the specific geometry of the used polyomino tiles and assemblies is important in allowing the yellow path in the left figure to be blocked without also blocking the blue path.

$= 1$,

2. a tile of some type $t \in T$ is placed at $(w + n, h)$, where $n, h \geq 1$, and

3. the tiles of α_0 are the only tiles of α in the first quadrant to the left of t,

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

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

We refer to α_0 and α_1 as the *bit writers*, and the paths which grow from t as the *bit readers*. Also, note that while this definition is specific to a bit-reader gadget in which the bit readers grow from right to left, any rotation of a bit reader is valid by suitably rotating the positions and directions of Definition 3.2.1. As mentioned in Figure 3.1, depending on the actual geometries of the polyominoes used and their careful placement, it may be possible to enforce the necessary blocking of all paths of the wrong type, while still allowing at least one path of the correct type to complete growth in any valid assembly sequence.

3.2.2 Turing-Machine Simulation

In order to show that a polyomino shape (i.e., a system composed of tiles of only that shape) is computationally universal at $\tau = 1$, we show how it is possible to simulate an arbitrary Turing machine using such a polyomino system. In order to simulate an arbitrary Turing machine, we show how to self-assemble a zig-zag Turing machine [12, 52]. A zig-zag Turing machine at $\tau = 1$ works by starting with its input row as the seed assembly, then growing rows one by one, alternating growth from left to right with growth from right to left. As a row grows across the top of the row immediately beneath it, it does so by forming a path of single tile width, with tiles connected by glues, which pass information horizontally through their glues, while the geometry of the row below causes only one of two choices of paths to grow at regular intervals, effectively passing information vertically via the geometry, using bit-reading gadgets.

Each cell of the Turing machine's tape is encoded by a series of bit-reader gadgets that encode in binary the symbol in that cell and, if the read/write head is located there, what state the machine is in. Additionally, as each cell is read by the row above, the necessary information must be geometrically written above it so that the next row can read it. See Figure 3.2 for an example depicting a high-level schematic without showing details of the individual polyominoes. Figure 3.3 shows the same system after two rows have completed growth.

For a more specific example that shows the placement of individual, actual polyomino tiles as well as the order of their growth, see Figure 3.4. Note that the simulation of a zig-zag Turing machine can be performed by horizontal or vertical growth, and in any orienta-

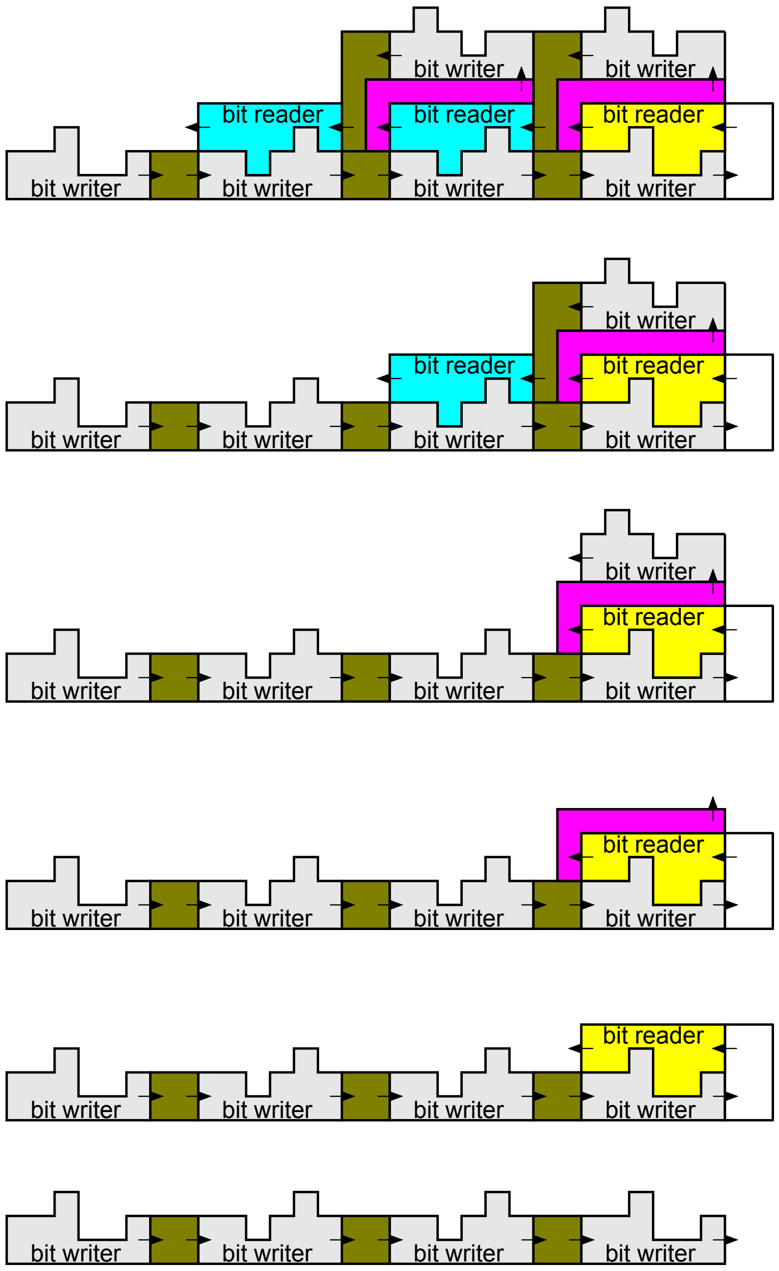


Figure 3.2: High-level schematic view of a zig-zag Turing machine and the bit-reading/writing gadgets that make up each row of the simulation. The bottom shows the seed row, consisting of bit-writer gadgets separated by spacers. Then, depicted as consecutive upward figures, the second row begins its growth. Yellow/blue portions depict locations of bit-reader gadgets (for 0 and 1, respectively), which grow pink paths upward after completing in order to grow bit writer gadgets (grey), and then gold spacers back down to the point where the next bit reader can grow.

tion.

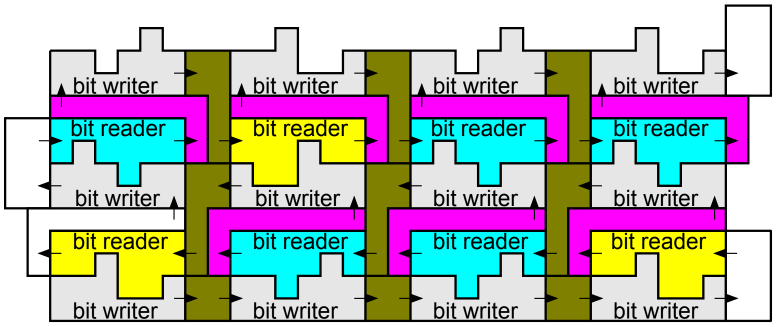


Figure 3.3: High-level schematic view of a zig-zag Turing machine and the bit-reading/writing gadgets that make up the first two rows of simulation.

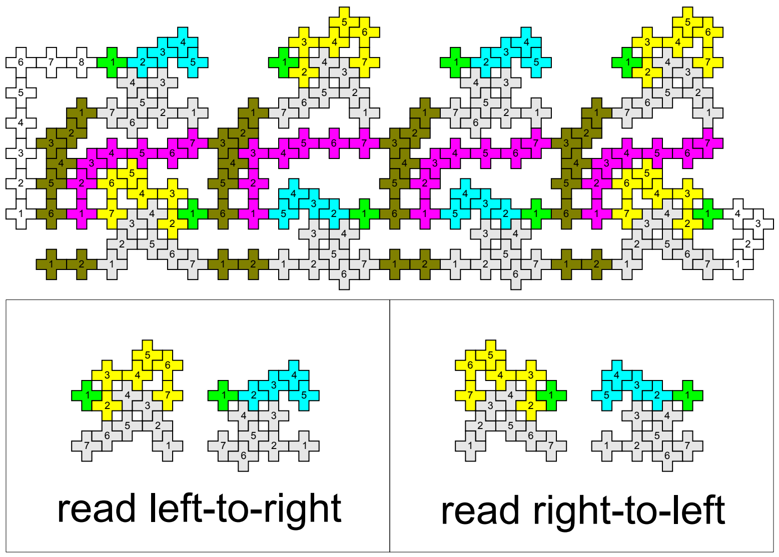


Figure 3.4: The system of Figure 3.2 after two rows of the zig-zag simulation have been completed (omitting the output bit writer gadgets of the second row), implemented with "plus-sign" polyominoes. The bottom left shows 0 and 1 bit-writer and reader combinations, with the writer having grown from right to left and the reader from left to right. The bottom right shows the same, but with growth directions reversed. Grey tiles represent bit-writer gadgets. Green tiles represent the beginning of bit-reader gadgets that are common to either bit; yellow represents the path that can grow to signify a 0 bit being read, and blue a 1 bit. Other colors correspond to those for the gadgets used in Figure 3.2, with numbers corresponding to the growth order of the tiles in each gadget.

3.3 Technical Lemmas: Grids of Polyominoes

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

The following lemma says that if a polyomino P can be translated by a vector $\vec{v}$ so that no pixel positions of the translated polyomino overlap the pixel positions of P, then for any integer $c \neq 0$, no pixel positions of P translated by $c \cdot \vec{v}$ overlap the pixel positions of P. The proof of Lemma 1 can be found in [17]; the statement of the lemma has been included for the sake of completeness.

Lemma 1. Consider a two-dimensional, bounded, connected, regular closed set S, i.e., S is equal to the topological closure of its interior points. Suppose S is translated by a vector v to obtain shape S_v, such that S and S_v have disjoint interiors. Then the shape S_{c*v} obtained by translating S by $c * v$ for any integer $c \neq 0$ and S have disjoint interiors.

Informally, the following lemma says that any polyomino gives rise to a polyTAM system that can produce an infinite "grid" of polyominoes, as shown in Figure 3.5.

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

Details of the proof are omitted for space reasons and can be found in [29].

Let $p, p' \in P$ be distinct pixels in the polyomino P at positions (x, y) and (x', y') respectively, let $\vec{r} = (x - x', y - y')$, and let $\vec{v}, \vec{w}$ be as defined in Lemma 2. Then, if there exists $c_1, c_2 \in \mathbb{Z}$ such that $(x, y) + c_1\vec{v} + c_2\vec{w} = (x', y')$, we say that the polyomino which

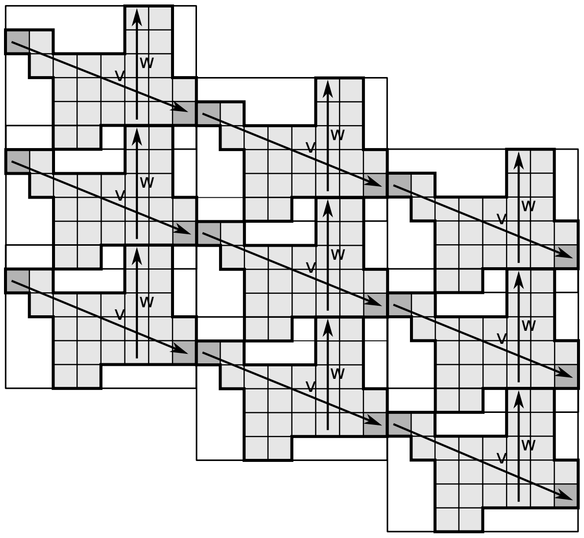


Figure 3.5: A lattice formed from an example polyomino, using the vectors $\vec{v}$ and $\vec{w}$.

occupies (x', y') is $\vec{r}$-shifted with respect to (or relative to) the polyomino at (x, y). If a polyomino at position (x', y') is 0-shifted with respect to a polyomino at (x, y), we say that the polyomino at position (x', y') is *on grid* with the polyomino at (x, y). If a polyomino is not on grid with a polyomino at (x, y), we say that the polyomino is *off grid*. Henceforth, if we do not mention the tile which another tile is shifted in respect to, assume that the tile is shifted with respect to the seed.

For the remainder of this section, for a polyomino P, we let $V \subset \mathbb{Z}^2$ denote the set of vectors such that $\vec{r} \in V$ provided that there exists some directed, singly seeded system, single shape $\mathcal{T}' = (T', \sigma')$ with shape given by P whose terminal assembly α' contains an $\vec{r}$-shifted polyomino tile, and we let B denote the subset of vectors $B \subset V$ such that $\vec{b} \in B$ provided that there exists a directed, singly seeded system, single shape $\mathcal{T} = (T, \sigma)$ with shape given by P such that the terminal assembly α of $\mathcal{T}$ consists of exactly two tiles: the seed tile σ and a $\vec{b}$-shifted tile. B can be thought of as the set of vectors such that the polyomino P and a copy of P shifted by a vector in B are non-overlapping and contain pixels that share a common edge. We can think of B as a set of basis vectors for V in the following sense. If $\vec{r} \in V$, then $\vec{r}$ can be written as a linear combination of shifts in B. The following lemma is a more formal statement of this fact.

Lemma 3. For any vector $\vec{r} \in V$, $\vec{r} = \Sigma c_i \vec{b_i}$ for some $c_i \in \mathbb{Z}$ and $\vec{b_i} \in B$.

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

For a rectangle R and a tile T, we say that T *lies in the southeast* (respectively northwest) *corner* of R iff the south and east edges of the bounding rectangle of the tile T lie on the south and east edges of R. Let V contain every linear combination $\Sigma c_i \vec{b_i}$ for $c_i \in \mathbb{Z}$ and $\vec{b_i} \in B$. The next two lemmas formalize the following notion. For $c \in \mathbb{Z}$ and $\vec{b} \in B$, the following lemma shows how if $\vec{r} = c \cdot \vec{b}$, we can give a system $\mathcal{T}_{\vec{r}}$ that contains an $\vec{r}$-shifted tile. Furthermore, the properties given in the lemma statement ensure that if we have two such system, $\mathcal{T}_{\vec{r}_1}$ and $\mathcal{T}_{\vec{r}_2}$, corresponding to two shift vectors $\vec{r}_1$ and $\vec{r}_2$, $\mathcal{T}_{\vec{r}_1}$ and $\mathcal{T}_{\vec{r}_2}$ can be "concatenated" to give a system that contains an $(\vec{r}_1 + \vec{r}_2)$-shifted polyomino tile. See Figure 3.6 for schematic depictions of the properties given in the following lemma.

Lemma 4. Let $\vec{r} = c \cdot \vec{b}$ for any $c \in \mathbb{Z}$ and $\vec{b} \in B$. Then there exists some directed, singly seeded system $\mathcal{T} = (T, \sigma)$ with all tiles shaped P such that the terminal assembly α of $\mathcal{T}$ contains an $\vec{r}$-shifted tile. Moreover, the system $\mathcal{T}$ and assembly α have the following properties.

1. There is a unique assembly sequence that yields α,

2. For some $m, n \in \mathbb{N}$, α is contained in an $m \times n$ rectangle R and the seed tile S lies in the southeast corner of R, and

3. the last tile, A, to attach to α lies in the northwest corner of R.

Please see the full paper [29] for the proof of Lemma 4.

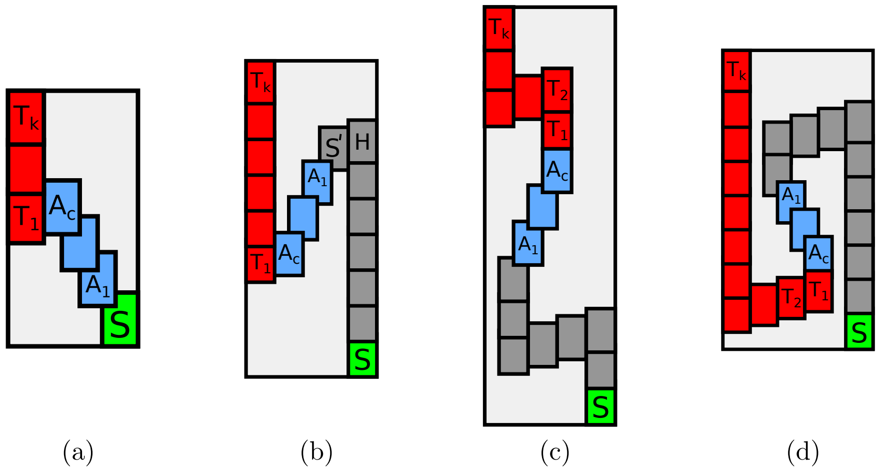


Figure 3.6: A depiction of the properties given in Lemma 4 for four different cases. Each of the small rectangles (red, green, blue or gray in color) serves as the bounding rectangle of some polyomino. Hence, α is contained in the union of all of the regions bounded by these rectangles. The seed tile S of α is contained in the green rectangle. As assembly proceeds from the seed the first r-shifted polyomino tile is A_c. From this point on in the assembly process, the rectangles are on grid with A_c.

As previously mentioned, the properties given in Lemma 4 ensure that if we have two such systems, $\mathcal{T}_{\vec{r}_1}$ and $\mathcal{T}_{\vec{r}_2}$ corresponding to two shift vectors $\vec{r}_1$ and $\vec{r}_2$, $\mathcal{T}_{\vec{r}_1}$ and $\mathcal{T}_{\vec{r}_2}$ can be "concatenated" to give a system that contains an $(\vec{r}_1 + \vec{r}_2)$-shifted polyomino tile. The following lemma formalizes this notion of "concatenation".

Lemma 5. Let $\vec{r} = \Sigma_{i=0}^{n} c_i \vec{b_i}$ for any $c_i \in \mathbb{Z}$ and $\vec{b_i} \in B$. Then there exists some directed, singly seeded system $\mathcal{T} = (T, \sigma)$ with all tiles shaped P such that the terminal assembly α of $\mathcal{T}$ contains an $\vec{r}$-shifted polyomino. Moreover, the system $\mathcal{T}$ and assembly α have the following properties.

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

Proof. This follows by applying Lemma 4 to each of the summands of $\vec{r} = \Sigma c_i \vec{b_i}$. □

In Lemma 5, we start with a seed tile in the southeast corner of a rectangular region R and proceed to place an r-shifted tile in the northwest corner of the rectangle. Note that by using the techniques used to prove Lemma 4 and Lemma 5, we can show analogous lemmas where the seed tile lies in any corner of R and an r-shifted tile lies in the opposite corner.

Now we state the main lemma for this section. This lemma will allow us to construct bit-writing gadgets used in the construction given in Section 3.4. Intuitively, the lemma states that given a polyomino P and vector $\vec{r} \in V$, we can define a polyTAM system that starts growth from a seed tile, S, in the southeast corner of a rectangle R, and without growing outside of R, places $\vec{r}$-shifted tiles on the west edge of R in such a way that it is possible to then continue growth to the west of R. The possibility of continuing growth to the west of R is formally stated as Property 4 in Lemma 6. This lemma will allow us to assemble a series of bit-writers while also resting assured that once these bit-writers have assembled, bit-reader assemblies can continue growth. It is helpful to see Figure 3.7 for an overview of the properties given in the following lemma.

Lemma 6. Let P be a polyomino. Let $\vec{r}$ be a vector in V. Then there exists a directed, singly seeded system $\mathcal{T} = (T, \sigma)$ with all tiles of shape P which produces α such that $\mathcal{T}$ and α have the following properties.

1. There is a unique assembly sequence that yields α,

2. for some $m, n \in \mathbb{N}$, α is contained in an $m \times n$ rectangle R and the seed tile S lies in the southeast corner of R,

3. for any tile A of α such that the west edge of the bounding rectangle of A lies on the west edges of R, A is $\vec{r}$-shifted, and

4. for any $m'', n'' \in \mathbb{N}$, we can choose $\mathcal{T}$ such that the last tile L to attach to α lies on grid in the northeast corner of a rectangle R' with dimensions $m' \times n'$ where $m' > m''$ and $n' > n''$. Moreover, the south and west edges of R' lie on the south and west

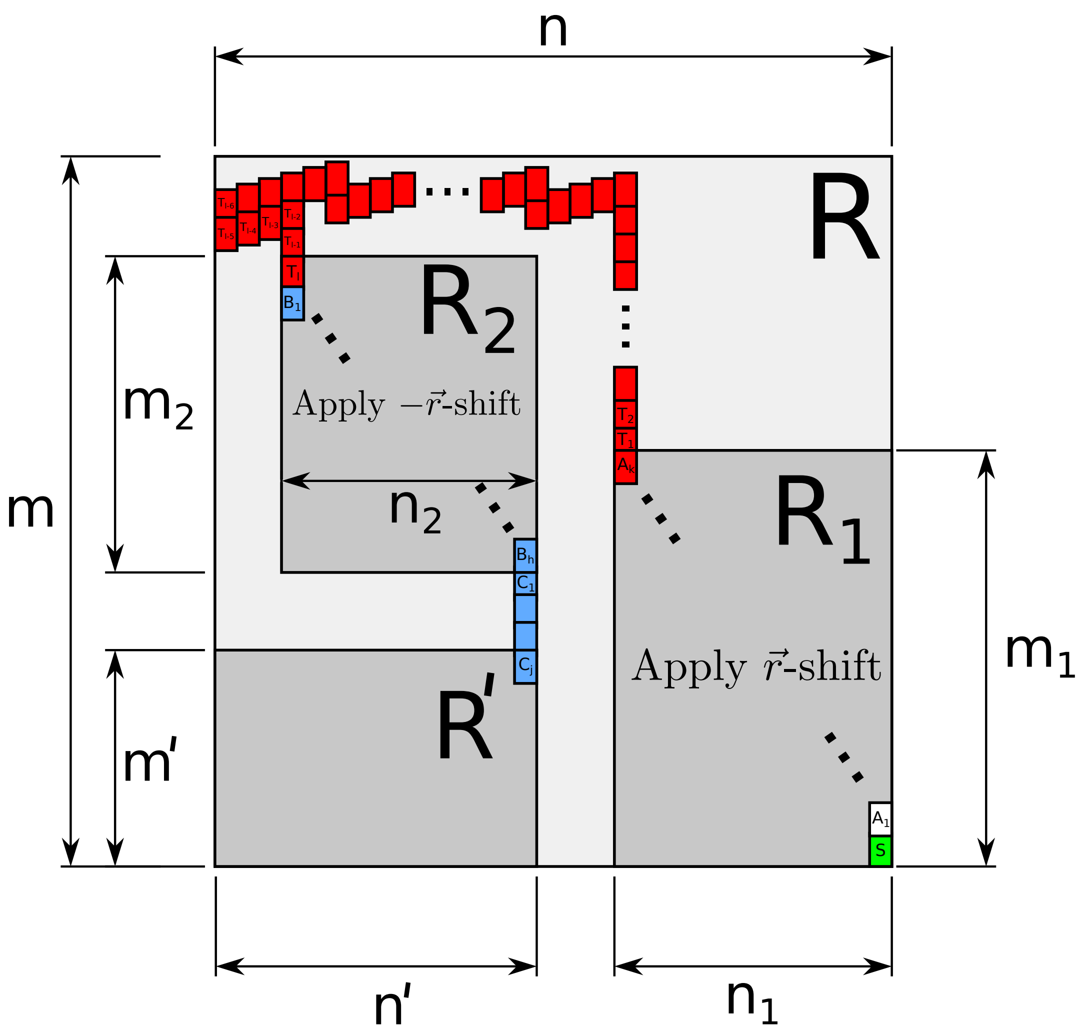


Figure 3.7: The rectangular region R_1 contains tiles of α defined using Lemma 5 with vector $\vec{r}$. The red rectangular regions each serve as the bounding rectangle of an $\vec{r}$-shifted tile of α. This path of tiles, $\{T_i\}_{i=1}^{l}$, places a final tile in the northwest corner of the region R_2. R_2 contains tiles of α defined using Lemma 5 with vector $-\vec{r}$-shifted. The blue rectangular regions each contain a single tile that is on grid with S. This path of tiles places a final tile in the northeast corner of the region R'. Note that by modifying the path of tiles $\{T_i\}_{i=1}^{l}$, we can make the dimension m' and n' of R' arbitrarily large.

edges of R, and no portion of any tile of α lies inside of R' and outside of the bounding rectangle of L.

Please see the full paper [29] for the proof of Lemma 6.

As in Lemma 5, we start with a seed tile in the southeast corner of a rectangular region R and proceed to place r-shifted tiles on the west edge of the rectangle. Note that we can show analogous lemmas where the seed tile starts in any corner of a rectangle R, and r-shifted tiles are placed on a chosen opposite edge.

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

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

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

It follows from the procedure outlined in Section 3.2.2 and the Lemmas of Section 3.3 that in order to simulate an arbitrary Turing Machine by a TAS consisting only of tiles of some polyomino shape P, it is sufficient to construct a system consisting only of tiles of shape P for which there exists a bit-reading gadget, because the additional paths required for a zig-zag Turing machine simulation are guaranteed to be producible by the lemmas of Section 3.3.

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

Definition (Basic polyomino)**.** A polyomino P is said to be a *basic* polyomino if and only if for every vector $\vec{x}$ modulo the polyomino grid for P, there exists a system $\mathcal{T}$ containing only tiles with shape P such that $\mathcal{T}$ produces α and α contains a $\vec{x}$-shifted polyomino. Otherwise we call P *non-basic*.

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

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

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

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

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

(4) P is non-basic.

The Lemmas of Section 3.3 provide us with the basic facilities to build paths of tiles which occupy particular points while avoiding others. By carefully designing the grids and offsets for the tiles of each polyomino P, we are able to construct the constituent paths of the bit-reading gadgets.

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

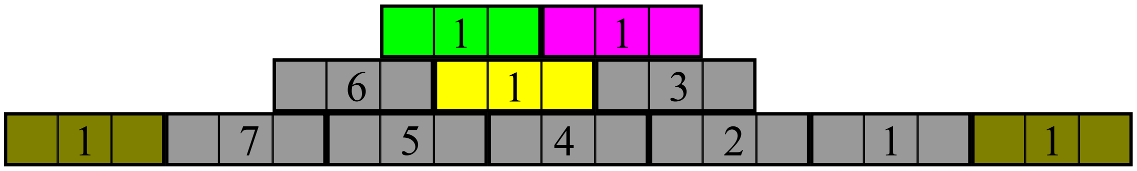
For any polyomino P, we know that tiles of shape P can produce a grid by Lemma 2; throughout this section, we simply refer to this as the grid (for P). We also note that the grid for a given P may be slanted as in Figure 3.5, and that the construction of the zig-zag Turing machine is simply slanted accordingly. If we say that a tile is $\vec{v}$-shifted for some vector $\vec{v}$, we mean that it is off grid by the vector $\vec{v}$.

For all figures in this section, we use the same color conventions as in Figure 3.1. Thus, the green tiles in this section represent the t tile; as discussed in the caption of the figure, the yellow and blue tiles represent the two potential paths grown from t, while the dark grey tiles represent tiles that prevent the growth of paths from t. We refer to these grey tiles as *blockers*. We use the convention that if a path of yellow polyominoes grows, then a 0 is being read. Similarly, if a path of blue polyominoes grows, then a 1 is being read. Consequently, we call polyominoes that prevent the growth of the blue path 1-*blockers* and polyominoes that prevent the growth of the yellow path 0-*blockers*. In addition, tiles of the same color are numbered in order to indicate the order of their placement where the higher numbered tiles are placed later in the assembly sequence.

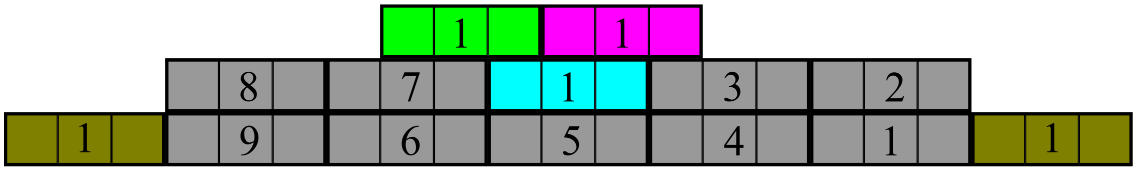
3.4.1 Case (1): P Is an $m \times 1$ Polyomino

If P is a straight line, and therefore $n = 1$, we can simply use a scheme as illustrated in Figure 3.8. In Figure 3.8a, a 0 bit is read as indicated by the placement of the yellow polyomino. Notice that the tile labeled 3 in Figure 3.8a prevents the attachment of the blue-colored tile. After the yellow tile attaches, a fuchsia tile attaches as shown in the figure which allows for growth to continue. Figure 3.8b shows a similar scenario in the case that a 1 is read. The key property of this bit reader is that the yellow and blue tiles have different offsets relative to the green tile, which is always possible if P is a line of length ≥ 3. Note that the bit reader shown in Figure 3.8 is a left-to-right bit-reading gadget. A right-to-left bit-reading gadget can be constructed in a similar fashion.

The case for any P which is an $n \times 1$ straight line can be handled in the same way. Thus, we now only need to consider the cases $m \geq n \geq 2$. Because P is connected, this implies both $d_x \geq 2$ and $d_y \geq 2$. Furthermore, we assume the that the grid constructed from Lemma 2 using P is created by attaching the southernmost pixel on the eastern edge of P to the northernmost pixel on the western side of the P, as suggested by Figure 3.5.



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



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

Figure 3.8: The two different bit-reading schemes for an $m \times 1$ polyomino. Note that the bit reader in this figure proceeds from left to right.

3.4.2 Case (2): P Is Such That $d_x = d_y = 2$

Before describing bit-reading constructions, we analyze the possible cases for the shape of P; refer to Figure 3.9. Also, we note that $d_x = d_y = 2$ implies that P is basic.

First consider the situation in which $|P|$ is even. If both $(1, 0)$ and $(0, 1)$ belong to P, the assumption $d_x = d_y = 2$ implies that $(1, 1)$ must also belong to P, but no further pixels. Thus, P is a 2×2-square, which will be treated as **Case (2a)**. Now, without loss of generality consider the case that $(0, 1)$ belongs to P, but $(1, 0)$ does not. It follows from $d_x = d_y = 2$ that $(1, 1)$ belongs to P, as well as $(1, 2)$. This conclusion can be repeated until all pixels of P are allocated. It follows that P is an even zig-zagging shape. This is shown as **Case (2b)** in Figure 3.9.

Now consider the case in which $|P|$ is odd. If both $(1, 0)$ and $(0, 1)$ belong to P, $(1, 1)$ cannot be part of P, and P is an L-shape consisting of three pixels. If without loss of generality $(0, 1)$ belongs to P, but $(1, 0)$ does not, we can conclude analogous to (2a) that P is an odd zig-zagging shape, shown as **Case (2c)** in Figure 3.9; this also comprises the case of an L-shape with three pixels.

Now we sketch the bit-reading schemes. As these cases are relatively straightforward, we simply refer to the corresponding figures. Note that the logic of the arrangement is color coded: the first polyomino we add to our tile set is the green polyomino along with

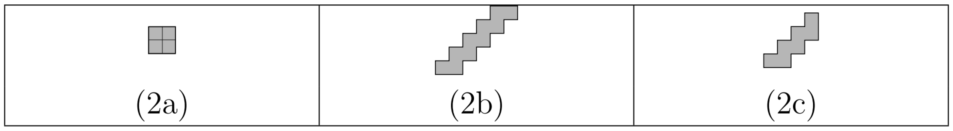
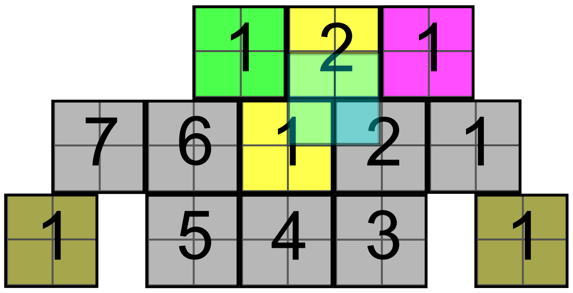


Figure 3.9: The possible shapes in Case (2), when $d_x = d_y = 2$.

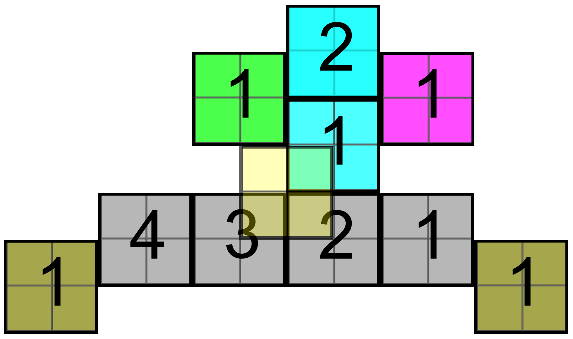
the blue and yellow polyominoes that allow for an blue tile to attach to the east of it in an on grid position and a yellow tile to attach to the east of it shifted $(-1, -1)$ relative to the polyomino grid.

Case (2a)

If P is a 2×2 square we use the scheme shown in Figure 3.10.



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



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

Figure 3.10: A general bit-reading scheme for a left-to-right bit reader in case (2a), in which P is a 2×2 square.

Case (2b)

If P is an even zig-zagging polyomino, as shown in part (2b) of Figure 3.9, we use the bit-reading schemes shown in Figure 3.11.

Case (2c)

If P is an odd zig-zagging polyomino, as shown in part (2c) of Figure 3.9, we use the bit-reading schemes shown in Figure 3.12.

This concludes Case (2).

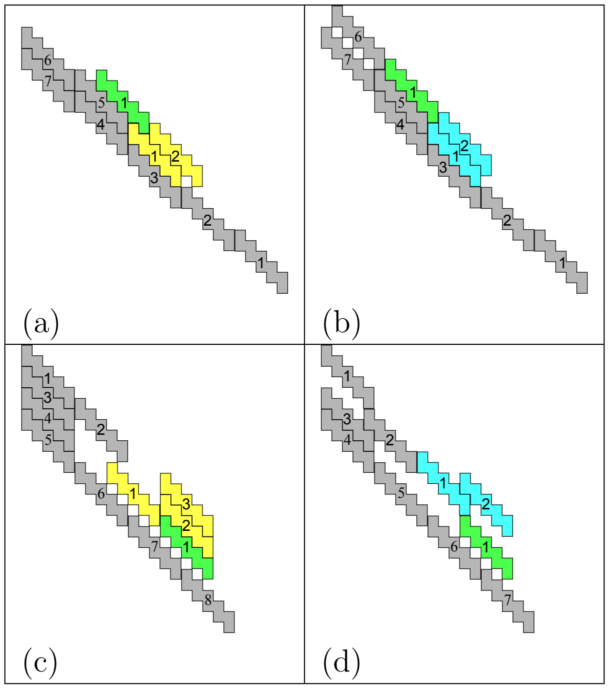


Figure 3.11: The bit-reading schemes for Case (2b) of Figure 3.9.

3.4.3 Case (3): P Is Basic And Not In Case (1) or (2)

We now describe how to construct a system that contains a bit-reading gadget in the case that $\max\{d_x, d_y\} \geq 3$, $\min\{d_x, d_y\} \geq 2$, and P is a basic polyomino. This means that it is possible to construct a path using tiles of shape P which place a tile at any possible offset in relation to the grid. We will use this ability to place blockers and bit-reader paths exactly where we need them, with those locations specified throughout the description of this case. Without loss of generality, assume that $\max\{d_x, d_y\} = d_y$.

Case (3) Overview

A schematic diagram showing the growth of the bit-reading gadget system we construct is shown in Figure 3.13. Note that the figure depicts what the bit-reader would look like if the grid formed by P was a square grid. In cases of a slanted grid (such as that shown in Figure 3.5), the bit-gadgets would be correspondingly slanted. Growth of the system begins with the seed as shown in Figure 3.13. From the seed, the system grows a path of tiles

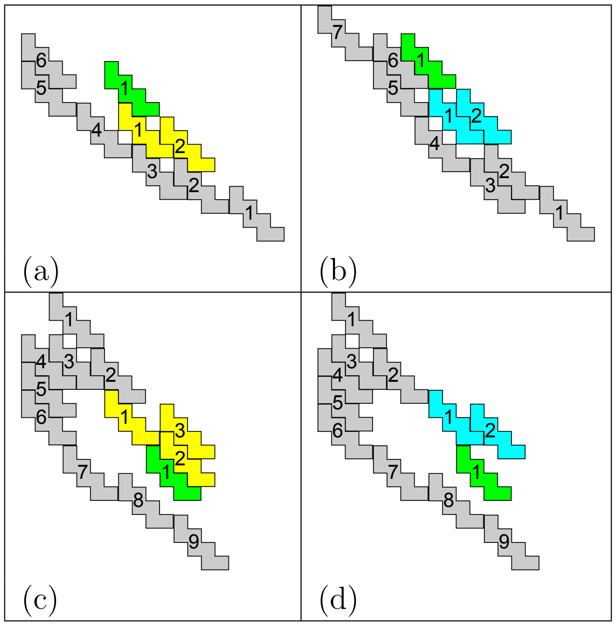


Figure 3.12: The bit-reading schemes for Case (2c) of Figure 3.9.

west (shown as a light grey path in the figure) to which one of the two bit writers attach (shown as dark grey in the figure). Once one of the bit writers assembles, growth proceeds as shown in the schematic view until the other bit writers assemble. Then growth continues upward to the next level (i.e. the seed row can be considered a “zig” row and the next row a “zag” row of the zig-zag Turing machine simulation) as shown in the figure until a green tile is placed. Depending on the bit writer gadget to the east of the green tile either a yellow path of tiles grows, indicating that a 0 has been read (as shown in the schematic view with the westernmost bit writer), or a blue path of tiles grows, indicating that a 1 has been read (as shown in the schematic view with the easternmost bit writer). Henceforth, we refer to the system described by the schematic view in Figure 3.13 as the bit-reading gadget.

The light grey tiles that compose the bit-reading gadget are easily constructed by placing glues on the polyomino P so that they grow the paths shown in Figure 3.13 (again, modulo the slant of the particular grid formed by P), which are on grid with the seed, where the grid is formed following the technique used in the proof of Lemma 2. The construction of the other tiles is now described. The green tile is constructed by placing a glue

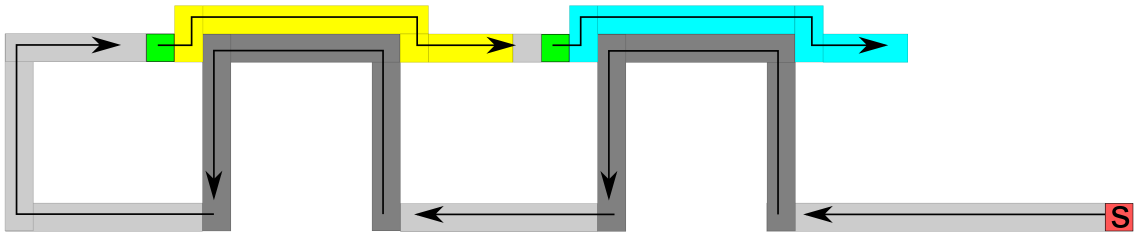


Figure 3.13: A schematic diagram showing the growth of the bit-reading gadget system for Case (3). Note that in the actual construction, a larger gap would exist between the two bit-readers to allow the path between them to first extend upward and create the necessary bit-writer, then come back down and continue growth of the next bit-reader.

on its western side so that it attaches to the grey tiles on grid as shown in the schematic view. Furthermore, glues are placed on the green tile and the first blue tile so that the blue tile attaches to the green tile in an on-grid manner. Glues are placed on the green tile and the first yellow tile so that the southern edge of the southernmost pixel on the east perimeter of the green tile attaches to the northern edge of the northernmost pixel on the western perimeter of the yellow tile (thus putting the yellow tile off grid).

Case (3) Bit-Writer Construction

First we describe the construction of the bit-writer subassemblies of the bit-reading gadget by describing the placement of the blockers in relation to the position of the green tile. We will discuss how to create tile sets which can create the necessary sets of paths for the gadgets, and then the final tile set will simply consist of a union of those tile sets. Suppose that the 0-blocker is a $\vec{x_1}$-shifted polyomino and the 1-blocker is a $\vec{x_2}$-shifted polyomino. (Recall that P is a basic polyomino, and thus it is possible to build a path such that a blocker can be at any shift relative to the grid.) We construct two separate systems, say $\mathcal{T}_{0B}$ and $\mathcal{T}_{1B}$ as described in Lemma 6 so that the 0-blocker and 1-blocker, respectively, are the northernmost tiles on the western edges of the assemblies (shown in part (a) and (b) of Figure 3.14). We denote the assemblies produced by these systems as α_0 and α_1, respectively. Next, extend the tile sets of the systems if needed so that the last tiles placed lie on grid in the same grid row as the seed as shown in part (c) and (d) of Figure 3.14. In ad-

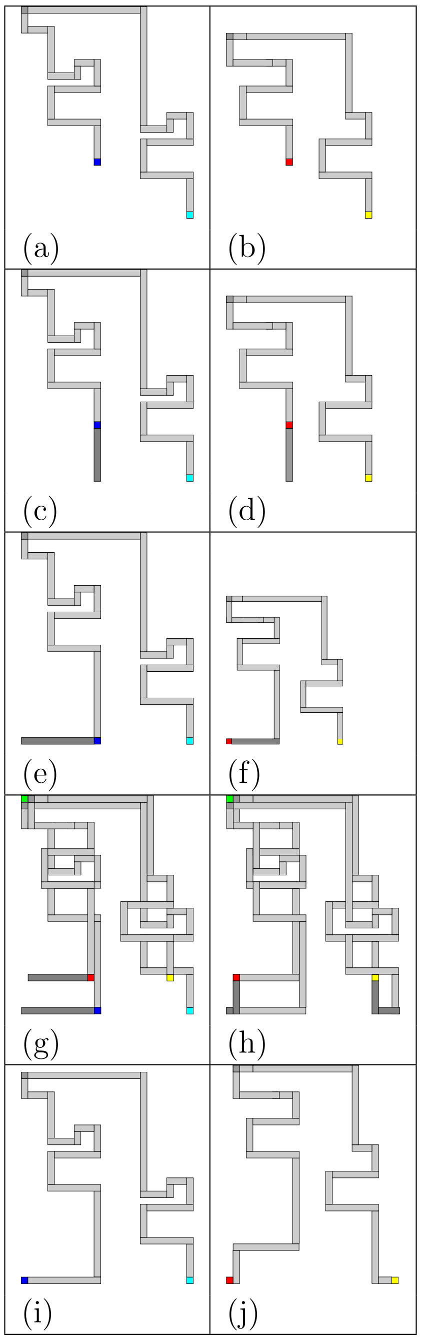
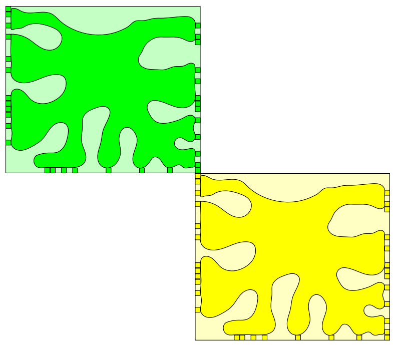


Figure 3.14: A schematic representation of the construction of the bit writers. The blockers are represented by dark grey squares in the northwest of the assemblies. The assembly which places the 0-blocker begins growth from the blue tile and the assembly which places the 1-blocker begins growth from the yellow tile. The last tile placed in the assembly that contains the 0-blocker is shown in blue and the last tile placed by the assembly containing the 1-blocker is shown in red. Paths of new tiles that are added at each step are dark grey.

dition, extend the tile sets of the two systems (if needed) so that the last tile placed has pixels that lie in the same column as the westernmost pixel in the blocker or to the west of that column. This is shown schematically in part (e) and (f) of the figure. Now, place the green polyomino so that its bounding rectangle's southwest corner lies at the origin, and place the assemblies α_0 and α_1 constructed above so that the 1-blocker and 0-blocker lie relative to the green tile as described above (shown in part (g) of Figure 3.14). Without loss of generality suppose that the seed of α_1 lies to the southeast of the seed of α_0 (as is the case in the figure). Then we can extend the tile set of $\mathcal{T}_{0B}$ so that whenever α_1 is placed as described above, the seeds of α_0 and α_1 lie at the same position, since both paths are on grid in those locations. In addition, without loss of generality suppose that the tile placed last in α_1 is further west than the last tile placed in α_0. Then we extend the tile set of $\mathcal{T}_{0B}$ so that the last tile placed in α_0 is at the same position as the last tile placed in α_1. These two steps are shown in part (h) of the figure. The construction of the bit writer gadgets is now complete and the schematic diagram of the completed bit writers is shown in parts (i) and (j) of Figure 3.14.

Case (3) Bit-Reader Construction


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

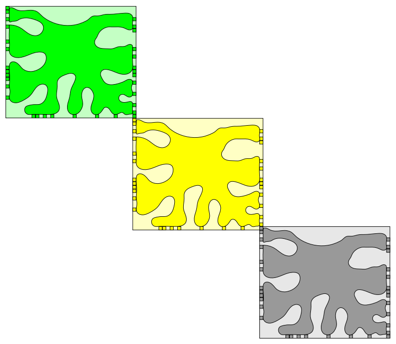
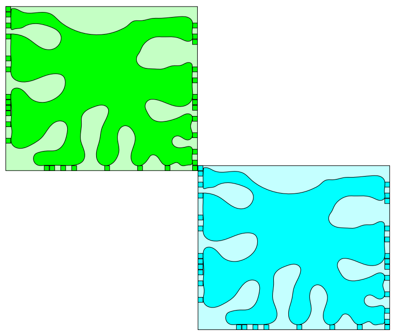

(b) The 0-blocker is placed so that the easternmost pixel on the north perimeter of the 0-blocker overlaps the westernmost pixel on the south perimeter of the yellow tile.

Figure 3.15: Placement of the 0-blocker, which blocks the yellow (i.e. 0-reader) path.

Figures 3.15 and 3.16 show the placement of the 0-blocker and 1-blocker, respectively. Figure 3.17 shows how the glues are placed on the first and second tiles in the yellow path


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

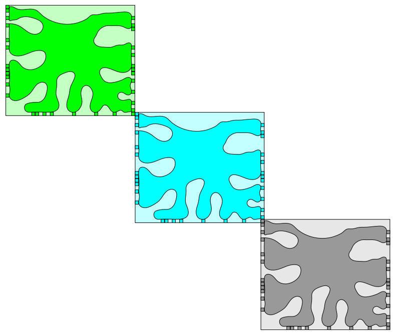

(b) The 1-blocker is placed so that the northernmost pixel on the western perimeter of the 1-blocker overlaps the southernmost pixel on the east perimeter of the blue tile.

Figure 3.16: Placement of the 1-blocker, which blocks the blue (i.e. 1-reader) path.

(in the figure the second yellow tile is shown as an orange tile for clarity) so that the second yellow tile binds to the first yellow tile in the system. In part (a) of Figure 3.17, an orange tile (representing the second tile to attach in the yellow path) is placed so that it now lays directly on top of the yellow tile. The d_y pixels which lie in the column with the most pixels are shown as a red column in part (b) of the figure. Notice that when the orange tile is translated by the vector $(1, 0)$ the m red pixels on the yellow tile now lay adjacent to the m red pixels on the orange tile (see part (c)). Now, we shift the orange tile by the vector $(0, 2)$ and make two observations: (1) the bounding rectangle of the orange tile now no longer overlaps the bounding rectangle of the grey tile, (2) the orange tile has a pixel which lies adjacent to a pixel in the yellow tile and/or a pixel which overlaps a pixel in the yellow tile as shown in part (d) of the figure. In the case that the orange tile contains pixels which overlap pixels in the yellow tile, we translate the orange tile to the north until no pixels overlap, but pixels lie adjacent to each in the two tile (shown in part (e) of the figure).

We now have a configuration as shown in part (f) of Figure 3.17 in which there are not any overlapping pixels and the yellow and orange tiles have pixels which lie adjacent to each other. We can now place glues on the green, yellow and orange tiles so that they assemble as shown with the yellow tile attaching to the green tile and the orange tile attaching to the yellow tile.

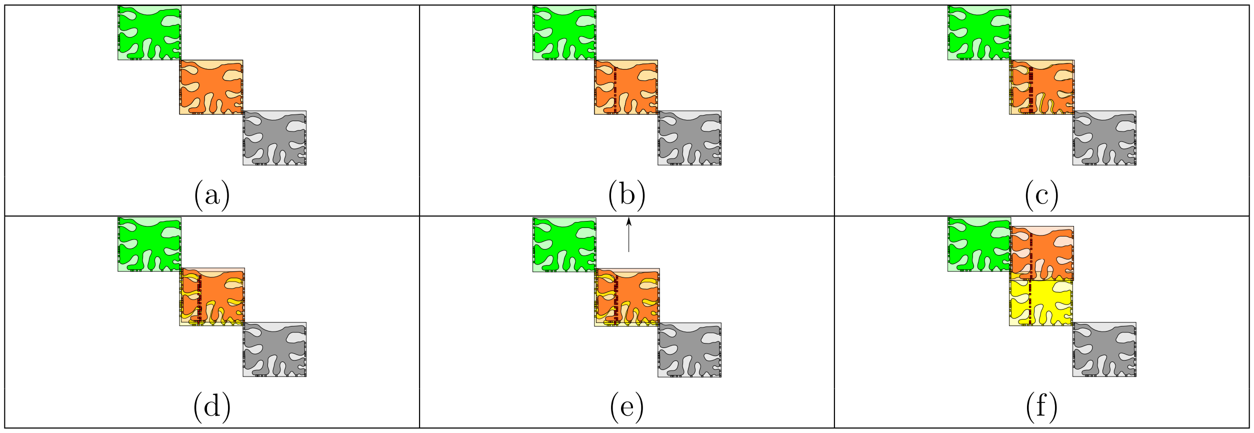
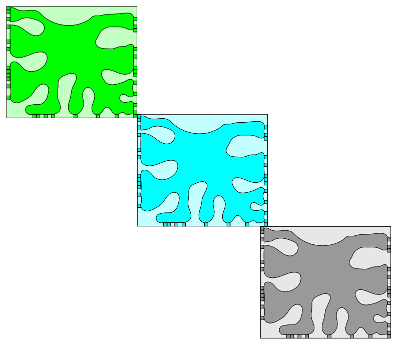


Figure 3.17: The steps involved in placing the grey blocker and yellow tiles so that an blue tile is prevented from binding to the green tile, but allows for a yellow tile to bind to the green and then continue growth of the yellow path on the grid. The orange tile represents the second tile of the yellow path (to make it easier to distinguish), and figures (a) through (e) show how it can be initially placed immediately on top of the first yellow tile, and then moved into a position which allows for correct binding.

We now describe how glues are placed on the first and second tiles to assemble in the path of blue-colored tiles. Figure 3.18a shows how the 0-blocker lies in relation to the blue and green tiles. Notice that a tile can attach to the north of the blue tile without overlapping any pixels on other tiles. Thus, the second tile to attach in the blue path is placed to the north of the first blue tile in the path such that it is on the grid. This is shown in Figure 3.18b where we use a purple tile to represent the second tile in the blue path for clarity. Consequently, we place glues on the first and second tiles to attach in the blue path in a manner such that the second tile in the path binds on grid with respect to the first tile.

Case (3) Right-to-Left Bit-Reading Gadget Construction

As the above sections describe how to build the left-to-right bit-reading gadget, we now construct the right-to-left bit-reading gadget by using mirrored versions of the arguments given above with a few small changes. In the left-to-right bit-reading gadget we can always place the yellow tile so that it is a $(-1,-1)$-shifted polyomino. Notice that this is not


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

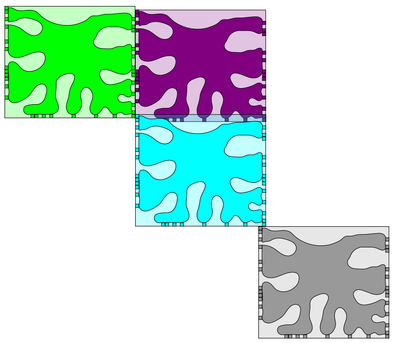

(b) The 0-blocker is placed so that the westernmost pixel on the north perimeter of the 0-blocker overlaps the easternmost pixel on the south perimeter of the yellow tile.

Figure 3.18: Placement of the second tile in the blue path (shown as purple to make it easier to distinguish).

the case when the bit reader is growing to the west. Thus we make the following changes to the argument above when constructing the right-to-left bit-reading gadget. For convenience, we call the first tile to attach in the blue path t_1 and the first tile to attach in the yellow path t_0. To begin, we attach t_1 to the green tile so that the northernmost pixel on the east perimeter of t_1 attaches to the southernmost pixel on the western perimeter of the green tile via their east/west glues. Say that this places the blue tile so that it is an (x_1, x_2)-shifted polyomino. Note that this means t_1 is not necessarily on grid since as noted above the grid we are using is formed by attaching the southernmost pixel on the east perimeter of P to the northernmost pixel on the western perimeter of P. Now, observe that this implies that we can also attach a $(x_1 + 1, x_2 - 1)$-shifted tile to the green tile (by the points that we used for their attachment at (x_1, x_2)). We thus construct glues so that t_0 attaches to the green tile such that it is a $(x_1 + 1, x_2 - 1)$-shifted polyomino. Now, we can construct the bit-writers as in Section 3.4.3 with the blockers shifted in the following ways: (1) the 1-blocker is shifted so that when it is placed its northernmost pixel on the east perimeter overlaps the southernmost pixel on the western perimeter of t_1, and (2) the 0-blocker is placed so that its easternmost pixel on its north perimeter overlaps the westernmost pixel on the south perimeter of t_0. We can then use the mirrored version of the construction in section 3.4.3 to grow the rest of the path of tiles composing the yellow and blue paths.

Case (3) Correctness of the Bit-Reading Gadget

Let us now examine what our constructed system will assemble. Growth will start with the seed and then grow two bit-writer subassemblies consecutively. For concreteness, suppose that α_1 is grown first and then α_0. After α_0 is assembled, a path of tiles will grow upward and over to place a green tile such that the green tile will be placed with its position relative to the grey tile as shown in part (a) of Figure 3.17. It then follows by the way we placed the green and yellow tiles and the location of α_0 that the yellow path will be able to assemble. This will eventually lead to the placement of the second green tile, which is placed to the west of the second bit writer. The relative placements of that green tile and the blocker of α_1 ensure that a blue path, and only a blue path, will assemble. This concludes the necessary demonstration of the correct growth of a bit-reader gadget. (The full Turing machine simulation also includes bit-writers, designed as previously described, to output between bit-readers and the necessary zig-zag paths.)

3.4.4 Case (4): P is non-basic

For this case, suppose that P is a non-basic polyomino. Note that we are not making the claim that non-basic polyominoes exist; in fact we conjecture that they do not. However, a proof seems at least as complicated as handling this case with an explicit construction. Details are omitted because of limited space; for details, see the full version of this paper [29].

3.5 Multiple Polyomino Systems that are Computationally Universal

In previous sections we have focussed on the computational power of systems consisting of singly shaped polyominoes and showed that single polyomino systems are universal for polyominoes of size $\geq$ 3, while monomino and domino systems are likely not capable of such computation. We now show that any multiple shape polyomino system (i.e. one that utilizes at least 2 distinct polyomino shapes, regardless of their size) is capable of universal computation. The following lemma was originally proven in [39] and follows from Theorem 1 present there.

Lemma 7. For every standard Turing Machine M and input w, there exists a TAS with $\tau = 1$ consisting only of tiles shaped as either a monomino or a domino that simulates M on w.

Proof. To see this we provide a sketch of a right-to-left bit reading gadget in Figure 3.20. A left-to-right gadget can be derived similarly, and together these gadgets can be used to construct a zig-zag Turing machine simulation in the same manner as single shaped poly-TAM systems. $\square$

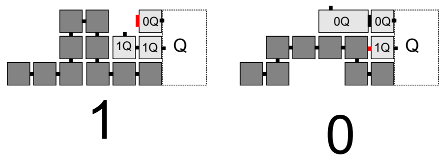


Figure 3.19: A single bit example of how the assembly is able to read geometry to gain information about a north glue and still retain information about the west glue. We use the following conventions. The small black rectangles represent glues which allow singletons to bind. The longer black rectangles represent glues that can potentially bind to a duple (note that these glues are the same types of glues as the others, just drawn differently for extra clarity). The red rectangles represent glues that have mismatched.

Lemma 8. For every standard Turing Machine M and input w, there exists a TAS with $\tau = 1$ consisting only of tiles shaped as dominoes, 2×1 and 1×2 polyominoes, that simulates M on w.

Proof. To see this we provide a sketch of a right-to-left bit reading gadget in Figure 3.20. A left-to-right gadget can be derived similarly, and together these gadgets can be used to construct a zig-zag Turing machine simulation in the same manner as single shaped poly-TAM systems. $\square$

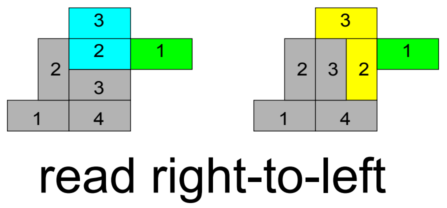


Figure 3.20: A 2-shape system consisting of the two distinct domino polyominoes can be designed to read bits for the simulation of a zig-zag Turing machine.

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

Proof. If either P or Q are size 3 or larger, we get the result from theorem 1. If one polyomino is a monomino and the other a domino, we get the result from Lemma 7, and if P and Q are the two distinct domino shapes, then we get the result by Lemma 8. □

Chapter 4

A Model of Self-assembly that is not Computationally Universal

4.1 Introduction

In the previous chapter we explored non-cooperative self-assembling systems in the polyTAM and showed how to use geometric bit-reading to simulate a Turing machine. We also showed that there are certain systems (for example aTAM systems) for which there does not exist a bit-reading gadget. While this shows that these systems cannot simulate a Turing machine in a "traditional" sense, it does not show that these systems cannot compute. Next we introduce a model where we can show more definitively that computation is simply not possible in this model. We introduce the *Reflexive Tile Assembly Model* (RTAM), which can be thought of as the aTAM with the exception that tiles in the RTAM are allowed to flip horizontally and/or vertically. Unlike the aTAM, where complementary strands of DNA are given the same glue label, the use of complementary glues plays an important role in the RTAM in that they prevent a tile from being able to flip and bind to itself. We then show a series of results within the RTAM. First we show that at temperature 1, the class of directed RTAM systems – systems that yield a single pattern upto rotation and ignoring tile orientation – is not computationally universal. We also show that like the aTAM at temperature 2, the class of directed RTAM systems at temperature 2 is computationally universal. This shows that there is a fundamental dividing line between temperature 1 and temperature 2 directed systems in the RTAM; whereas such a dividing line in the aTAM is still a long-standing open problem. In order to further compare the RTAM with the aTAM, we then turn our attention to the self-assembly of finite shapes in the RTAM. First, we consider the assembly of squares as this is a common benchmark in self-assembly for measuring the strength of a model. We show that for n even, it is impossible to self-assemble any $n \times n$ square, and for n odd, any $n \times n$ square can be self-assembled using only n tile types, but cannot be self-assembled using less than n tile types. This is in contrast to the aTAM at temperature 1, where the conjectured lower bound for assembling an $n \times n$ square is $2n-1$. Finally, we give a series of results that about which shapes finite shapes self-assemble from a single seed in the RTAM including a complete classification of which finite shapes self-assemble from a single seed tile at temperature 1.

4.2 Preliminaries

Definition of the Reflexive Tile Assembly Model

As in the aTAM, we work in the 2-dimensional discrete space $\mathbb{Z}^2$. Define the set $U_2 = \{(0,1),(1,0),(0,-1),(-1,0)\}$ to be the set of all *unit vectors* in $\mathbb{Z}^2$. Intuitively, a tile type t is a unit square that can be translated and flipped across its vertical and/or horizontal axes, but not rotated. This provides each tile type with a pair of North-South (NS) sides and a pair of East-West (EW) sides, such that either side $s \in NS$ may be facing north while the other is facing south (and vice versa for the EW glues). For ease of discussion, however, we will talk about tile types as being defined in fixed orientations, but then allow them to attach to assemblies in possibly flipped orientations. Therefore, we define each t as having a well-defined "side $\vec{u}$" for each $\vec{u} \in U_2$. Each side $\vec{u}$ of t has a "glue" with "label" $\text{label}_t(\vec{u})$–a string over some fixed alphabet–and "strength" $\text{str}_t(\vec{u})$–a nonnegative integer–specified by its type t. Let $R = \{D, V, H, B\}$ be the set of permissible reflections for a tile which is assumed to begin in the default orientation, where D corresponds to no change from the default, V a single vertical flip (i.e. a reflection across the x-axis), H a single horizontal flip (i.e. a reflection across the y-axis), and B a single horizontal flip and a single vertical flip. (Note that the ordering of flips for B does not matter as either ordering results in the same orientation, and also that all combinations of possibly many flips across each axis result only in tiles of the orientations provided by R.) See Figure 4.1 for an example of each. Let $S : R \times U_2 \to U_2$ be a function which takes a type of reflection $r \in R$ and a side $s \in U_2$, and which returns the side of a tile in its default orientation which would appear on side s of the tile when it has been reflected according to r. (E.g. for the tile type shown in Figure 4.1, $S(H, W) = E$, and $S(H, N) = N$.) It is important to note that a glue does not have any particular orientation along the edge on which it resides, and so remains unchanged throughout reflections.

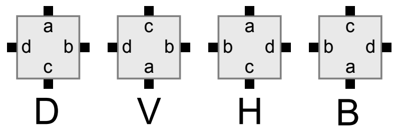


Figure 4.1: Left to right: (1) Default orientation of an example tile type t, (2) t flipped vertically, (3) t flipped horizontally, (4) t flipped across both axes

Two tiles t and t' that are placed at the points $\vec{a}$ and $\vec{a}+\vec{u}$ and reflected by $r \in R$ and $r' \in R$, respectively, *bind* with *strength* $\text{str}_t\left(S(r,\vec{u})\right)$ if and only if

$$\left(\text{label}_t\left(S(r,\vec{u})\right), \text{str}_t\left(S(r,\vec{u})\right)\right) = \left(\overline{\text{label}_{t'}\left(S(r',-\vec{u})\right)}, \text{str}_{t'}\left(S(r',-\vec{u})\right)\right).$$

That is, the glues on adjacent edges of two tiles bind iff they have complementary labels and the same strength.

In the subsequent definitions, given two partial functions f, g, we write $f(x) = g(x)$ if f and g are both defined and equal on x, or if f and g are both undefined on x.

Fix a finite set T of tile types. A T-*assembly*, sometimes denoted simply as an *assembly* when T is clear from the context, is a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T \times R$ defined on at least one input, with points $\vec{x} \in \mathbb{Z}^2$ at which $\alpha(\vec{x})$ is undefined interpreted to be empty space, so that dom α is the set of points with *oriented* tiles. We write $|\alpha|$ to denote $|\text{dom } \alpha|$, and we say α is *finite* if $|\alpha|$ is finite. For a given location $\vec{v} \in \mathbb{Z}^2$, we denote the tile in α at location $\vec{v}$ by $\alpha(\vec{v})$ (if no tile exists there, $\alpha(\vec{v})$ is undefined). Let F be a function which takes as input an assembly α, a reflection $r \in R$, and a translation vector $\vec{v} \in \mathbb{Z}^2$, and which returns the assembly α^r, corresponding to α reflected according to r then translated by $\vec{v}$. We say that two assemblies, α and β, are equivalent iff there exists some reflection $r \in R$ and translation vector $\vec{v} \in \mathbb{Z}^2$ such that for $\beta' = F(\beta, r, \vec{v})$, $|\alpha| = |\beta'|$ and for all $\vec{v} \in |\alpha|$, $\alpha(\vec{v}) = \beta'(\vec{v})$. That is, α and β are equivalent iff one of them can be flipped and translated so that they perfectly match at all locations. For assemblies α and α', we say that α is a *subassembly* of α', and write $\alpha \sqsubseteq \alpha'$, if dom $\alpha \subseteq$ dom α' and $\alpha(\vec{x}) = \alpha'(\vec{x})$ for all $x \in$ dom α. An assembly is *τ-stable* if it cannot be broken up into smaller assemblies without breaking bonds of total strength at least τ, for some $\tau \in \mathbb{N}$. For a tile set T, we let $p_T : T \times R \to T$ be the projection map onto T (i.e. $p_T((t,r)) = t$). A *configuration* given by an assembly α is defined to be the map from $\mathbb{Z}^2$ to T given by $p_T \circ \alpha$.

Self-assembly begins with a *seed assembly* σ, in which each tile has a specified and fixed orientation, and proceeds asynchronously and nondeterministically, with tiles in any valid reflection in R adsorbing one at a time to the existing assembly in any manner that preserves τ-stability at all times. A *tile assembly system* (*TAS*) is an ordered triple $\mathcal{T} = (T, \sigma, \tau)$, where T is a finite set of tile types, σ is a seed assembly with finite domain in which each tile is given a fixed orientation, and $\tau \in \mathbb{N}$. A *generalized tile assembly system* (*GTAS*) is defined similarly, but without the finiteness requirements. We write $\mathcal{A}[\mathcal{T}]$ for the set of all assemblies that can arise (in finitely many steps or in the limit) from $\mathcal{T}$. An assembly $\alpha \in \mathcal{A}[\mathcal{T}]$ is *terminal*, and we write $\alpha \in \mathcal{A}_\Box[\mathcal{T}]$, if no tile can be τ-stably added to it. It is clear that $\mathcal{A}_\Box[\mathcal{T}] \subseteq \mathcal{A}[\mathcal{T}]$.

An assembly sequence in a TAS $\mathcal{T}$ is a (finite or infinite) sequence $\vec{\alpha} = (\alpha_0, \alpha_1, \ldots)$ of assemblies in which each α_{i+1} is obtained from α_i by the addition of a single tile. The *result* $\mathrm{res}(\vec{\alpha})$ of such an assembly sequence is its unique limiting assembly. (This is the last assembly in the sequence if the sequence is finite.) The set $\mathcal{A}[\mathcal{T}]$ is partially ordered by the relation $\longrightarrow$ defined by

$$\begin{aligned} \alpha \longrightarrow \alpha' \quad \text{iff} \quad & \text{there is an assembly sequence } \vec{\alpha} = (\alpha_0, \alpha_1, \ldots) \\ & \text{such that } \alpha_0 = \alpha \text{ and } \alpha' = \mathrm{res}(\vec{\alpha}). \end{aligned}$$

We say that $\mathcal{T}$ is *strongly directed* if and only if either $|\mathcal{A}_\Box[\mathcal{T}]| = 1$ or if for every pair of terminal assemblies $\alpha, \beta \in \mathcal{A}_\Box[\mathcal{T}]$, there exists a reflection $r \in R$ and a translation vector $\vec{v} \in \mathbb{Z}^2$ such that $\alpha = F(\beta, r, \vec{v})$. Furthermore, we say that $\mathcal{T}$ is *directed* if and only if for all $\alpha, \beta \in \mathcal{T}$, there exists a reflection $r \in R$ and a translation vector $\vec{v} \in \mathbb{Z}^2$ such that

$p_T \circ \alpha = p_T \circ F(\beta, r, \vec{v})$. In other words, all of the assemblies of a directed systems give the same configuration.

A set $X \subseteq \mathbb{Z}^2$ *weakly self-assembles* if there exists a TAS $\mathcal{T} = (T, \sigma, \tau)$ and a set $B \subseteq T$ such that for each $\alpha \in \mathcal{A}_\square[\mathcal{T}]$ there exists a reflection $r \in R$ and a translation $\vec{v} \in \mathbb{Z}^2$ such that $\alpha_r = F(\alpha, r, \vec{v})$ and $\alpha_r^{-1}(B) = X$ holds. Essentially, weak self-assembly can be thought of as the creation (or "painting") of a pattern of tiles from B (usually taken to be a unique "color" such as black) on a possibly larger "canvas" of un-colored tiles.

A set X *strictly self-assembles* if there is a TAS $\mathcal{T}$ such that for each assembly $\alpha \in \mathcal{A}_\square[\mathcal{T}]$ there exists a reflection $r \in R$ and a translation $\vec{v} \in \mathbb{Z}^2$ such that $\alpha_r = F(\alpha, r, \vec{v})$ and dom $\alpha_r = X$. Essentially, strict self-assembly means that tiles are only placed in positions defined by the shape. Note that if X strictly self-assembles, then X weakly self-assembles. (Let all tiles be in B.)

4.2.1 Paths in the Binding Graph and as Assemblies

Given an assembly α and locations $\vec{x}$ and $\vec{y}$ such that $\vec{x}$,$\vec{y} \in$ dom α, we define a *path in α from $\vec{x}$ to $\vec{y}$* (or simply a *path from $\vec{x}$ to $\vec{y}$*) as a simple directed path in the binding graph of α with the first location being $\vec{x}$ and the last $\vec{y}$. We refer to such a path as π_x^y, and for $k = |\pi_x^y|$ (i.e. k is the length of, or number of tiles on, π_x^y) and $0 \leq i < k$, let $\pi_x^y(i)$ be the ith location of π_x^y. Thus, $\pi_x^y(0) = \vec{x}$, and $\pi_x^y(k-1) = \vec{y}$. We can thus refer to the ith tile on π_x^y and its reflection as $\alpha(\pi_x^y(i))$, and as shorthand will often refer to locations and/or tiles along a path. Regardless of the order in which the tiles of π_x^y were placed in α, we define *input* and *output* sides for each tile in π_x^y (except for the first and last, respectively) in relation to their position on π_x^y. The input side of the ith tile of π_x^y, $\alpha(\pi_x^y(i))$, is that which binds to $\alpha(\pi_x^y(i-1))$, and the output side is that which binds to $\alpha(\pi_x^y(i+1))$. We denote these sides as $IN(\pi_x^y(i))$ and $OUT(\pi_x^y(i))$, respectively. (Thus, $\alpha(\pi_x^y(0))$ has no input side, and $\alpha(\pi_x^y(k-1))$ has no output side.) Note that in a $\tau = 1$ system, an assembly α' exactly representing π_x^y would be able to grow solely from $\alpha(\pi_x^y(0))$, in the order of π_x^y, with each tile having input and output sides as defined for π_x^y.

4.3 Computation in the RTAM

4.3.1 The RTAM is Not Computationally Universal at $\tau = 1$

In this section, we show that directed RTAM systems are not computationally universal by showing that any shape weakly assembled by a directed RTAM system is "simple". We will first define our notion of simple. Many of the definitions used can also be found in [26].

Definition. A set $X \subseteq \mathbb{Z}^2$ is *semi-doubly periodic* if there exist three vectors $\vec{b}$, $\vec{u}$, and $\vec{v}$ in $\mathbb{Z}^2$ such that

$$X = \left\{ \left. \vec{b} + n \cdot \vec{u} + m \cdot \vec{v} \;\right|\; n, m \in \mathbb{N} \right\}.$$

Less formally, a semi-doubly periodic set is a set that repeats infinitely along two vectors (linearly independent vectors in the non-degenerate case), starting at some base point $\vec{b}$. Now, let $\mathcal{T} = (T, \sigma, 1)$ refer to a directed, temperature-1 RTAM system. We show that any such $\mathcal{T}$ weakly self-assembles a set $X \subseteq \mathbb{Z}^2$ that is a finite union of semi-doubly periodic sets.

Theorem 3. Let $\mathcal{T} = (T, \sigma, 1)$ be a directed RTAM system. If a set $X \subseteq \mathbb{Z}^2$ weakly self-assembles in $\mathcal{T}$, then X is a finite union of semi-doubly periodic sets.

Proof. (sketch) Here we give a high-level sketch of the proof of Theorem 3. The basic idea of the proof is as follows. For an RTAM system $\mathcal{T} = (T, \sigma, 1)$ we consider all of the paths of n tiles (for n to be defined) that can assemble from each exposed glue of σ such that each consecutive tile that binds forming the path attach via a north or west glue (and we say that such a path "extends to the north-west").

Any finite path is trivially the union of semi-doubly periodic sets. Then, for n sufficiently large, a path of n tiles that extends to the north-west must contain two distinct tiles t_1 and t_2 of the same tile type in the same orientation. If for every such path, every two distinct tiles t_1 and t_2 of the same tile type in the same orientation lie on a horizontal or vertical line, then it can be argued that the terminal configuration of $\mathcal{T}$ must consist of

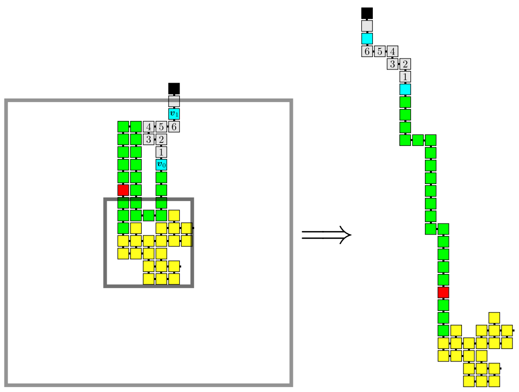


Figure 4.2: A depiction of tiles for a path that can assemble in $\mathcal{T}$. The original path is on the left. The path on the right is a modification to the path on the left that must also be able to assemble in $\mathcal{T}$. The blue tiles labeled $\vec{v}_0$ and $\vec{v}_1$ are of the same tile type and orientation. The tiles labeled 1 through 6 can be repeated indefinitely as depicted in Table 4.1(a).

finitely many infinitely long horizontal or vertical paths connected to σ, and is therefore the finite union of semi-doubly periodic sets. On the other hand, if there is a path such that the two distinct tiles t_1 and t_2 of the same tile type in the same orientation do not lie on a horizontal or vertical path, then we argue that the terminal assembly of $\mathcal{T}$ is the finite union of semi-doubly periodic sets as follows. First, we note that for such a path, π say, the tiles between t_1 and t_2 can be repeated indefinitely. This is shown in Table 4.1(a). Then we show how to modify π by reflecting tiles to obtain an infinite family of paths. Ex-

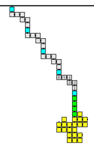
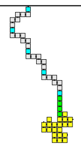
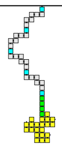
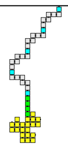
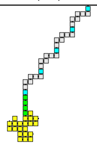
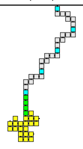
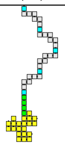
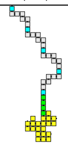
 (a)	 (b)	 (c)	 (d)
 (e)	 (f)	 (g)	 (h)

Table 4.1: Each figure in this table depicts a possible path that can assemble in $\mathcal{T}$. The yellow tiles make up the seed and the green tiles are a path leading to the repeated tile that allows the path to repeat.

amples of such modified paths are shown in Table 4.1(b)-(h). Now, if a tile t belongs to one of these paths and has location l say, then, since $\mathcal{T}$ is directed the terminal assembly of $\mathcal{T}$ must contain a tile of the same type as t at each such location l. Finally, we note that

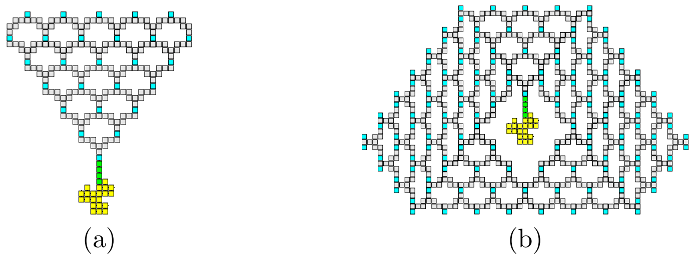


Figure 4.3: (a) A configuration that can be thought of as the "union" of the type-consistent assemblies depicted in Table 4.1. (b) A configuration of tiles that must weakly self-assemble in $\mathcal{T}$.

all of these paths taken together form a semi-doubly periodic set. This is depicted in Figure 4.3a. Continuing this line of reasoning, we show that the terminal assembly of $\mathcal{T}$ is the finite union of semi-doubly periodic sets. A portion of such an assembly is shown in Figure 4.3b.

$\square$

An intuitive reason that Theorem 3 supports the conclusion that the RTAM is not computationally universal is as follows. Let

$$H = \{(i, x) \mid \text{program } i \text{ halts when run on input } x\}$$

be the halting set and let M be a Turing machine that outputs a 1 if $(i, x) \in H$. The typical way of expressing the computation of M in tile assembly is as follows. For a fixed tileset T, a seed assembly σ encodes an "input" to the computation, while the "output" of 1 by M corresponds to the translation of some configuration being contained in the terminal assembly α_σ of $(T, \sigma, 1)$. For this sense of computation, the following corollary says that the set of seed assemblies that "output" a 1 is a recursive set (not just a recursively enumerable set). This would contradict the fact that the halting set is not recursive. This is stated in the following corollary. For more details see [36].

Corollary 1. For any RTAM tileset T and fixed finite configuration C, let S be the set of seed assemblies σ such that (1) the RTAM system $(T, \sigma, 1)$ is directed and (2) the terminal assembly of $\mathcal{T}$ contains C. Then, S is a recursive set.

4.3.2 Universal computation at $\tau = 2$

In this section give a theorem that states that universal computation is possible in the RTAM at temperature 2. We give an example of simulating a binary counter in the RTAM and give the general proof in [36].

Theorem 4. The RTAM is computationally universal at $\tau = 2$. Moreover, the class of directed RTAM systems is computationally universal at $\tau = 2$.

First we note that given a Turing machine M, we use Lemma 7 of [12] to obtain a tile set which simulates M using a *zig-zag system*. In fact, as noted in [52], we can find a singly seeded *compact zig-zag system* $\mathcal{T} = (T, \sigma, 2)$ with $\mathcal{A}_{\Box}[\mathcal{T}] = \{\alpha\}$ which simulates M. Then the proof of Theorem 4 relies on showing that any compact zig-zag system in the aTAM at temperature 2 can be converted into a directed RTAM system $\mathcal{S}$ that is "almost" compact zig-zag. The RTAM system constructed differs from a compact zig-zag system in that when the length of a row of the growing assembly increases by a tile, a strength-2 glue is exposed allowing a tile to bind below the row, resulting in the possibility of a single "misplaced" tile per row, however, this suffices to simulate a Turing machine.

4.4 Self-assembly of shapes in the RTAM at $\tau = 1$

In this section, we discuss the self-assembly of shapes in the RTAM, especially the commonly used benchmark of squares. At temperature 2, using a zig-zag binary counter similar to that used in Section 4.3.2, $n \times n$ squares can be built using the optimal $\log n / (\log \log n)$ tile types following the construction of [59] with only trivial modifications. Similarly, the majority of shapes which can be weakly self-assembled in the temperature-2 aTAM can be built in the temperature-2 RTAM, although shapes with single-tile-wide branches which are not symmetric are impossible to strictly self-assemble in the RTAM.

At temperature-1, however, the differences between the powers of the aTAM and RTAM appear to increase. Here we will demonstrate that squares whose sides are of even length cannot weakly (or therefore strictly) self-assemble in the RTAM at $\tau = 1$, although any square can strictly self-assemble in the $\tau = 1$ aTAM. We then prove a tight bound of n tile types required to self-assemble an $n \times n$ square for odd n in the RTAM at $\tau = 1$. (Which is, interestingly, better than the conjectured lower bound of $2n - 1$ for the $\tau = 1$ aTAM.)

4.4.1 For even n, no $n \times n$ square self-assembles in the $\tau = 1$ RTAM

Theorem 5. For all $n \in \mathbb{Z}^+$ where n is even, there exists no RTAM system $\mathcal{T} = (T, \sigma, 1)$ where $|\sigma| = 1$ and $\mathcal{T}$ weakly (or strictly) self-assembles an $n \times n$ square.

Proof. We prove Theorem 5 by contradiction. Therefore, assume that for some $n \in \mathbb{Z}^+$ such that $(n \mod 2) = 0$, there exists an RTAM system $\mathcal{T} = (T, \sigma, 1)$ such that $|\sigma| = 1$ and $\mathcal{T}$ weakly self-assembles an $n \times n$ square S. It must therefore be the case that for some subset of tile types $B \subseteq T$, for all $\alpha \in \mathcal{A}_\Box[\mathcal{T}]$ there exist $r \in R$ and $\vec{v} \in \mathbb{Z}^2$ such that for $\alpha_r = F(\alpha, r, \vec{v})$, $\alpha_r^{-1}(B) = S$ (that is, every terminal assembly contains an $n \times n$ square of "black" tiles, and no "black" tiles anywhere else).

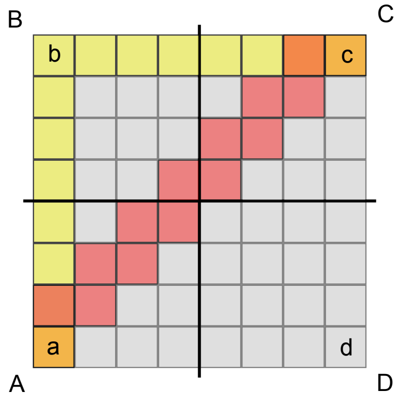


Figure 4.4: The shortest path between two opposite corners of an $n \times n$ square has length $2n - 1$.

We choose an arbitrary $\alpha \in \mathcal{A}_{\Box}[\mathcal{T}]$ and refer to the four corners of S in α as $\vec{a}$, $\vec{b}$, $\vec{c}$, and $\vec{d}$, with $\vec{a}$ being the southwest and moving clockwise to name the remaining corners. (See Figure 4.4 for an example using an 8×8 square.) Since α must be a stable assembly, there must exist a set of (possibly overlapping) paths P in the binding graph of α which contain the tiles at all 4 corners. Let π_a^c be a shortest path which contains both corners $\vec{a}$ and $\vec{c}$ (there may be more than one), and note that $|\pi_a^c| \geq 2n - 1$ because the tiles are located on a grid. (Two possible such paths can be seen in Figure 4.4 as the red and yellow paths.) Let the uppercase letters A, B, C, and D represent the quadrants which contain the corners with the corresponding lowercase labels. It must be the case that π_a^c contains at least one tile in either quadrant B or D, else it could not reach quadrant C. This is due to the facts that n is even and that π_a^c is a path through a grid graph where diagonal travel is not allowed. Without loss of generality, assume that π_a^c enters quadrant B.

We now know that some path π_a^c through the binding graph of α connects $\vec{a}$ and $\vec{c}$ and also contains at least one point in quadrant B. Additionally, we know that within the overall binding graph of α, π_a^c must somehow also be connected to the tile at $\vec{d}$. Let π_d^{ac} be a path in the binding graph which contains both $\vec{d}$ and some tile in π_a^c. Note that this may just be the tile at $\vec{d}$ since π_a^c may contain that location. We now define path π' as the longest of the following two paths formed by possibly combining subpaths of π_a^c and π_d^{ac}: 1) a path which connects $\vec{d}$ and $\vec{c}$ and contains a tile in B, or 2) a path which connects $\vec{a}$ and $\vec{d}$ and contains a tile in quadrant B. For ease of discussion, we will talk about π_a^c as being a directed path from $\vec{a}$ to $\vec{c}$, while noting that we don't know the actual ordering of its growth. Such a π' must exist because the point at which π_d^{ac} intersects π_a^c must be either 1) before it has entered B, which yields case 1 for π', 2) after it has left B, which yields case 2 for π', or 3) within B in which case either holds. Note that π_a^c could perhaps enter and leave B multiple times, which would only strengthen our argument, but for simplicity we will simply make use of the first time it enters quadrant B and the last time it leaves quadrant B for the above argument.

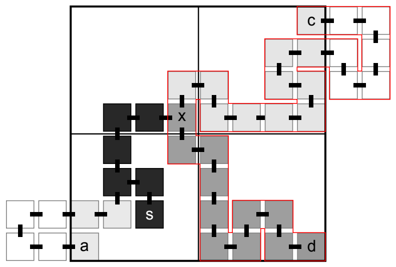


Figure 4.5: Example paths in an 8×8 square (i.e. of even dimension). White tiles represent tile types not in B (and thus tiles not in S), black tiles represent path π'_σ, dark grey tiles represent path π_d^{ac}, and π_a^c is represented by a combination of tiles of all colors. Path π' connects $\vec{d}$ and $\vec{c}$ by passing through $\vec{x}$ (which is in quadrant B) and is outlined in red.

Without loss of generality, we will assume case 1 for path π', and note that we now have found the following path in the binding graph of α. Path π' includes the tile at location $\vec{d}$, contains at least one tile in quadrant B, and also includes the tile at location $\vec{c}$. (See Figure 4.5 for an example.) For a square of dimension n, it must be the case that path π' contains a minimum of n pairs of tiles bound to each other on their east/west edges.

Now we check to see if the seed σ is contained within π'. If not, we define π'_σ as the minimum path in α's binding graph which contains σ and some tile in π', else π'_σ is simply σ. Now we define a new valid assembly sequence, $\vec{\alpha'}$ for $\mathcal{T}$ as follows.

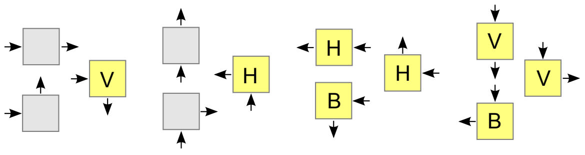


Figure 4.6: Every possible pairing of an input and output side of a tile on a path. Those in grey have as input sides either south or west, and as output sides either north or east. Those in yellow do not, but are labeled with the minimal reflection necessary to orient them so that they do.

Let $\vec{x}$ denote the location of the intersection of π'_σ and π', and π_x^c be the portion π' from $\vec{x}$ to $\vec{c}$ and let π_x^d be the portion of π' from $\vec{x}$ to $\vec{d}$. Without loss of generality, we will assume that the tile on π'_σ which binds to σ attaches to the north of σ. (If this is not the case, then all following directions can simply be rotated by the same angle as the difference of the direction of the first binding from north.) Starting from σ, $\vec{\alpha'}$ attaches one tile at a time from π'_σ so that output glues are exposed on the north or east edge of each tile. In other words, as tiles attach, flip them so that the next tile to attach does so at a tile location that is north or east of the previously attached tile. (Figure 4.6 shows how this must be possible for each tile.) This causes all inputs along π'_σ to be on the south or west sides of tiles, and outputs on the north and east. Furthermore, place the tile from location $\vec{x}$ so that the exposed glue used for the binding of π^c_x is on its the north or west, and the exposed glue used for the binding of π^d_x is on its south or east. Figure 4.7 shows how this can be done. (Note that depending on where and from which direction π'_σ intersects π', it could be the case that the output sides which bind to π^c_x and π^d_x are instead south or east, and north or west, respectively. However, again this doesn't change the argument as the following directions can be rotated appropriately, so we assume the output for π^c_x is on the north or west for ease of discussion.)

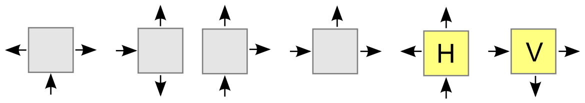


Figure 4.7: Every possible pairing of a south or west input side and 2 output sides of the final tile of π'_σ. Those in grey have one output side on the north or west, and the other on the south or east. Those in yellow do not, but are labeled with the minimal reflection necessary to orient them so that they do.

Now we assemble the path π^c_x as follows. Starting from the tile of location $\vec{x}$, attach one tile at a time so that the output glues are exposed on the north or west edge of each tile. This results in a version of π^c_x which grows strictly up and to the left. Such growth is possible because of the set of reflections possible and the fact that no tile previously placed from π^c_x or π'_σ can block the placement. The fact that blocking can't occur is due to the fact that the stretched version of π^c_x grows strictly up and/or to the left so all previous tiles placed for π^c_x cannot block, and after the first tile placed after the final tile of π'_σ, the newly forming path is either completely above π'_σ or to the left of the topmost tile of that path. Next we assemble the path π^d_x in an analogous manner, but down and to the right. Starting from the tile of location $\vec{x}$, attach one tile at a time so that the output glues are exposed on the south or east edge of each tile. For the same but reflected reasons as for the growth of the modified version of π_x^c, π_x^d can be grown in this way. See Figure 4.8 for an example of these "stretched" paths.

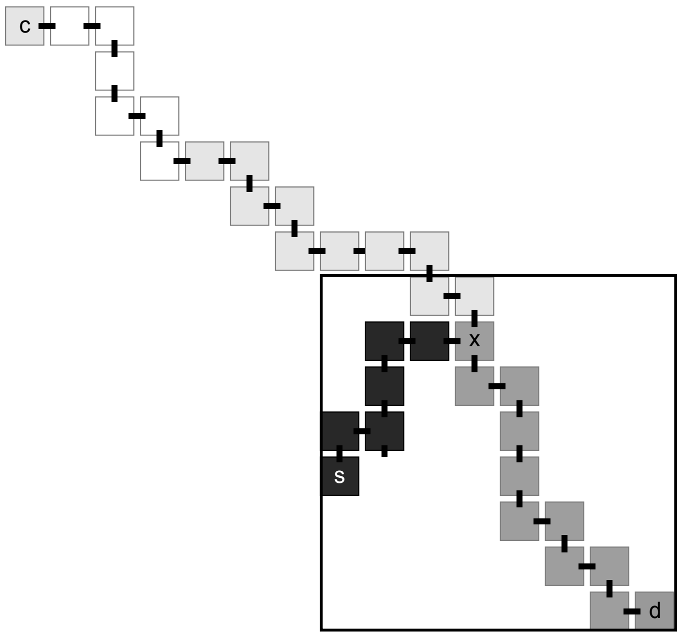


Figure 4.8: Example paths π_σ^x, π_x^c, and π_x^d stretched out when assembled by $\vec{\alpha}$.

The tile types which are at the ends of π_x^c and π_x^d were at locations $\vec{c}$ and $\vec{d}$ in α, so they must be of types which are in $B \subseteq T$ (i.e. "black" tile types). However, since there are $\geq n$ pairs of tiles connected via east/west glues in π' and it is now maximally stretched, it must be $\geq n + 1$ tiles wide (i.e. from leftmost to rightmost tiles). Thus, this is a producible assembly which has "black" tiles at a distance which is further apart than any two points in the square S. Therefore $\vec{\alpha'}$ does not weakly (or, therefore, strictly) self-assemble the $n \times n$ square S and so neither does $\mathcal{T}$. This is a contradiction, and thus $\mathcal{T}$ does not exist. □

4.4.2 Tight bounds on the tile complexity of squares of odd dimension

In this section, we prove tight bounds on the number of tiles necessary to self-assemble a square of odd dimension.

Theorem 6. For all $n \in \mathbb{Z}^+$ where n is odd, an $n \times n$ square strictly self-assembles in an RTAM system $\mathcal{T} = (T, \sigma, 1)$ where $|T| = n$ and $|\sigma| = 1$.

Proof. The general scheme for the construction can be seen in Figure 4.9. Essentially, the seed, which is in the center of the square, presents some glue on both the north and south, and presents some second glue on both the east and west. Above the seed, a series of $(n-1)/2$ hard-coded tile types form a vertical column, with the east and west edges of all tiles exposing the same glue. Using the same tile types, a reflected version of the upper column grows downward to the bottom of the square. To both the east and west of that column, a hard-coded series of another $(n-1)/2$ tile types form reflected rows out to the sides of the square. This requires $1 + 2((n-1)/2) = n$ tile types and strictly self-assembles an $n \times n$ square.

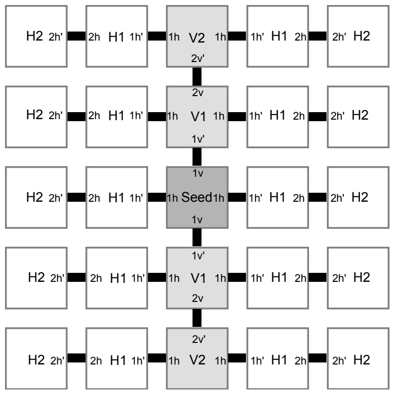


Figure 4.9: A 5×5 square built by an RTAM TAS at $\tau = 1$ using only 5 tile types.

To prove Theorem 6, we provide the following construction which demonstrates how to build an RTAM TAS which strictly self-assembles an $n \times n$ square for any odd n. The basic idea is that we place the seed tile in the middle of the square, and from both its top and bottom it grows copies of a reflected column of tiles which grow to the top and bottom of the square, respectively. This requires one tile type for the seed and $(n-1)/2$ tile types for the column (which grows in both directions). Each of the seed and the tiles of the vertical column have the same glue on both east and west sides. From each of those a row of $(n-1)/2$ unique tile types grows to the left or right boundary of the square. This construction is robust to any valid rotation of the tiles and strictly self-assembles an $n \times n$ square using exactly $1 + 2(n-1)/2 = n$ tile types.

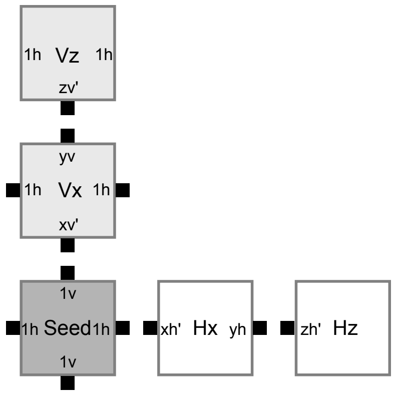


Figure 4.10: The tile type templates for building a square whose sides are odd length in the RTAM at $\tau = 1$ using n unique tile types.

In Figure 4.10 we show the templates used to generate the necessary tile types to self-assemble an $n \times n$ square for odd n, and in Figure 4.9 we show an example of the terminal assembly for $n = 3$. For all odd n, the seed is exactly as in Figure 4.10. Then for each m where $0 < m < (n-1)/2$, we create a unique tile type from the tile type template labeled "Vx". We do this by replacing each "x" (i.e. those of the tile label and the south glue) with the value "m", and the "y" of the north glue with the value "$m + 1$". (Note the "prime" markings of some glues which denote they are complementary to the "un-primed" versions, and for all tile types created from these templates we keep those markings unchanged, which is important in restricting the potential interactions.) In this way, we create the tile types for the central column between the seed and the top or bottom most locations. To make the top/bottom tile type, we start with the "Vz" template and replace the "z" of the label and south glue with the value $(n-1)/2$. To make the tile types for the horizontal rows which grow outward from the central column, for each m where $0 < m < (n-1)/2$ we create a unique tile type from the tile type template labeled "Hx". We do this by replacing each "x" (i.e. those of the tile label and the west glue) with the value "m", and the "y" of the east glue with the value "$m+1$". In this way we create the tile types for the horizontal row between the central column and left or right most locations. To make the left/right tile type, we start with the "Hz" template and replace the "z" of the label and west glue with the value $(n-1)/2$. This completes the creation of exactly n tile types: 1 for the seed, $2(((n-1)/2)-1)$ for the interiors of the column and rows, and 1 each for the top/bottom tile type and left/right tile type. The RTAM TAS $\mathcal{T}$ consisting of this tile set, a seed consisting of a copy of the seed tile in any valid orientation at the origin, and $\tau = 1$ strictly self-assembles an $n \times n$ square and is directed. This is because the only tiles which can attach to the north and south of the seed are properly reflected versions of the "V1" tile type, and then to the north/south of those two tiles properly reflected copies of "V2", etc. until copies of the "V$(n-1/2)$" tile type cap off the north and south growth of the column. Regardless of the reflections of the "Vx" and "Vz" tiles across the vertical axis, the only subassemblies that can grow to the left and right are the arms consisting of the "Hx" and "Hz" tile types. All producible terminal assemblies of $\mathcal{T}$ are equivalent and in the shape of an $n \times n$ square. □

We prove Theorem 6 by giving a scheme for obtaining the tileset for any given n that exploits the fact that for n odd, an $n \times n$ square in $\mathbb{Z}^2$ is symmetric across a row and column points. Although Theorem 6 pertains to squares, a simple modification of the proof shows the following corollary.

Corollary 2. For all $n, m \in \mathbb{Z}^+$ where n is odd, an $n \times m$ square strictly self-assembles in an RTAM system $\mathcal{T} = (T, \sigma, 1)$ where $|T| = \frac{n+m}{2}$ and $|\sigma| = 1$.

We also prove that the upper bound of Theorem 6 is tight, i.e. an $n \times n$ square, where n is odd, cannot be self-assembled using less than n tile types.

Theorem 7. For all $n \in \mathbb{Z}^+$ where n is odd, there exists no RTAM system $\mathcal{T} = (T, \sigma, 1)$ where $|T| < n$ and $|\sigma| = 1$ such that $\mathcal{T}$ weakly (or strictly) self-assembles an $n \times n$ square.

Proof. We prove Theorem 7 by contradiction. Therefore, assume that for some $n \in \mathbb{Z}^+$ such that $(n \mod 2) = 1$, there exists an RTAM system $\mathcal{T} = (T, \sigma, 1)$ such that $|\sigma| = 1$ and $|T| < n$, and $\mathcal{T}$ weakly self-assembles an $n \times n$ square S. It must therefore be the case that for some subset of tile types $B \subseteq T$, for all $\alpha \in \mathcal{A}_{\Box}[\mathcal{T}]$ there exist $r \in R$ and $\vec{v} \in \mathbb{Z}^2$ such that for $\alpha_r = F(\alpha, r, \vec{v})$, $\alpha_r^{-1}(B) = S$.

We choose an arbitrary $\alpha \in \mathcal{A}_{\Box}[\mathcal{T}]$ and, using the notation of Figure 4.4, note that in the binding graph of α, there is a path π_a^c which connects the tiles in locations $\vec{a}$ and $\vec{c}$ (i.e. opposite corners of the square). Furthermore, the tile types of the tiles at locations $\vec{a}$ and $\vec{c}$ must be in $B \subseteq T$, and tiles of types in B can be no further apart from each other (in the plane) than a Manhattan distance of $2n - 1$. Since the seed σ may not be on π_a^c, define the location $\vec{x}$ to be either 1) the location of σ if it is on π_a^c, or 2) the location of the intersection of π_a^c and the shortest path in the binding graph which contains σ and some tile in π_a^c. Note that such a location $\vec{x}$ must exist since α must be connected. Define the path π_σ^x as the path from σ to $\vec{x}$ and note that if σ is on π_a^c then π_σ^x consists of a single tile. Also, we can now define π_x^a and π_x^c as the paths from $\vec{x}$ to $\vec{a}$ and $\vec{c}$, respectively. (See Figure 4.11 for a possible example of such paths in α.)

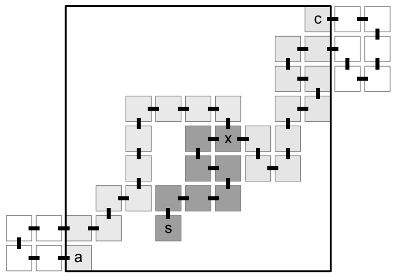


Figure 4.11: Example paths π_σ^x (darkest grey) and π_a^c (a combination of light grey, dark grey, and white tiles) in α which weakly self-assembles a 9×9 square (i.e. of odd dimension). White tiles represent tile types not in $B \subseteq T$, while grey and dark grey tiles represent "black" tile types in B.

We will now define a valid assembly sequence $\vec{\alpha'}$ in $\mathcal{T}$ which builds an assembly α' which consists of modified versions of the paths π_σ^x and π_a^c, and which places tiles of types in $B \subseteq T$ (i.e. "black" tiles) at points further apart than $2n - 1$, proving that $\mathcal{T}$ does not weakly self-assemble S. Let $k = |\pi_\sigma^x|$ and let $\vec{\alpha'}$ start from σ and first build a modified version of π_σ^x in the following way. If $k = 1$, stop. Else, without loss of generality assume that $\alpha(\pi_\sigma^x(1))$ attaches to the north of σ in α. (If it binds on the east, south, or west, for the following argument used to construct α', rotate all directions used clockwise by 90, 180, or 270 degrees respectively.) For each i where $0 < i < k$, place a tile of the same type as at $\alpha(\pi^x_\sigma(i))$ but reflected so that its input side is either south or west, and its output side is either north or east. Note that for any pair of input and output sides, it is possible to reflect a tile to meet this criteria (see Figure 4.6). Place the tile of that type and orientation so that it binds with the $(i-1)$th tile of the newly created path. Note that this also must be possible because, following from σ, every tile is oriented so that the side used as its output side in π^x_σ is north or east and has a glue which matches that of the newly placed tile on its (now) south or west side, respectively, and since the newly forming path grows only up and/or to the right, no previously placed tile can block it from binding. The result of this sequence of placing the tiles from π^x_σ is a "stretched" version of the path π^x_σ which grows only up and/or right from the seed σ, grown by the valid assembly sequence $\vec{\alpha'}$.

We now extend $\vec{\alpha'}$ to grow a stretched version of π^c_a by growing stretched versions of π^a_x and π^c_x. We will build one of π^a_x or π^c_x by stretching it up and to the left, and the other by stretching it down and to the right. (See Figure 4.12 for an example.) Since the final tile of π^x_σ is included in π^c_a, it must be the case that it has two output sides (to bind to each of π^a_x and π^c_x). Also, recall that its input side must have been either south or west. We modify $\vec{\alpha'}$ so that the final tile of π^x_σ is oriented with one output side on either the north or west, and the other on either the south or east. See Figure 4.7 for all possible orientations of this tile. Without loss of generality, assume that in this orientation, the side which serves as input to π^a_x is on either the north or west, and that serving as input to π^c_x is on either the south or east. (If it is the opposite, the directions used to grow the stretched π^a_x and π^c_x can simply be swapped and the argument will still hold.) We will now build a stretched version of π^a_x up and to the left, and π^c_x down and to the right. To do this, we extend $\vec{\alpha'}$ to attach the tiles of π^a_x, moving along that path as it appears in α and placing the same tiles in the same order, but always orienting them so that each has as its input side either its south or east, and as its output side either its north or west, allowing each successive tile to correctly bind to its predecessor on the path. As before, this is possible because of the set of reflections possible and the fact that no tile previously placed from π_x^a or π_σ^x can block the placement. The fact that blocking can't occur is due to the fact that the stretched version of π_x^a grows strictly up and/or to the left so all previous tiles placed for π_x^a cannot block, and after the first tile placed after the final tile of π_σ^x, the newly forming path is either completely above π_σ^x or to the left of the topmost tile of that path. The stretched version of π_x^c is grown analogously, with all input sides being on the north or west and output sides to the south or east.

Note that in the special case where $\vec{x} = \vec{a}$ or $\vec{x} = \vec{c}$ in α, meaning that either the seed is in one of those corners, or the path from the seed to the path π_a^c terminates at one of those corners, we can simplify our construction and simply grow a single direction from $\vec{x}$.

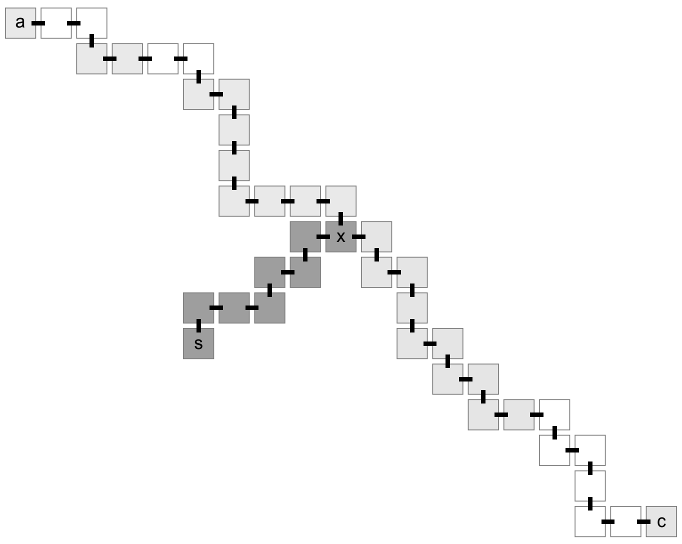


Figure 4.12: Example paths π_σ^x and π_x^c stretched out in $\hat{\pi}_\sigma^c$.

At this point, $\vec{\alpha'}$ grows assembly α' which is a path that looks like that in Figure 4.12 and which starts from the seed and grows a stretched version of π_σ^x, then grows a stretched version of π_a^c (as connected stretched versions of π_x^a and π_x^c) from the end of that path. Since in α, π_a^c connects the bottom-left corner of the square S with the top-right corner, it must be the case that $|\pi_a^c| \geq (2n-1)$. Since $|T| \leq (n-1)$, and $2(n-1) = 2n-2$, by the pigeonhole principle we know that along π_a^c (and therefore also its stretched version), it must be the case that at least 3 tiles of some same type appear.

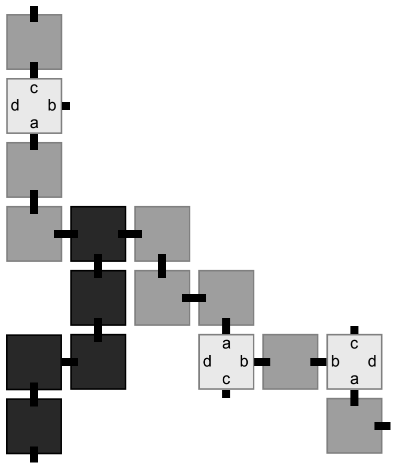


Figure 4.13: One possible configuration of a portion of the stretched paths π_σ^x (black) and π_x^a and π_x^c (grey). Three reflected copies of the same tile type appear on π_x^a and π_x^c (lightest grey). A path using different input sides can be formed between the top and bottom copies of the repeated tile type, entering from the a and b sides, respectively.

With 3 copies of the same tile type appearing along π_a^c, it must be the case that a path can be formed which connects two of the copies in such a way that it enters each of the two tiles along different sides of that tile type when it's in its default configuration. (See Figure 4.13 for an example where a path can be formed between the top copy and the bottom copy, with the top being entered via the a side and the bottom via the b side, while the path between the top and middle copies enters each from the a side, and the path between the middle and bottom copies enters each from the b side.) Let $\vec{f}$, $\vec{g}$, and $\vec{h}$ be the locations of the repeated tile type in α', and let $\vec{f}$ and $\vec{g}$ be the locations which can be connected by a path entering the tiles at those locations from different sides, and let $\vec{f}$ be the furthest of $\vec{f}$ and $\vec{g}$ from the final tile of π_σ^x (or the southernmost if they tie). Without loss of generality, assume that $\vec{f}$ is south of $\vec{g}$ (otherwise, rotate the following directions).

We now modify $\vec{\alpha'}$ so that it builds the stretched copies of π_σ^x, π_x^a, and π_x^c until it is about to place the tile at $\vec{f}$. At this point, it rotates the tile for that position so that the side which serves as input for the tile at $\vec{g}$ is facing either south or east, and places that tile. Then, since the tile at $\vec{g}$ was above and/or to the left of that at $\vec{f}$, the side it used as output is now oriented so that it can be used as output from the copy at $\vec{g}$. We add to $\vec{\alpha'}$ the tile placements which now make a copy of the path π_g^f, again orienting the final tile to allow yet another copy. We make $\vec{\alpha'}$ add $2n$ copies of π_g^f. Then, we let the final tile on the final copy of π_g^f be oriented so that the final portion of the original stretched copy of π_x^c can grow. At this point, $\vec{\alpha'}$, which is still a valid assembly sequence in $\mathcal{T}$, forms the stretched copy of π_σ^x, as well as the stretched copy of π_a^c, with the addition that some interior portion of π_a^c has been "pumped" so that the overall distance between the tiles at the end of the path are at a Manhattan distance at least $2n$ from each other. This is because the stretched path π_a^c is at least that long, containing $2n$ copies of π_g^f, and it grows so that each tile placed makes the endpoints strictly further from each other. Further, the tiles placed at the endpoints of the stretched version of π_a^c are of the same types as those placed at $\vec{a}$ and $\vec{c}$ in α. Since those were at the corners of the square S, they were tile types from $B \subseteq T$ and must therefore be inside the square S. However, in S, no two points are at a Manhattan distance greater than $2n - 1$ from each other. Thus, $\vec{\alpha'}$ does not weakly (or therefore strictly) self-assemble S and neither does $\mathcal{T}$. This is a contradiction, and thus Theorem 7 is proven. □

4.4.3 Assembling finite shapes in the RTAM

In this section we first give a corollary of Theorem 6 showing that sufficiently symmetric shapes weakly self-assemble in the RTAM. Then we prove 3 theorems about assembling finite shapes in $\mathbb{Z}^2$ in the RTAM. These theorems show that the assembly of finite shapes in singly seeded RTAM systems is quite a bit different than the assembly of finite shapes in the aTAM by singly seeded systems.

Given a shape S in $\mathbb{Z}^2$, let χ_S denote the characteristic function of the set S. That is, $\chi_S(x, y) = 1$ if $x \in S$ and $\chi_S(x, y) = 0$ otherwise. Then, we say that a shape S is *odd-symmetric* with respect to a horizontal line $y = l$ (respectively, vertical line $x = l$) for $l \in \mathbb{Z}^2$ iff for all $(a, b) \in \mathbb{Z}^2$, $\chi_S(a, l-b) = \chi_S(a, l+b)$ (respectively, $\chi_S(l-a, b) = \chi_S(l+a, b)$).

If there exists a line such that a shape, S, is odd-symmetric with respect to this line, we say that the shape is odd-symmetric. Given a shape S, we call the smallest rectangle of points in $\mathbb{Z}^2$ containing S the *bounding-box* for S.

Let R denote the bounding box of an odd-symmetric shape S, and let n and m in $\mathbb{N}$ be the dimensions of R. A simple modification to the proof of Theorem 6 where a tile is labeled with a label B if and only if it corresponds to points in S, shows that odd-symmetric shapes can weakly assemble in the RTAM.

Corollary 3. Given a shape S in $\mathbb{Z}^2$, if S is odd-symmetric, then there exists an RTAS $\mathcal{T} = (T, \sigma, \tau)$ such that $|\sigma| = 1$, $\tau \geq 1$, and $\mathcal{T}$ weakly assemblies S.

Additionally, if one is willing to build 2 mirrored copies of the shape in each assembly, then any finite shape can be weakly self-assembled in the RTAM at $\tau = 1$, along with its mirrored copy (at a cost of tile complexity approximately equal to the number of points in the shape) by simply building a central column (or row) from which identical copies of hardcoded rows (or columns) grow, so that each side grows a reflected copy of the shape in hardcoded slices.

We say that a TAS (in either the aTAM or the RTAM) $\mathcal{T}$ is called *mismatch-free* if for every producible assembly $\alpha \in \mathcal{A}[\mathcal{T}]$ with two neighboring tiles with abutting edges e_1 and e_2, either e_1 and e_2 do not have glues or e_1 and e_2 have glues with matching labels and strengths. Then, for singly seeded aTAM systems, any finite connected shape can be strictly assembled by a mismatch-free system. Theorems 8, 9, and 10 show that assembling shapes in the RTAM is more complex.

Theorem 8. There exists a finite connected shape S in $\mathbb{Z}^2$ that weakly self-assembles in a singly seeded RTAM system such that there exists no singly seeded RTAM system that strictly self-assembles S.

Proof. Consider the shape S depicted in Figure 4.14a. By Corollary 3, there is a singly seeded RTAM system that weakly self-assembles S. Therefore, to complete the proof, we show that this shape cannot be strictly self-assembled by a singly seeded system in the RTAM by contradiction. Suppose that $\mathcal{T} = (T, \sigma, \tau)$ is a singly seeded system that strictly assembles S, and let α in $\mathcal{A}_\Box[\mathcal{T}]$ be a terminal assembly such that dom $(\alpha) = S$. Without loss of generality, assume that the seed for $\mathcal{T}$ is at a location shown in red in Figure 4.14a. Then, by applying a single tile reflection, we can modify the assembly sequence of α to obtain a producible assembly β depicted in Figure 4.14b, which contradicts the fact that S strictly assembles in $\mathcal{T}$.

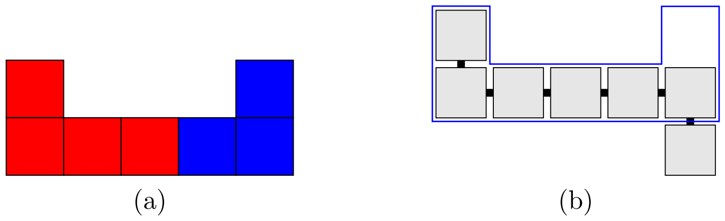


Figure 4.14: (a) A shape S that cannot strictly self-assemble in the RTAM. (b) A producible assembly that can assemble in any system which can also produce an assembly whose domain is S (upto translation and reflection).

$\Box$

Theorem 9. There exists a finite shape S in $\mathbb{Z}^2$ that can be strictly self-assembled by some singly seeded RTAM system such that every singly seeded RTAM system at temperature 1 which strictly self-assembles S is not directed.

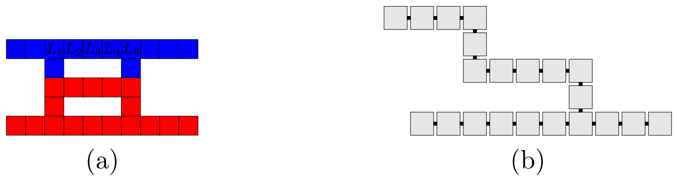


Figure 4.15: (a) A shape S that cannot be self-assembled in the RTAM by a directed system.

Proof. Consider the shape S depicted in Figure 4.15a. First, to see that S can be strictly self-assembled in the RTAM by a singly seeded system, consider the tile set shown in Figure 4.16.

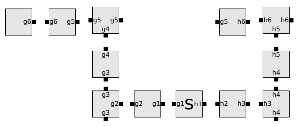


Figure 4.16: A tile set T such that a system $\mathcal{T} = (T, \sigma, \tau)$, with σ a single tile assembly consisting of the tile labeled S, strictly self-assembles the shape S shown in Figure 4.15a. Note that all of the glues depicted in this figure have strength τ, and that assembly of the shape S requires the reflection of these tiles.

Then, to complete the proof, we show by contradiction that S cannot be strictly self-assembled by a directed singly seeded system in the RTAM. Suppose that $\mathcal{T} = (T, \sigma, \tau)$ is a directed singly seeded system that strictly assembles S, and let α in $\mathcal{A}_{\Box}[\mathcal{T}]$ be a terminal assembly such that dom $(\alpha) = S$. Without loss of generality, assume that the location of the seed in α is at a location shown in red in Figure 4.15a.

Let t_i denote the tile of α located at positions L_i. Note that the south glue of either t_1 or t_5 is bound to α with strength τ. Without loss of generality, assume that the south glue of t_1 is bound to α with strength τ. If t_1 is bound to t_2 with strength τ, t_2 is bound to t_3 with strength τ, and t_3 is bound to t_4 with strength τ, then by modifying the assembly sequence for α and applying the appropriate reflections, we can can give an assembly that is producible in $\mathcal{T}$, and is a total of 11 tiles wide. An example of such an assembly is shown in Figure 4.15b. This contradicts the assumption that $\mathcal{T}$ strictly self assembles S. On the other hand, if the strength of the bond between t_i and t_{i+1} is less than τ for $1 \leq i \leq 4$, then the south glues of t_1 and t_5 must each be bound with strength τ. Therefore, the assembly α' depicted as gray tiles in Figure 4.17 must be producible in $\mathcal{T}$. In the following

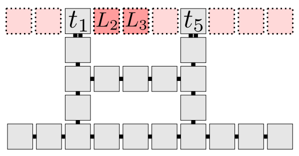


Figure 4.17: A shape S that cannot be self-assembled in the RTAM by a directed system. Strength-τ glues are shown with two bars, the rest of the glues can be either strength-1 or strength-2 as long as the assembly is τ-stable.

argument, without loss of generality, we assume that for every glue, g, on a tile in T, there is a matching glue g' on some (possibly the same) tile in T. Then, since two tiles must be able to attach so that they occupy the two tile locations to the west of t_1, after applying a reflection of t_1, these same tiles, call them a and b, must be able to attach with strength τ to the east to t_1 with the easternmost tile, b, located at L_3. Similarly, (prior to a and b binding) three tiles, call them a', b', and c', must be able to attach with strength τ to the west of t_5, with a' located at L_4, b' located at L_3, and c' located at L_2. Notice that b' must have two strength-τ glues. Hence, nondeterministically, either b or b' will be at location L_3. Finally, note that b and b' cannot have the same tile type. This follows by contradiction; suppose that b and b' are of the same type. Then, after binding, b must expose a strength-τ glue. However, when the westernmost tile that attaches to the west of t_1 is b, this strength-τ glue would allow for a tile to bind, giving a producible assembly with a domain that is not contained in S. This contradicts the assumption that $\mathcal{T}$ strictly self-assembles S. Finally, since in any terminal assembly of $\mathcal{T}$, either b or b' will be at location L_3, and b and b' do not have the same tile type, we conclude that $\mathcal{T}$ is not directed.

□

Theorem 10. There exists a finite shape S in $\mathbb{Z}^2$ such that every singly seeded RTAM system that strictly self-assembles S is not mismatch-free.

Proof. Let S denote the shape shown in Figure 4.15a. Then, the proof follows from the proof of Theorem 4.15. Note that in proof of Theorem 4.15, if b is placed at L_3, then the west edge of a' and the east edge of b must have different glues. Similarly, if b' is placed at L_3, then the east edge of a and the west edge of b' must have different glues. Therefore, any singly seeded RTAM system that strictly self-assembles S is not mismatch-free. □

4.4.4 Mismatch-free assembly of finite shapes in the RTAM

Given a shape S, i.e. a finite connected subset of $\mathbb{Z}^2$, we say that a *graph of* S is a graph $G_S = (V, E)$ with a vertex at the center of each point in S and an edge between every pair of vertices at adjacent points of S. A *tree of* S, T_S, is a graph of S which is a tree. (See Figure 4.18 for examples of S, G_S, and G_T.) Given a graph $G = (V, E)$, we say that an *axis* of G is a horizontal or vertical line of vertices such that there is an edge between each pair of adjacent points on that line. Notice that two distinct axes can be collinear. Given an axis a, an *axial branch* of T_S is a branch of T_S containing exactly 1 vertex v on a and all vertices and edges of T_S which are connected to a vertex that does not lie on a and is adjacent to v. We say that the branch *begins from* v. Intuitively, an axial branch is a connected component extending from an axis. (See the pink highlighted portion of Figure 4.18c for an example axial branch off the green axis.)

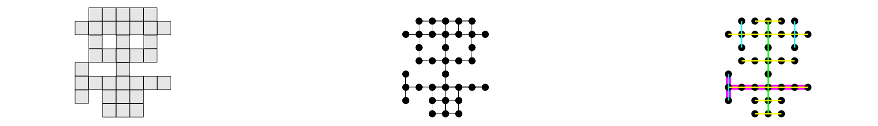

(a) Example shape S (b) Graph of S, G_S (c) Tree of S, T_S

Figure 4.18: An example ϵ-symmetric shape.

A tree T_S is symmetric across an axis a if, for every vertex v contained on a, the branches of a which begin from v are symmetric across a. A tree T_S is *off-by-one symmetric* across an axis a if, for every vertex v except for at most 1, the branches of a which begin from v are symmetric across a. See Figure 4.18c for an example of such a tree, with the axis a shown in green.

Definition. A tree T is *ϵ-symmetric* if and only if for any axis a of T, T is off-by-one symmetric across a.

Definition. Given a shape S with graph G_S, we say that S is *ϵ-symmetric* if and only if there exists a spanning tree, T_S, of G_S such that T_S is ϵ-symmetric.

For an example of an ϵ-symmetric shape S, see Figure 4.18a. The tree T_S is off-by-one symmetric across the vertical green axis, the branches off of that axis are symmetric across the horizontal yellow axes, and the branches off of those axes are symmetric across the vertical blue axes. The following theorem gives a complete classification of finite connected shapes which can be assembled by temperature-1 singly seeded mismatch-free RTAM systems.

Theorem 11. Let $S \subset \mathbb{Z}^2$ be a finite connected shape. There exists a mismatch-free RTAM system $\mathcal{T} = (T, \sigma, 1)$ with $|\sigma| = 1$ that strictly assembles S if and only if S is ϵ-symmetric.

Proof. First, we show that if S is ϵ-symmetric, then there is a singly seeded mismatch-free RTAM system that strictly assembles S. Let T_S be a tree for S which satisfies Definition 4.4.4. We use T_S to define a tile set T and give a seed tile σ such that the system $\mathcal{T} = (T, \sigma, 1)$ strictly self-assembles S, and then we argue that for any terminal assembly α in $\mathcal{T}$, the domain of α is equal to S (up to translation and reflection). T and σ are obtained as follows.

First, notice that we can give a temperature-1 aTAM system $\mathcal{T}' = (T', \sigma', 1)$ that assembles the shape S with the properties (1) $\mathcal{T}'$ is a singly seeded directed system, (2) for $\alpha' \in \mathcal{A}[\mathcal{T}']$, the binding graph of α' is exactly the tree T_S, and (3) when the seed tile σ' is the single tile located at some point v, then for each point p of S, there is a unique tile t' in T' such that $\alpha(p) = t'$. More intuitively, when the seed tile of $\mathcal{T}'$ is placed at v, then assembly proceeds by the binding of unique tiles equipped with glues specific to each point of S.

Now, we define the tiles of T to be copies of the tiles in T' and define σ' to be σ. Notice that in the RTAM, if we assume that no tiles are reflected before binding as assembly proceeds, then the terminal assembly, α, that results has a domain equal to S. Furthermore, since the binding graph of α is T_S, a reflected tile, t, of α lies on some axis, a_t say, and corresponds to some vertex v_t of that axis. Then, since T_S is ϵ-symmetric and appropriately reflected tiles of T can still bind to t even when t is reflected prior to binding, the domain of a terminal assembly of $\mathcal{T}$ will either (1) be equal to S (This is the case that the axial branches beginning from v_t are equivalent upto reflection about the axis a_t. See Figure 4.19b.), or (2) be equal to a reflection of S about the line corresponding to the axis a_0 (This is the special case that the axial branches beginning from v_t are not equivalent after a reflection about a_t. See Figure 4.19c.). In either case, it follows that S strictly assembles in $\mathcal{T}$.

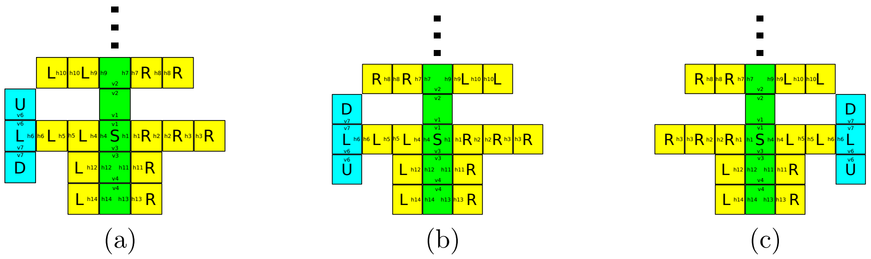


Figure 4.19: An example of the assembly of a portion of an ϵ-symmetric shape which assembles in $\mathcal{T}$. Relative to Figure (a), Figures (b) and (c) give two different producible assemblies of $\mathcal{T}$.

Now we show that if there is a singly seeded mismatch-free RTAM system that strictly assembles S, then S is ϵ-symmetric. Let $\mathcal{T} = (T, \sigma, 1)$ be a singly seeded mismatch-free RTAM system that strictly self-assembles S, and let $\alpha \in \mathcal{A}_{\Box}[\mathcal{T}]$ be some terminal assembly of $\mathcal{T}$.

First, we note that the binding graph of α must be a tree, which we denote by T_α. This follows from the fact that if the binding graph contains a cycle, then there are infinitely many producible assemblies in $\mathcal{T}$. See Figure 4.20 for an example. Moreover, T_α must be ϵ-symmetric. This can be shown by contradiction as follows.

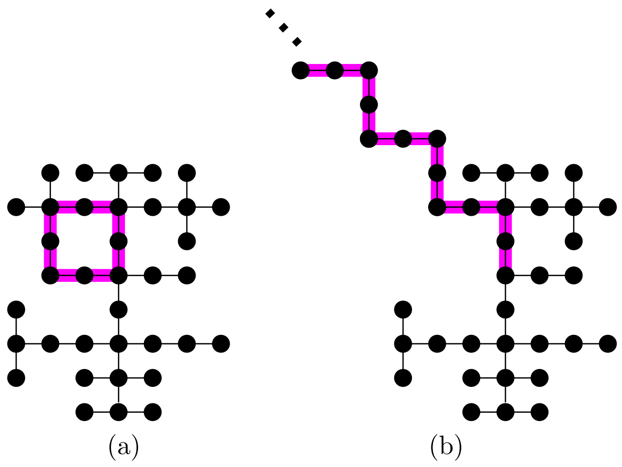


Figure 4.20: (a) An example of a binding graph that contains a cycle. (b) An example of a binding graph of an assembly where tiles with type of those tiles belonging to the cycle shown in (a) are repeated an arbitrary number of times using appropriately reflected tiles.

Suppose that T_α is not ϵ-symmetric. Then, there exists an axis a of T_α such that T_α is not off-by-one symmetric across a. In other words, there are at least two vertices, which we denote by v_1 and v_2, of T_α which lie on a such that the branches of a which begin from v_1 are not symmetric across a and the branches of a which begin from v_2 are not symmetric across a. Now, we can modify the assembly sequence of α by reflecting the tile, t, located at v_1 prior to binding, allowing reflected tiles to assemble reflections of the subassemblies which are bound to t, and keeping the original assembly sequence otherwise. Now, as tiles of the reflected subassemblies originating from t bind, either a mismatch occurs or the reflected subassemblies completely assemble. If a mismatch occurs, we arrive at a contradiction since $\mathcal{T}$ is mismatch-free. If the reflected subassemblies completely assemble, then we note that no translation and reflection of dom (β) is equal to S. (This is because we essentially chose a sequence where the two asymmetric branches across a adopted reflections across a such that the sides of each which would be on the same side of a in S are now on different sides of a.) This contradicts the assumption that $\mathcal{T}$ strictly assemblies S. Therefore, T_α is ϵ-symmetric. Hence S is ϵ-symmetric.

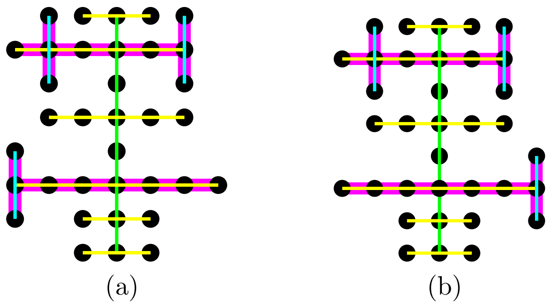


Figure 4.21: An example of possible binding graphs of a shape that is not ϵ-symmetric. (a) and (b) depict binding graphs of two distinct assemblies.

□

The following theorem shows that with cooperation, RTAM systems can assemble arbitrary scale factor 2 shapes.

Theorem 12. Let $S \subset \mathbb{Z}^2$ be a finite connected shape, and S^2 be S at scale factor 2. There exists a mismatch-free RTAM system $\mathcal{T} = (T, \sigma, 2)$ with $|\sigma| = 1$ that strictly self-assembles S^2.

Proof. Let S be a finite connected shape in $\mathbb{Z}^2$ and S^2 be S at scale factor 2. Note that we can give a temperature-1 aTAM system $\mathcal{T}' = (T', \sigma', 1)$ that assembles the shape S with the properties (1) $\mathcal{T}'$ is a singly seeded directed system, (2) for α' in $\mathcal{A}_{\Box}[\mathcal{T}]'$, the binding graph of α' is a tree $T_{\alpha'}$, (3) the location of the seed tile σ' corresponds to a leaf of $T_{\alpha'}$, and (4) for each point p of S, there is a unique tile t' in T' such that $\alpha(p) = t'$. For an example of such an assembly α' see Figure 4.22.

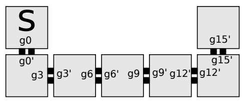


Figure 4.22: An example terminal assembly giving a shape assembled by an aTAM system $\mathcal{T}'$.

We now give an RTAM system $\mathcal{T}$ based on $\mathcal{T}'$ such that a terminal assembly α of $\mathcal{T}$ can be obtained from α' by replacing single tiles of α' by 2×2 blocks of tiles, and thus assembles S^2. aTAM tiles and corresponding RTAM tiles that give rise to this block replacement scheme are described in Figure 4.23.

As 2×2 blocks assemble, cooperation is used to ensure that the orientations of the tiles of a 2×2 block are fixed relative to the orientation of the seed. Figure 4.24 depicts a terminal assembly of the RTAM system $\mathcal{T}$ based on the aTAM system $\mathcal{T}'$ whose terminal assembly is shown in Figure 4.22.

□

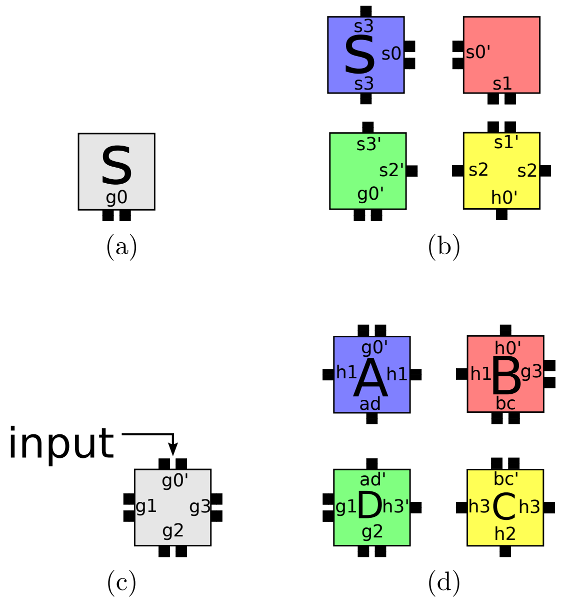


Figure 4.23: (a) An example seed tile of the aTAM system $\mathcal{T}'$. (b) Tiles that represent the seed tile of (a) used to define the RTAM system $\mathcal{T}$. The seed of $\mathcal{T}$ consists of only the tile labeled S. (c) An example tile of the aTAM system $\mathcal{T}'$ that attaches via the glue $g0'$. Note that it may be the case that not all glues g_1, g_2, and g_3 shown here are exposed. (d) Tiles that represent the tile of (c) used to define the RTAM system $\mathcal{T}$. Notice that tiles bind in the order A, B, C and D. The tile shown in (c) has "output" glues $g1$, $g2$, and $g3$ exposed. In the case where one or more of these glues is not exposed by a tile, we obtain tiles analogous to those in (d) that represent this single tile as follows. The glues $g1$, $g2$, and $g3$ of the tiles of (d) are exposed if and only if the respective glues of the tile in (c) are exposed. For example, if the tile in (c) exposes a $g1$ glue, then so will the tile D of (d). Otherwise, D will not expose a $g1$ glue. Note that the cooperative binding of B fixes the orientation of B. Similarly, the cooperative binding of D fixes the orientation of the tile D.

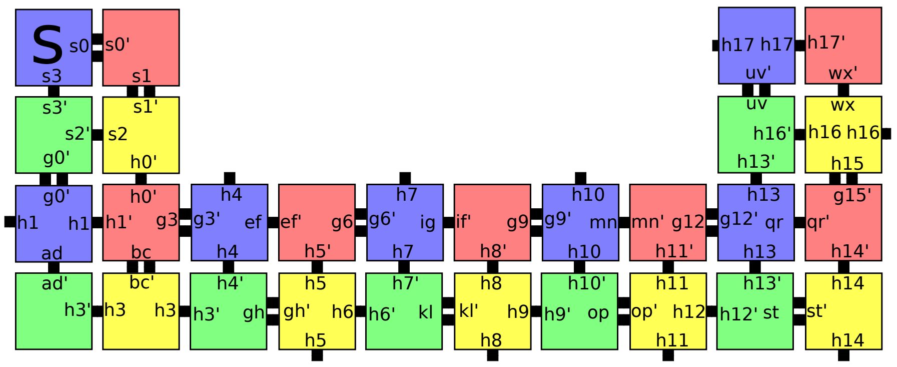


Figure 4.24: An example terminal assembly giving a shape assembled by an RTAM system $\mathcal{T}$ based on the aTAM system $\mathcal{T}'$ whose terminal assembly is shown in Figure 4.22.

Chapter 5

Equivalence of CA and aTAM

5.1 Introduction

Mathematical models of systems composed of large, distributed collections of simple components which are guided by only local interactions between neighboring elements have demonstrated the rise of emergent complexity, and provided enormous insight into the behaviors of many naturally occurring systems, while also guiding the modeling and development of complex artificial systems. Two such notable models are cellular automata (CA) and the astract Tile Assembly Model (aTAM). In this chapter, we seek to explore the relationship between these two models.

Introduced by Stanislaw Ulam and John von Neumann in the 1940's, CA consist of an infinite grid of cells which can each sense their immediate neighborhoods and then all independently but synchronously update their states based on a finite set of rules and the state of their neighborhoods. Since their introduction, CA have provided a rich theoretical framework for studying the power of systems governed by local interactions. Much of that study has included classifications of the relative powers of various CA systems with differing neighborhoods and rules governing their state changes, and a large amount of this classification has been the result of various demonstrations of the ability of one system to simulate another, including very importantly the definitions of what it means for one system to simulate another. A key notion developed during this study was that of intrinsic universality [2, 14, 16, 47, 33], which was designed to capture a strong notion of simulation, in which one particular automaton is capable of simulating the *behavior* of any automaton within a class of automata. Furthermore, to simulate the behavior of another automaton, the simulating automaton must evolve in such a way that a translated rescaling (rescaled not only with respect to rectangular blocks of cells, but also with respect to time) of the simulator can be mapped to a configuration of the simulated automaton. The specific rescaling depends on the simulated automaton and gives rise to a global rule such that each step of the simulated automaton's evolution is mirrored by the simulating automaton, and vice versa via the inverse of the rule. In this way, it is said that the simulator captures the dynamics of the simulated system, acting exactly like it, modulo rescaling. This is in contrast to a computational simulation, for example when a general purpose digital computer runs a program to simulate a cellular automata while the processor's components don't actually arrange themselves as, and behave like, a grid of cellular automata. Such computational simulations of computable systems can be performed by systems in any Turing universal model. However, as we will discuss shortly, Turning universality does not imply the simulation capabilities necessary for intrinsic universality.

Introduced by Erik Winfree in 1998 [64], the abstract Tile Assembly Model (aTAM) is a mathematical model in which the individual components are square "tiles", with "glues" on their edges, which are able to autonomously bind together to form structures based only on the amount and strengths of matching glues on edges of adjacent tiles. The aTAM was inspired by Wang tiling [63], but provides a model for the dynamic growth of tilings. Like various CA, the aTAM has been proven to be computationally universal and capable of quite powerful behavior. Recently, taking example from the work on CA, much work has been done to classify the power of the aTAM and derivative tile assembly models based on their powers of simulation [23, 9, 31, 17, 19]. In fact, [22] showed that the aTAM is intrinsically universal, which means that there is a single tile set U such that, for any aTAM tile assembly system $\mathcal{T}$ (of any temperature), the tiles of U can be arranged into a seed structure dependent upon $\mathcal{T}$ so that the resulting system (at temperature 2), using only the tiles from U, will faithfully simulate the behaviors of $\mathcal{T}$. In contrast, in [19] it was shown that no such tile set exists for the 2HAM since, for every temperature, there is a 2HAM system which cannot be simulated by any system operating at a lower temperature. Thus no tile set is sufficient to simulate 2HAM systems of arbitrary temperature, despite the fact that the 2HAM is computationally universal, and can also simulate any arbitrary aTAM system as shown in [9]. Furthermore, it was shown in [46] that although the aTAM in 3 dimensions is computationally universal at temperature 1 (see [12]), it is unable to simulate the behavior of the majority of temperature 2 aTAM systems. These results from [46] and [12] prove that Turing universality does not imply the simulation power necessary for intrinsic universality.

As early as the aTAM's initial introduction, its power to simulate CA was explored. Winfree et al. showed that the 2-D aTAM can be used to simulate 1-D CA [67], and Winfree [64] showed that the 3-D aTAM can simulate 2-D CA. Furthermore, the aTAM is commonly colloquially referred to as an asynchronous, nondeterministic CA in which quiescent states that change to "tile" states never change again (analogous to write-once memory). These comparisons led naturally to our exploration of simulations between the two models using the same dimensions for each, namely 2-D. However, even between systems within the same model, defining a satisfactory notion of simulation, namely one which captures the essence of one system "behaving" like the other while also generating analogous results, or output, can be difficult. While the definition of a CA system simulating an aTAM system may be in some sense rather natural, the definition of an aTAM system simulating a CA system must take into account the write-once nature of tile assembly systems. To account for this, we modify the standard notions of simulation used in intrinsic universality to allow for an increasing scale factor during simulation. Essentially, such simulation definitions typically make use of a standard block replacement scheme in which, throughout the simulation, each constant sized block of the simulator can be directly mapped to an element of the simulated system. To allow a static model such as the aTAM to simulate a dynamic model such as CA, we allow the scale factor of the simulation to increase after each time step of the simulated system is completed.

For our main results, we present the following. First, a single nondeterministic, synchronous CA which, for any arbitrary aTAM system $\mathcal{T}$, can be given an initial configuration dependent upon $\mathcal{T}$ so that it will exactly simulate $\mathcal{T}$, producing the same output patterns (modulo rescaling) and preserving the dynamics of $\mathcal{T}$. Second, we exhibit a single aTAM tile set which, for any nondeterministic, synchronous CA $\mathcal{C}$ which begins with a finite initial configuration (i.e. all but a finite number of cells begin in a quiescent state), can be given an initial seed configuration dependent upon $\mathcal{C}$ so that it will exactly simulate $\mathcal{C}$, producing the same output patterns (modulo rescaling) and preserving the dynamics of $\mathcal{C}$.

5.2 Preliminaries

5.2.1 Cellular Automata

In our discussion about cellular automata, we will use the following definitions. Most of the conventions used in these definitions come from [14].

Definition. A 2-*dimensional nondeterministic cellular automata* $\mathcal{A}$ is a 4-tuple $(\mathbb{Z}^2, S, N, \delta)$ where

1. S is a finite set of states.
2. $N \subset \mathbb{Z}^2$ is a finite set defining the neighborhood of $\mathcal{A}$.
3. $\delta : S^{|N|} \rightarrow 2^S$ is the local rule of $\mathcal{A}$. δ maps a neighborhood defined by N and a point in $\mathbb{Z}^2$, usually referred to as a *cell*, to a set of states.

Note that a *deterministic cellular automata* is simply a special case of a nondeterministic CA in which $\delta : S^{|N|} \rightarrow S$, i.e. it maps each neighborhood to a single state.

A *configuration* c is a mapping from $\mathbb{Z}^2$ to S. Let C be the set of configurations in $\mathcal{A}$. The *global rule* G is obtained as follows. For $p \in \mathbb{Z}^2$, $G : C \rightarrow 2^C$ such that $c' \in G(c) \iff c'(p) \in \delta(c_{p+v_1}, \ldots, c_{p+v_k})$ where $\{v_1, \ldots, v_k\} = N$.

We assume that S contains a unique *quiescent state* where a quiescent state q is a state such that δ maps a neighborhood of cells in this state to a singleton set containing only the quiescent state. In this chapter, we only consider finite initial configurations (which we will typically denote by c_0) where all but finitely many cells are quiescent. In this chapter we are concerned with the CA-initial configuration pair $(\mathcal{A}, c_0)$ and refer to such pairs as CA *systems*.

There are many interesting examples of cellular automata. One of particular interest here is John Conway's Game of Life. (See [32].) This is a 2D cellular automata where each cell is in one of two states `alive` or `dead`. Local rules are given for a 3×3 squares for cells according to the following.

1. An `alive` cell with less than two neighbors becomes `dead`.
2. An `alive` cell with two or three neighbors stays `alive`.
3. An `alive` cell with more than three neighbors becomes `dead`.
4. A `dead` cell with three `alive` neighbors becomes `alive`.

These simple rules give rise to an amazing amount of complexity and structure. In fact, in [56] a universal Turing machine built in Conway's Game of Life is presented that starts from a finite configuration that encodes another Turing machine and its tape and simulates the execution of the encoded Turing machine with the encoded tape as input.

5.2.2 CA simulation of a TAS

For S as in Definition 5.2.1 and k a vector of $\mathbb{Z}^2$, let $\psi^k : S^{\mathbb{Z}^2} \to S^{\mathbb{Z}^2}$ be the bijection mapping a configuration c to the configuration c' such that for each cell i, $c'_{i+k} = c_i$. ψ^k is called the *shift* operator. Now let $m = (m_1, m_2)$ be a pair of strictly positive integers. $o^m : S^{\mathbb{Z}^2} \to (S^{[0,m_1]\times[0,m_2]})^{\mathbb{Z}^2}$ is the bijection such that for all $c \in S^{\mathbb{Z}^2}$, $z \in \mathbb{Z}^2$ and $p \in [0, m_1] \times [0, m_2]$, $o^m(c)(z)(p) = c(mz + p)$. o^m is called the *packing* map. Let $\mathcal{A} = (\mathbb{Z}^2, S, N, \delta)$ be a cellular automaton. An $\langle m, n, k\rangle$*-rescaling* of $\mathcal{A}$ is a cellular automaton $\mathcal{A}^{\langle m,n,k\rangle}$ with states set $S^{[0,m_1]\times[0,m_2]}$ and global transition function $o^m \circ \psi^k \circ G^n_{\mathcal{A}} \circ o^{-m}$, where $G^n_{\mathcal{A}}$ is the composition of the global function for $\mathcal{A}$ n times.

We now define what it means to say that a synchronous nondeterministic 2D cellular automata with an initial configuration *simulates* an aTAM system. First we let R be the partial function that maps individual cells in some state to single tiles with some tile type. R is the *representation* function. In the following definitions, $\mathcal{A} = (\mathbb{Z}^2, S, N, \delta)$ is a synchronous nondeterministic CA with C denoting the set of configurations and $\mathcal{T} = (T, \sigma, \tau)$ is an aTAM system. We denote by c_0 a finite initial configuration in C and let $\tilde{C} = \cup_{n=0}^{\infty} G^n(c_0)$. In other words, $\tilde{C}$ is all of the configurations obtained by applying the global rule some number of times to c_0. Let $R^* : \tilde{C} \to \mathcal{A}[\mathcal{T}]$ be the canonical extension of R. Finally, we let $(\mathcal{A}, c_0)$ be the pair consisting of the CA $\mathcal{A}$ and the initial configuration c_0.

Definition. We say that $\mathcal{T}$ *follows* $(\mathcal{A}, c_0)$ iff for all $c \in \tilde{C}$, $\alpha, \beta \in \mathcal{A}[\mathcal{T}]$ and $n \geq 0$, if $R^*(c) = \alpha$ and $\beta \in R^*[G^n(c)]$ then $\alpha \longrightarrow \beta$.

Note that $R^*[G^n(c)]$ denotes the image of the set $G^n(c)$ under R^*. Informally, Definition 5.2.2 means that if a configuration represents an assembly α, then anything this configuration maps to under applications of the global rule represents some assembly that α can grow into. The following definition captures the idea that for an assembly α represented by a configuration c, any assembly that α grows into is represented by a configuration obtained from c by applications of the global rule.

Definition. We say that $(\mathcal{A}, c_0)$ *models* $\mathcal{T}$ if $\forall \alpha \in \mathcal{A}[\mathcal{T}], \exists c \in \tilde{C}$ such that $R^*(c) = \alpha$ and $\forall \beta \in \mathcal{A}[\mathcal{T}]$, if $\alpha \longrightarrow \beta$ then $\exists n \geq 0$ such that $\beta \in R^*[G^n(c)]$.

Note that a configuration representing some terminal assembly α must transition to configurations that still represent α. Finally, we give the definition of simulation.

Definition. $(\mathcal{A}, c_0)$ *simulates* $\mathcal{T}$ iff there is an $\langle m, n, k \rangle$-rescaling $\mathcal{A}'$ of $\mathcal{A}$ such that $\mathcal{T}$ follows $\mathcal{A}'$ and $\mathcal{A}'$ models $\mathcal{T}$.

5.2.3 TAS simulation of a CA

As in the previous section, $\mathcal{A} = (\mathbb{Z}^2, S, N, \delta)$ denotes a synchronous nondeterministic CA with a finite initial configuration c_0, C denotes the set of configuration and $\mathcal{T} = (T, \sigma, \tau)$

denotes an aTAM system. Again, let $\tilde{C} = \cup_{n=0}^{\infty} G^n(c_0)$ where c_0 is the finite initial configuration of $\mathcal{A}$ and let $(\mathcal{A}, c_0)$ be the pair consisting of the CA $\mathcal{A}$ and the initial configuration c_0.

Because any aTAM system produces static assemblies and the state of a cell of a CA may change multiple times, it would be impossible to represent a cell of a configuration in $\tilde{C}$ with fixed block assemblies over T. Therefore, we introduce the notion of a *scalable representation function.*

For $n \in \mathbb{Z}^+$, an *n-block supertile* over T is a partial function $\alpha : \mathbb{Z}_n^2 \dashrightarrow T$, where $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$. Let B_n^T be the set of all n-block supertiles over T. The n-block with no domain is said to be *empty.* For a general assembly $\alpha : \mathbb{Z}^2 \dashrightarrow T$, define $\alpha_{x,y}^n$ to be the n-block supertile defined by $\alpha_{x,y}^n(i,j) = \alpha(nx+i, ny+j)$ for $0 \leq i, j < n$.

Let R_n for $n \in \mathbb{N}$ be a partial function that maps assemblies over T to configurations in C with the following property. If $\alpha \in \mathcal{A}[\mathcal{T}]$ and $R_n(\alpha) = c$, then for some n, α can be broken into n-block supertiles such that R_n maps these supertiles to cells of c. In other words, for a given assembly α, the partial function R_n either is not defined on α or maps α to $c \in C$ by mapping n-block supertiles of α to cells of c. Then the scalable representation function is defined as $R : \mathbb{N} \times \mathcal{A}[\mathcal{T}] \dashrightarrow \tilde{C}$ where $R(n, \beta) = R_n(\beta)$. Finally, we define simulation of a CA with initial configuration c_0 by an aTAM system.

Definition. $\mathcal{T}$ *simulates* $(\mathcal{A}, c_0)$ (under scalable representation function R) iff there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that the following hold.

1. $\forall n \in \mathbb{N}$, $R_{f(n)}[\mathcal{A}[\mathcal{T}]] = G^n(c_0)$.

2. $\forall \alpha \in \mathcal{A}[\mathcal{T}]$ such that $R_{f(n)}(\alpha) = c_n \in G^n(c_0)$, for any $\beta \in \mathcal{A}[\mathcal{T}]$ in the domain of $R_{f(n+1)}$ such that $\alpha \longrightarrow \beta$ it must be the case that $R_{f(n+1)}(\beta) \in G^{n+1}(c_0)$.

3. $\forall c_n \in G^n(c_0)$ such that $R_{f(n)}(\alpha) = c_n$ for some $\alpha \in \mathcal{A}[\mathcal{T}]$, if $\alpha \longrightarrow \beta$ where β is in the domain of $R_{f(n+1)}$ then $R_{f(n+1)}(\beta) \in G^{n+1}(c_0)$.

In Definition 5.2.3, f can be thought of as taking a time step n and determines a block size for the representation. Then R takes $f(n)$ and an assembly and either returns a configuration in $G^n(c_0)$ or nothing if the assembly has not fully simulated the n^{th} time step. This is necessary to simulate the dynamics of a synchronous CA, in which all cells simultaneously update their states. Basically statement 1 of Definition 5.2.3 says that starting with an initial configuration, every configuration obtained by applying the global rule is represented by some assembly over T and that any step-assembly pair (n, α) in the domain of R represents some configuration. Moreover, statements 2 and 3 of Definition 5.2.3 implies that these representations follow the action of the global rule.

5.3 A Nondeterministic CA Which Can Simulate Any aTAM System

In Theorem 13, we show that for any aTAM system, there is a synchronous nondeterministic CA such that for an appropriate choice of finite initial configuration, this CA simulates the aTAM system. This gives some sense of a synchronous nondeterministic CA being *intrinsically universal for* the aTAM.

Theorem 13. There exists a synchronous nondeterministic CA $\mathcal{A} = (\mathbb{Z}^2, S, N, \delta)$ such that for any aTAM system $\mathcal{T} = (T, \sigma, \tau)$ there exists a finite initial configuration c_0 of $\mathcal{A}$ so that $(\mathcal{A}, c_0)$ simulates $\mathcal{T}$.

To show this Theorem, we appeal to the following Lemma. The construction in Section 5.3.1, proves this Lemma.

Lemma 9. For any aTAM system $\mathcal{T} = (T, \sigma, \tau)$, there exists a synchronous nondeterministic CA $\mathcal{A} = (\mathbb{Z}^2, S, N, \delta)$ and an initial configuration c_0 such that $(\mathcal{A}, c_0)$ simulates $\mathcal{T}$.

Then Theorem 13 is proven as follows. First, there is a tile set U which is intrinsically universal for the aTAM and can be used at temperature $\tau = 2$ to simulate any aTAM system. Therefore we let $\mathcal{U}$ be an aTAM system that uses U at $\tau = 2$. By Lemma 9, we can then give a CA that suffices for Theorem 13 by constructing a CA that simulates an arbitrary $\mathcal{U}$ (i.e. one with an arbitrary seed), which we give in the following section.

5.3.1 CA Construction

The goal of this construction is to give a synchronous nondeterministic CA $\mathcal{B} = (\mathbb{Z}^2, S, N, \delta)$ and initial configuration that simulates an arbitrary aTAM system $\mathcal{T} = (T, \sigma, \tau)$. The neighborhood of $\mathcal{B}$ is the Moore neighborhood and the states and local rules for $\mathcal{B}$ are obtained as follows. First, we add a state to S for each tile type of $\mathcal{T}$ we call these states `tile_states`. We also add states `token_state_up`, `token_state_left`, `token_state_down` and `token_state_right` and we use `token_states` to refer to any of these 4 states. We refer to any cell in a `token_state` as the *token*. This token moves one cell counterclockwise at each time step and only one cell is in a `token_state` at any given time. At each time step, the cell in a `token_state` moves one cell either up, left, down or right in an effort to traverse the *surfaces* of an existing configuration, where a surface of a configuration is the connected set of quiescent cells that neighbor (using the Moore neighborhood) at least one non-quiescent state. (See Figure 5.1.) Note that a configuration may have many disjoint surfaces.

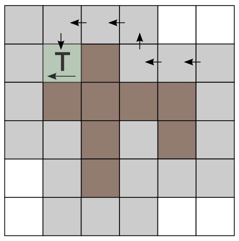


Figure 5.1: A token traversing the surface of a configuration. The surface of the configuration is denoted by light grey tiles. The cell labeled T is in `token_state_left` as indicated by the arrow depicted on the cell.

At each time step, a cell in a `token_state` can nondeterministically transition to a `tile_state` if and only if the tile corresponding to this state could bind in the simulated aTAM system. This ensures that at any given time step, at most one cell transitions from a quiescent state to a `tile_state`. Figure 5.2 shows an example of a local rule obtained from a tile set.

The idea is that the token can put cells in a `tile_state` on the surface of a configura-

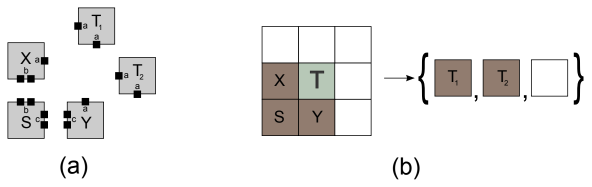


Figure 5.2: **(a)** A tile set consisting of 5 tile types. Glues b and c have strength 2 and a glues have strength 1. **(b)** A local rule corresponding to the tile set in **(a)**. Cells in the Moore neighborhood that are in `tile_states` are labeled with the label of the corresponding tile type in **(a)**. The cell labeled T is in a `token_state`. Blank cells are quiescent.

tion. However a configuration may have many disjoint surfaces. Non-quiescent states of a configuration can break the $\mathbb{Z}^2$ lattice into disjoint regions of cells in quiescent states. For example, this can occur when the CA is simulating a tile set that assembles a frame, i.e. tiles around some square of empty tiles. This leads to disjoint surfaces that the token must traverse. Therefore, care must be taken in order to allow the token to traverse surfaces separated by non-quiescent states. This is accomplished by adding a `bridge_tile_state` to S for each tile type of the simulated aTAM system. The token is allowed to "pass over" these `bridge_tile_states`. Passing over a cell in `bridge_tile_state` is done by adding a `bridge_tile_token_state` to S for each tile type in T and each direction `up`, `left`, `down` and `right`. Figure 5.3 shows the token and its path as it traverses two surfaces of the configuration by crossing `bridge_tile_states` . When transitioning to a `tile_state` or `bridge_tile_state`, we can determine which state to transition to by using Moore neighborhoods. For more details on how `token_states` or `bridge_tile_states` work, see the full paper [35].

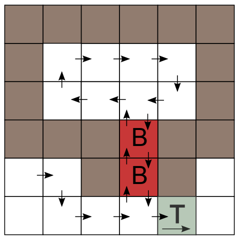


Figure 5.3: Traversing two disjoint surfaces using a single token and `bridge_tile_states`

If and when a path of cells in a `bridge_tile_state` no longer leads to quiescent states and the final quiescent state transitions to a `tile_state`, the token traverses the cells in `bridge_tile_states` as it continues its counterclockwise traversal of a configuration. Since the path of cells in `bridge_tile_states` no longer leads to any quiescent states, as the token traverses cells in `bridge_tile_states`, these cells transition to `tile_states` that correspond to their `bridge_tile_state` counterparts. As a result, the token no longer unnecessarily traverses a path of cells in `bridge_tile_states` that would only lead to other cells in `bridge_tile_states`.

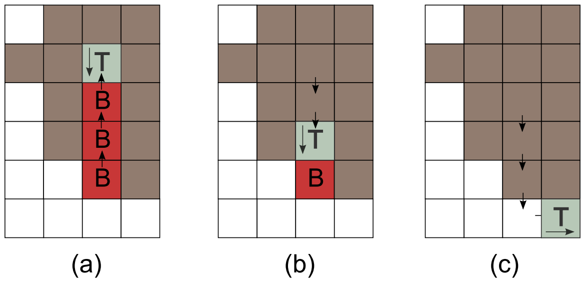


Figure 5.4: **(a)**The token prior to transitioning to a `tile_state`. **(b)** Two time steps later we see a cell in `bridge_tile_state` has transitioned to just a `tile_state`. **(c)** The entire path of cells in `bridge_tile_states` have transitioned to corresponding `tile_states`.

An example of a CA simulating an aTAM system can be found at http://self-assembly.net/CASimTAS. There are also instructions for creating a CA based on an aTAM system.

5.4 An aTAM Tile Set Which Can Simulate Any Nondeterministic CA

Theorem 14. There exists an aTAM tile set U which is able to simulate the entire class of nondeterministic CA systems with finite initial configurations.

Theorem 14 states that there is a single tile set U in the aTAM which can be used to form a TAS $\mathcal{U} = (U, \sigma_{\mathcal{C}}, 2)$ which is dependent upon a given CA $\mathcal{C}$, for any arbitrary nondeterministic CA $\mathcal{C}$ and a finite initial configuration for $\mathcal{C}$, where the seed to the aTAM system encodes information about $\mathcal{C}$ and its initial configuration, so that $\mathcal{U}$ simulates $\mathcal{C}$. In order to prove Theorem 14, we will progress in two steps, first proving the following Lemma.

Lemma 10. Let CA $\mathcal{A}$ be Conway's Game of Life. There exists an aTAM tile set U and a scalable representation function R such that, given c_0 as an arbitrary but finite initial configuration of $\mathcal{A}$, there exists an aTAM TAS $\mathcal{T} = (U, \sigma_{c_0}, 2)$ such that $\mathcal{T}$ simulates $(\mathcal{A}, c_0)$.

Lemma 10 states that there exists a single tile set U in the aTAM which can be used to simulate the Game of Life CA given any finite initial configuration. We now present a construction to prove this.

5.4.1 Overview of construction to prove Lemma 10

The system $\mathcal{T} = (U, \sigma_{c_0}, 2)$ will be designed so that the seed is a single line of tiles which encodes the initial configuration, c_0, of $\mathcal{A}$. Assume that all non-quiescent cells within c_0 can fit into an $n \times n$ square. (Throughout this discussion, we will refer to a *cell* as exactly one of the cells of $\mathcal{A}$ and the *grid* as the full set of cells being simulated at a given time. A *step* refers to a single time step of $\mathcal{A}$ and a *stage* refers to the assembly representing the entire grid at a particular step.) The encoding of the initial configuration consists of a listing of the states of each of the n^2 cells within that box. Since it is possible that, at each time step $0 < t < \infty$, a cell which was previously quiescent and which was just outside the boundary of the currently simulated grid switches its state to a non-quiescent value, to accurately simulate the full behavior of $\mathcal{A}$ we must simulate an increasingly larger grid at each time step. In order to assure that no (non-quiescent) behavior of $\mathcal{A}$ could occur beyond the bounds of our simulation, at each stage we increase the dimensions of the grid by 2, adding a row of cells to each of the top and bottom, and a column to each of the left and right. We say that we perform a recursive, "in-place" simulation of $\mathcal{A}$, namely one in which every subassembly which maps to a single cell at some time step t contains within it, at smaller scale factors, the entire configuration of $\mathcal{C}$ at *every* time step $t' < t$ (recursive), and also that the subassembly mapping to any single cell at any time step t is contained within an infinite hierarchy of subassemblies which each map to a unique cell at some time step t' where $t < t' < \infty$, i.e. each simulated cell and grid is fully encapsulated within the simulation of a single cell at the next greater time step (in-place).

See Figures 5.5 and 5.6 for high-level depictions of the simulation of time steps 0 (the initial configuration of $\mathcal{A}$) and 1 (the first transition of $\mathcal{A}$). Details of the construction can be found in [35].

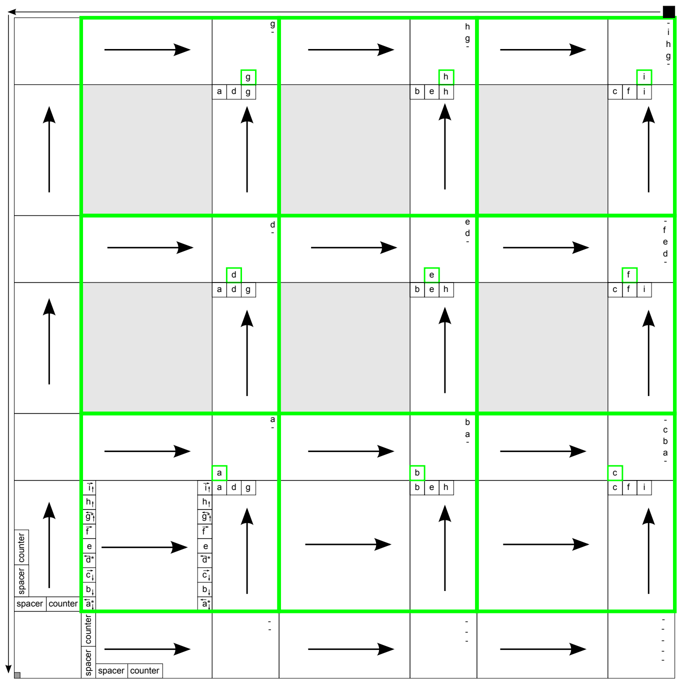


Figure 5.5: Completed formation of the representation of time step 0.

5.4.2 Overview of construction to prove Theorem 14

Now that we have defined the above construction for a tile set which can simulate the Game of Life CA given an arbitrary finite initial configuration, we sketch the necessary extensions to provide a tile set which can simulate *any* synchronous nondeterministic CA.

Let $\mathcal{C} = (\mathbb{Z}^2, S, N, \delta)$ be an arbitrary synchronous nondeterministic CA, d be the maximum unit distance of any element of N from the center position (i.e. the distance of a cell's furthest neighbor in its neighborhood), b be the number of bits required to represent S, and c_0 be the initial configuration of $\mathcal{C}$. Now, define M as a nondeterministic Turing machine which takes as input the encoding of a synchronous nondeterministic CA (in some standard encoding) and the representation of a grid of cells in the same format they are

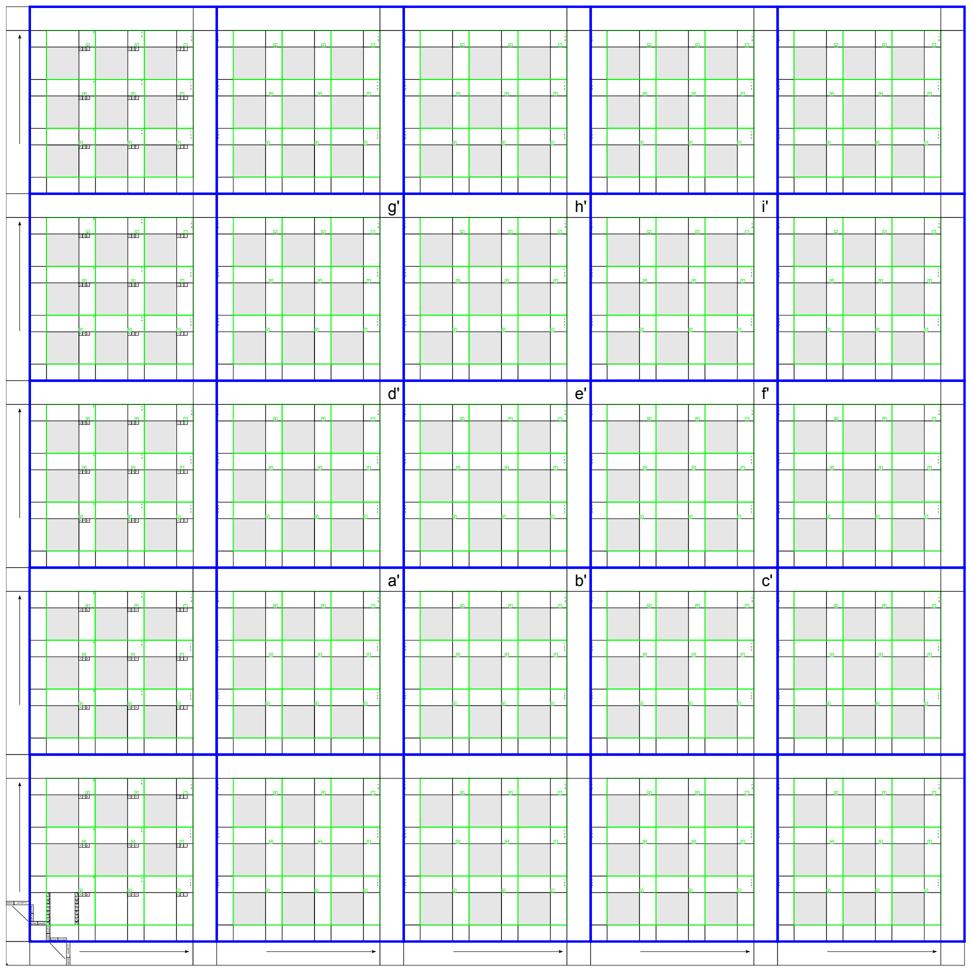


Figure 5.6: Completed growth of the second stage. Blue squares depict the representations of the cells of that stage.

represented in the previous construction (i.e. as they appear in the seed assembly or along the western edges of the stage fibers of a given stage), with the only difference being that the states are not now restricted to only single bits, but instead may be series of bits (with delimiters between the bits representing the states of different cells), and outputs the new cell state for the one cell marked with "$*$" and "$+$". Note that M must be nondeterministic to simulate $\mathcal{C}$, and in order to randomly select from a set of s possible states it simply chooses the bits of a binary number i between 0 and $s-1$ (the random choice of bits will be performed by the nondeterministic attachment of one of two tiles in each bit position) and then selects the ith of the possible states. For details on how to approach uniform distribution across the selection of possible choices, various gadgets of increasing but bounded space complexity can be used (see [24]). Define r to be the longest running time of M when given any single neighborhood $S^{|N|}$ for $\mathcal{C}$, and let m be the maximum amount of tape space used. Note that both r and m can be easily (if exhaustively) determined by simply running M for each of the $S^{|N|}$ possible neighborhoods.

Now we adjust the previous construction as follows. To create c_0' from c_0, we do as before, but we add an additional d rings of quiescent cells around c_0' to account for the fact that $\mathcal{C}$ may have an arbitrarily large neighborhood and we want to simulate enough quiescent cells around the border of our grid at all times to ensure that we are simulating all non-quiescent cells. In the seed assembly, in place of the "spacer", encode the definition of $\mathcal{C}$ (in the encoding used by M) plus $r+m$ spacer tiles to provide enough space for M's tape and running time (the space provided here will ensure that rotations of these values provide the necessary space throughout the assembly). In place of the transition computation gadget, M will be run. In order to do this, whenever the boundary fiber reaches a point at which it would have formerly grown a square which initiates the transition computation gadget, instead of growing the square, M is simulated in a standard zig-zag manner. Additionally, M only computes the transition for a single cell (beginning with the bottom one marked with "$*$") and then passes the newly computed cell value along with the rest of the (unchanged) cell values for that column upward. As before, all needed information is also passed through the simulation of M to the right. Now, rather than just receiving the new cell values and passing them along, the squares at the intersections of vertical and horizontal stage fibers also execute M, whose definition is passed in from the west via the western boundary fiber. This allows all of the same information flow, but splits up the computations of new cell values in such a way that each simulated cell computes at most a single new cell value, which can be done within the time and space bounds, r and m. In order to provide consistency of scale, a counter is embedded within the running of M so that, in the case of computations which require variable amounts of time relative to each other, the counter ensures that $r + m$ space is always used. Again, as in the previous construction, once the value for a cell at a particular time step is computed, it is continually passed along into all other representations of the same cell within other larger cells, but never recomputed.

Finally, the representation function R now must be adjusted to take into account the fact that cell states are now binary strings, and also take into account the new scaling factor due to the padding provided to run M. Nonetheless, this construction retains the same properties such as the previous, in terms of completing each stage before beginning the next.

Chapter 6

Dupled aTAM

6.1 Introduction

We first introduce the *Dupled abstract Tile Assembly Model* (DaTAM), which is essentially the aTAM extended to allow both square and rectangular, duple, tile types. We then show a series of results within the DaTAM which prove that at temperature 1 it is quite powerful: it is computationally universal and able to build $N \times N$ squares using $O(\log N)$ tile types. We next demonstrate that, while the addition of duples does provide significant power to temperature-1 systems, it doesn't allow for asymptotic gains over the aTAM in terms of the tile complexity required to self-assemble thin rectangles, with the lower bound for an $N \times k$ rectangle being $\Omega\left(\frac{N^{1/k}}{k}\right)$. We then provide a series of results which show that the neither the aTAM at temperature-2 nor the DaTAM at temperature-1 is strictly more powerful than the other, namely that in each there are shapes which can be self-assembled which are impossible to self-assemble in the other, and that there are also systems in each which cannot be simulated by the other. These mutually exclusive powers provide a very interesting framework for further study of the unique abilities provided by the incorporation of duples into self-assembling systems. Furthermore, as previously mentioned, the use of duples has already been proven possible in laboratory experiments, providing even further motivation for the model.

6.2 Preliminaries

6.2.1 Informal description of the Dupled abstract Tile Assembly Model

In this section, we give a very brief, informal description of the abstract Tile Assembly Model (aTAM) and the Dupled abstract Tile Assembly Model (DaTAM). For a more detailed, technical definition please refer to [39].

The abstract Tile Assembly Model (aTAM) was introduced by Winfree [64]. In the aTAM, the basic components are translatable but non-rotatable *tiles* which are unit squares with *glues* on their edges. Each glue consists of a string *label* value and an integer *strength* value. A *tile type* is a unique mapping of glues (including possibly the *null* glue) to 4 sides, and a tile is an instance of a tile type. Assembly begins from a specially designated *seed* which is usually a single tile but maybe be a pre-formed collection of tiles, and continues by the addition of a single tile at a time until no more tiles can attach. A tile is able to bind to an adjacent tile if the glues on their adjacent edges match in label and strength, and can attach to an assembly if the sum of the strengths of binding glues meets or exceeds a system parameter called the *temperature* (which is typically set to either 1 or 2). A *tile assembly system* (TAS) is an ordered 3-tuple (T, σ, τ) where T is the set of tile types (i.e. tile set), σ is the seed configuration, and τ is the temperature.

The Dupled abstract Tile Assembly Model (DaTAM) is an extension of the aTAM which allows for systems with square tiles as well as rectangular tiles. The rectangular tiles are 2×1 or 1×2 rectangles which can logically be thought of as two square tiles which begin pre-attached to each other along an edge, hence the name *duples*. A *dupled tile assembly system* (DTAS) is an ordered 5-tuple (T, S, D, σ, τ) where T, σ, and τ are as for a TAS, and S is the set of singleton (i.e. square) tiles which are available for assembly, and D is the set of duple tiles. The tile types which make up S and D all belong to T, with those in D each being a combination of two tile types from T.

6.2.2 Zig-zag tile assembly systems

Originally defined in [12], we define zig-zag tile assembly systems and compact zig-zag tile assembly systems in the same manner as [52]. In [52] they called a system $\mathcal{T} = (T, \sigma, \tau)$ a zig-zag tile assembly system provided that (1) $\mathcal{T}$ is directed, (2) there is a single sequence $\vec{\alpha} \in \mathcal{T}$ with $\mathcal{A}_{\Box}[\mathcal{T}] = \{\vec{\alpha}\}$, and (3) for every $\vec{x} \in \text{dom } \alpha, (0, 1) \notin \text{IN}^{\vec{\alpha}}(\vec{x})$. More intuitively, a zig-zag tile assembly system is a system which grows to the left or right, grows up some amount, and then continues growth again to the left or right. Again, as defined in [52], we call a tile assembly system $\mathcal{T} = (T, \sigma, \tau)$ a *compact zig-zag tile assembly system* if and only if $\mathcal{A}_\Box[\mathcal{T}] = \{\vec{\alpha}\}$ and for every $\vec{x} \in \text{dom } \alpha$ and every $\vec{u} \in U_2$, $\text{str}_{\alpha(\vec{x})}(\vec{u}) + \text{str}_{\alpha(\vec{x})}(-\vec{u}) < 2\tau$. Informally, this can be thought of as a zig-zag tile assembly system which is only able to travel upwards one tile at a time before being required to zig-zag again.

6.2.3 Simulation

In this section, we present a high-level sketch of what we mean when saying that one system *simulates* another. Please see [39] for complete, technical definitions, which are based on those of [46].

For one system $\mathcal{S}$ to simulate another system $\mathcal{T}$, we allow $\mathcal{S}$ to use square (or rectangular when simulating duples) blocks of tiles called *macrotiles* to represent the simulated tiles from $\mathcal{T}$. The simulator must provide a scaling factor c which specifies how large each macrotile is, and it must provide a *representation function*, which is a function mapping each macrotile assembled in $\mathcal{S}$ to a tile in $\mathcal{T}$. Since a macrotile may have to grow to some critical size (e.g. when gathering information from adjacent macrotiles about the simulated glues adjacent to its location) before being able to compute its identity (i.e. which tile from $\mathcal{T}$ it represents), it's possible for non-empty macrotile locations in $\mathcal{S}$ to map to empty locations in $\mathcal{T}$, and we call such growth *fuzz*. In standard simulation definitions (e.g. those in [46, 19, 34, 22]), fuzz is restricted to being laterally or vertically adjacent to macrotile positions in $\mathcal{S}$ which map to non-empty tiles in $\mathcal{T}$. We follow this convention for the definition of simulation of aTAM systems by DaTAM systems. However, since duples occupy more than a unit square of space, for our definition of aTAM systems simulating DaTAM systems, we allow fuzz to extend to a Manhattan distance of 2 from a macrotile which maps to a non-empty tile in $\mathcal{T}$. As a further concession to the size of duples, for that simulation definition we also allow empty macrotile locations in $\mathcal{S}$ to map to tiles in $\mathcal{T}$, provided they are half of a duple for which the other half has sufficiently grown. Thus, while our result for aTAM systems simulating DaTAM systems (Theorem 17) shows its impossibility in general, our intent with the simulation definitions is to relax them sufficiently that, if simulation equivalent to the standard notions of simulation were possible, these definitions would allow it.

Given the notion of block representations, we say that $\mathcal{S}$ simulates $\mathcal{T}$ if and only if (1) for every producible assembly in $\mathcal{T}$, there is an equivalent producible assembly in $\mathcal{S}$ when the representation function is applied, and vice versa (thus we say the systems have *equivalent productions*), and (2) for every assembly sequence in $\mathcal{T}$, the exactly equivalent assembly sequence can be followed in $\mathcal{S}$ (modulo the application of the representation function), and vice versa (thus we say the systems have *equivalent dynamics*). Thus, equivalent production and equivalent dynamics yield a valid simulation.

6.3 Mutually Exclusive Powers

In this section, we demonstrate a variety of shapes and systems in the DaTAM at $\tau = 1$ and the aTAM at $\tau = 2$ which can be self-assembled and simulated, respectively, by only one of the models.

6.3.1 A shape in the DaTAM but not the aTAM

In this section, we show that there exists an infinite shape which can self-assemble in the DaTAM at $\tau = 1$ but not in the aTAM at $\tau = 2$. Figure 6.1 shows a high-level sketch of a portion of this shape.

Theorem 15. There exists a shape $W \subset \mathbb{Z}^2$ such that there exists DTAS

$$\mathcal{D} = (T_{\mathcal{D}}, S, D, \sigma, 1)$$

in the DaTAM which self-assembles W, but no TAS $\mathcal{T} = (T, \sigma', 2)$ in the aTAM which self-assembles W.

Here we give an intuitive overview of why the aTAM cannot simulate the shape depicted in Figure 6.1. See [39] for the full proof. First, we call the shape in Figure 6.1 W.

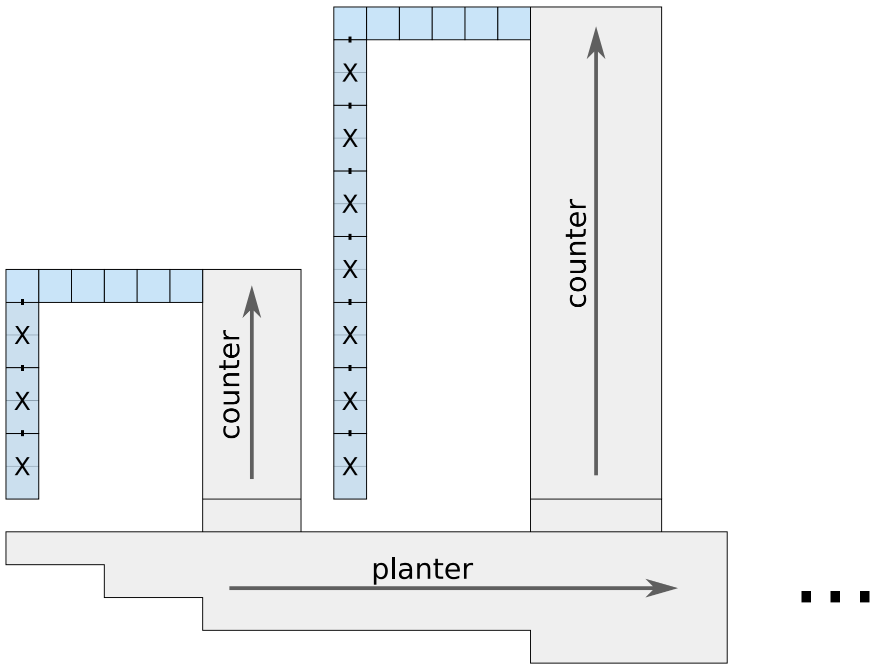


Figure 6.1: A high-level sketch of a portion of the infinite shape which can self-assemble in the DaTAM at $\tau = 1$ but not in the aTAM at $\tau = 2$. (Modules not to scale.)

Since, by the proof of Theorem 2, DaTAM systems are capable of simulating compact zig-zag systems, W assembles in the DaTAM as follows. A horizontal counter called the `planter` begins growth from a single tile seed and continues to grow indefinitely. The top-most tiles of the `planter` expose glues that allow vertical counters to grow. Each of these vertical counters is a finite subassembly whose height is an even number of tile locations and, from left to right, each successive counters grows to a height that is greater than the previous counter. When a vertical counter finishes upward grow, a single tile wide path of 6 tiles binds to the left of the counter. The leftmost tile of this single tile wide path exposes a south glue that allows for duples to attach. Equipped with matching north and south glues, these duples form a single tile wide path of duples, called a `finger`, that grows downward toward the `planter`. Since the height of each vertical counter is even and the first duple of a `finger` is placed 1 tile location below this height, there are an odd number of tile locations for the duples of a `finger` to occupy. As a result, each finger is forced to cease growth exactly 1 tile location away from the `planter`.

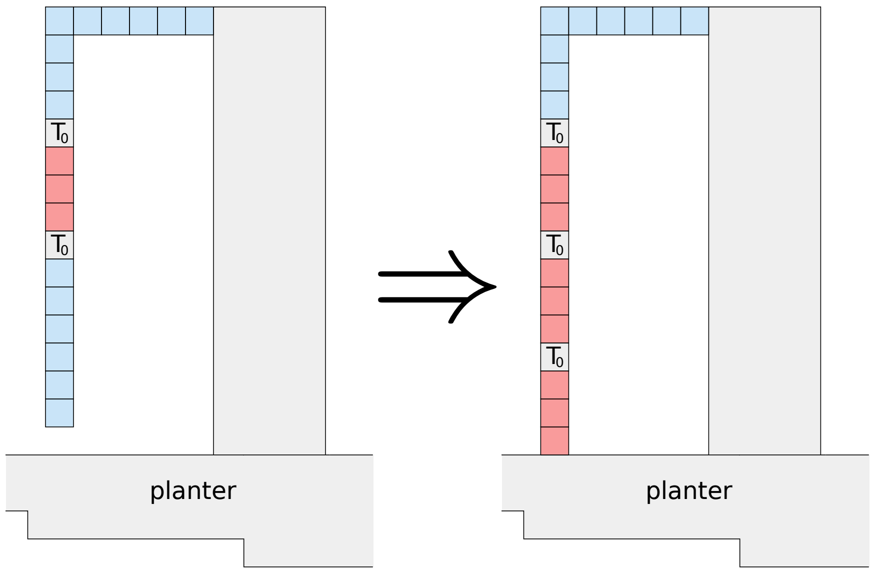


Figure 6.2: Left: A `finger` containing two occurrences of a tile of type T_0. Right: A valid producible assembly that results in a shape that differs from W.

Since in an aTAM system, any tile of an assembly takes up a single location of the infinite grid-graph, it is impossible to grow the `finger` component of the shape W. This essentially follows from the fact that for a single tile wide line of length l assembled in a TAS, if the number of tiles in l is greater than the number of tile types in the TAS, then at least two tiles of l must have the same type. Therefore, by repeating the tiles between these two tiles of the same type, we can attempt to grow a line indefinitely. Hence, when a TAS attempts to grow a `finger` that is longer than the number of tile types in the TAS, we can always find an assembly sequence such that the line forming this `finger` places a tile one tile location above the tiles forming the `planter`. Figure 6.2 depicts this invalid assembly. Therefore, no TAS can assemble W.

6.3.2 A shape in the aTAM but not the DaTAM

In this section, we give a high-level sketch of the proof that there exists a shape which can self-assemble in the aTAM at $\tau = 2$ but not in the DaTAM at $\tau = 1$. Please see [39] for the full details.

Theorem 16. There exists a shape $S \subset \mathbb{Z}^2$ such that there exists a TAS $\mathcal{T} = (T, \sigma, 2)$

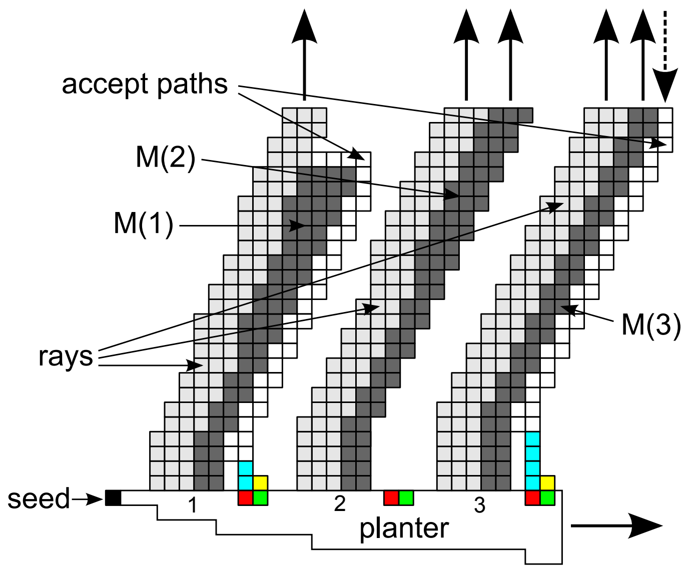


Figure 6.3: Schematic depiction of a portion of the infinite shape which can self-assemble in the aTAM at $\tau = 2$ but not in the DaTAM at $\tau = 1$.

in the aTAM which self-assembles S, but no DTAS $\mathcal{D} = (T_{\mathcal{D}}, S_{\mathcal{D}}, D_{\mathcal{D}}, \sigma', 1)$ in the DaTAM which self-assembles S.

See Figure 6.3 for a high-level sketch of a portion of the infinite shape, which is based on the shape used in the proof of Theorem 4.1 of [8] (which in turn is based on that of Theorem 4.1 of [41]). Essentially, $\mathcal{T}$ assembles S in the following way. Beginning from the seed, it grows a module called the `planter` eastward. The `planter` is a modified log-height counter which counts from 1 to ∞, and for each number - at a well-defined location - places a binary representation of that number on its north side. From each such location, modules called `rays` and Turing machine simulations begin. Each `ray` grows at a unique and carefully defined slope so that it can direct the growth of its adjacent Turing machine simulation in such a way the no Turing machine simulation will collide with another `ray`, but it also potentially has infinite tape space for its computation. The infinite series of Turing machine computations each run the same machine, M, on input n where n is the value presented by the `planter` at that location. If and only if each computation halts and accepts, a path of tiles grows down along the right side of the computation until it reaches a position from which it grows a vertical path of tiles directly downward to crash into the `planter` (blue in Figure 6.3). It's important that the height of the vertical portions (blue) of the paths increase for each. If and when a path places a tile adjacent to the `planter`, glue cooperation between the final tile of the path and a `planter` tile allow for the placement of a final tile (yellow in Figure 6.3). S is the infinite shape resulting from the growth of all portions.

The reason that S cannot assemble in the DaTAM at $\tau = 1$ is that glue cooperation cannot be used to place the yellow tiles, so each must be able to attach to just a tile in the blue portion of a path or the `planter` tile in a green location. It is impossible for all yellow tiles to be placed correctly because if they attach to 1) the blue portions of paths, since those get arbitrarily long, they must have repeating tile types which could be used to grow blue paths of the wrong height which allow yellow tiles to attach too far above the planter, or 2) the `planter` tiles, then the `planter` would have to be able to allow yellow tiles to attach exactly in all positions corresponding to halting and accepting computations, but the Turing machine being simulated accepts a language which is computably enumerable but not decidable, thus that is impossible. Thus, no DaTAM system can assemble S.

6.3.3 A DaTAM system which cannot be simulated by the aTAM

In this section, we give a single directed DaTAM system $\mathcal{D}$ at $\tau = 1$ which cannot be simulated by any aTAM system at $\tau = 2$. The fact that the aTAM at temperature 2 is incapable of simulating a single directed temperature 1 DTAS shows that the addition of duples fundamentally changes the aTAM model. The DTAS constructed in this section is similar to the system given in Section 6.3.1. See Figure 6.4 for a depiction of a producible assembly of $\mathcal{D}$. In order to show that the aTAM cannot simulate this DTAS, we use a technique used in [46]. This technique relies on the notion of a *window movie*. Please see [39] for details of window movies as they pertain to the DaTAM.

Theorem 17. There exists a single directed DaTAM system $\mathcal{D} = (T_{\mathcal{D}}, S, D, \sigma, \tau)$ such that $\mathcal{D}$ cannot be simulated by any temperature 2 aTAM system.

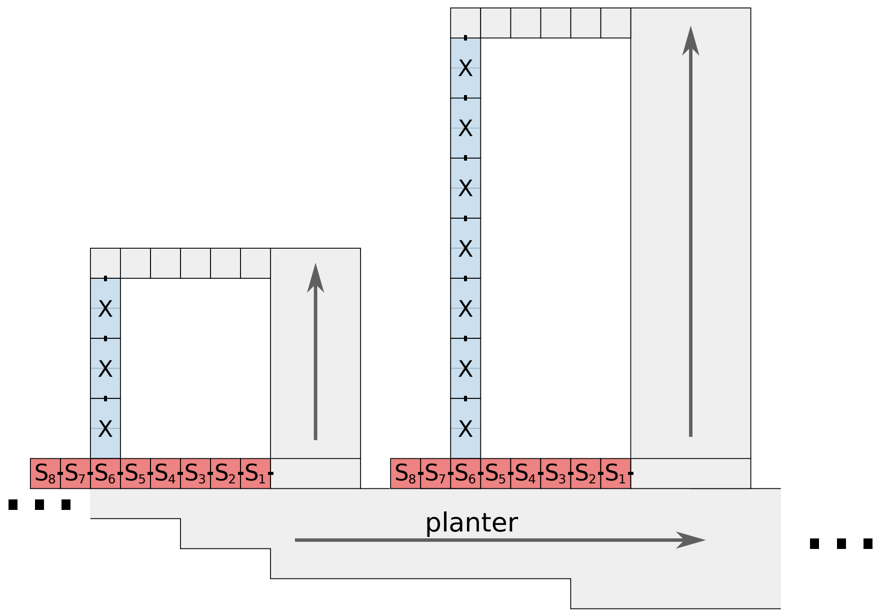


Figure 6.4: A portion of a producible assembly of the temperature 1 DaTAM system which cannot be simulated in the aTAM at $\tau = 2$.

To prove Theorem 17, we prove that there is no TAS that can simulate the DTAS, $\mathcal{D}$, described as follows. First, the system is identical to the DTAS described in Section 6.3.1 with one exception. Just as with the DTAS in Section 6.3.1, $\mathcal{D}$ grows a `planter`, vertical counters and `fingers`. In addition to these subassemblies, $\mathcal{D}$ grows an 8 tile long, single tile wide line, l, of tiles from the base of each vertical counter. As seen in Figure 6.4, l, consisting of tiles S_1, S_2, …, S_8, grows to the left of each vertical counter and extends past the single tile wide gap between a `finger` and the `planter`. The intuitive idea behind the proof of Theorem 17 is that for any aTAM system, $\mathcal{T}$, that attempts to simulate $\mathcal{D}$, it must be able to simulate the growth of `fingers`. Therefore, for any $n > 0$, $\mathcal{T}$ must be able to grow a subassembly that simulates a `finger` consisting of n duples. This subassembly must have a constant width based on the block replacement scheme used in the simulation with block size m say, and a length of roughly $2nm$. We show that any such system $\mathcal{T}$ capable of such growth must also grow a simulated `finger` that crashes into the simulation of the `planter`. To show this, we use a window movie lemma (similar to Lemma 3.1 in [46]). The lemma shown here holds for closed rectangular windows. (For details and a formal statement of the window movie lemma used here, see [39].) Then, since the simulated `planter`, `finger`, and vertical counter separate the infinite grid-graph into two disjoint sets, there is no way to ensure that a subassembly representing the tile labeled S_8 of $\mathcal{T}$ grows *only* after the subassemblies representing the tiles labeled S_1, S_2, $\ldots$, S_5 grow. In other words, there is no way to ensure that $\mathcal{T}$ and $\mathcal{D}$ have equivalent dynamics, and therefore $\mathcal{T}$ does not simulate $\mathcal{D}$. For the full proof Theorem 17, please see [39].

6.3.4 An aTAM system which cannot be simulated by the DaTAM

In Section 6.3.3 we showed the aTAM can't simulate every DaTAM system. Here we show the converse; the DaTAM can't simulate all aTAM systems. The particular aTAM system that we show can't be simulated by the DaTAM is the same given in [46] that is used to show that temperature 1 aTAM systems cannot simulate every temperature 2 aTAM system. Intuitively, this shows that cooperation, which is possible for temperature 2 aTAM systems, cannot be simulated using duples when temperature is restricted to 1. (See Figure 6.5 for the tile set.)

Theorem 18. There exists a temperature 2 aTAM system $\mathcal{T} = (T, \sigma, 2)$ such that $\mathcal{T}$ cannot be simulated by any temperature 1 DaTAM system.

Here we give a brief overview of the TAS, $\mathcal{T}$, that we show cannot be simulated by any DTAS and provide a sketch of the proof. Please see [39] for the full details.

See Figure 6.5 for an overview of the TAS $\mathcal{T}$. The proof that there is no DTAS that simulates $\mathcal{T}$ is briefly described as follows. For any DTAS, $\mathcal{D}$, that attempts to simulate $\mathcal{T}$, it is shown that $\mathcal{D}$ is capable of an invalid assembly sequence. Intuitively, the idea is that when an arm (the bottom arm say) is sufficiently long, an assembly sequence in $\mathcal{D}$ of a subassembly α that represents this arm must contain repetition. Using a window movie lemma similar to Lemma 3.3 in [46], this repetition is removed to produce an assembly in $\mathcal{D}$ that is essentially equivalent to removing a section of tiles from α and splicing together the exposed ends along matching glues. This results in a shorter arm α' that still attempts

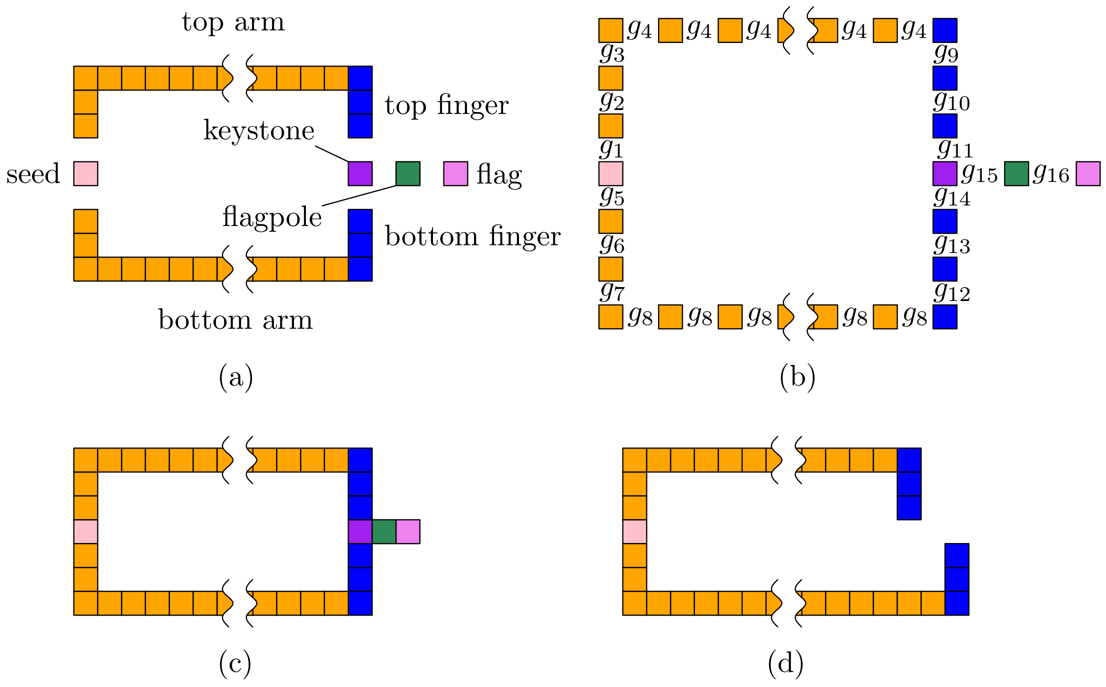


Figure 6.5: (Figure taken from [46]) (a) An overview of the tile assembly system $\mathcal{T} = (T, \sigma, 2)$. $\mathcal{T}$ runs at temperature 2 and its tile set T consists of 18 tiles. (b) The glues used in the tileset T. Glues g_{11} and g_{14} are strength 1, all other glues are strength 2. Thus the keystone tile binds with two "cooperative" strength 1 glues. Growth begins from the pink seed tile σ: the top and bottom arms are one tile wide and grow to arbitrary, nondeterministically chosen, lengths. Two blue figures grow as shown. (c) If the fingers happen to meet then the keystone, flagpole and flag tiles are placed, (d) if the fingers do not meet then growth terminates at the finger "tips".

to grow a keystone and flagpole, and hence leads to an invalid simulation of $\mathcal{T}$.

Chapter 7

DrgTAM

7.1 Introduction

The contributions of this chapter are threefold. First, we show that the rgTAM is also not capable of simulating glue cooperation. Second, we introduce the Dupled restricted glue TAM (DrgTAM) which allows for both square tiles and "duple" tiles, which are simply pre-formed pairs of 2 tiles joined along one edge before assembly begins, and it allows for glues with negative strength (i.e. those which exert repulsive force). However, it is restricted similar to the rgTAM in that the magnitude of glue strengths cannot exceed 1 (i.e. only strengths 1 and -1 are allowed). Third, we show that by creating the DrgTAM by combining two models (the rgTAM and the Dupled aTAM) which are computationally universal at temperature 1 but which cannot independently simulate glue cooperation, the result is a model which in some measures is greater than the sum of its parts. That is, the resulting DrgTAM is capable of both universal computation *and* the simulation of glue cooperation. This is the first such result for passive (i.e. non-active) tile assembly systems. In fact, we show the stronger result that there is a single tile set in the DrgTAM which can be configured to, in a temperature-1 system, simulate any arbitrary aTAM system, making it intrinsically universal for the aTAM. Coupled with the result in [39] which proves that there are temperature-1 systems in the DTAM, which are thus also in the DrgTAM, that cannot be simulated by the aTAM at temperature-2, this actually implies that the DrgTAM is more powerful than the temperature-2 aTAM.

The chapter is organized as follows. In Section 8.2 we give high-level sketches of the definitions of the models and of the concepts of simulation used throughout the chapter. In Section 7.3 we prove that rgTAM systems cannot simulate the glue cooperation of temperature 2 aTAM systems, and in Section 7.4 we present the proof that the DrgTAM can simulate the temperature-2 aTAM and in fact contains a tile set which is intrinsically universal for it.

7.2 Preliminaries

Throughout this chapter, we use three tile assembly models: 1. the aTAM, 2. the restricted glue TAM (rgTAM), and 3. the dupled rgTAM (DrgTAM). We now informally describe these models.

Informal description of the restricted glue Tile Assembly Model

The rgTAM was introduced in [52] where it was shown that the rgTAM is computationally universal even in the case where only a single glue has strength -1. The definition used in [52] and the definition given here are similar to the irreversible negative glue tile assembly model given in [21].

The restricted glue Tile Assembly Model (rgTAM) can be thought of as the aTAM where the temperature is restricted to 1 and glues may have strengths $-1, 0$, or 1. A system in the rgTAM is an ordered pair (T, σ) where T is the *tile set*, and σ is a stable *seed assembly*. We call an rgTAM system an rgTAS. *Producible* assemblies in an rgTAS can be defined recursively as follows. Let $\mathcal{T} = (T, \sigma)$ be an rgTAS. Then, an assembly α is producible in $\mathcal{T}$ if 1. $\alpha = \sigma$, 2. α is the result of a stable attachment of a single tile to a producible assembly, or 3. α is one side of a cut of strength ≤ 0 of a producible assembly.

In [21], Doty et al. give a list of the choices that can be made when defining a model with negative glues. These choices are (1) seeded/unseeded, (2) single-tile addition/two-handed assembly, (3) irreversible/reversible, (4) detachment precedes attachment/detachment and attachment in arbitrary order, (5) finite tile counts/infinite tile counts, and (6) tagged result/tagged junk. Here we have chosen the rgTAM to be a seeded, single-tile addition, irreversible model that uses infinite tiles. We also assume that attachment and detachment in the model occur in arbitrary order, however the results presented here also hold in the case where detachment precedes attachment. Finally, the definition of simulation (see Section 7.2.3) implicitly enforces a notion of tagged result and tagged junk. In particular, if detachment occurs in a simulating system, of the two resulting assemblies one contains the seed and represents some assembly in the simulated system, while the other resulting assembly must map to the empty tile.

Informal description of the Dupled restricted glue Tile Assembly Model

The DrgTAM is an extension of the rgTAM which allows for systems with square tiles as well as rectangular tiles. The rectangular tiles are 2×1 or 1×2 rectangles which can logically be thought of as two square tiles which begin pre-attached to each other along an edge, hence the name *duples*. A *DrgTAM system* (DrgTAS) is an ordered 4-tuple (T, S, D, σ) where, as in a TAS, T is a tile set and σ is a seed assembly. S is the set of singleton (i.e. square) tiles which are available for assembly, and D is the set of duple tiles. The tile types making up S and D all belong to T, with those in D each being a combination of two tile types from T.

It should be noted that the glue binding two tiles that form a duple must have strength 1, and the glues exposed by a duple may have strength -1, 0, or 1. Also notice that for an assembly α in a DrgTAS, a cut of strength ≤ 0 may separate two nodes of the grid graph that correspond to two tiles of a duple. Then, the two producible assemblies on each side of this cut each contain one tile from the duple.

We now give the formal definitions of the tile assembly models.

7.2.1 Formal description of the restricted glue Tile Assembly Model

In this section we formally define the restricted glue Tile Assembly Model (rgTAM). Since the rgTAM is based on the aTAM, most of the formal definition of Section 2.2 apply here. The rgTAM can be thought of as the aTAM where every system (rgTAS) in the rgTAM has the properties that $\tau = 1$ and glues may have strengths $-1, 0,$ or 1. A system in the rgTAM is defined as an ordered pair (T, σ) where T is a set of tile types, and σ is a stable seed assembly.

An assembly sequence in an rgTAS $\mathcal{T}$ is a (finite or infinite) sequence $\vec{\alpha} = (\alpha_0, \alpha_1, \ldots)$ of assemblies in which each α_{i+1} is obtained from α_i in one of two ways. First, α_{i+1} can obtained from α_i by the addition of a single tile such that the sum of the strengths of bound glues of this single tile in α_{i+1} is ≥ 1. Second, α_{i+1} can obtained from α_i if α_{i+1} lies on one side of a cut of α_i such that the strength of this cut is ≤ 0. Unlike an assembly sequence in the aTAM, assembly sequences in the rgTAM may not have a unique limiting assembly, and therefore, may not have a result. However, given an assembly α in an rgTAS, and an assembly sequence $\vec{\alpha}$ if the limit of $\vec{\alpha}$ is α, then we say that the *result* (denoted $\text{res}(\vec{\alpha})$) of $\vec{\alpha}$ is α. The notations used in Section 2.2 apply to the rgTAM. In addition to these notations, we distinguish between tile attachment and assemblies produced by a cut of strength ≤ 0 as follows.

$$\begin{aligned}
\alpha \to_+ \alpha' \quad &\text{iff} \quad \alpha' \text{ is obtained from } \alpha \text{ by a single stable tile addition} \\
\alpha \to_- (\alpha'_1, \alpha'_2) \quad &\text{iff} \quad \alpha'_1 \text{ and } \alpha'_2 \text{ lie on each side of a cut of } \alpha \text{ such that the} \\
&\qquad \text{strength of this cut is } \leq 0
\end{aligned}$$

7.2.2 Formal description of the Dupled restricted glues Tile Assembly Model

This section gives a formal definition of the Dupled restricted glues Tile Assembly Model (DrgTAM). First, we define the dupled aTAM (DaTAM), which is a mild extension of Winfree's abstract tile assembly model [64]. Then, as in 7.2.1, we define the DrgTAM by restricting temperature to 1 and glues strengths to -1, 0, or 1.

Given $V \subseteq \mathbb{Z}^2$, the *full grid graph* of V is the undirected graph $G^{\mathrm{f}}_V = (V, E)$, and for all $\vec{x}, \vec{y} \in V$, $\{\vec{x}, \vec{y}\} \in E \iff \|\vec{x} - \vec{y}\| = 1$; i.e., if and only if $\vec{x}$ and $\vec{y}$ are adjacent on the 2-dimensional integer Cartesian space. Fix an alphabet Σ. Σ^* is the set of finite strings over Σ. Let $\mathbb{Z}$, $\mathbb{Z}^+$, and $\mathbb{N}$ denote the set of integers, positive integers, and nonnegative integers, respectively.

A *square tile type* is a tuple $t \in (\Sigma^* \times \mathbb{N})^4$; i.e. a unit square, with four sides, listed in some standardized order, and each side having a *glue* $g \in \Sigma^* \times \mathbb{N}$ consisting of a finite string *label* and nonnegative integer *strength.* Let $T \subseteq (\Sigma^* \times \mathbb{N})^4$ be a set of tile types. We define a set of *singleton types* to be any subset $S \subseteq T$. Let

$$t = ((g_N, s_N), (g_S, s_S), (g_E, s_E), (g_W, s_W)) \in T,$$

$d \in \{N, S, E, W\} = \mathcal{D}$, and write $Glue_d(t) = g_d$ and $Strength_d(t) = s_d$. A *duple type* is defined as an element of the set $\{(x, y, d) \mid x, y \in T, \ d \in \mathcal{D}, \ Glue_d(x) = Glue_{-d}(y), \text{ and } Strength_d(x) = Strength_{-d}(y) \geq \tau\}$.

A *configuration* is a (possibly empty) arrangement of tiles on the integer lattice $\mathbb{Z}^2$, i.e., a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$. Two adjacent tiles in a configuration *interact*, or are *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* G^{b}_α, a grid graph whose vertices are positions occupied by tiles, according to α, with an edge between two vertices if the tiles at those vertices interact. An *assembly* is a connected, non-empty configuration, i.e., a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$ such that $G^{\mathrm{f}}_{\mathrm{dom}\ \alpha}$ is connected and dom $\alpha \neq \varnothing$. The *shape* $S_\alpha \subseteq \mathbb{Z}^d$ of α is dom α. Let α be an assembly and $B \subseteq \mathbb{Z}^2$. α *restricted to* B, written as $\alpha \upharpoonright B$, is the unique assembly satisfying $(\alpha \upharpoonright B) \sqsubseteq \alpha$, and dom $(\alpha \upharpoonright B) = B$

Given $\tau \in \mathbb{Z}^+$, α is τ-*stable* if every cut of G^{b}_α has weight at least τ, where the weight of an edge is the strength of the glue it represents. When τ is clear from context, we say α is *stable.* Given two assemblies α, β, we say α is a *subassembly* of β, and we write $\alpha \sqsubseteq \beta$, if $S_\alpha \subseteq S_\beta$ and, for all points $p \in S_\alpha$, $\alpha(p) = \beta(p)$. Let $\mathcal{A}^T$ denote the set of all assemblies of tiles from T, and let $\mathcal{A}^T_{<\infty}$ denote the set of finite assemblies of tiles from T.

A *dupled tile assembly system* (DTAS) is a tuple $\mathcal{T} = (T, S, D, \sigma, \tau)$, where T is a finite tile set, $S \subseteq T$ is a finite set of singleton types, D is a finite set of duple tile types, $\sigma : \mathbb{Z}^2 \dashrightarrow T$ is the finite, τ-stable, *seed assembly*, and $\tau \in \mathbb{Z}^+$ is the *temperature.*

Given two τ-stable assemblies α, β, we write $\alpha \to^{\mathcal{T}}_1 \beta$ if $\alpha \sqsubseteq \beta$, $0 < |S_\beta \setminus S_\alpha| \leq 2$. In this case we say α $\mathcal{T}$*-produces* β *in one step.* The $\mathcal{T}$*-frontier* of α is the set $\partial^{\mathcal{T}} \alpha = \bigcup_{\alpha \to^{\mathcal{T}}_1 \beta} S_\beta \setminus S_\alpha$, the set of empty locations at which a tile could stably attach to α.

A sequence of $k \in \mathbb{Z}^+ \cup \{\infty\}$ assemblies $\alpha_0, \alpha_1, \ldots$ over $\mathcal{A}^T$ is 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 (for a finite sequence, this is the final assembly in the sequence). If $\vec{\alpha} = (\alpha_0, \alpha_1, \ldots)$ is an assembly sequence in $\mathcal{T}$ and $\vec{m} \in \mathbb{Z}^2$, then the *$\vec{\alpha}$-index* of $\vec{m}$ is $i_{\vec{\alpha}}(\vec{m})$ $=\min\{i \in \mathbb{N} | \vec{m} \in \text{dom } \alpha_i\}$. That is, the $\vec{\alpha}$-index of $\vec{m}$ is the time at which any tile is first placed at location $\vec{m}$ by $\vec{\alpha}$. For each location $\vec{m} \in \bigcup_{0 \leq i \leq l} \text{dom } \alpha_i$, define the set of its input sides $\text{IN}^{\vec{\alpha}}(\vec{m}) = \{\vec{u} \in U_2 | \text{str}_{\alpha_{i_\alpha(\vec{m})}}(\vec{u}) > 0\}$.

We write $\alpha \to^{\mathcal{T}} \beta$, and we say α *$\mathcal{T}$-produces* β (in 0 or more steps) if there is a $\mathcal{T}$-assembly sequence $\alpha_0, \alpha_1, \ldots$ of length k such that (1) $\alpha = \alpha_0$, (2) $S_\beta = \bigcup_{0 \leq i < k} S_{\alpha_i}$, and (3) for all $0 \leq i < k$, $\alpha_i \sqsubseteq \beta$. If k is finite then it is routine to verify that $\beta = \alpha_{k-1}$.

We say α is *$\mathcal{T}$-producible* if $\sigma \to^{\mathcal{T}} \alpha$, and we write $\mathcal{A}[\mathcal{T}]$ to denote the set of $\mathcal{T}$-producible assemblies. An assembly α is *$\mathcal{T}$-terminal* if α is τ-stable and $\partial^{\mathcal{T}} \alpha = \varnothing$. We write $\mathcal{A}_\Box[\mathcal{T}] \subseteq \mathcal{A}[\mathcal{T}]$ to denote the set of $\mathcal{T}$-producible, $\mathcal{T}$-terminal assemblies. If $|\mathcal{A}_\Box[\mathcal{T}]| = 1$ then $\mathcal{T}$ is said to be *directed*.

We say that a DTAS $\mathcal{T}$ *strictly (a.k.a. uniquely) self-assembles* a shape $X \subseteq \mathbb{Z}^2$ if, for all $\alpha \in \mathcal{A}_\Box[\mathcal{T}]$, $S_\alpha = X$; i.e., if every terminal assembly produced by $\mathcal{T}$ places tiles on – and only on – points in the set X.

Now, the DrgTAM is defined a in Section 7.2.1 and a DrgTAS is defined to be a system in the DrgTAM. Note that the glue binding two tiles that form a duple must have strength 1, and the glues exposed by a duple may have strength -1, 0, or 1. Also notice that for an assembly α in a DrgTAS, a cut of strength ≤ 0 may separate two nodes of the grid graph that correspond to two tiles of a duple. Then, the two producible assemblies on each side of this cut each contain one tile from the duple.

7.2.3 Informal Definitions of Simulation

In this section, we present a high-level sketch of what we mean when saying that one system *simulates* another. Please see [38] for complete, technical definitions, which are based on those of [46].

For one system $\mathcal{S}$ to simulate another system $\mathcal{T}$, we allow $\mathcal{S}$ to use square (or rectangular when simulating duples) blocks of tiles called *macrotiles* to represent the simulated tiles from $\mathcal{T}$. The simulator must provide a scaling factor c which specifies how large each macrotile is, and it must provide a *representation function*, which is a function mapping each macrotile assembled in $\mathcal{S}$ to a tile in $\mathcal{T}$. Since a macrotile may have to grow to some critical size (e.g. when gathering information from adjacent macrotiles about the simulated glues adjacent to its location) before being able to compute its identity (i.e. which tile from $\mathcal{T}$ it represents), it's possible for non-empty macrotile locations in $\mathcal{S}$ to map to empty locations in $\mathcal{T}$, and we call such growth *fuzz*. We follow the standard simulation definitions (see [46, 19, 34, 22]), and restrict fuzz to being laterally or vertically adjacent to macrotile positions in $\mathcal{S}$ which map to non-empty tiles in $\mathcal{T}$.

Given the notion of block representations, we say that $\mathcal{S}$ simulates $\mathcal{T}$ if and only if (1) for every producible assembly in $\mathcal{T}$, there is an equivalent producible assembly in $\mathcal{S}$ when the representation function is applied, and vice versa (thus we say the systems have *equivalent productions*), and (2) for every assembly sequence in $\mathcal{T}$, the exactly equivalent assembly sequence can be followed in $\mathcal{S}$ (modulo the application of the representation function), and vice versa (thus we say the systems have *equivalent dynamics*). Thus, equivalent production and equivalent dynamics yield a valid simulation.

We say that a tile set U is *intrinsically universal* for a class $\mathfrak{C}$ of tile assembly systems if, for every system $\mathcal{T} \in \mathfrak{C}$ a system $\mathcal{U}_{\mathcal{T}}$ can be created for which: 1. U is the tile set, 2. there is some initial seed assembly consisting of tiles in U which is constructed to encode information about the system $\mathcal{T}$ being simulated, 3. there exists a representation function R which maps macrotiles in the simulator $\mathcal{U}_{\mathcal{T}}$ to tiles in the simulated system, and 4. under $R, \mathcal{U}_{\mathcal{T}}$ has equivalent productions and equivalent dynamics to $\mathcal{T}$. Essentially, there is one tile set which can be used to simulate any system in the class, using only custom configured input seed assemblies. For formal definitions of intrinsic universality in tile assembly, see [22, 46, 34].

7.3 A temperature-2 aTAM system that cannot be simulated by any rgTAS

In this section we show that there exists a temperature-2 aTAM system that cannot be simulated by any rgTAM system. Here we give an overview of the TAS, $\mathcal{T}$, that we show cannot be simulated by any rgTAS, and an overview of the proof. For sake of brevity, more rigorous details of the following proof can be found in [38].

Theorem 19. There exists a temperature-2 aTAM system $\mathcal{T} = (T, \sigma, 2)$ such that $\mathcal{T}$ cannot be simulated by any rgTAS.

Let $\mathcal{T} = (T, \sigma, 2)$ denote the system with T and σ given in Figure 6.5. The glues in the various tiles are all unique with the exception of the common east-west glue type used within each arm to induce non-deterministic and independent arm lengths. Glues are shown in part (b) of Figure 6.5. Note that cooperative binding happens at most once during growth, when attaching the keystone tile to two arms of identical length. All other binding events are noncooperative and all glues are strength 2 except for g_{11}, g_{14} which are strength 1.

The TAS $\mathcal{T}$ was used in [46] to show that there is a temperature-2 aTAM system that cannot be simulated by a temperature-1 aTAM system. To prove that there is no rgTAS that simulates $\mathcal{T}$, we use a similar proof to the proof for aTAM systems, however, we must take special care to show that allowing for a single negative glue does not give enough strength to the model to allow for simulation of cooperative glue binding.

The proof is by contradiction. Suppose that $\mathcal{S} = (S, \sigma_S)$ is an rgTAS that simulates $\mathcal{T}$. We call an assembly sequence $\vec{\alpha} = (\alpha_0, \alpha_1, \dots)$ in an rgTAS *detachment free* if for all $i \geq 0$, α_{i+1} is obtained from α_i by the stable attachment of a single tile. The following lemma gives sufficient conditions for the existence of a detachment free assembly sequence.

Lemma 11. Let $\mathcal{S} = (S, \sigma_S)$ be an rgTAS and let $\alpha \in \mathcal{A}[\mathcal{S}]$ be a finite stable assembly. Furthermore, let β be a stable subassembly of α. Then there exists a detachment free assembly sequence $\vec{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_n)$ such that $\alpha_1 = \beta$, and $\alpha_n = \alpha$.

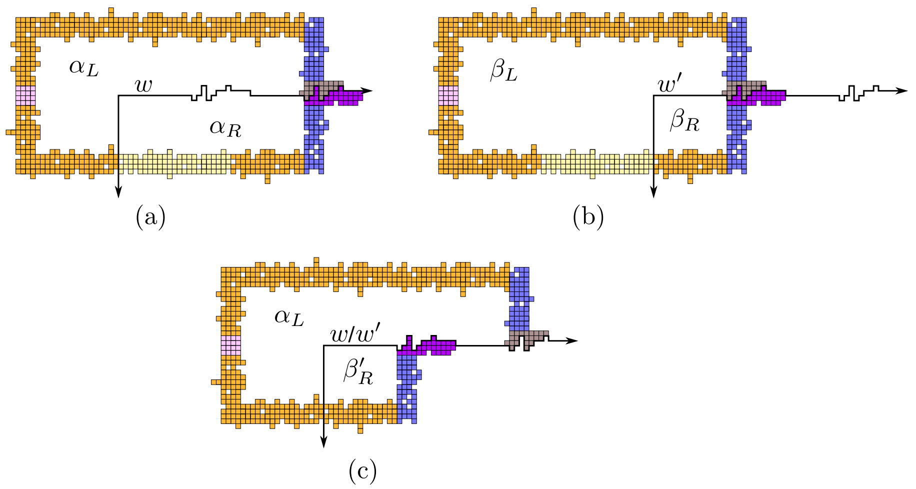


Figure 7.1: An example assembly formed by S simulating $\mathcal{T}$ – (a) and (b), and the resulting producible assembly (c) constructed via a "splicing" technique that uses the window movie lemma. The assembly in (c) shows that $\mathcal{S}$ is incapable of valid simulation of $\mathcal{T}$.

A corollary of this lemma is that if an rgTAS gives a valid simulation of $\mathcal{T}$, it can do so using detachment free assembly sequences. Using detachment free assembly sequences, it is possible to use a technique for "splicing" subassemblies of producible assemblies of $\mathcal{S}$. This technique uses a lemma referred to as the "window movie lemma". For aTAM systems, this lemma is shown in [46] (Lemma 3.1). We give a version of the window movie lemma that holds for detachment free assembly sequences. See [38] for the formal definitions of windows and window movies, and for a formal statement of the window movie lemma that we use. Figure 7.1 gives a depiction of this splicing technique. Here we use this lemma for detachment free assembly in the rgTAM. Then, using this splicing technique, we show that if $\mathcal{S}$ can simulate $\mathcal{T}$, it can also produce assemblies that violate the definition of simulation. In other words, we arrive at our contradiction and conclude that there is no rgTAS that can simulate $\mathcal{T}$.

7.4 Simulation of the aTAM with the DrgTAM

In this section, given an aTAM system $\mathcal{T} = (T, \sigma, 2)$, we describe how to simulate $\mathcal{T}$ with a DrgTAS at temperature 1 with $O(1)$ scale factor and tile complexity $O(|T|)$. It will then follow from [22] that there exists a tile set in the DrgTAM at $\tau = 1$ which is intrinsically universal for the aTAM at any temperature, i.e. it can be used to simulate any aTAM system of any temperature.

Theorem 20. For every aTAM system $\mathcal{T} = (T, \sigma, 2)$, there exists a DrgTAS

$$\mathcal{D} = (T_{\mathcal{D}}, S, D, \sigma', 1)$$

such that $\mathcal{D}$ simulates $\mathcal{T}$ with $O(1)$ scale factor and $|S \cup D| = O(|T|)$.

We now provide a high-level overview of the construction. For the remainder of this section, $\mathcal{T} = (T, \sigma, 2)$ will denote an arbitrary TAS being simulated, $\mathcal{D} = (T_{\mathcal{D}}, S, D, \sigma', 1)$ the simulating DrgTAS, and R the representation function which maps blocks of tiles in $\mathcal{D}$ to tiles in $\mathcal{T}$. The system $\mathcal{T}$ is simulated by a DrgTAS through the use of macrotiles which consist of the components shown in Figure 7.2. Note that macrotiles are not necessarily composed of all of the components shown in Figure 7.2, but will consist of at least one of the subassemblies labeled probe. Informally, the subassemblies labeled probe, which we

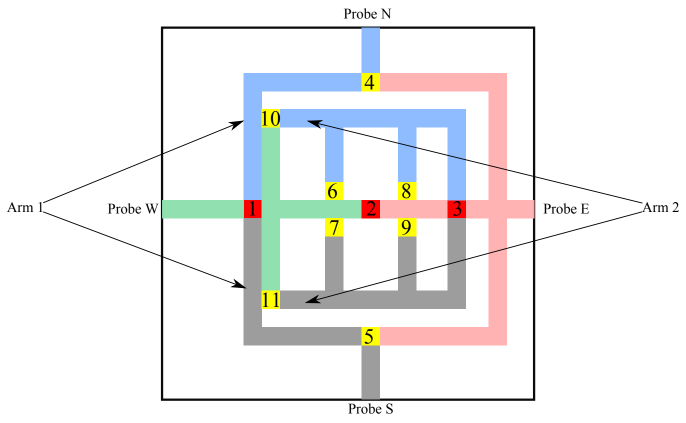


Figure 7.2: Macrotile probes, points of cooperation, and points of competition

will now refer to as probes, "simulate" the glues of the tiles in T. If a probe is simulating a glue which is of strength 2, then it does not require the assistance of any other probes in order to complete the macrotile containing it. On the other hand, if the glue which the probe is simulating is of strength 1, then the probe cannot assemble a new macrotile until another probe arrives which simulates a glue with which the other glue can cooperate and place a new tile in $\mathcal{T}$. Before probes can begin the growth of a new macrotile, they must claim (i.e. place a tile in) one of the *points of competition* (shown as red in Figure 7.2) depending on the configuration of the macrotile. Once a special tile is placed in one of the points of competition, the representation function R maps the macrotile to the corresponding tile in T, and the growth of the macrotile can begin.

We use the following conventions for our figures. All duples are shown in darker colors (even after they are broken apart) and singletons are shown in lighter colors. Negative glues are represented by red squares protruding from tiles, and positive glues are represented by all other colored squares protruding from tiles. We represent glue mismatches (a glue mismatch occurs when two different glues are adjacent or a glue is adjacent to a tile side that does not have a glue) by showing the mismatching glues receded into the tiles from which they would normally protrude. A red box enclosing a subassembly indicates that subassembly has total binding strength 0.

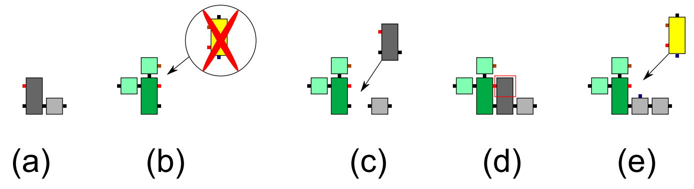


Figure 7.3: An assembly sequence of an adjacent cooperator gadget.

The cooperator gadget is the underlying mechanism that allows for the DrgTAM to simulate the cooperative placement of a tile in a $\tau \geq 2$ TAS. We consider two cases of cooperative tile placement: 1) the tiles that cooperatively contribute to the placement of a tile have adjacent corners (e.g. one is north of the location to be cooperatively tiled while the other is to the east or west), and 2) the tiles that cooperatively contribute to the placement of a tile are non-adjacent, that is there is a tile wide gap between the two tiles. We create a cooperator gadget for each of these two cases. Not surprisingly, we call the cooperator gadget that mimics the former case the *adjacent cooperator gadget* and the cooperator gadget that mimics the latter case the *gap cooperator gadget*. Each of these two gadgets is asymmetric in nature and consists of two parts: 1) a finger and 2) a resistor. The function of the resistor is to cause a duple that is attached to the finger gadget to break apart and expose the internal glue of the duple which can then be used for binding of another tile.

An adjacent cooperator gadget is shown in Figure 7.3. Part (a) of this figure depicts the finger part of the gadget, and the subassembly labeled (b) is the resistor. Note that the only tiles which have the ability to bind to the exposed glues are duples with a negative glue that is aligned with the negative glue that is adjacent to the exposed glues. This means that neither subassembly can grow any further until its counterpart arrives. In Figure 7.3 parts (c) - (e) we see the assembly sequence showing the interaction between the two parts of the cooperator gadget. In this particular assembly sequence we have assumed that the resistor piece of the gadget has arrived first. In part (c), we see the arrival of a tile (presumably from a probe) which allows for the duple that is a part of the finger gadget to bind with total strength 1. The 0 strength cut that is induced by this binding event is shown by the red box in part (d) of the figure. Since the tile encapsulated in the red box is bound with total strength 0, it eventually detaches which leads us to part (e) of the figure. Notice that the dissociation event has caused a new glue to be exposed. This glue now allows for the binding of a duple as shown in part (e) of Figure 7.3.

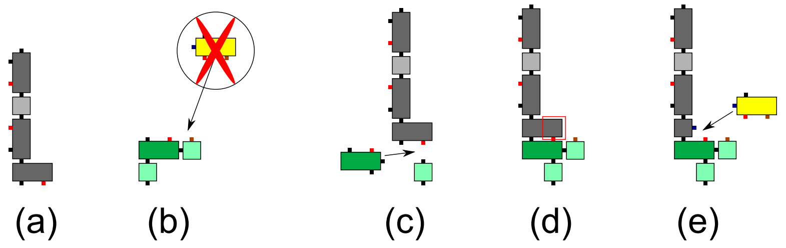


Figure 7.4: An assembly sequence of a gap cooperator gadget.

Figure 7.4 shows a gap cooperator gadget which is a simple extension of the adjacent cooperator gadget. This extension of the adjacent cooperator gadget allows for a crosser gadget (described below) to grow a path of tiles in between the two parts of the gadget. This gadget allows a new glue to be exposed upon the arrival of a negative glue (Figure 7.4 part (c)) which causes half of the duple to detach (shown in part (d) of the figure). This allows a duple to attach as shown in Figure 7.4(e) which depends on both of the glues exposed by the two parts of the gadget. Notice that the binding of this tile cannot occur unless both parts of the gadget are present.

The previous gadgets showed that in order for two probes to cooperate, they must be connected by a path of tiles. In order for other probes to cross in between these connected probes we utilize what we call a “crosser gadget”. The assembly sequence for a crosser is shown in Figure 7.5. Growth of the gadget begins with the placement of a singleton which is prevented from growing further. This singleton exposes glues which allow for duples to bind (Figure 7.5(b) and (c)) that cause the path of tiles blocking the singleton’s growth to detach (Figure 7.5(d)). Note that the attachment of these duples cannot occur before the singleton arrives since they would only have total binding strength zero. [38] offers a more in-depth description of the gadgets described above.

We can now use these gadgets to give a more complete description of the probes which are shown in Figure 7.2. All of the numbered regions represent gadgets. Gadgets labeled 1-3 in the figure represent gap cooperator gadgets which allow for cooperation between the probes to which they are attached. The gadgets labeled 5-9 denote adjacent cooperator gadgets which allow for the potential of cooperation between the probes to which they are attached. Finally, the gadgets labeled 10 and 11 are cooperator gadgets which allow for Probe W to trigger the growth of the second arms of Probe N and Probe S. See [38] for more details about the structure of probes and their accompanying gadgets.

The output of the representation function for a particular macrotile depends on the three regions labeled 1-3 in Figure 7.2. If a special tile is placed in region 1, then the macrotile region is mapped to the tile in T that corresponds to the special tile regardless of the tiles in the other regions. Similarly, region 3 takes precedence over region 2. Finally, if a spe-

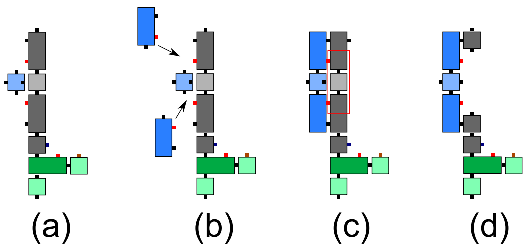


Figure 7.5: An assembly sequence of a crosser gadget.

cial tile has not been placed in either region 1 or 3, then the output of the representation function depends on the tile placed in region 2.

The seed of our simulator is formed from a set of tiles in $S \cup D$ which have been hard-coded. For a more detailed explanation of the representation function and regions 1-3 see, a case analysis of how our construction handles all possible binding scenarios, and a more detailed explanation about the construction of the seed in the simulator, see [38].

Corollary 4. There exists a DrgTAM tile set U which, at temperature-1, is intrinsically universal for the aTAM. Furthermore, the sets of singletons and duples, S and D, created from U are constant across all simulations.

As mentioned above this result follows from [22]. See [38] for more details.

Chapter 8

Simulating Active Tiles with 3-D Static Tiles

8.1 Introduction

A newly developed model, the Signal-passing Tile Assembly Model (STAM) [50], is based upon the 2HAM but with a powerful and important difference. Tiles in the aTAM and 2HAM are static, unchanging building blocks which can be thought of as analogous to write-once memory, where a location can change from empty to a particular value once and then never change again. Instead, the tiles of the STAM each have the ability to undergo some bounded number of transformations as they bind to an assembly and while they are connected. Each transformation is initiated by the binding event of a tile's glue, and consists of some other glue on that tile being turned either "on" or "off". By chaining together sequences of such events which propagate across the tiles of an assembly, it is possible to send "signals" which allow the assembly to adapt during growth. Since the number of transitions that any glue can make is bounded, this doesn't provide for "fully reusable" memory, but even with the limited reuse it has been shown that the STAM is more powerful than static models such as the aTAM and 2HAM (in 2D), for instance being able to strictly self-assemble the Sierpinski triangle [50]. A very important feature of the STAM is its asynchronous nature, meaning that there is no timeframe during which signals are guaranteed to fully propagate, and no guaranteed ordering to the arrival of multiple signals. Besides providing a useful theoretical framework of asynchronous behavior, the design of the STAM was carefully aligned to the physical reality of implementation by DNA tiles using cascades of strand-displacement. Capabilities in this area are improving, and now include the linear transmission of signals, where one glue binding event can activate one other glue on a DNA tile [49].

Although the STAM is intended to provide both a powerful theoretical framework and a solid basis for representing possible physical implementations, often those two goals are at odds. In fact, in the STAM it is possible to define tiles which have arbitrary *signal com-*

plexity in terms of the numbers of glues that a tile may have on any given side and the number of signals that each tile can initiate. Clearly, as the signal complexity of tiles increase, the ease of making these tiles in the laboratory diminishes. Therefore, in this chapter, our first set of results provide a variety of methods for simplifying the tiles in STAM systems. Besides reducing just the general signal complexity of tiles, we also seek to reduce and/or remove certain patterns of signals which may be more difficult to build into DNA-based tiles, namely *fan-out* (which occurs when a single signal must split into multiple paths and have multiple destinations), *fan-in* (which occurs when multiple signals must converge and join into one path to arrive at a single glue), and *mutual activation* (which occurs when both of the glues participating in a particular binding event initiate their own signals). By trading signal complexity for tile complexity and scale factor, we show how to use some simple primitive substitutions to reduce STAM tile sets to those with much simpler tiles. Note that while in the general STAM it is possible for signals to turn glues both "on" and "off", our results pertain only to systems which turn glues "on" (which we call $STAM^+$ systems).

In particular, we show that the tile set for any temperature 1 STAM$^+$ system, with tiles of arbitrary complexity, can be converted into a temperature 1 STAM$^+$ system with a tile set where no tile has greater than 2 signals and either fan-out or mutual activation, but not both, are completely eliminated. We show that any temperature 2 STAM$^+$ system can be converted into a temperature 2 STAM$^+$ system where no tile has greater than 1 signal and both fan-out and mutual activation are eliminated. Importantly, while both conversions have a worst case scale factor of $|T^2|$, where T is the tile set of the original system, and worst case tile complexity of $|T^2|$, those bounds are required for the extremely unrealistic case where *every* glue is on *every* edge of some tile and also sends signals to *every* glue on *every* side of that tile. Converting from a more realistic tile set yields factors which are on the order of the square of the maximum signal complexity for each side of a tile, which is typically much smaller. Further, the techniques used to reduce signal complexity and remove fan-out and mutual activation are likely to be useful in the original design of tile sets rather than just as brute force conversions of completed tile sets.

We next consider the topic of intrinsic universality, which was initially developed to aid in the study of cellular automata [14, 16]. The notion of intrinsic universality was designed to capture a strong notion of simulation, in which one particular automaton is capable of simulating the *behavior* of any automaton within a class of automata. Furthermore, to simulate the behavior of another automaton, the simulating automaton must evolve in such a way that a translated rescaling (rescaled not only with respect to rectangular blocks of cells, but also with respect to time) of the simulator can be mapped to a configuration of the simulated automaton. The specific rescaling depends on the simulated automaton and gives rise to a global rule such that each step of the simulated automaton's evolution is mirrored by the simulating automaton, and vice versa via the inverse of the rule.

In this way, it is said that the simulator captures the dynamics of the simulated system, acting exactly like it, modulo scaling. This is in contrast to a computational simulation, for example when a general purpose digital computer runs a program to simulate a cellular automata while the processor's components don't actually arrange themselves as, and behave like, a grid of cellular automata. In [22], it was shown that the aTAM is intrinsically universal, which means that there is a single tile set U such that, for any aTAM tile assembly system $\mathcal{T}$ (of any temperature), the tiles of U can be arranged into a seed structure dependent upon $\mathcal{T}$ so that the resulting system (at temperature 2), using only the tiles from U, will faithfully simulate the behaviors of $\mathcal{T}$. In contrast, in [19] it was shown that no such tile set exists for the 2HAM since, for every temperature, there is a 2HAM system which cannot be simulated by any system operating at a lower temperature. Thus no tile set is sufficient to simulate 2HAM systems of arbitrary temperature.

For our main result, we show that there is a 3-D 2HAM tile set U which is intrinsically universal (IU) for the class $\mathfrak{C}$ of all STAM$^+$ systems at temperature 1 and 2. For every $\mathcal{T} \in \mathfrak{C}$, a single input supertile can be created, and using just copies of that input supertile and the tiles from U, at temperature 2 the resulting system will faithfully simulate $\mathcal{T}$. Furthermore, the simulating system will use only 2 planes of the third dimension. (The signal tile set simplification results are integral in the construction for this result, especially in allowing it to use only 2 planes.) This result is noteworthy especially because it shows that the dynamic behavior of signal tiles (excluding glue deactivation) can be *fully duplicated* by static tile systems which are allowed to "barely" use three dimensions. Furthermore, for every temperature $\tau > 1$ there exists a 3-D 2HAM tile set which can simulate the class of all STAM$^+$ systems at temperature τ. Note that this version of the chapter is an extension of [34], and here we present the additional results that, for every $\tau \geq 1$ there exists a tile set U_τ which is IU for the STAM$^+$ at temperature τ. However, due to space constraints the majority of construction and proof details, as well as formal definitions are omitted from this version and can instead be found in [30].

8.2 Preliminaries

Here we provide descriptions of the models and terms used in this chapter.

8.2.1 Informal description of the STAM

In the STAM, tiles are allowed to have sets of glues on each edge (as opposed to only one glue per side as in the TAM and 2HAM). Tiles have an initial state in which each glue is either "`on`" or "`latent`" (i.e. can be switched `on` later). Glues can later be switched "`off`" in response to glue binding events. Tiles each implement a transition function which is executed upon the binding of any glue on any edge of that tile. The transition function specifies, for each glue g on a tile, a set of glues (along with the sides on which those glues are located) and an action, or *signal* which is *fired* by g's binding, for each glue in the set. The actions specified may be to: 1. turn the glue `on` (only valid if it is currently `latent`), or 2. turn the glue `off` (valid if it is currently `on` or `latent`). This means that glues can only be `on` once (although may remain so for an arbitrary amount of time or permanently), either by starting in that state or being switched `on` from `latent` (which we call *activation*), and if they are ever switched to `off` (called *deactivation*) then no further transitions are allowed for that glue. This essentially provides a single "use" of a glue and the signal sent by its binding. Note that turning a glue `off` breaks any bond that that glue may have formed with a neighboring tile. Also, since tile edges can have multiple active glues, when tile edges with multiple glues are adjacent, it is assumed that all matching glues in the `on` state bind (for a total binding strength equal to the sum of the strengths of the individually bound glues). The transition function defined for each tile type is allowed a unique set of output actions for the binding event of each glue along its edges, meaning that the binding of any particular glue on a tile's edge can initiate a set of actions to turn an arbitrary set of the glues on the sides of the same tile either `on` or `off`.

As the STAM is an extension of the 2HAM, binding and breaking can occur between tiles contained in pairs of arbitrarily sized supertiles. In order to allow for physical mechanisms which implement the transition functions of tiles but are arbitrarily slower or faster than the average rates of (super)tile attachments and detachments, rather than immediately enacting the outputs of transition functions, each output action is put into a set of "pending actions" which includes all actions which have not yet been enacted for that glue (since it is technically possible for more than one action to have been initiated, but not yet enacted, for a particular glue). Any event can be randomly selected from the set, regardless of the order of arrival in the set, and the ordering of either selecting some action from the set or the combination of two supertiles is also completely arbitrary. This provides fully asynchronous timing between the initiation, or firing, of signals (i.e. the execution of the transition function which puts them in the pending set) and their execution (i.e. the changing of the state of the target glue), as an arbitrary number of supertile binding events may occur before any signal is executed from the pending set, and vice versa.

An STAM system consists of a set of tiles and a temperature value. To define what is producible from such a system, we use a recursive definition of producible assemblies which starts with the initial tiles and then contains any supertiles which can be formed by doing the following to any producible assembly: 1. executing any entry from the pending actions of any one glue within a tile within that supertile (and then that action is removed from the pending set), 2. binding with another supertile if they are able to form a τ-stable supertile, or 3. breaking into 2 separate supertiles along a cut whose total strength is $< \tau$.

The STAM, as formulated, is intended to provide a model based on experimentally plausible mechanisms for glue activation and deactivation. However, while the model allows for the placement of an arbitrary number of glues on each tile side and for each of them to signal an arbitrary number of glues on the same tile, this is currently limited in practice. Therefore, each system can be defined to take into account a desired threshold for each of those parameters, not exceeding it for any given tile type, and so we have defined the notion of *full-tile signal complexity* as the maximum number of signals on any tile in a set (see [30]) to capture the maximum complexity of any tile in a given set.

Definition. We define the *STAM*$^+$ to be the STAM restricted to using only glue activation, and no glue deactivation. Similarly, we say an STAM$^+$ tile set is one which contains no defined glue deactivation transitions, and an STAM$^+$ system $\mathcal{T} = (T, \tau)$ is one in which T is an STAM$^+$ tile set.

As the main goal of this chapter is to show that self-assembly by systems using active, signalling tiles can be simulated using the static, unchanging tiles of the 3D 2HAM, since they have no ability to break apart after forming τ-stable structures, all of our results are confined to the STAM$^+$.

8.2.2 Informal definitions for simulation

Here we informally describe what it means for one 2HAM or STAM TAS to "simulate" another. Formal definitions, adapted from those of [19], can be found in [30].

Let $\mathcal{U} = (U, S_U, \tau_U)$ be the system which is simulating the system $\mathcal{T} = (T, S_T, \tau_T)$. There must be some scale factor $c \in \mathbb{N}$ at which $\mathcal{U}$ simulates $\mathcal{T}$, and we define a *repre-*

sentation function R which maps each $c \times c$ square (sub)assembly in $\mathcal{U}$ to a tile in $\mathcal{T}$ (or empty space if it is incomplete). Each such $c \times c$ block is referred to as a *macrotile*, since that square configuration of tiles from set U represent a single tile from set T. We say that $\mathcal{U}$ simulates $\mathcal{T}$ under representation function R at scale c.

To properly simulate $\mathcal{T}$, $\mathcal{U}$ must have 1. *equivalent productions*, meaning that every supertile producible in $\mathcal{T}$ can be mapped via R to a supertile producible in $\mathcal{U}$, and vice versa, and 2. *equivalent dynamics*, meaning that when any two supertiles α and β, which are producible in $\mathcal{T}$, can combine to form supertile γ, then there are supertiles producible in $\mathcal{U}$ which are equivalent to α and β which can combine to form a supertile equivalent to γ, and vice versa. Note that especially the formal definitions for equivalent dynamics include several technicalities related to the fact that multiple supertiles in $\mathcal{U}$ may map to a single supertile in $\mathcal{T}$, among other issues. Please see [30] for details.

We say that a tile set U is *intrinsically universal* for a class of tile assembly systems if, for every system in that class, a system can be created for which 1. U is the tile set, 2. there is some initial configuration which consists of supertiles created from tiles in U, where those "input" supertiles are constructed to encode information about the system being simulated, and perhaps also singleton tiles from U, 3. a representation function which maps macrotiles in the simulator to tiles in the simulated system, and 4. under that representation function, the simulator has equivalent productions and equivalent dynamics to the simulated system. Essentially, there is one tile set which can simulate any system in the class, using only custom configured input supertiles.

8.3 Formal definitions for simulation

In this section, we formally define what it means for one 2HAM or STAM TAS to "simulate" another 2HAM or STAM TAS. Therefore, all tilesets may be either 2HAM or STAM tile sets, and supertiles refer to active supertiles when built from STAM tile sets.

For a tileset T, let A^T and $\tilde{A}^T$ denote the set of all assemblies over T and all supertiles over T respectively. Let $A^T_{<\infty}$ and $\tilde{A}^T_{<\infty}$ denote the set of all finite assemblies over T and all finite supertiles over T respectively.

In what follows, let U be a d-dimensional tile set and let $m \in \mathbb{Z}^+$. An *m-block assembly*, or *macrotile*, over tile set U is a partial function $\gamma : \mathbb{Z}^d_m \dashrightarrow U$, where $\mathbb{Z}_m = \{0, 1, \dots m-1\}$. Note that the dimension of the m-block is implicitly defined by U. Let B^U_m be the set of all m-block assemblies over U. The m-block with no domain is said to be *empty*. For an arbitrary assembly $\alpha \in A^U$ and $(x_0, \dots x_{d-1}) \in \mathbb{Z}^d$, define $\alpha^m_{x_0,\dots x_{d-1}}$ to be the m-block supertile defined by $\alpha^m_{x_0,\dots,x_{d-1}}(i_0, \dots, i_{d-1}) = \alpha(mx_0 + i_0, \dots, mx_{d-1} + i_{d-1})$ for $0 \leq i_0, \dots, i_{d-1} < m$.

For some tile set T of dimension $d' \leq d$, where $d \in \{2, 3\}$ and $d' \in \{d-1, d\}$, and a partial function $R : B^U_m \dashrightarrow T$, define the *assembly representation function* $R^* : A^U \dashrightarrow A^T$ such that $R^*(\alpha) = \beta$ if and only if $\beta(x_0, ..., x_{d'-1}) = R(\alpha^m_{x_0,...,x_{d-1}})$ for all $(x_0, ..., x_{d-1}) \in \mathbb{Z}^{d-1}$. Let $f : \mathbb{Z}^d \to \mathbb{Z}^{d'}$, where $f(x_0, \dots, x_{d-1}) = (x_0, \dots, x_{d-1})$ if $d = d'$ and $f(x_0, \dots, x_{d-1}) = (x_0, \dots, x_{d'-1}, 0)$ if $d' = d - 1$, and undefined otherwise. If either U or T is an STAM tile set, note that the assembly representation function does not consider the pending sets Π of any constituent tiles, and thus treats active supertiles with different pending sets but otherwise identical as identical supertiles. α is said to map *cleanly* to β under R^* if for all non empty blocks $\alpha'^m_{x_0,\dots,x_{d-1}}$, $(f(x_0, \dots, x_{d-1}) + f(u_0, \dots, u_{d-1})) \in \text{dom } \alpha$ for some $u_0, \dots, u_{d-1} \in \{-1, 0, 1\}$ such that $u_0^2 + \dots + u_{d-1}^2 \leq 1$, or if α' has at most one non-empty m-block $\alpha^m_{0,\dots,0}$. In other words, α' may have tiles on supertile blocks representing empty space in α, but only if that position is adjacent to a tile in α. We call such growth "around the edges" of α' *fuzz* and thus restrict it to be adjacent to only valid macrotiles, but not diagonally adjacent (i.e. we do not permit *diagonal fuzz*).

For a given assembly representation function R^*, define the *supertile representation function* $\tilde{R} : \tilde{A}^U \dashrightarrow \mathcal{P}(A^T)$ such that $\tilde{R}(\tilde{\alpha}) = \{R^*(\alpha) | \alpha \in \tilde{\alpha}\}$. $\tilde{\alpha}$ is said to *map cleanly* to $\tilde{R}(\tilde{\alpha})$ if $\tilde{R}(\tilde{\alpha}) \in \tilde{A}^T$ and α maps cleanly to $R^*(\alpha)$ for all $\alpha \in \tilde{\alpha}$.

In the following definitions, let $\mathcal{T} = (T, S, \tau)$ be a 2HAM or STAM TAS and, for some initial configuration $S_{\mathcal{T}}$, that depends on $\mathcal{T}$, let $\mathcal{U} = (U, S_{\mathcal{T}}, \tau')$ be a 2HAM or STAM TAS, and let R be an m-block representation function $R : B^U_m \dashrightarrow T$.

Definition. We say that $\mathcal{U}$ and $\mathcal{T}$ have *equivalent productions* (at scale factor m), and we write $\mathcal{U} \Leftrightarrow_R \mathcal{T}$ if the following conditions hold:

1. $\left\{\tilde{R}(\tilde{\alpha}) | \tilde{\alpha} \in \mathcal{A}[\mathcal{U}]\right\} = \mathcal{A}[\mathcal{T}]$.
2. $\left\{\tilde{R}(\tilde{\alpha}) | \tilde{\alpha} \in \mathcal{A}_{\Box}[\mathcal{U}]\right\} = \mathcal{A}_{\Box}[\mathcal{T}]$.
3. For all $\tilde{\alpha} \in \mathcal{A}[\mathcal{U}]$, $\tilde{\alpha}$ maps cleanly to $\tilde{R}(\tilde{\alpha})$

Definition. We say that $\mathcal{T}$ *follows* $\mathcal{U}$ (at scale factor m), and we write $\mathcal{T} \dashv_R \mathcal{U}$ if, for any $\tilde{\alpha}, \tilde{\beta} \in \mathcal{A}[\mathcal{U}]$ such that $\tilde{\alpha} \to^1_{\mathcal{U}} \tilde{\beta}$, $\tilde{R}(\tilde{\alpha}) \to^{\leq 1}_{\mathcal{T}} \tilde{R}\left(\tilde{\beta}\right)$.

Definition. We say that $\mathcal{U}$ *strongly models* $\mathcal{T}$ (at scale factor m), and we write $\mathcal{U} \models^+_R \mathcal{T}$ if for any $\tilde{\alpha}$, $\tilde{\beta} \in \mathcal{A}[\mathcal{T}]$ such that $\tilde{\gamma} \in C^{\tau}_{\tilde{\alpha},\tilde{\beta}}$, then for all $\tilde{\alpha}', \tilde{\beta}' \in \mathcal{A}[\mathcal{U}]$ such that $\tilde{R}(\tilde{\alpha}') = \tilde{\alpha}$ and $\tilde{R}\left(\tilde{\beta}'\right) = \tilde{\beta}$, it must be that there exist $\tilde{\alpha}'', \tilde{\beta}'', \tilde{\gamma}' \in \mathcal{A}[\mathcal{U}]$, such that $\tilde{\alpha}' \to_{\mathcal{U}} \tilde{\alpha}''$, $\tilde{\beta}' \to_{\mathcal{U}} \tilde{\beta}''$, $\tilde{R}(\tilde{\alpha}'') = \tilde{\alpha}$, $\tilde{R}\left(\tilde{\beta}''\right) = \tilde{\beta}$, $\tilde{R}(\tilde{\gamma}') = \tilde{\gamma}$, and $\tilde{\gamma}' \in C^{\tau'}_{\tilde{\alpha}'',\tilde{\beta}''}$.

Note that *strongly models* is in contrast to *weakly models* as defined in [19].

Definition. Let $\mathcal{U} \Leftrightarrow_R \mathcal{T}$ and $\mathcal{T} \dashv_R \mathcal{U}$. $\mathcal{U}$ *simulates* $\mathcal{T}$ (at scale factor m) if $\mathcal{U} \models^+_R \mathcal{T}$.

Throughout this chapter, we use the above definition of "simulation", which in [19] is referred to by the term "strong simulation", while they use the term "simulation" to refer to a weaker definition which make use of "weakly models". Thus, here, for simulation we use the intuitively "stronger" notion of simulation where the simulator must in some sense more strictly model the simulated system.

8.3.1 Intrinsic universality

Let REPR denote the set of all m-block (or macrotile) representation functions (i.e., m-block supertile representation functions for some $m \in \mathbb{Z}^+$). For some $d \in \{2, 3\}$, let $\mathfrak{C}$ be a class of d-dimensional tile assembly systems, and let U be a d'-dimensional tile set for $d' \geq d$. Note that every element of $\mathfrak{C}$, REPR, and $\mathcal{A}^U_{<\infty}$ is a finite object, hence can be represented in a suitable format for computation in some formal system such as Turing machines. We say U is *intrinsically universal* for $\mathfrak{C}$ *at temperature* $\tau' \in \mathbb{Z}^+$ if there are computable functions $\mathcal{R} : \mathfrak{C} \to \mathsf{REPR}$ and $\mathcal{S} : \mathfrak{C} \to \left(A^U_{<\infty} \to \mathbb{N} \cup \{\infty\}\right)$ such that, for each $\mathcal{T} = (T, S, \tau) \in \mathfrak{C}$, there is a constant $m \in \mathbb{N}$ such that, letting $R = \mathcal{R}(\mathcal{T})$, $S_{\mathcal{T}} = S(\mathcal{T})$, and $\mathcal{U}_{\mathcal{T}} = (U, S_{\mathcal{T}}, \tau')$, $\mathcal{U}_{\mathcal{T}}$ simulates $\mathcal{T}$ at scale m and using supertile representation function R. That is, $\mathcal{R}(\mathcal{T})$ outputs a representation function that interprets macrotiles (or m-blocks) of $\mathcal{U}_{\mathcal{T}}$ as assemblies of $\mathcal{T}$, and $S(\mathcal{T})$ gives the initial state used to create the necessary macrotiles from U to represent $\mathcal{T}$ subject to the constraint that no macrotile in $S_{\mathcal{T}}$ can be larger than a single $m \times m$ square. We say that U is *intrinsically universal* for $\mathfrak{C}$ if it is intrinsically universal for $\mathfrak{C}$ at some temperature $\tau' \in Z^+$.

8.4 Transforming STAM$^+$ Systems From Arbitrary to Bounded Signal Complexity

In this section, we demonstrate methods for reducing the signal complexity of STAM$^+$ systems with $\tau = 1$ or $\tau > 1$ and results related to reducing signal complexity. First, we define terms related to the complexity of STAM systems, and then state our results for signal complexity reduction.

We now provide informal definitions for fan-in, fan-out and mutual activation. For more rigorous definitions, see [30].

Definition. For an STAM system $\mathcal{T} = (T, \sigma, \tau)$, we say that $\mathcal{T}$ contains **fan-in** iff there exist glues $g_1 \ldots g_i$ on a tile $t \in T$ such that only when all of these glues bind, the activation or deactivation of more than 1 glue on t is triggered.

Definition. For an STAM system $\mathcal{T} = (T, \sigma, \tau)$, we say that $\mathcal{T}$ contains **fan-out** iff there exists a glue g on a tile $t \in T$ such that whenever g binds, it triggers the activation or deactivation of more than 1 glue on t.

Definition. For an STAM system $\mathcal{T} = (T, \sigma, \tau)$, we say that $\mathcal{T}$ contains **mutual activation** iff $\exists t_1, t_2 \in T$ with glue g on adjacent edges of t_1 and t_2 such that whenever t_1 and t_2 bind by means of glue g, the binding of g causes the activation or deactivation of other glues on both t_1 and t_2.

8.4.1 Impossibility of eliminating both fan-out and mutual activation at $\tau = 1$

We now discuss the impossibility of completely eliminating both fan-out and mutual activation at temperature 1. Consider the signal tiles in Figure 8.1 and let $\mathcal{T} = (T, 1)$ be the STAM$^+$ system where T consists of exactly those tiles. Theorem 21 shows that at temperature 1, it is impossible to completely eliminate both fan-out and mutual activation. In other words, any STAM$^+$ simulation of $\mathcal{T}$ must contain some instance of either fan-out or mutual activation. The intuitive idea is that the only mechanism for turning on glues is binding, and at temperature 1 we cannot control when glues in the `on` state bind. Hence any binding pair of glues that triggers some other glue must do so by means of a sequence of glue bindings leading from the source of the signal to the signal to be turned `on`. The tile set depicted in Figure 8.1 grows infinitely to the right and each B tile must have two glues activated, where in this tile set or any simulation thereof, each glue activation in the entire assembly must trace back to the A tile. However, the tile set must have a bounded signal complexity, thus, without fan-out or mutual activation the assembly will grow to a size where both glues on the next B tile to bind cannot be activated. Hence there must be paths to both of the triggered glues from the single originating glue where at some point a single binding event fires two signals. We will see that this is not the case at temperature 2 since we can control glue binding through cooperation there.

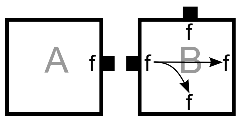


Figure 8.1: An example of a tile set where fan-out and mutual activation cannot be completely removed. The glue f on the west edge of tile type B signals two other glues.

Theorem 21. At temperature 1, there exists an STAM$^+$ system $\mathcal{T}$ such that any STAM$^+$ system $\mathcal{S}$ that simulates $\mathcal{T}$ contains fan-out or mutual activation.

8.4.2 Eliminating either fan-out or mutual activation, reducing fan-in.

In this section we will discuss the possibility of eliminating fan-out from an STAM$^+$ system. We do this by simulating a given STAM$^+$ system with a simplified STAM$^+$ system where fan-in is reduced to at most 2, and the system contains no fan-out, but does contain mutual activation. A slight modification to the construction that we provide then shows that mutual activation can be swapped for fan-out of at most 2.

Definition. An *n-simplified STAM tile set* is an STAM tile set which has the following properties: (1) the full-tile signal complexity is limited to a fixed constant $n \in \mathbb{N}$, (2) there is no fan-out, and (3) fan-in is limited to 2. We say that an STAM system $\mathcal{T} = (T, \sigma, \tau)$ is n-simplified if T is n-simplified.

Theorem 22. For every STAM$^+$ system $\mathcal{T} = (T, \sigma, \tau)$, there exists a 2-simplified STAM$^+$ system $\mathcal{S} = (S, \sigma', \tau)$ which simulates $\mathcal{T}$ with scale factor $O(|T|^2)$ and tile complexity $O(|T|^2)$.

To prove Theorem 22, we construct a macrotile such that every pair of signal paths that run in parallel are never contained on the same tile. This means that at most two signals are ever on one tile since it is possible for a tile to contain at most two non-parallel (i.e. crossing) signals. In place of fan-out, we use mutual activation gadgets (see Figure 8.3) within the *fan-out zone*. Similarly, we use a *fan-in zone* consisting of tiles that merge incoming signals two at a time, in order to reduce fan-in. For examples of these zones, see Figure 8.2. Next, we print a circuit (a system of signals) around the perimeter of the macrotile which ensures that the external glues (the glues on the edges of the macrotiles that cause macrotiles to bind to one another) are not turned on until a macrotile is fully assembled. More details of the construction can be found in [30].

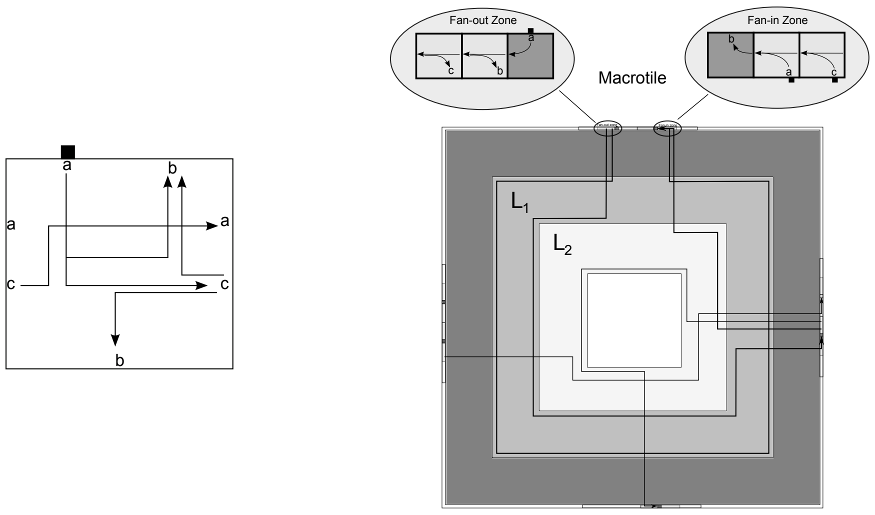


Figure 8.2: A tile with 5 signals (left) and the STAM$^+$ macrotile that simulates it (right). Each frame, labeled L_1 and L_2, corresponds to a glue. For example on the tile to be simulated (left) there is a signal that runs from glue a to glue c. In order to simulate this signaling, a signal runs from the fan-out zone of glue a to the frame associated with glue a, labeled L_1, on the north edge. The signal then wraps around the frame until it reaches the east side on which glue c lies. Then the signal enters the fan-in zone of glue c.

To further minimize the number of signals per tile at $\tau > 1$, cooperation allows us to reduce the number of signals per tile required to just 1. To achieve this result, we modify the construction used to show Theorem 22, and prove Theorem 23. The details of the modification can be found in [30].

Theorem 23. For every STAM$^+$ system $\mathcal{T} = (T, \sigma, \tau)$ with $\tau > 1$, there exists a 1-simplified STAM$^+$ system $\mathcal{S} = (S, \sigma', \tau)$ which simulates $\mathcal{T}$ with scale factor $O(|T|^2)$ and tile complexity $O(|T|^2)$.

8.4.3 Summary of Results

At temperature 1, the minimum signal complexity obtainable in general is 2 and while it is possible to eliminate either fan-out or mutual activation, it is impossible to eliminate both. For temperatures greater than 1, cooperation allows for signal complexity to be reduced to just 1 and for both fan-out and mutual activation to be completely eliminated. Table 8.1 gives a summary of these two cases of reducing signal complexity and shows the cost of

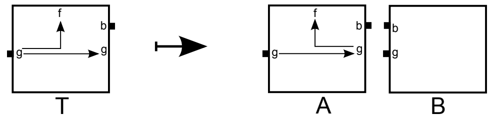


Figure 8.3: An example of a mutual activation gadget consisting of tiles A and B without fan-out simulating, at $\tau = 1$, the functionality of tile T which has fan-out. The glue b represents the generic glues which holds the macrotile together. The idea is to "split" the signals from the west glue g on tile A into two signals without using fan-out. Once the west glue g on tile A binds, it turns on the east glue g on tile A. Then, when the east glue g on tile A binds to tile B, it triggers glue f. Thus, the east glue g triggers both the west glue g and glue f without fan-out.

Temperature	Signal per Tile	Scale Factor	Tile Complexity	Contains Fan-In / Mutual Activation
1	2	$O(\|T\|^2)$	$O(\|T\|^2)$	one or the other
> 1	1	$O(\|T\|^2)$	$O(\|T\|^2)$	neither

Table 8.1: The cost of reducing signal complexity at $\tau = 1$ and at $\tau > 1$.

such reductions in terms of scale factor and tile complexity.

8.5 A 3D 2HAM Tile Set which is IU for the STAM$^+$

In this section we present our main result, namely a 3D 2HAM tile set which can be configured to simulate any temperature 1 or 2 STAM$^+$ system, at temperature 2. It is notable that although three dimensions are fundamentally required by the simulations, only two planes of the third dimension are used.

Theorem 24. There is a 3D tile set U in the 2HAM such that U is intrinsically universal at temperature 2 for the class of all 2D STAM$^+$ systems where $\tau \in \{1,2\}$. Further, U uses no more than 2 planes of the third dimension.

To prove Theorem 24, we let $\mathcal{T}' = (T', S', \tau)$ be an arbitrary STAM$^+$ system where $\tau \in \{1,2\}$. For the first step of our simulation, we define $\mathcal{T} = (T, S, \tau)$ as a 2-simplified STAM$^+$ system which simulates $\mathcal{T}'$ at scale factor $m' = O(|T'|^2)$, tile complexity $O(|T'|^2)$, as given by Theorem 22, and let the representation function for that simulation be $R' : B^{T}_{m'} \dashrightarrow T'$. We now show how to use tiles from a single, universal tile set U to form an initial configuration $S_\mathcal{T}$ so that the 3D 2HAM system $\mathcal{U}_\mathcal{T} = (U, S_\mathcal{T}, 2)$ simulates $\mathcal{T}$ at scale factor $m = O(|T| \log |T|)$ under representation function $R : B^U_m \dashrightarrow T$. This results in $\mathcal{U}_\mathcal{T}$ simulating $\mathcal{T}'$ at a scale factor of $O(|T'|^4 \log(|T'|^2))$ via the composition of R and R'. The exponent of 4 results because U simulates $\mathcal{T}$ using scale factor $|T| \log |T|$ where $|T| = O(|T'|^2)$, hence the scale factor for this part of the simulation is $O(|T'|^2 \log |T'|^2)$, but $\mathcal{T}$ is also simulating $\mathcal{T}'$ with scale factor $|T'|^2$. Thus the total scale factor will be $O(|T'|^4 \log(|T'|^2))$. Note that throughout this section, τ refers to the temperature of the simulated systems $\mathcal{T}$ and $\mathcal{T}'$, while the temperature of $\mathcal{U}_\mathcal{T}$ is always 2.

8.5.1 Construction overview

In this section, due to restricted space we present the 3D 2HAM construction at a very high level. Please see [30] for more details.

Assuming that T is a 2-simplified STAM$^+$ tile set derived from T', we note that for each tile in T: 1. glue deactivation is not used, 2. it has ≤ 2 signals, 3. it has no fan-out, and 4. fan-in is limited to 2. To simulate $\mathcal{T}$, we create an input supertile $\sigma_\mathcal{T}$ from tiles in U so that $\sigma_\mathcal{T}$ fully encodes $\mathcal{T}$ in a rectangular assembly where each row fully encodes the definition of a single tile type from T. Beginning with an initial configuration containing an infinite count of that supertile and the individual tile types from U, assembly begins with the growth of a row on top of (i.e. in the $z = 1$ plane) each copy of $\sigma_\mathcal{T}$. The tiles forming this row nondeterministically select a tile type $t \in T$ for the growing supertile to simulate, allowing each supertile the possibility of simulating exactly one $t \in T$, and each such t to be simulated. Once enough tiles have attached, that supertile maps to the selected t via the representation function R, and at this point we call it a *macrotile.*

Each such macrotile grows as an extension of $\sigma_\mathcal{T}$ in $z = 0$ in a zig-zag fashion (see Figure 8.8 for an indication of the direction of this growth) to form a square ring with a hole in the center. This zigzag growth occurs using cooperativity as described in [12]. The growth occurs clockwise from $\sigma_\mathcal{T}$, creating the west, north, east, then south sides, in that order. As each side grows, the information from the definition of t which is relevant to

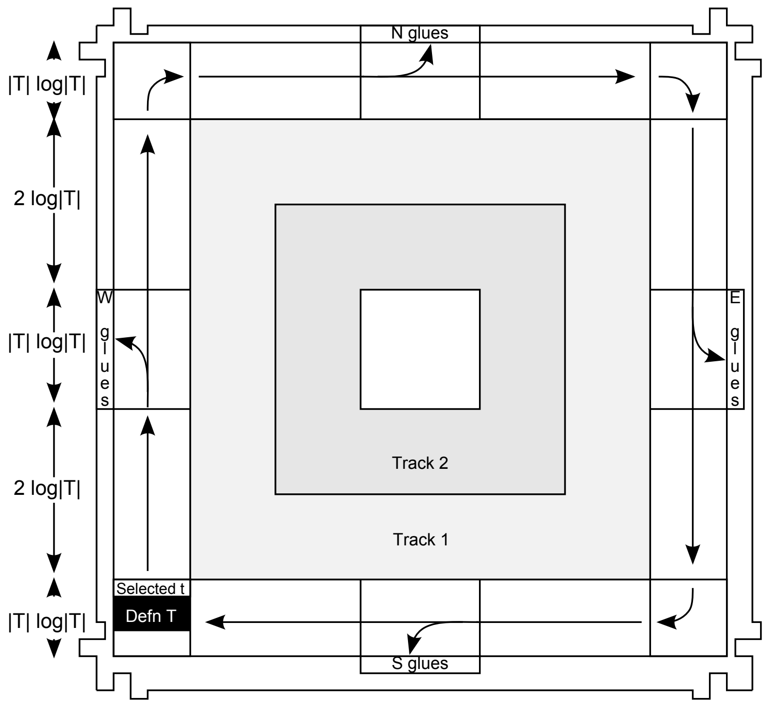


Figure 8.4: A high level sketch of the components and formation of a macrotile, including dimensions, not represented to scale. The grey portions labeled “Track 1” and “Track 2” represent initially empty space that is later populated with tiles as needed to simulate signals in the macrotile.

that side is rotated so that it is presented on the exterior edge of the forming macrotile. The second to last stage of growth for each side is the growth of geometric "bumps and dents" near the corners, which ensure that any two macrotiles which attempt to combine along adjacent edges must have their edges in perfect alignment for any binding to occur. The final stage of growth for each side is to place the glues which face the exterior of the macrotile and are positioned correctly to represent the glues which begin in the `on` state for that side. The macrotile is poised to simulate the signals of the STAM$^+$ tile it represents through a process of tile assembly into rows that carry the signal information to the correct simulated glue position(s). In the following description, this process of signal simulation will be referred to as the "signal" in the context of the macrotile.

Once the first side of a macrotile completes (which is most likely to be the west side, but due to the nondeterministic ordering of tile additions it could potentially be any side), that macrotile can potentially bind to another macrotile, as long as the tiles that they represent would have been able to do so in $\mathcal{T}$. Whenever macrotiles do bind to each other, the points at which any binding glues exist allow for the attachment of duples (supertiles consisting of exactly 2 tiles) on top of the two binding tiles (in $z = 1$). These duples initiate the growth of rows in $z = 1$ which move inward on each macrotile to determine if there is information encoded which specifies a signal for that simulated glue to fire. If not, that row terminates. If so, it continues growth by reading the information about that signal (i.e. the destination side and glue), and then growth continues which carries that information inward to the hole in the center of the macrotile. Once there, it grows clockwise in $z = 0$ until arriving at the correct side and glue, where it proceeds to initiate the growth of a row in $z = 1$ out to the edge of the macrotile in the position representing the correct glue. Once it arrives, it initiates the addition of tiles which effectively change the state of the glue from `latent` to `on` by exposing the necessary glue(s) to the exterior of the macrotile.

The width of the center hole is carefully specified to allow for the maximum necessary 2 "tracks" along which fired signals can travel, and growth of the signal paths is carefully designed to occur in a zig-zag pattern such that there are well-defined "points of competition" which allow two signals which are possibly using the same track to avoid collisions, with the second signal to arrive growing over the first, rotating toward the next inward track, and then continuing along that track. Further, the positioning of the areas representing the glues on each edge is such that there is always guaranteed to be enough room for the signals to perform the necessary rotations, inward, and outward growth. If it is the case that both signals are attempting to activate the same glue on the same side, when the second signal arrives, the row growing from the innermost track toward the edge of the macrotile will simply run into the "activating" row from the first signal and halt, since there is no need for both to arrive and in the STAM such a situation simply entails that signal being discarded. (Note that this construction can be modified to allow for any arbitrary full-tile signal complexity n for a given tile set by simply increasing the number of tracks to n, and all growth will remain correct and restricted to $z \in \{0, 1\}$.)

This construction allows for the faithful simulation of $\mathcal{T}$ by exploiting the fact that the activation of glues by fired signals is completely asynchronous in the STAM, as is the attachment of any pair of supertiles, and both processes are being represented through a series of supertile binding events which are similarly asynchronous in the 2HAM. Further, since the signals of the STAM$^+$ only ever activate glues (i.e. change their states from `latent` to `on`), the constantly "forward" direction of growth (until terminality) in both models ensures that the simulation by $\mathcal{U}_\mathcal{T}$ can eventually produce representations of all supertiles in $\mathcal{T}$, while never generating supertiles that don't correctly map to supertiles in $\mathcal{T}$ (equivalent production), and also that equivalent dynamics are preserved.

Theorem 25. For every $\tau > 1$, there is a 3D tile set U_τ such that, in the 2HAM, U_τ is intrinsically universal at temperature τ for the class of all 2D STAM$^+$ systems of temperature τ. Further, U uses no more than 2 planes of the third dimension.

To prove Theorem 25, we create a new tile set U_τ for each τ from the tile set of Theorem 24 by simply creating $O(\tau)$ new tile types which can encode the value of the strength of the glues of T in $\sigma_{\mathcal{T}}$, and which can also be used to propagate that information to the edges of the macrotiles. Signal complexity can be reduced to 1 using substitution gadgets which require that glue representations be split[30]. For the exterior glues of the macrotiles, we split τ-strength glues into one of strength $\lceil \tau/2 \rceil$ and the other $\lfloor \tau/2 \rfloor$. In this way, the new tile set U_τ will form macrotiles exactly as before, while correctly encoding glues of strengths 1 through τ on their exteriors, and the systems using it will correctly simulate STAM$^+$ systems at temperature τ.

8.6 For Each τ, an STAM$^+$ Tile Set Which is IU for the STAM$^+$ at τ

In this section, we show that for every temperature τ there is an STAM$^+$ tile set which can simulate any STAM$^+$ system at temperature τ. The constructions for $\tau > 1$ and $\tau = 1$ differ slightly, and are presented in that order, and both are based heavily upon the construction in Section 8.5.

Let $\mathcal{T}' = (T', S_{\mathcal{T}'}, \tau)$ be an arbitrary STAM$^+$ system and let $\mathcal{T} = (T, S_{\mathcal{T}}, \tau)$ be the STAM$^+$ system constructed in Section 8.4 that is 1-simplified in the case where $\tau > 1$ and 2-simplified in the case where $\tau = 1$. The constructions here yield STAM$^+$ systems $\mathcal{U}_{\mathcal{T}} = (U_\tau, S_{\mathcal{U}_{\mathcal{T}}}, \tau)$ that simulate each $\mathcal{T}$, and therefore prove Theorem 26.

Theorem 26. For each temperature τ, there is an STAM$^+$ tile set U_τ such that U_τ is intrinsically universal for the class of all STAM$^+$ systems at temperature τ.

8.6.1 STAM$^+$ Tile Sets which are IU for the STAM$^+$ at each $\tau > 1$

This construction makes use of macrotiles very similar to those of the construction in Section 8.5 to simulate the tiles of $\mathcal{T}$. To see how they work, note that in the 2D STAM$^+$, growth of macrotiles can proceed in exactly the same way as most of the macrotile growth in the 3D 2HAM construction of Section 8.5 with a few exceptions. First, since we cannot take advantage of multiple planes, the simulated signals must propagate using actual signals of active tiles. This can be done as follows. First, we queue a signal to be fired along the necessary path through the macrotile. In the 3D 2HAM construction, as a macrotile grows, glues that can potentially propagate a simulated signal upon binding expose glues that allow for a duple to be placed above them in the $+z$-direction. In the STAM$^+$ construction, we can simply replace these tiles with signal tiles such that the binding of a glue would fire cascading signals toward a signal track of the macrotile. See Figure 8.8 for an example of a signal track and placement of glues that aid the simulated signal propagation. The entire sequence of firing a simulated signal, resulting in a glue going from `latent` to `on`, is as follows. Following the construction of Section 8.5, glues that allow for two macrotiles to bind are assumed to have strength less than τ, and a strength τ glue g is represented by two glues g_1 and g_2. This means that when we turn a glue `on`, we really expose glues on the exterior of a macrotile that allow for tiles to bind to the macrotile that *present* either a single glue or two glues such as g_1 and g_2. Let α be a macrotile producible in $\mathcal{U}_\mathcal{T}$, let a be a glue of the simulated tile that fires a signal, and let b (respectively b_1 and b_2 in the case that we are turning `on` a strength τ glue) be a glue that turns `on` as a result of the binding of a. When α binds to another macrotile β (producible in $\mathcal{U}_\mathcal{T}$) using a, the binding event for a initiates the propagation of signals that eventually expose a strength τ glue, g_s, on the inside of the signal track (which is initially empty). Edges of the macrotile that border the signal track expose glues that allow for cooperative binding in a clockwise manner around the track. All but one of these glues – all but the glue on the tile abutting the signal track at the location directly across from the tile location for the simulated glue to be turned `on` – are the same. That final glue, call it g_e, initiates a sequence of signals that turns `on` a glue g_h on the exterior of α. Now, when a tile binds to g_s, a single tile wide path of tiles grows along the signal track using cooperation until binding with g_e. The binding event involving g_e fires a signal that initiates the propagation of signals that expose g_h on the exterior of α (Figure 8.8). Then, g_h allows for tiles to attach that present b (respectively b_1 and b_2), completing the simulation of the signal from a turning `on` b. To finish the construction, we show how to obtain $S_{\mathcal{U}_\mathcal{T}}$ from $S_\mathcal{T}$.

Creating the initial configuration

Here we give a conversion of the initial configuration outlined in Section 8.5 that works for STAM$^+$ systems with $\tau > 1$. As in Section 8.5, suppose that the initial configuration $S_{\mathcal{T}}$ consists of the singleton tiles in T. We create the initial configuration $S_{\mathcal{U}_{\mathcal{T}}}$ by letting $S_{\mathcal{U}_{\mathcal{T}}} = U_\tau \cup \{\sigma_{\mathcal{T}}\}$ where $\sigma_{\mathcal{T}}$ is a rectangle consisting of tiles from U_τ which has a single row encoding the definition of each tile $t \in T$. The specific encoding that is used is identical to the encoding used in the 3D 2HAM construction and can be found in [30]. Figure 8.5 depicts one of these rectangular supertiles, along with selection tiles. To select a row from one of these rectangular supertiles, tiles cooperatively bind to the east edge of tiles of the rectangle and nondeterministically choose a row. In Figure 8.5, the tiles labeled N are skipping rows, the tile labeled Y selects the row and the tiles labeled D mark the fact that a row has been selected. Care must be taken to ensure that some row is always selected. Once a row is selected, the binding of glues g_s in Figure 8.5 fires a signal that turns `on` either a glue g_0 or a glue g_1 depending on the row and the encoding of the tile $t \in T$ and also turns `on` another g_s glue should the signal need to be propagated. The binding of g_0 or g_1 glues fires signals that propagate to the north edge of the tiles located on the top of the rectangular supertile, thus completing the nondeterministic selection of some tile $t \in T$ for the macrotile to simulate.

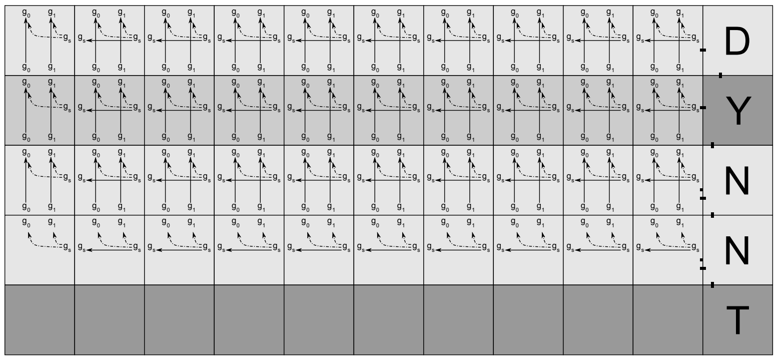
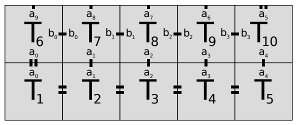


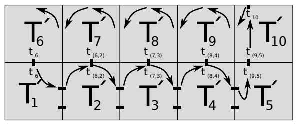
Figure 8.5: A schematic figure for a seed. Each tile west of N,Y or D represents either a 1 or 0 depending on the encoding of the tile set to be simulated, and has only one of the two signals depicted with dashed lines.

8.6.2 An STAM$^+$ Tile Set which is IU for the STAM$^+$ at $\tau = 1$

The construction for $\tau = 1$ is very similar to the previous, essentially including a few relatively minor changes to adapt the construction of Section 8.5. We begin by noting that by Theorem 21, since $\tau = 1$ the macrotiles that are simulating tiles in $\mathcal{T}$ must be capable of simulating two signals, since at $\tau = 1$, the tiles of T cannot be less than 2-simplified. We also note that at $\tau = 1$, STAM$^+$ systems are capable of simulating temperature-2 zig-zag growth, which can be used throughout the construction of Section 8.5 to build the macrotiles. Figure 8.6 shows how, starting from a single row of tiles, one additional row of tiles can be added in a zig-zag fashion. Additional rows can then be grown similarly. In Figure 8.6, note that the zig-zag growth is deterministic. For example, T_6' and T_2' determine a single type of tile to be placed to the north of T_2'. In our case, deterministic zig-zag growth is all that is needed. Thus, additional glue and signal complexity allow each tile to attach with a single strength-1 glue, which causes a signal cascade that exposes a glue combining information about both glues adjacent to the next location for a tile to attach. By combining information about both adjacent glues, this strength-1 glue effectively simulates them and thus their cooperative behavior.



(a)



(b)

Figure 8.6: (a) depicts zig-zag growth for $\tau = 2$. With tiles T_1 through T_5 in place, tiles T_6 through T_{10} must be placed in that order. Moreover, tile T_6 depends only on T_1, while tiles T_i for $i \in \{7, 8, 9, 10\}$ depend on T_{i-1} and T_{i-5}. (b) depicts zig-zag growth for $\tau = 1$ using active tiles. Again, with tiles T_1' through T_5' in place, tiles T_6' through T_{10}' must be placed in that order. Moreover, tile T_6' depends only on T_1', while tiles T_i' for $i \in \{7, 8, 9, 10\}$ depend on T_{i-1}' and T_{i-5}'. Also, tiles in each consecutive row contain signals that allow for another row to grow in a zig-zag pattern. Note that only the signals which activate glues which are eventually bound to are shown, while the tiles would actually need to accommodate glues and signals which could represent all possible pairs of input glues for the next growth location.

Macrotile formation

Now, recall that the growth of the macrotiles described in Section 8.4.2 that occurs in the $z = 0$ plane follows a zig-zag pattern. Therefore, we can use active tiles such that similar growth can occur in 2D STAM$^+$ systems with $\tau = 1$. Figure 8.8 shows a scheme for how this zig-zag growth works. Starting from a seed, a tile type $t \in T$ is selected. This seed and selection process is described in Section 8.6.2 that follows. Then, following a zig-zag growth pattern, each side of the macrotile forms. As the sides of the macrotiles form, special care is taken in presenting the glues exposed by each edge of the macrotile. The glues that will be exposed on the exterior of a macrotile are `latent` initially. This ensures proper growth of the macrotile by postponing macrotile binding until the macrotile has formed. As zig-zag growth forming the west edge of a macrotile completes, a glue is exposed on the west edge of the northwest most tile, allowing for the attachment of a *corner gadget* at the northwest corner. See figure 8.7 for an example of a corner gadget. When the northwest corner gadget binds, it initiates the successive firing of signals that eventually expose a glue that allows for the growth of the north side of the macrotile and then a second corner gadget to attach at the northeast corner. Similarly, when the northeast corner gadget binds, it initiates the successive firing of signals that eventually expose a glue that allows for a third corner gadget to attach at the southeast corner and in turn fires a signal that eventually allows for the final corner gadget to be placed at the southwest corner. Just as in the construction of Section 8.5, the corner gadgets give the macrotile bumps and dents which ensure proper alignment when two macrotiles bind. Finally, the binding of the final corner gadget initiates a sequence of active glues that trigger the simulated glues exposed by the macrotile to turn `on`. Attaching corner gadget in this manner guarantees us that the bumps of the macrotile are in place prior to the exposure of any `on` glues on the exterior of the macrotile.

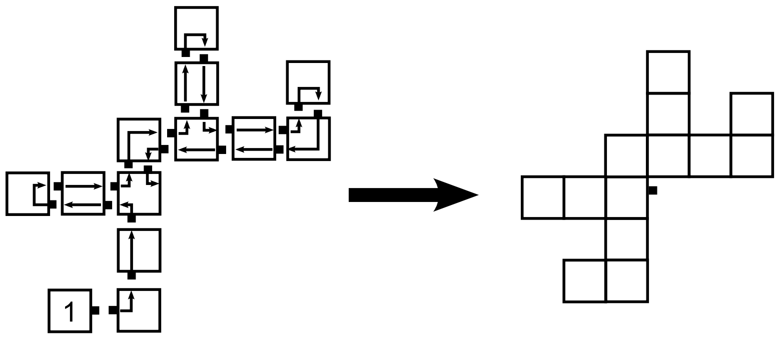


Figure 8.7: A corner gadget for macrotiles in $\tau = 1$ systems. Note that the formation of the entire corner gadget must take place prior to the exposure of the glue that allows the corner gadget to bind to a macrotile.

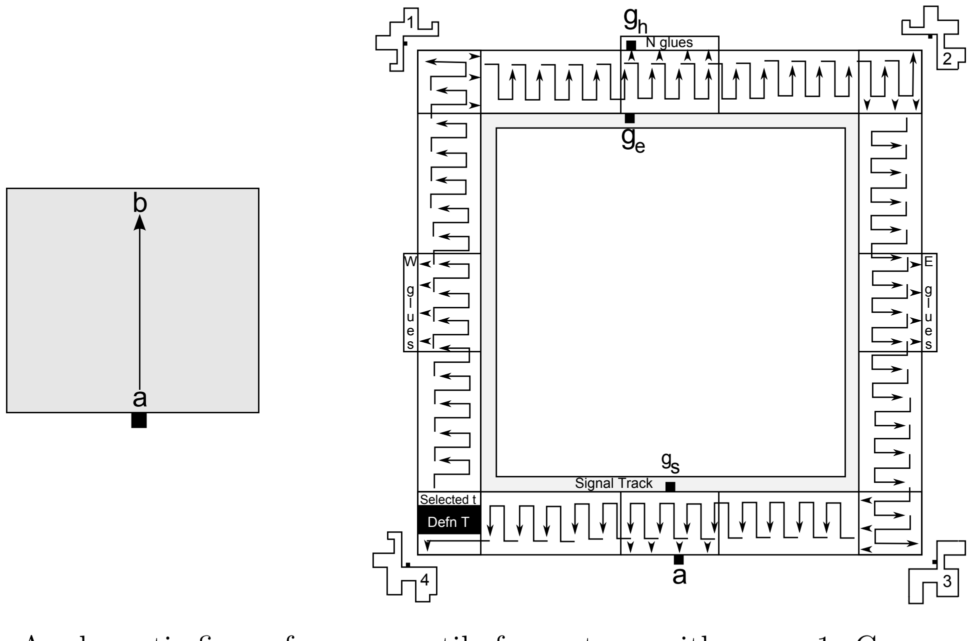


Figure 8.8: A schematic figure for a macrotile for systems with $\tau = 1$. Corner gadgets are labeled in the order that they must attach. An example of the placement of glues g_s, g_e, and g_h that aid the propagation of the simulated signal from a to b are shown. The glue g_h recruits tiles that will expose one or two glues representing b.

Handling signals

To permit the firing of signals by glues on the exterior of a macrotile, we proceed as in the $\tau > 1$ case. Except here, the formation of the signal track and placement of any tile that may expose a glue on the exterior of a macrotile takes place during the zig-zag formation of a macrotile α. Again, note that for any macrotile, we need only fire at most 2 signals. We define a *wire* to be a path of active glues through tiles of a macrotile such that the signals along this path propagate through adjacent tiles starting at a preset initial tile until the signal reaches a preset ending tile. For the $\tau = 1$ construction we will turn glues on macrotile edges `on` using wires. A binding event involving a glue g_1 can initiate the propagation of signals that turn `on` a glue g_2. In this case, we say that there is a *wire* from g_1 to g_2. Now, in the case where $\tau = 1$, glues that allow for two macrotiles to bind are `latent` initially. Let α be a macrotile producible in $\mathcal{U}_\mathcal{T}$, let a be a glue of the macrotile that fires a signal and let b be a glue that turns `on` as a result of the binding of a. As α grows, wires can be placed on tiles using the information encoded in the seed by simply counting to the correct glue locations. These wires start from the tile with the a glue exposed and end at the tile with the glue b glue exposed. The placement of tiles with the signals making up the wire from a to b is as follows. First, if a is exposed on a tile, t_a, belonging to the east or west edge (resp. the north or south edge) of α, tiles containing signals making up the wire from a to b are placed horizontally (resp. vertically) as a row (resp. column) of the zig-zag pattern from t_a to a tile t_s lying on the signal track. The signal track is depicted in Figure 8.8. Similarly, if b is a glue on a tile, t_b, belonging to the east or west edge (resp. the north or south edge) of α, tiles containing signals making up the wire from a to b are place horizontally (resp. vertically) as a row (resp. column) of the zig-zag pattern from t_b to a tile t_e lying on the signal track. The final tiles containing signals making up the wire from a to b are the tiles along the signal track that run clockwise from t_s to a tile t_e. Notice that we can have two such wires from any two glues exposed on the exterior of α. Now, when α binds to another macrotile β producible in $\mathcal{U}_\mathcal{T}$ using a, the binding event for *a* initiates the propagation of signals that eventually expose a *b* via a wire from *a* to *b*.

The initial configuration for $\tau = 1$

In this section, we show how to modify the seed given in Section 8.6.1 so that it can be used in systems where $\tau = 1$. The main idea here is to use signals to mimic the behavior of cooperation that can occur in systems with temperature greater than 1. Figure 8.9 gives an example of such a seed supertile. The rows of the supertile that encode tiles of T and the selection process remain unchanged, however the nondeterministic selection of a row changes slightly. The binding of successive selector tiles (tiles labeled N, Y and D) use signals to ensure that they are placed from bottom to top. Also, a stopper tile, labeled S in the figure, prevents the placement of an extra selector tile. Finally, as in the case for $\tau > 1$, we must ensure that some row is selected. To do this we place a glue d on the south edge of tile S and a d glue on the north edge of each tile labeled N so that if no Y attaches and a tile labeled N is placed below S, binding of d will fire a signal to turn `on` g_s on the west edge of the topmost N tile, selecting the tile type encoded in the top row.

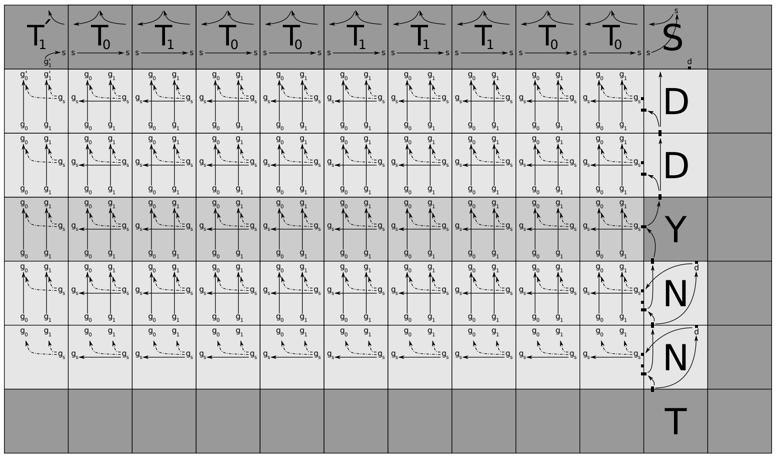


Figure 8.9: A schematic figure for a seed for $\tau = 1$. For each tile, depending on the encoding of the tile set to be simulated, only one of the signals depicted with dashed lines is present per tile. Tiles labeled T_0, T_1, T_1', N, Y and D are not part of the seed assembly and are only shown to illustrate the selection of a row.

Once a row is selected, glues that have been turned `on` during the selection process allow for tiles, labeled T_0, T_1 and T_1' in the figure, to be placed. A tile T_0' analogous to the tile T_1' is placed instead of T_1' if the left most bit of the selected row is 0. T_0' is not shown in the figure. When T_1' (or T_0') binds, signals propagated by glue pair bindings, corresponding to glues labeled s in the figure, are used to ensure that each T_0, T_1, T_0' or T_1' is in place. Once each of these tiles is in place, a glue is presented on the north edge of the rectangular supertile grown from the seed that allows for zig-zag growth of the macrotile to start.

Finally, we make a general observation about the seed supertile formation for systems with $\tau = 1$. Consider the glues that hold the initial configuration together. Such glues cannot be exposed initially, for if they were, tiles that make up the initial configuration could spontaneously bind and possibly create arbitrary seeds containing encodings of tile types that should not be simulated. "Junk" supertiles that do not take part in any simulation could also be created. Therefore, for systems with $\tau = 1$ we form the seed from tiles which initially have all glues in the `latent` state, and first manually turn `on` the necessary set of glues on each tile used for the seed, and then use these glues to attach those tiles together.

8.6.3 Complexity analysis

First, the tile complexity is $O(1)$ as there is a single universal tile set U_τ which simulates any T'. Let k_τ denote the tile complexity for a fixed τ. Now, since macrotiles form in the same way as the $z = 0$ plane growth of the 3D macrotiles given in Section 8.5, the scale factor of the macrotiles in the STAM$^+$ construction is the same. In particular, the scale factor is $O(|T| \log |T|)$, while using a constant sized tile set and the scale factor for $\mathcal{U}_\mathcal{T}$ simulating $\mathcal{T}'$ is $O(|T'|^4 \log(|T'|^2))$. The techniques to reduce signal complexity given in Section 8.4 can be used to reduce the signal complexity of tiles in our macrotiles. We claim that this only contributes a constant size increase in scale-factor. First, note that the most signal complexity for any one tile of the construction at $\tau > 1$ is 4, and the most glue complexity of any one tile is bounded by a constant that depends only on k_τ. In particular, the glue complexity is independent of $|T'|$. Hence, the tile set U_τ can be 1-simplified with only a constant sized increase in scale-factor. Therefore, the scale factor for $\mathcal{U}_\mathcal{T}$ simulating $\mathcal{T}'$ is $O(|T'|^4 \log(|T'|^2))$ even for a 1 simplified tile set. For $\tau = 1$, the most signal complexity for any one tile of the construction at $\tau = 1$ is also a constant that is independent of the tile set T. In fact, to mimic zig-zag growth for $\tau = 1$ each tile contains a number of signals that depends on k_τ plus some signals that are used for the firing of signals once macrotiles have formed. Overall, this will be a constant amount of signal complexity that is independent of $|T|$. The glue complexity of each tile is also bounded by a similar constant that is independent of $|T|$. Therefore, when we apply the simplification techniques given in Section 8.4, the scale-factor increase will be a constant independent of $|T|$. Hence, the tile set U_τ can be 2-simplified with only a constant sized increase in scale-factor, and the scale factor for $\mathcal{U}_\mathcal{T}$ simulating $\mathcal{T}'$ remains to be $O(|T'|^4 \log(|T'|^2))$ for $\tau = 1$.

Note that the representation function used for this construction as well as the proof of correctness are nearly identical to those for the construction of Section 8.5, and they can be found in [30].

8.7 Conclusion

We have shown how to transform STAM$^+$ systems (at temperature 1 or > 1) of arbitrary signal complexity into STAM$^+$ systems which simulate them while having signal complexity no greater than 2 and 1, respectively. However, if the original tile set being simulated is T, the scale factor and tile complexity of the simulating system are approximately $O(|T|^2)$. It seems that these factors cannot be reduced in the worst case, i.e. when a tile of T has a copy of every glue of the tile set on each side, and each copy of each glue on the tile activates every other, yielding a signal complexity of $O(|T|^2)$. However, whether or not this is a true lower bound remains open, as well as what factors can be achieved for more "typical" systems with much lower signal complexity.

A significant open problem which remains is that of generalizing both constructions (the signal reduction and the 3D 2HAM simulation) to the unrestricted STAM. Essentially, this means correctly handling glue deactivation and possible subassembly dissociation. While this can't be handled within the standard 3D 2HAM where glue bonds never change or break, it could perhaps be possible if negative strength (i.e. repulsive) glues are allowed (see [21] for a discussion of various formulations of models with negative strength glues). However, it appears that since both constructions use scaled up macrotiles to represent individual tiles of the systems being simulated, there is a fundamental barrier. The STAM assumes that whenever two tiles are adjacent, all pairs of matching glues across the adjacent edge which are both currently `on` will immediately bind (which is in contrast to other aspects of the model, which are asynchronous). Since both constructions trade the ability of individual tile edges in the STAM to have multiple glues with scaled up macrotiles which distribute those glues across individual tiles of the macrotile edges, it appears to be difficult if not impossible to maintain the correct simulation dynamics. Basically, a partially formed side of a macrotile could have only a subset of its initially `on` glues in place, but enough to allow it to bind to another macrotile. At that point, if glue deactivations are initiated which result in the dissociation of the macrotile before the remaining glues of the incomplete macrotile side assemble, then in the simulating system, those additional glues won't ever bind. However, in the simulated system they would have. This results in a situation where, after the dissociation, the simulated system would potentially have additional pending glue actions (initiated by the bindings of the additional glues) which the simulating system would not, breaking the simulation.

Overall, laboratory experiments continue to show the plausibility of physically implementing signalling tiles [49], while previous theoretical work [50] shows some of their potential, and the results in this chapter demonstrate how to obtain much of that power with simplified tiles. We feel that research into self-assembly with active components has a huge amount of potential for future development, and continued studies into the various tradeoffs (i.e. complexity of components, number of unique component types, scale factor, etc.) between related models provide important context for such research. We hope that our results help to contribute to continued advances in both theoretical and experimental work along these lines.

Chapter 9

Conclusion

We have introduced many new theoretical models of self-assembly. We have also defined notions of simulation between these models as well as notions of simulation between existing models. As we continue to advance the theory of self-assembly, we find that in such a relatively budding field, as fast as questions can be answered, new ones arise, and that working on open problems inevitably leads to more open problems. Therefore, it seems fitting that the conclusion of this thesis simply be a series of open questions and possible directions for future research. We begin with questions about the temperature 1 aTAM which are subject of much discussion in the self-assembly community.

9.1 Universality in subclasses of the class of all aTAM systems

As previously mentioned, in [64], Winfree showed that the class of directed aTAM systems at temperature 2 is computationally universal, and in [22], Doty et al. showed that the aTAM is intrinsically universal for itself by showing that there is a temperature 2 system that when appropriately seeded can simulate any given aTAM system (even those with temperature $\neq 1$). The following questions regarding the aTAM remain open. (1) Is the class of directed aTAM systems at temperature 1 is computationally universal? (2) Is the class of temperature 1 aTAM systems intrinsically universal? It is conjectured that the answer to both of these questions is "no". In the proof that the aTAM is intrinsically universal for itself, one can observe that even for directed systems, the intrinsically universal tile set fundamentally relies on nondeterminism. It is unknown if this relying on nondeterminism is necessary when simulating directed systems. Hence we have the following question. Is the class of directed aTAM systems intrinsically universal for itself?

9.2 Simulation in the polyTAM

We have seen that the geometry of tiles can be utilized to achieve computation, but many questions regarding simulation remain open. Therefore, it would be interesting to seek a classification of which polyTAM systems can simulate other polyTAM systems. For example, we have seen that any polyomino gives rise to a single shaped polyTAM system such that this system forms an infinite grid. Then, given an aTAM system, with the appropriate assignment of glues to edges of polyomino tiles with a single shape one could deduce that given any aTAM system $\mathcal{T}$ with temperature τ and any polyomino P, there exists a single shape polyTAM system $\mathcal{S}$ with shape P and temperature τ such that $\mathcal{S}$ simulates $\mathcal{T}$. Along similar lines, one can conclude that the polyTAM at temperature 2 is IU for the aTAM since the aTAM at temperature 2 is IU for itself. While this simple line of reasoning shows that the single shaped polyTAM with tiles shaped like a unit square can be simulated by some single shaped polyTAM system with tiles shaped like any given polyomino, this is far from a complete picture of how polyTAM systems are related via simulation.

9.3 Extending the RTAM

The RTAM at temperature 1 was shown to be non-Turing-complete. Insofar as this model accurately capture the physical system which it sets out to describe, this seems to cast doubt on the effectiveness of this physical system to algorithmically drive self-assembly. Assuming the RTAM is a sufficiently accurate model, is there an extension of the RTAM that would be feasible to implement physically and able to perform computation? A candidate for such an extension would be something like the Dupled RTAM (analogous to the Dupled aTAM). In addition, we have seen that at temperature 2, the RTAM is Turing-complete. In order to simulate a Turing machine we showed that the RTAM can simulate zig-zag systems as defined in the aTAM. A more general question remains open. Is the RTAM IU for the aTAM? Essentially, this question is asking if cooperation is powerful enough to completely overcome the fact that in the RTAM tiles may be reflected before binding.

9.4 Is the STAM IU?

Finally, we have seen that the STAM$^+$ is IU for itself and that the 3-D 2HAM is IU for the STAM$^+$. Is the STAM (which allows supertiles to disassociate) IU for itself? Since the 3-D 2HAM does not allow for tile detachment, it trivially follows that the 3-D 2HAM is not IU for the STAM. However, the 3-D 2HAM with negative glues could allows for tile detachment and therefore, the result that the 3-D 2HAM is IU for the STAM$^+$ may generalize, which leads us to the following question. Is the 3-D 2HAM with negative glues IU for the STAM?

References

[1] Leonard Adleman, Qi Cheng, Ashish Goel, and Ming-Deh Huang. Running time and program size for self-assembled squares. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 740–748, Hersonissos, Greece, 2001.

[2] Jürgen Albert and Karel Culik II. A simple universal cellular automaton and its one-way and totalistic version. *Complex Systems*, 1:1–16, 1987.

[3] Pablo Arrighi and Jonathan Grattage. Intrinsically universal¡ i¿ n¡/i¿-dimensional quantum cellular automata. *Journal of Computer and System Sciences*, 78(6):1883–1898, 2012.

[4] Pablo Arrighi, Nicolas Schabanel, and Guillaume Theyssier. Intrinsic simulations between stochastic cellular automata. Technical Report 1208.2763, Computing Research Repository, 2012.

[5] Robert D. Barish, Rebecca Schulman, Paul W. Rothemund, and Erik Winfree. An information-bearing seed for nucleating algorithmic self-assembly. *Proceedings of the National Academy of Sciences*, 106(15):6054–6059, March 2009.

[6] Robert D. Barish, Rebecca Schulman, Paul W. K. Rothemund, and Erik Winfree. An information-bearing seed for nucleating algorithmic self-assembly. *Proceedings of the National Academy of Sciences*, 106(15):6054–6059, April 2009.

[7] Bahar Behsaz, Ján Maňuch, and Ladislav Stacho. Turing universality of step-wise and stage assembly at temperature 1. In Darko Stefanovic and Andrew Turberfield, editors, *DNA Computing and Molecular Programming*, volume 7433 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin Heidelberg, 2012.

[8] Nathaniel Bryans, Ehsan Chiniforooshan, David Doty, Lila Kari, and Shinnosuke Seki. The power of nondeterminism in self-assembly. *Theory of Computing*, 9:1–29, 2013.

[9] Sarah Cannon, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Matthew J. Patitz, Robert Schweller, Scott M. Summers, and Andrew Winslow. Two hands are better than one (up to constant factors). Technical Report 1201.1650, Computing Research Repository, 2012.

[10] Ho-Lin Chen and Ming-Yang Kao. Optimizing tile concentrations to minimize errors and time for dna tile self-assembly systems. In Yasubumi Sakakibara and Yongli Mi, editors, *DNA*, volume 6518 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2010.

[11] 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.

[12] 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.

[13] Marianne Delorme, Jacques Mazoyer, Nicolas Ollinger, and Guillaume Theyssier. Bulking i: an abstract theory of bulking. *Theoretical Computer Science*, 412(30):3866–3880, 2011.

[14] Marianne Delorme, Jacques Mazoyer, Nicolas Ollinger, and Guillaume Theyssier. Bulking i: An abstract theory of bulking. *Theor. Comput. Sci.*, 412(30):3866–3880, 2011.

[15] Marianne Delorme, Jacques Mazoyer, Nicolas Ollinger, and Guillaume Theyssier. Bulking II: Classifications of cellular automata. *Theoretical Computer Science*, 412(30):3881–3905, 2011.

[16] Marianne Delorme, Jacques Mazoyer, Nicolas Ollinger, and Guillaume Theyssier. Bulking ii: Classifications of cellular automata. *Theor. Comput. Sci.*, 412(30):3881–3905, 2011.

[17] E. D. Demaine, M. L. Demaine, S. P. Fekete, M. J. Patitz, R. T. Schweller, A. Winslow, and D. Woods. One tile to rule them all: Simulating any tile assembly system with a single universal tile. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP 2014),* IT University of Copenhagen, Denmark, July 8-11, 2014, volume 8572 of *LNCS*, pages 368–379. Springer Berlin Heidelberg, 2014.

[18] Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Mashhood Ishaque, Eynat Rafalin, Robert T. Schweller, and Diane L. Souvaine. Staged self-assembly: nanomanufacture of arbitrary shapes with $O(1)$ glues. *Natural Computing*, 7(3):347–370, 2008.

[19] Erik D. Demaine, Matthew J. Patitz, Trent A. Rogers, Robert T. Schweller, Scott M. Summers, and Damien Woods. The two-handed assembly model is not intrinsically universal. In *40th International Colloquium on Automata, Languages and Programming, ICALP 2013, Riga, Latvia, July 8-12, 2013*, Lecture Notes in Computer Science. Springer, 2013.

[20] David Doty. Theory of algorithmic self-assembly. *Commun. ACM*, 55(12):78–88, December 2012.

[21] David Doty, Lila Kari, and Benoît Masson. Negative interactions in irreversible self-assembly. *Algorithmica*, 66(1):153–172, 2013.

[22] 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.

[23] David Doty, Jack H. Lutz, Matthew J. Patitz, Scott M. Summers, and Damien Woods. Intrinsic universality in self-assembly. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science*, pages 275–286, 2009.

[24] David Doty, Jack H. Lutz, Matthew J. Patitz, Scott M. Summers, and Damien Woods. Random number selection in self-assembly. In *Proceedings of The Eighth International Conference on Unconventional Computation (Porta Delgada (Azores), Portugal, September 7-11, 2009)*, 2009.

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

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

[27] B. Durand and Zs. Róka. The game of life: universality revisited. In M. Delorme and J. Mazoyer, editors, *Cellular Automata*. Kluwer, 1999.

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

[29] 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.

[30] Tyler Fochtman, Jacob Hendricks, Jennifer E. Padilla, Matthew J. Patitz, and Trent A. Rogers. Signal transmission across tile assemblies: 3D static tiles simulate active self-assembly by 2D signal-passing tiles. Technical Report 1306.5005, Computing Research Repository, 2013.

[31] Bin Fu, Matthew J. Patitz, Robert T. Schweller, and Robert Sheline. Self-assembly with geometric tiles. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming*, ICALP, pages 714–725, 2012.

[32] Martin Gardner. Mathematical games - the fantastic combinations of john conway's new solitaire game "life". (223):120–123, 1970.

[33] Eric Goles Ch., Pierre-Étienne Meunier, Ivan Rapaport, and Guillaume Theyssier. Communication complexity and intrinsic universality in cellular automata. *Theoretical Computer Science*, 412(1-2):2–21, 2011.

[34] Jacob Hendricks, Jennifer E. Padilla, Matthew J. Patitz, and Trent A. Rogers. Signal transmission across tile assemblies: 3D static tiles simulate active self-assembly by 2D signal-passing tiles. In David Soloveichik and Bernard Yurke, editors, *DNA Computing and Molecular Programming*, volume 8141 of *Lecture Notes in Computer Science*, pages 90–104. Springer International Publishing, 2013.

[35] Jacob Hendricks and Matthew J. Patitz. On the equivalence of cellular automata and the tile assembly model. In Turlough Neary and Matthew Cook, editors, Proceedings *Machines, Computations and Universality 2013,* Zürich, Switzerland, 9/09/2013 - 11/09/2013, volume 128 of *Electronic Proceedings in Theoretical Computer Science*, pages 167–189. Open Publishing Association, 2013.

[36] Jacob Hendricks, Matthew J. Patitz, and Trent Rogers. Reflections on tiles (in self-assembly). Technical Report 1404.5985, Computing Research Repository, 2014.

[37] Jacob Hendricks, Matthew J. Patitz, and Trent A. Rogers. Doubles and negatives are positive (in self-assembly). In *Proceeding of Unconventional Computation and Natural Computation 2014 (UCNC 2014),* University of Western Ontario, London, Ontario, Canada, 7/14/2014 - 7/18/2014, 2014. to appear.

[38] Jacob Hendricks, Matthew J. Patitz, and Trent A. Rogers. Doubles and negatives are positive (in self-assembly). *CoRR*, abs/1403.3841, 2014.

[39] 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. In *Proceedings of 20th International Computing and Combinatorics Conference (COCOON 2014),* Atlanta, Georgia, USA, 8/04/2014 - 8/06/2014, 2014. to appear.

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

[41] 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.

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

[43] Chengde Mao, Thomas H. LaBean, John H. Relf, and Nadrian C. Seeman. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature*, 407(6803):493–6, 2000.

[44] Ján Maňuch, Ladislav Stacho, and Christine Stoll. Two lower bounds for self-assemblies at temperature 1. *Journal of Computational Biology*, 17(6):841–852, 2010.

[45] Jacques Mazoyer and Ivan Rapaport. Inducing an order on cellular automata by a grouping operation. In *STACS 98*, pages 116–127. Springer, 1998.

[46] 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.

[47] 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.

[48] Nicolas Ollinger and Gaétan Richard. Four states are enough! *Theoretical Computer Science*, 412(1):22–32, 2011.

[49] Jennifer E. Padilla. personal communication, 2013.

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

[51] Matthew J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, 2014.

[52] Matthew J. Patitz, Robert T. Schweller, and Scott M. Summers. Exact shapes and turing universality at temperature 1 with a single negative glue. In *Proceedings of the 17th international conference on DNA computing and molecular programming*, DNA'11, pages 175–189, Berlin, Heidelberg, 2011. Springer-Verlag.

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

[54] Matthew J. Patitz and Scott M. Summers. Self-assembly of decidable sets. *Natural Computing*, 10(2):853–877, 2011.

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

[56] Paul Rendell. A universal turing machine in conway's game of life. In *HPCS*, pages 764–772, 2011.

[57] Paul W. K. Rothemund. *Theory and Experiments in Algorithmic Self-Assembly*. PhD thesis, University of Southern California, December 2001.

[58] Paul W. K Rothemund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of dna sierpinski triangles. *PLoS Biol*, 2(12):e424, 12 2004.

[59] 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.

[60] Rebecca Schulman and Erik Winfree. Synthesis of crystals with a programmable kinetic barrier to nucleation. *Proceedings of the National Academy of Sciences*, 104(39):15236–15241, 2007.

[61] David Soloveichik, Matthew Cook, and Erik Winfree. Combining self-healing and proofreading in self-assembly. *Natural Computing*, 7(2):203–218, 2008.

[62] David Soloveichik and Erik Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36(6):1544–1569, 2007.

[63] Hao Wang. Proving theorems by pattern recognition – II. *The Bell System Technical Journal*, XL(1):1–41, 1961.

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

[65] Erik Winfree and Renat Bekbolatov. Proofreading tile sets: Error correction for algorithmic self-assembly. In Junghuei Chen and John H. Reif, editors, *DNA*, volume 2943 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.

[66] Erik Winfree, Furong Liu, Lisa A. Wenzler, and Nadrian C. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394(6693):539–44, 1998.

[67] Erik Winfree, Xiaoping Yang, and Nadrian C. Seeman. Universal computation via self-assembly of dna: Some theory and experiments. In *DNA Based Computers II, volume 44 of DIMACS*, pages 191–213. American Mathematical Society, 1996.

[68] Damien Woods. Intrinsic universality and the computational power of self-assembly. In *MCU: Proceedings of Machines, Computations and Universality*, volume 128, pages 16–22, Univ. of Zürich, Switzerland. Sept. 9-12, 2013. Open Publishing Association. `dx.doi.org/10.4204/EPTCS.128.5`.