

Asynchronous Signal Passing for Tile Self-Assembly: Fuel Efficient Computation and Efficient Assembly of Shapes

Jennifer E. Padilla^{*} Matthew J. Patitz[†] Raul Pena[‡] Robert T. Schweller[§]
Nadrian C. Seeman[¶] Robert Sheline^{||} Scott M. Summers^{**} Xingsi Zhong^{††}

Abstract

In this paper we demonstrate the power of a model of tile self-assembly based on active glues which can dynamically change state. We formulate the Signal-passing Tile Assembly Model (STAM), based on the model of Padilla, et al. [20] to be asynchronous, allowing any action of turning a glue on or off, attaching a new tile, or breaking apart an assembly to happen in any order. Within this highly generalized model we provide three new solutions to tile self-assembly problems that have been addressed within the abstract Tile Assembly Model and its variants, showing that signal passing tiles allow for substantial improvement across multiple complexity metrics. Our first result utilizes a recursive assembly process to achieve *tile-type efficient* assembly of linear structures, using provably fewer tile types than what is possible in standard tile assembly models. Our second system of signal-passing tiles simulates any Turing machine with high *fuel efficiency* by using only a constant number of tiles per computation step. Our third system assembles the discrete Sierpinski triangle, demonstrating that this pattern can be *strictly self-assembled* within the STAM. This result is of particular interest in that it is known that this pattern cannot self-assemble within a number of well studied tile self-assembly models. Notably, all of our constructions are at temperature 1, further demonstrating that signal-passing confers the power to bypass many restrictions found in standard tile assembly models.

1 Introduction

The abstract Tile Assembly Model (aTAM) [32] created a paradigm for computation to be carried out by a physical assembly process that captured the essence of the Wang tiling model [30]. Turing machine simulation within the aTAM demonstrated its capacity for universal computation, and many subsequent assembly programs shifted focus to patterns, shapes and structures as the output of tile computation [13, 24, 25, 29]. Many modifications to the standard aTAM have been investigated, including several variants that capture the notion of hierarchical assembly [2–5, 7, 20]. Previous work introduced the notion of using signaled glue activation [18, 20], in particular to guide hierarchical assembly, enabling recursive self assembly [20]. Here

^{*}Department of Chemistry, New York University, jp164@nyu.edu This author’s research was supported by National Science Foundation Grant CCF-1117210.

[†]Department of Computer Science and Computer Engineering, University of Arkansas, mpatitz@self-assembly.net This author’s research was supported in part by National Science Foundation Grant CCF-1117672.

[‡]Dept of Computer Science, University of Texas–Pan American, nb-raul@utpa.edu. This author’s research was supported in part by National Science Foundation Grant CCF-1117672.

[§]Department of Computer Science, University of Texas–Pan American, schweller@cs.panam.edu This author’s research was supported in part by National Science Foundation Grant CCF-1117672.

[¶]Department of Chemistry, New York University, ned.seeman@nyu.edu This author’s research was supported by National Science Foundation Grant CCF-1117210.

^{||}Dept of Computer Science, University of Texas–Pan American, b.sheline@utpa.edu. This author’s research was supported in part by National Science Foundation Grant CCF-1117672.

^{**}Department of Computer Science, University of Wisconsin–Oshkosh, summer@uwosh.edu

^{††}Department of Computer Science, Clemson University, zhongxingsi@gmail.com. This author’s research was supported in part by National Science Foundation Grant CCF-1117672.

we develop a general model of signaled tile self-assembly to enrich the tile assembly paradigm with greater capabilities that anticipate advancing techniques in the field of DNA nanotechnology and allow tile assembly to more closely emulate biological and natural processes.

Signaled glue activation was introduced in [20] for the purpose of establishing communication inside an assembly so that by activating glues at the exterior of the assembly it may take on new identities or roles. Interactions between assemblies, as described in hierarchical models such as the 2-HAM [1, 5], can simulate the interactions of individual tiles, coordinated in the STAM by the introduction of signals. In particular, recursive assembly results when supertiles simulate the original tiles of the tile set [20], a strategy outlined here in section 4 in a scheme for efficient line assembly.

Cooperativity, where more than one tile must be in place to determine which tile may be added next, is an important aspect of tile assembly systems. A binding threshold is included in the aTAM, given as the temperature, τ of the system. Tiles can bind only if the sum of glue interactions at a particular site meet or exceed τ , thus a system at temperature 2 readily includes cooperativity, whereas it is not as readily achieved at temperature 1. In the STAM, cooperativity can occur at temperature 1 via a query process, where a tile binds to an assembly at one edge, then queries a neighboring tile by turning on a set of glues. Information about the neighboring tile is gained based on which of these glues binds to its match. This cooperative effect differs from the aTAM in that STAM tiles may also respond to the identities and binding events of more distant tiles, enabling the constructions given in this paper to operate at temperature 1. Though the constructions here do not demonstrate a full simulation of the aTAM at temperature 2 by the STAM at temperature 1, the results here are significant given the known and conjectured limitations to temperature 1 computation in the aTAM and its variants [6, 12].

We expect glue deactivation to be as easy to implement as glue activation based on plausible DNA strand displacement mechanisms. Therefore, we utilize this new capability to design a Turing Machine that is fuel efficient (Section 5), and to implement the strict self-assembly of the Sierpinski triangle (Section 6). While on first consideration, glue deactivation might be thought to be on par with negative glues, it never requires repulsive forces between tiles, a necessity for negative glues that to our knowledge has yet to be implemented. Glue deactivation serves here to produce a fuel efficient Turing Machine that does not need to copy the state of unchanged positions on the tape. A halting computation produces an output tape, not a transcript of the computation as in the traditional aTAM simulation of a Turing Machine. Strict self assembly of the Sierpinski triangle is achieved by releasing tiles that are not part of the target structure.

The addition of signaled glue activation and deactivation to tile assembly brings it one step closer to emulating biological processes of self assembly, where communication within a developing and growing living organism are crucial to its survival and success. In this construction, it becomes easier to view the Turing Machine plus its tape as a developing entity, that by following its input instructions, much as a cell follows its genetic recipe, goes through a metamorphosis and emerges from a halting computation as a new entity. The asynchronous growth process of the strict Sierpinski triangle in this model resembles the growth of something such as coral, where the “living” functional part of the system inhabits the growing frontier of the structure, laying down an enduring structure before dying and being washed away. The constructions presented in this paper demonstrate not only a more efficient use of materials (Table 1) in certain cases, but also serve to make the model more relevant in a biological context. The STAM anticipates the increasing sophistication of molecular computation systems, as described in the next section.

2 Physical Basis for the Model

The generalized model presented here has been designed to take into consideration a DNA implementation of all aspects of signaled assembly. We envision a physical implementation where Watson-Crick DNA base pairing provides specific glue interactions as it has done before for DNA implementations of the aTAM [25, 32]. Additionally, we suggest that toehold-mediated DNA strand displacement reactions [35] may be the basis for the new elements of our model: binding-induced signaling, and glue activation or deactivation. The physical body of a tile might be composed entirely from a DNA origami structure [16, 26] in order to provide more room for signal pathways than the smaller DNA structures that have been used to implement the passive tiles of the aTAM [33]. Many known and tested DNA strand exchange mechanisms [23, 34, 36], cascades [9],

$n \times 1$ Lines	Tiles	Signals	Temperature	Glue Activation	Glue Deactivation
aTAM	n	-	1	-	-
STAM (Thm.4.2)	$O(1)$	$O(\log n)$	1	Yes.	No.

Turing Machine	Space	Fuel	Temp	Tiles	Signals	Glue Act.	Glue Deact.
aTAM ([24])	$O(\sum S_i)$	$O(S_i)$	2	$O(Q)$	-	-	-
3D aTAM ([6])	$O(\sum S_i)$	$O(S_i)$	1	$O(Q)$	-	-	-
Negative Glues ([11])	$O(S_i)$	$O(S_i)$	2	$O(Q)$	-	-	-
Negative Glues ([27])	$O(S_i)$	$O(1)$	8	$O(Q)$	-	-	-
STAM (Thm.5.1)	$O(S_i)$	$O(1)$	1	$O(Q)$	$O(1)$	Yes.	Yes.

Sierpinski Triangle	Strict	Tiles	Signals	Temp	Scale	Glue Act.	Glue Deact.
aTAM ([25])	No.	7	-	2	1	-	-
STAM (Thm.6.2)	Yes.	19	5	1	2	Yes.	Yes.
STAM (Thm.6.1)	No.	5	4	1	1	Yes.	No.

Table 1: Summary of our Results in the context of previous work in the field. In the Turing machine results, Q is the number of states of the Turing machine being simulated and S_i is the length of the tape at step i in the computation.

and walkers [17, 19, 31, 34] suggest possibilities for implementing the signal pathway, including logic gates for responding to multiple inputs and transducers for ensuring that the activated glue can have a different sequence from the glues on the input edges [28]. Details of a plausible DNA origami tile construction are given in [20].

Details of our model come from a consideration of possible DNA strand displacement mechanisms that might be used to implement signal-passing tiles. The physical basis of the three glue states are illustrated in Figure 1, where a glue may be considered ‘latent’ or ‘off’ (Figure 1c) if enough of its DNA sticky end, including the toehold, is blocked by being bound to a complementary DNA protection strand. Activation of a latent glue involves the toe-hold mediated removal of that strand as a result of a signal cascade (Figure 1d). The difference between the ‘on’ state (Figure 1a) and the ‘latent’ or ‘off’ state (Figure 1c) is based on the availability of toeholds and whether or not there could be any opportunity for them to initiate binding to another tile in the system. Deactivation is merely a reversal of this process, making it just as feasible as activation.

The limitation to pass a given signal only once, and to activate and deactivate a specific glue on a single tile only once comes from the inherent consumption of “DNA fuel” associated with DNA strand displacement cascades, where the production of more DNA base pairs in the products drives the process forward. Thus, the model is designed to emphasize this fuel-consumption aspect by making each use of a signal count as another signal path. The asynchronous nature of our model reflects the likewise asynchronous nature of the molecular events we envision for the physical implementation. In particular, DNA strand displacement reactions proceed via a branch migration resembling a random walk at a rate that can vary widely depending on toehold length and composition [36]. We allow for the time taken by these signaling or glue activation events by storing actions triggered by various events in STAM assembly in a queue for asynchronous processing. Together, these model details capture to the best of our ability and in the most general way possible the capabilities reported in the field of DNA nanotechnology that we see as plausible additions to the molecular implementation of tile assembly, particularly if DNA continues to play a significant role in these implementations.

3 STAM Notation and Model

In this section we define the Signal-passing Tile Assembly Model (STAM), an extension of the 2-Handed Assembly Model (2HAM) [3–5, 7], which itself is an extension of the Tile Assembly Model (TAM) [32]. The STAM is a refined version of the model presented in [20], in which the basic tiles of the TAM are augmented with the ability to receive information, in the form of *signals*, from neighboring tiles in an assembly, and to pass signals to neighboring tiles. A very important feature of signals is that each signal can only move

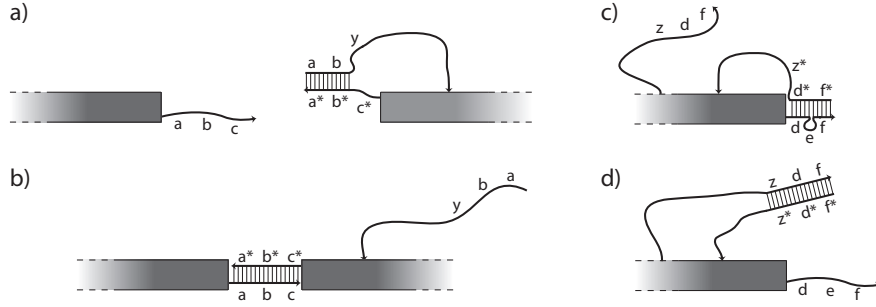


Figure 1: DNA mechanisms for triggering a signal cascade and glue activation. Single-stranded DNA is depicted as a line with an arrow at the 3' end and double-stranded DNA is shown as two parallel lines connected by a ladder representing base pairs. Letters such as a and y indicate short sequences of DNA bases and a star indicates the complementary sequence. Rectangles indicate the edges of structures, possibly composed of DNA origami, to which the relevant strands of DNA are attached. a) Complementary DNA strands of active glues can initiate binding at the toehold c^* . Both glues here are in the “on” state because complementary toehold sequences are exposed on both tiles. b) Toehold binding initiates a 3-strand DNA branch migration in which the abc strand displaces the yba strand as the tiles bind to one another, freeing the yba strand to trigger a signal cascade. c) The strand zdf is presumed to be unavailable until freed by a previous step in the signal cascade (not shown). The sequence labeled z may then bind at the toehold z^* on the protection strand $f^*d^*z^*$, removing it from the glue strand def via strand displacement. The glue strand def is in the “latent” state in which the entire glue strand is protected either by the complementary sequence on the protection strand $f^*d^*z^*$, or by inclusion in a small bulge loop, a structure known to protect short DNA sequences from initiating base-pairing with a complementary sequence [9]. This scheme prevents the signal strand zdf from containing a full copy of (and therefore being functionally equivalent to) the glue strand. d) Once the protection strand is removed, the glue strand abc is active or *on*.

across any given tile one time - they are not reusable.

The STAM that we define is a highly generalized model which imposes minimal restrictions on aspects such as the speed of signals and orderings of events. This generalized version, while perhaps difficult to create well-behaved constructions within, provides a framework that is intended to be maximally independent of the specific details of potential physical implementations of actual signal tiles, such as those using mechanisms suggested in [20]. Valid constructions within this model, such as all of the constructions presented within the following sections of this paper, will therefore also work correctly within more restricted versions of the model (for instance, where signal timing or ordering can be guaranteed).

3.1 Informal Description of the 2HAM

Since the STAM is an extension of the 2HAM, we now give a brief, informal description of the 2HAM.

A *tile type* is a unit square with four sides, each having a *glue* consisting of a *label* (a finite string) and *strength* (a positive integer value). 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* (a.k.a., *assembly*) is a positioning of tiles on the integer lattice \mathbb{Z}^2 . Two adjacent tiles in a supertile *interact* if the glues on their abutting sides are equal (in both label and strength) and *bind* with that shared strength. Each supertile induces a *binding graph*, a grid graph whose vertices are tiles, with an edge between two tiles if they interact and where the weight of that edge is the strength of their bond. The supertile is τ -*stable* if every cut of its binding graph has strength at least τ . That is, the supertile is stable if at least energy τ (i.e. a cut across bonds whose strengths sum to at least τ) is required to separate the supertile into two parts. A *tile assembly system* (TAS) is a pair $\mathcal{T} = (T, \tau)$, where T is a finite tile set (or more generally a finite set of supertiles) and τ is the *temperature*, a parameter specifying the minimum binding energy required for a supertile to be stable. Given a TAS $\mathcal{T} = (T, \tau)$, a supertile is *producible* if either it is an element of T , or it is the τ -stable result of translating two producible assemblies without overlap or rotation. A supertile α is *terminal* if for every producible supertile β , α and β cannot be τ -stably attached. A TAS is *directed* (a.k.a., *deterministic*, *confluent*) if it has only one terminal, producible supertile. Given a connected shape $X \subseteq \mathbb{Z}^2$, a TAS \mathcal{T} *strictly self-assembles* X (also *produces* X *uniquely*) if every producible, terminal supertile places tiles exactly on those positions in X (appropriately translated if necessary). Given a pattern $Y \subseteq \mathbb{Z}^2$ (which must not necessarily be connected), a TAS \mathcal{T} *weakly self-assembles* Y if every producible, terminal supertile places a subset of tiles $B \subseteq T$ exactly on those positions in Y (appropriately translated if necessary). Weak self-assembly can be thought of as using a subset of tile types to “paint a picture” of Y on a possibly larger canvas of tiles composing a terminal assembly.

3.2 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). Tiles also each implement a transition function which is executed upon the binding of any glue on any edge of that tile. The transition function specifies a set of glues (along with the sides on which those glues are located) and an action for each: 1. turn that glue **on** (only valid if it is currently **latent**), or 2. turn that 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**, and if they are ever switched to **off** then no further transitions are allowed for that glue. This essentially provides a single “use” of a glue (and thus the implicit signal sent by its activation and binding). Note that turning a glue **off** breaks any bond that the 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 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).

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 includes 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 apart into two separate supertiles along a cut whose total strength is less than τ .

As an overview, tiles in the STAM pass *signals* to neighboring tiles simply by activating glues which can bind with glues on adjacent edges of neighboring tiles. The information content of a signal is dependent upon the transition function of the receiving tile, that is, by what glue actions the receiving tile initiates upon the binding of its glue. By subsequently activating and deactivating its own glues, the receiving tile can propagate the signal to any of its neighbors. Solely by utilizing the mechanism of glue activation and deactivation initiated and carried out on individual tiles but chained together through series of glue bindings, a global network which is capable of passing information across entire assemblies (and also of allowing them to selectively enable sites for further growth or to discard arbitrary portions of the assembly), is created. However, an important restriction is the “fire once” nature of the signals, meaning that each glue can only transition to any given state once, and therefore the number of signals which a tile can process is a constant dependent upon the definition of the tile type.

The STAM, as formulated, is intended to provide a model based on experimentally plausible mechanisms for glue activation and deactivation, but to abstract them in a manner which is implementation independent. Therefore, no assumptions are made about the speed or ordering of the completion of signaling events (i.e. the execution of the transition functions which activate and deactivate glues and thus communicate with other tiles via binding events). This provides a highly asynchronous framework in which care must be made to guarantee desired results, but which then provides robust behavior independent of the actual parameters realized by a physical system. Furthermore, 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 would obviously be limited in physical implementations. 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 we have also defined the notion of *signal complexity*, as the maximum number of glues on any side of any tile in a given set, to capture the complexity of a tile set.

3.3 Formal Definition of the Signal Passing Tile Assembly Model

This section presents a formal definition of the STAM. In order to help clarify many aspects, a consistent example is provided and referenced throughout.

3.3.1 Basic Notation

Let D denote the set of labels $\{\textit{north}, \textit{south}, \textit{east}, \textit{west}\}$, or $\{N, S, E, W\}$.

Active Glues and Glue States Let Γ denote a set of *glue types*. A *glue* is an ordered pair (g, s) where $g \in \Gamma$ is the glue type and $s \in \mathbb{Z}$ is the glue *strength*. Note that throughout the remainder of this paper, unless specifically stated, all glues will have strength 1 and as shorthand will be denoted simply by their glue types with the strength omitted.

Let $Q = \{\textit{latent}, \textit{on}, \textit{off}\}$ be the set of possible *states* for a glue. Intuitively, **on** is the “normal”, active state of a glue, meaning that it is either able to bind or currently bound. A glue in the **latent** state is inactive (and therefore unable to bind), and also has never been **on** (or **off**). A glue in the **off** state is also

inactive and unable to bind, but can never be (re)activated. We define an *active glue* as an ordered pair (g, q) where $g \in \Gamma$ is a glue type and $q \in Q$ is a state.

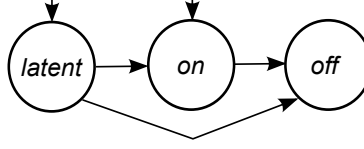


Figure 2: The valid state transitions for active glues

Active Labels As in the original TAM, we define a *label* as an arbitrary string over some fixed alphabet and labels will be used as non-functional (i.e. they don't participate in tile bindings) means of identifying tile types. Denote as Σ the set of all valid labels. For the self-assembly of patterns (e.g. the weak self-assembly of the Sierpinski triangle), tile labels are the mechanism used for distinguishing between groups of tiles, i.e. those “in” the pattern and those “outside” it. Experimentally, labels can be implemented as DNA loops which protrude above the surfaces of tiles for imaging purposes [15], and thus the motivation exists to allow for the modification of tile labels along with glues. Therefore, we define an *active label* is an ordered pair (x, q) where $x \in \Sigma$ and $q \in Q$.

Active Tiles and Transition Functions A *generalized active tile type* t is a unit square defined as a 4-tuple $t = (G, L, \delta, \Pi)$ where $G : D \rightarrow \mathcal{P}(\Gamma \times Q)$ is a function specifying the set of all active glues present on a given side, L is the set of all active labels, $\delta : D \times \Gamma \rightarrow \mathcal{P}(((D \times \Gamma \times \{\text{on}, \text{off}\}) \cup (\Sigma \times \{\text{on}, \text{off}\})))$ is the *transition function* and Π is a multiset over $(D \times \Gamma \times \{\text{on}, \text{off}\}) \cup (\Sigma \times \{\text{on}, \text{off}\})$. A generalized active tile type $t = (G, L, \delta, \Pi)$ is an *active tile type* if, for all $d \in D$ and for all active glues $(g, q), (g', q') \in G(d)$, $g \neq g'$. In other words, while a tile side may have multiple glues, there cannot be multiple copies of the same type of glue on a single side.

A transition function δ takes as input a direction $d \in D$ and a glue type $g \in \Gamma$ (i.e. we say that it is “fired” by the binding of glue type g on side d of t), and outputs a (possibly empty) set of *glue* or *label actions*, i.e., elements of $(D \times \Gamma \times \{\text{on}, \text{off}\}) \cup (\Sigma \times \{\text{on}, \text{off}\})$. Consider an active tile type $t = (G, L, \delta, \Pi)$ and suppose that $(g, q) \in G(d)$ for some $d \in D$ and $(d', g', q') \in \delta(d, g)$. Intuitively, if $m = \Pi(d', q')$ (i.e., the *multiplicity* of $(d', q') \in \Pi$) *before* δ “fires,” then $\Pi(d', q') = m + 1$ *after* executing δ . We assume for the sake of convenience that δ outputs the empty set for any pair of direction and glue on which δ is not explicitly defined. In other words, the binding of a glue on t fires the transition function, which can result in a set of “requests” for specific active glues and labels on t to transition into specified states.

As shorthand, let “−” represent “latent”, “1” represent “on”, and “0” represent “off”. Let $t = (G, L, \delta, \Pi)$ be an active tile type. For notational convenience, we will denote as $g_{\{q|\exists(d,g,q) \in \Pi\}}^p$ the active glue (g, p) of t satisfying, for some $d \in D$, $(g, p) \in G(d)$. We purposely omit the direction d from our shorthand notation because it will always be clear from the context. Note that, in our notation, the superscript specifies its current state (e.g. if the active glue is *on*, we write g^1), and the subscript represents its set of *pending state transitions* (i.e., the set of all glue actions $(d, g, q) \in \Pi$). For example, if $\{(N, g, \text{on}), (N, g, \text{off}), (N, g, \text{off})\} \subseteq \Pi$, then we write $g_{1,0,0}^p$. If there are no pending state transitions for g in Π , then we omit a subscript. Figure 2 shows the valid state transitions for active glues, which is restricted to *latent* being able to transition to either *on* or *off*, and “on” being able to transition to “off”. Note that once a glue is in the *off* state, there is no valid transition out of that state. Also, the only valid initial states for an active glue are *latent* or *on* since a glue which started in the *off* state could never interact and therefore could simply be removed.

Figure 3 shows an example active tile type with $G(N) = \{a^1, b^1\}$, $G(W) = \{bl^-, br^-\}$, $G(E) = \{bl^-, br^-\}$, $G(S) = \emptyset$ and $L = \{\text{Bot}\}$. Note that for the sake of convenience, in figures we tend to omit superscripts unless the glue state is *off*, so a glue with no superscript and a rectangular tab next to it represents that the glue is *on*, and with no such a tab is *latent*. Glues which are *off* will contain the “0” superscript, or be removed completely, to remove ambiguity. Figure 3 also includes two depictions of the tile’s transition

function. In this example, when the glue b on the North side of the tile binds, the tile's transition function specifies that glues b_l on the West side and b_r on the East side are requested to turn **on**.

An *active tile* is an instance of an active tile type which has its own pending sets for each of its active glues and labels—all of which sets must be initially empty.

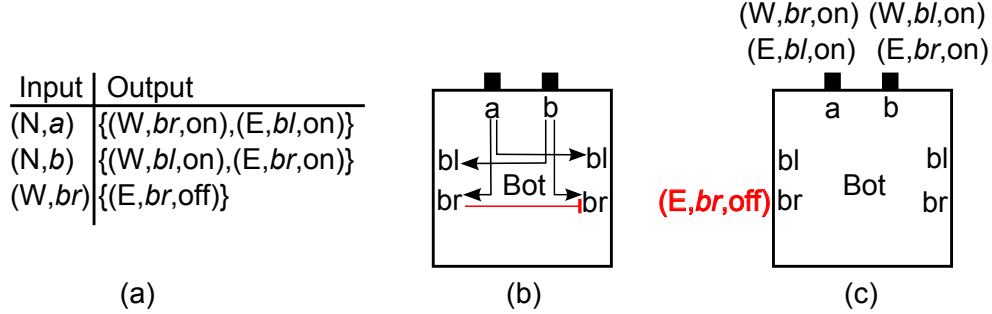


Figure 3: An example active tile with (a) a tabular representation of its transition function, (b) a graphical depiction of its transition function using lines and arrows, and (c) a semi-graphical depiction. In (b), an arrowed line pointing from glue g_1 to glue g_2 denotes that the binding of g_1 causes the glue action turning g_2 on to be put into g_2 's pending set P . A red line indicates that the action is to turn the destination glue off. In (c), the text descriptions of output actions for a given glue are placed next to its location on the tile edge. Note that for compactness we can also combine multiple actions. For example, if glue a also had output (W,bl,on) we could write $(W,(bl,br),on)$.

Active Supertiles Active supertiles in the STAM are defined similarly to supertiles in the 2HAM. Note that only glues which are in the **on** state can bind, and only those which are bound contribute to the τ -stability of a supertile. When two τ -stable supertiles can be translated so that they are non-overlapping and the abutting **on** glues can bind to create a cut strength of at least τ , we say that they are τ -combinable. A supertile A is said to be τ -breakable into supertiles B and C if there exists a strength $\tau' < \tau$ strength cut of the stability graph G_A that separates the tiles of A into supertiles B and C .

Active Supertile Combination Two active supertiles A and B may combine into active supertile C if the underlying supertiles for A and B are τ -combinable into the underlying supertile for C . When supertiles combine, all matched glues in the **on** state on the boundary between A and B are said to bind, and thus fire the transition functions of their respective tiles, causing the necessary states to be added to the pending sets of the targeted tiles.

Active Supertile Breaking An active supertile A can break into active supertiles B and C if the underlying supertile for A has a cut of less than τ -strength dividing A into the underlying supertiles for B and C .

An STAM system is a tuple (T, τ) where T is a set of initial active supertiles referred to as the *initial assembly set* of the system, and τ is a positive integer referred to as the *temperature* of the system. We further restrict that the initial assembly set only contains active super tiles $t = (G, L, \delta, \Pi)$ such that $\Pi = \emptyset$. For some problems such as the unique assembly of shapes and lines (see section 4), T is further restricted to contain only active supertiles that consist of singleton active tiles.

3.3.2 Producible Assemblies

The signal passing tile assembly model is defined in terms of the set of *producible* active supertiles $P_{T,\tau}$. $P_{T,\tau}$ is defined recursively as follows:

Base Set of Assemblies $T \subseteq P_{T,\tau}$

Combination Transition For any 2 active supertiles $A, B \in P_{T,\tau}$, if A and B are τ -combinable into active supertile C , then $C \in P_{T,\tau}$.

Break Transition For any supertile $A \in P_{T,\tau}$, if A is τ -breakable into active supertiles B and C , then $B, C \in P_{T,\tau}$.

Active Glue Transition Consider an active supertile $A \in P_{T,\tau}$. For each active tile $t \in A$ with $A(x, y) = t = (G, L, \delta, \Pi)$, for all $d \in D$, for all $(g, p) \in G(d)$ and for all $(d, g, q) \in \Pi$, there is a supertile $B \in P_{T,\tau}$ such that A and B differ only at location (x, y) as follows: $B(x, y) = t' = (G', L, \delta, \Pi')$ satisfying

1. $\Pi' = \Pi - \{(d, g, q)\}$ and
2. if $p = \text{on}$ and $q = \text{off}$ or $p = \text{latent}$ and $q = \text{off}$ or $p = \text{latent}$ and $q = \text{on}$, then for all $d \neq d' \in D$, $G'(d') = G(d')$, $(g, q) \in G'(d)$ and for all $q \neq q' \in Q$, $(g, q') \notin G'(d)$.

(This simply says that any supertile B which can be created from another producible supertile A by simply executing one of A 's pending actions is also producible.)

Active Label Transition Active label transitions are defined similar to active glue transitions.

3.3.3 Terminal Assemblies

The set of producible assemblies of an STAM system defines the collection of active supertiles that can occur throughout the assembly process. The *terminal* set of assemblies $\mathcal{A}_{\square}[T, \tau]$ of an STAM system (T, τ) defines the subset of producible assemblies for which no combination, break, glue transition, or label transition is possible. Conceptually, the terminal assembly set represents the sink state of the assembly system in which the system has been given enough time for all assemblies to reach a terminal state. The terminal set of assemblies is considered the output of an STAM system. In our constructions we are interested in designing systems that will have either 1) a unique terminal assembly that has a desired shape or 2) has a collection of terminal assemblies in which the desired target assembly is the largest or 3) the desired target assembly is the only assembly which contains a specially designated *marker* tile.

Please see Section 3.4 for an example STAM system and several transitions and producible assemblies.

3.4 Example STAM System

Here we give a small example STAM system along with descriptions of several transitions and producible assemblies.

3.4.1 Example tile types

Figure 4 shows a tile set that will be used for the following example. We will refer to active tiles by their labels for convenience. The “Bot” tile has four glue actions defined for its transition function, all of which turn **on** glues. The “MM” tile includes two glue actions which turn **off** glues: when glue e on the North binds, both that glue and glue b on the South get **off** transitions placed into the Π multiset of the active tile, which will eventually cause them to be turned **off**. The transition function for the “MR” tile contains a label action, namely when the c glue on the South binds.

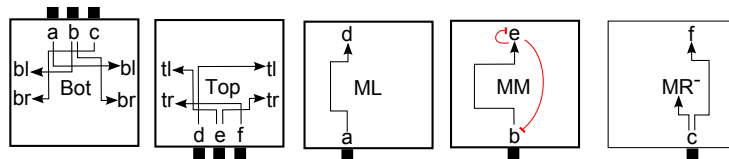


Figure 4: An example active tile set.

3.4.2 Example STAM transitions

Define an STAM system $(T, 2)$ where T is the tile set shown in Figure 4. Figure 5 shows a set of transitions and producible assemblies within that system. Figure 5(a) shows a “Top” and “MM” tile in their initial configurations (with empty Π sets for all active tiles as required, and all superscripts explicitly shown for reference). Figure 5(b) shows the supertile resulting from a combination transition performed on one “Top” and one “MM” tile. Note that the binding of the two e glues causes four glues to have state transitions added to the Π multisets the belong to their respective active tile. Figure 5(c) shows the supertile resulting from a glue flip transition performed on the supertile in Figure 5(b), namely the tl glue on the West side of the “Top” tile transitioned to state on and that pending transition was removed from the Π multiset belonging to its active tile. Figure 5(d) shows the supertile resulting after two glue flip transitions are performed on the supertile in Figure 5(c), namely glue tr on the East of “Top” is turned on and e on the North of “MM” is turned off.

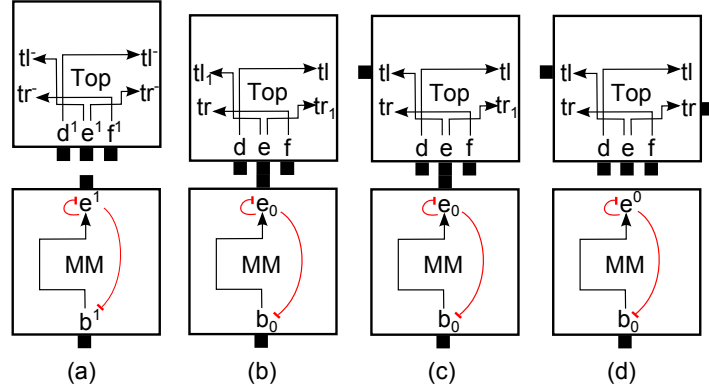


Figure 5: Example transitions for the tile types in Figure 4 assuming a temperature $\tau = 1$ system

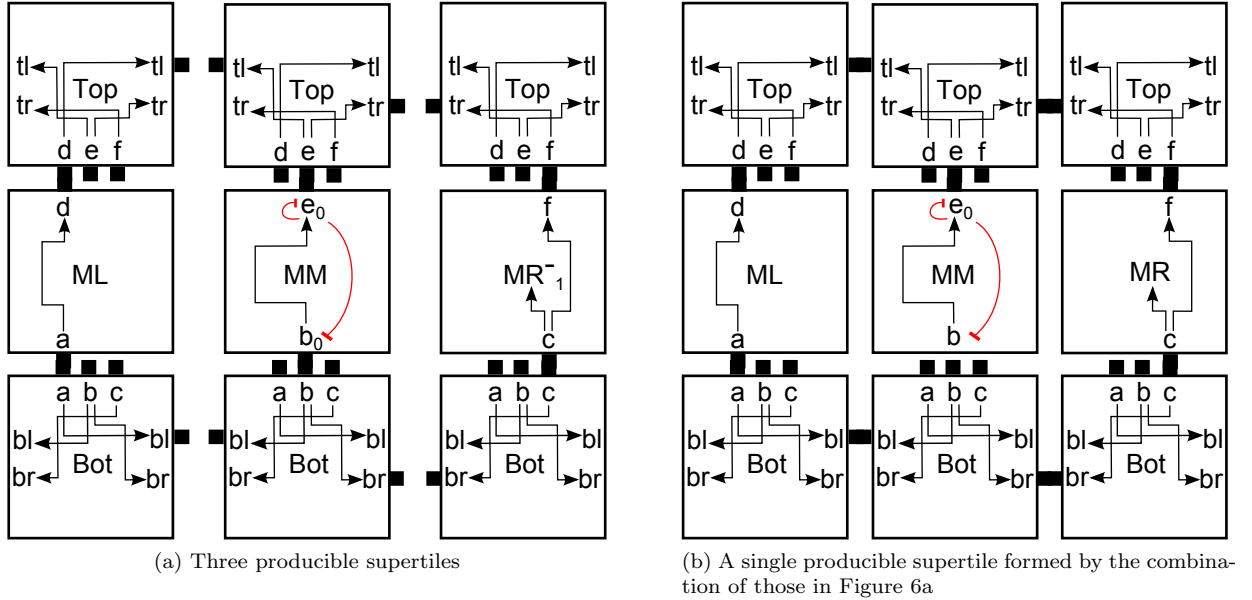


Figure 6: Producible supertiles in the temperature $\tau = 1$ system defined with the tile types in Figure 4

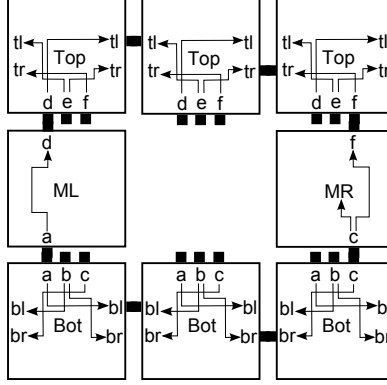


Figure 7: Another producible supertile in the temperature $\tau = 1$ system defined with tile types from Figure 4. Note that it is not terminal as a new “MM” tile could attach in the center, repeatedly falling off and being replaced an infinite number of times.

3.5 STAM Metrics

Within this paper we consider the problem of assembling precise shapes, simulating Turing machines, and the strict self-assembly of infinite fractals. To measure how efficiently we can solve these assembly problems within the STAM model, as opposed to alternate models, we consider a number of natural metrics designed to measure experimental feasibility with respect to likely STAM implementations, the most important being DNA origami based tiles which implement glue actions through DNA strand displacement.

3.5.1 Tile Complexity

The tile complexity of an STAM system (T, τ) is $|T|$, the number distinct active supertiles that serve as the initial set of assemblies for the system. In the case that T is restricted to contain active supertiles consisting of only singleton active tiles, this metric denotes the number of distinct tile types that must be engineered to implement the STAM system. In general, the metric describes the number of distinct assemblies that must be engineered to implement the system.

3.5.2 Signal Complexity

The signal complexity of an STAM system is the maximum number of glues that occur on the face of any tile from T . Within the STAM system, each glue along a tile face is potentially set up to fire a transition function that triggers a set of local glue actions upon binding. In practice, such transition triggers can be set up through a sequence of DNA strand displacements. Setting up a small network of such displacement chains on the face of a DNA origami tile is experimentally quite feasible. However, the cost and complexity grow substantially as the number of distinct displacement reactions grows, making the signal complexity a key metric for experimental feasibility.

3.5.3 Fuel Efficiency

Fuel efficiency is a metric that is considered in the context of simulating a Turing Machine and denotes the number of tiles that are used up (and cannot be reused) per computation step of the Turing machine being simulated. Our result constitutes the first Turing simulation self-assembly system that achieves $O(1)$ fuel efficiency.

3.5.4 Space Complexity

Space complexity is a metric that is considered in the context of simulating a Turing Machine and denotes the size of the assembly that represents the current tape and state of the Turing machine after a given computation step.

3.5.5 Temperature

For an STAM system (T, τ) , the temperature value τ denotes the number of glue bonds required for two assemblies to combine. Higher temperature systems realize finer grained bonding strength differences than lower temperature systems and may be comparably harder to implement in practice. One of the most straightforward class of systems to implement may be temperature $\tau = 1$ systems in which any single bond is sufficient to cause two assemblies to attach. In particular, strength $\tau = 1$ systems do not require the implementation of error prone *cooperative bonding* in which attachment may be based on 2 or more glues spread across 2 or more distinct tiles.

4 Efficient Construction of Linear Assemblies

In the aTAM, $n \times 1$ lines are inherently inefficient to self-assemble, requiring the worst possible tile complexity of n . However, using signal-passing tiles it is possible to create $n \times 1$ lines using no more than 6 tile types, regardless of the value of n . Of course, the value of n must somehow be encoded in the system, but rather than requiring n tile types as in the aTAM, in the STAM the value of n can be efficiently encoded using $\log n$ bits to require $O(1)$ tile types with $O(\log n)$ signal complexity. The construction we use makes use of a recursive doubling strategy where random tile binding events randomly assign the fate of each supertile at each stage, and is of independent interest in terms of using signal-passing tiles to perform recursive assembly of structures.

We first state our result for the special case where n is a power of 2 and provide a detailed construction. The exact tile set is given in Figure 9. A high-level description of the recursive doubling technique is given in Figure 8, and a small example of the assembly process for a length 16 line is given in Figures 10-11. This result can be generalized to work for any positive integer n by increasing the glues on the tile types of the powers of 2 construction and adding 3 more tile types, while keeping the signal complexity at $O(\log n)$, yielding the final result stated in Theorem 4.2.

Theorem 4.1. For every $k \in \mathbb{N}$, there exists an STAM system $(T, 1)$ which uniquely assembles a $2^k \times 1$ line. Moreover, T contains 4 tiles, has signal complexity $O(k)$, and does not use glue deactivation.

4.1 Proof of Theorem 4.1

Proof. Define $f(A)$ as a naming function that takes as input a supertile A and returns a string $s \in \{a_k, b_k, x_k, y_k | k \in \mathbb{N}\} \cup \lambda$. For an arbitrary input A , if the height is 1 and the width is 2^n for some $n \in \mathbb{N}$, $f(A)$ returns a_k if A has active glues labeled ax_k and ay_k on its eastern edge, b_k if A has active glues labeled bx_k and by_k on its eastern edge, x_k if A has active glues labeled ax_k and bx_k on its western edge, and y_k if A has active glues labeled ay_k and by_k on its western edge. Otherwise, $f(A)$ returns λ .

Initially, the system consists only of copies of tiles from T , namely the four singleton tiles whose names correspond to their identities as defined by f : a_0, b_0, x_0 , and y_0 . Assembly proceeds by recursive combination of these tiles into larger supertiles. Figure 8 shows how producible assemblies can join; for example, a_0 can attach to the western edge of either x_0 or y_0 , producing a_1 and y_1 , respectively. For a detailed view of the internal signal structure of the tiles that allows for this behavior, see Figure 9. Note that each assembly possesses two exposed external glues which allow non-deterministic binding to occur, a critical aspect of the construction.

The binding event that joins two assemblies together will propagate a signal either left or right (via L_k or R_k glues), depending on what identity the newly formed assembly is supposed to take on. For example: if

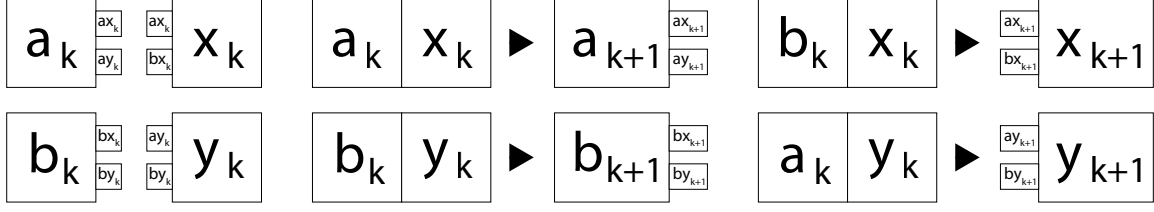


Figure 8: Four recursively defined types of assemblies: a_k, b_k, x_k, y_k . For example: a_k is composed of a_{k-1} and x_{k-1} and has glues ax_k and ay_k activated on its eastern edge.

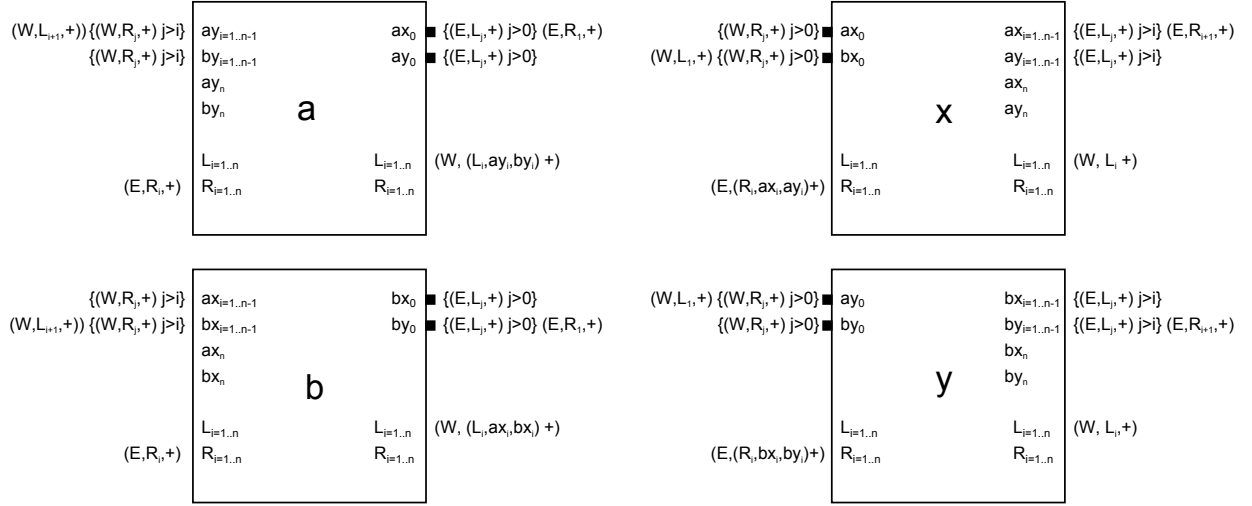


Figure 9: The tileset for producing a $1 \times n$ line, where n is a power of 2.

tile a_0 attaches to tile x_0 , it can only do so by virtue of glue ax_0 . At this point, the assembly “knows” that it is now supposed to be a_1 , and activates a glue on its right face: R_1 . Each tile in the assembly can receive R_1 on its left edge (because it is already activated), and activate it on its right edge, effectively propagating a signal across multiple tiles. When the rightmost tile (in this case, x_0) receives the signal, it activates glues corresponding to the appropriate identity, i.e., ax_1 and ay_1 . At this point, the assembly can properly be called a_1 . The tileset is designed in such a way that the external edge tiles in any particular assembly are known. For example, it can be observed that the leftmost and rightmost tiles in any a_k will be a_0 and x_0 , respectively. This fact allows a simple signal transduction: when an edge tile receives a particular signal, there is exactly one identity that particular assembly can adopt via glue activation.

The assembly process can be thought of as being isomorphic to a staged assembly process, where at every stage k such that $0 < k \leq \log_2 n$, four new assemblies are formed from assemblies produced at stage $k - 1$. These can combine with one another in four ways, and so on. This process terminates at the final mixing, when $k = \log_2 n$, because the final binding event does not trigger signal propagation, resulting in terminal $n \times 1$ lines. Note that there are 4 unique terminal assemblies, all of which are $n \times 1$ lines. \square

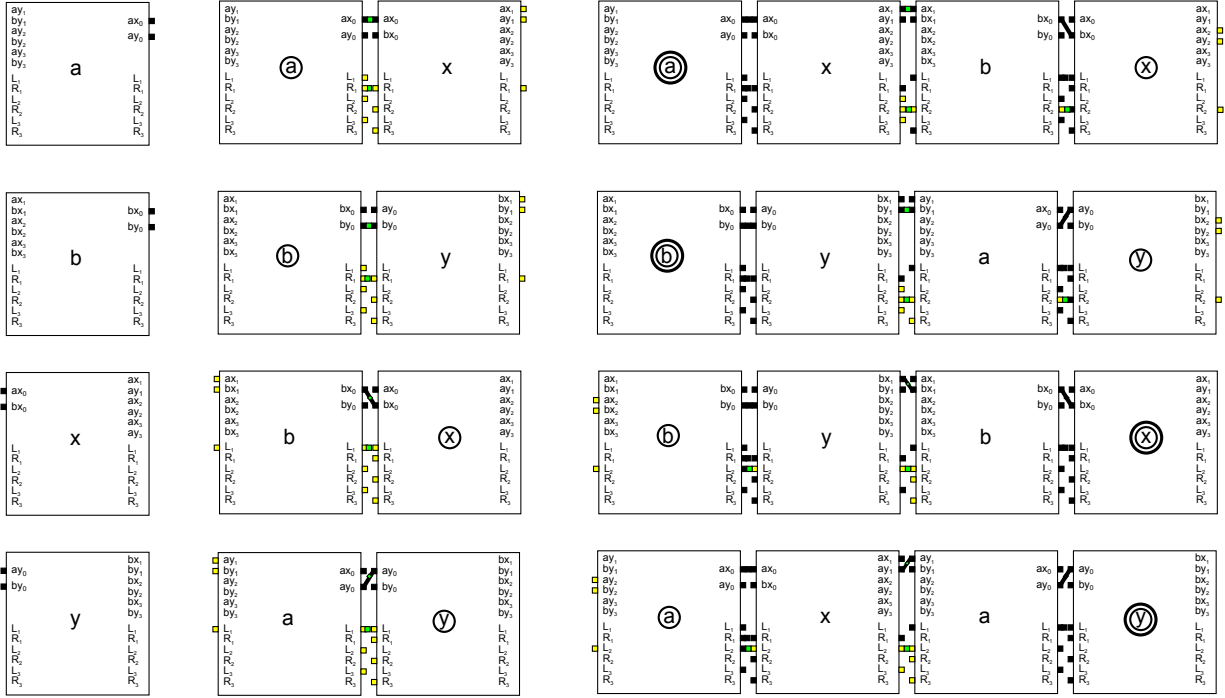


Figure 10: Example assembly of a line of length $2^4 = 16$, or $n = 4$ (part 1 of 2). Circles indicate the identity of a block of tiles at a particular stage given by the number of circles. For example, in the upper right, a is circled twice to indicate that this block of four tiles has identity a_2 .

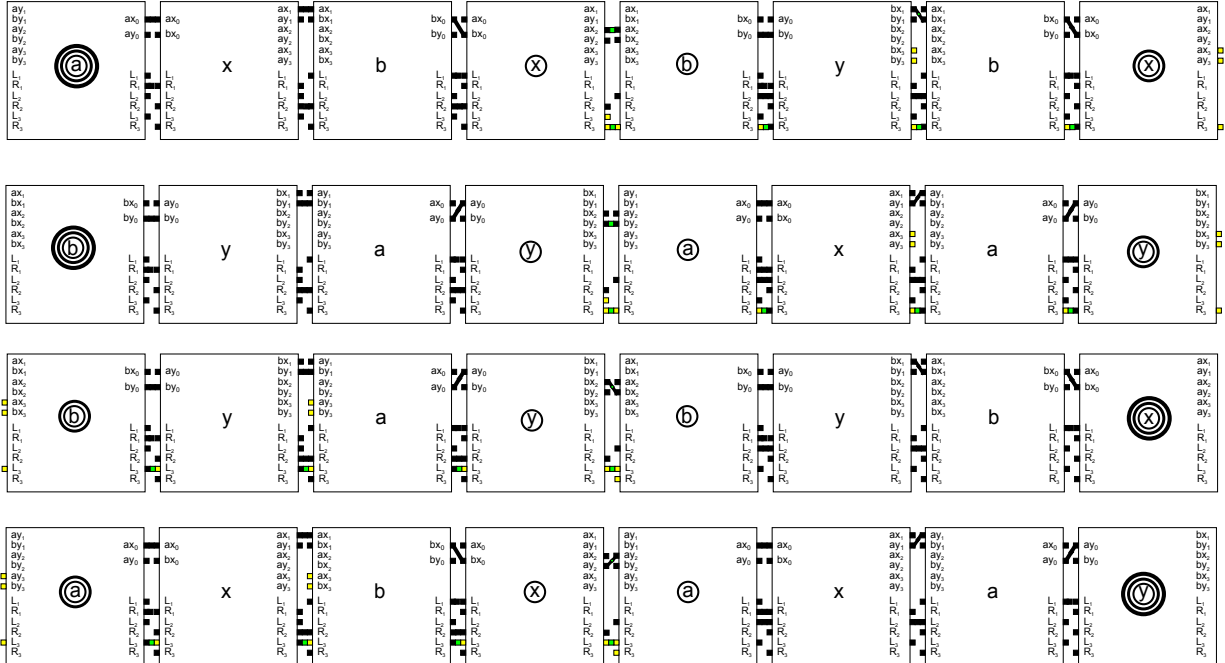


Figure 11: Example assembly of a line of length $2^4 = 16$, or $n = 4$ (part 2 of 2). The final step of combinations, where each length 8 segment combines with another to form a terminal assembly of length 16 and which cause no glue activations, is not shown.

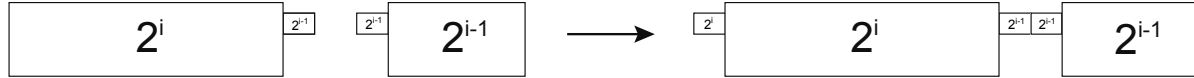


Figure 12: Line segments of lengths corresponding to the powers of 2 that sum to n must be joined to produce line of length n . Once line segments of unequal length have joined, signals ensure that growth continues only by addition of the correct unequal line segments.

4.2 Proof of Theorem 4.2

Theorem 4.2. For every $n \in \mathbb{N}$, there exists an STAM system $\mathcal{T} = (T, 1)$, with $|T| = O(1)$, which uniquely assembles an $n \times 1$ line. Moreover, the signal complexity of T is $O(\log n)$ and \mathcal{T} does not use glue deactivation.

Proof. Let $n \in \mathbb{N}$ be an arbitrary positive integer. Note that n can be written as a sum of powers of 2, i.e. $n = \sum_{i=0}^{\lfloor \log n \rfloor} 2^i b_i$ where b_i is the i th bit of the binary representation of n . Just as n can be written this way, the line of length n can be composed of segments with power of 2 lengths, each composed during the construction of the line of length 2^k where $k = \lfloor \log n \rfloor$, as described in the previous section. The joining of line segments corresponding to the powers of 2 that sum to n is shown in Figure 12.

The tiles from the previous section are modified to include $O(\log n)$ more glues each that direct the assembly of the powers of 2 shown in Figure 12. The resulting tile set still has signal complexity $O(\log n)$ and is shown in Figures 13-14. Depending on the specific bit sequence of n , the tile types a' , x' , and x'' may or may not be included in the tile set for that n . For each power of 2 which is a component of the binary representation of n , there is an assembly which stops doubling at that length and exposes a j_i glue on its eastern edge that allows it to bind to the bar of length corresponding to the next lower power of 2. If n is odd, then the x_0 tile is the only tile with an initially exposed j glue (namely, j_0) on its west side. If n is even and $b_1 = 1$ (the second bit in its binary representation is 1), then tile a' is used in the construction and will present the first west facing j glue, namely j_1 . If neither b_0 or b_1 are 1, then tile type x'' is used to ensure that there will be lines of length equal to the smallest included power of two which will expose a j glue on the west and will stop any continued doubling. For all other included powers of 2 (assuming that n is not a power of 2), the x' tile will be included in the tile set and will ensure that bars of lengths corresponding to each such power of 2 will stop doubling at those lengths and present j glues on their east side.

Once a bar binds via a j glue on its east, it sends a signal to the west side to open the next j glue and allow subsequent attachment. Finally, the bar of the largest power of 2 will also stop doubling and send a signal (starting from either an a or b tile) to the east allowing for the attachment of the line consisting of all smaller powers of 2.

□

5 Fuel Efficient Turing Machines

Showing that the original abstract Tile Assembly Model is computationally universal is a simple matter of designing a tile assembly system which can simulate a universal Turing machine, as originally shown in [32], and later expanded upon in [6, 8, 11, 22]. While displaying the computational power of the aTAM (and variants prior to the STAM), a common drawback of the constructions has been the number of tiles utilized during the formation of the assembly which simulates the computation, which, in this paper, is referred to as the fuel efficiency of the simulation.

For prior constructions, it has been necessary to make a new copy of the entire tape of the Turing machine between each computational step, with the new copy identical to the original except for the slight difference of a mere two tape cells indicating: 1. the output value in the tape cell left by the tape head, and 2. the tape cell marking the current location of the tape head. This full-scale copying of the tape, including the vast majority of cells which are unchanged, is wasteful in terms of the number of tiles required, experimentally very error prone due to the huge number of tile attachments required, and results in enormous assemblies.

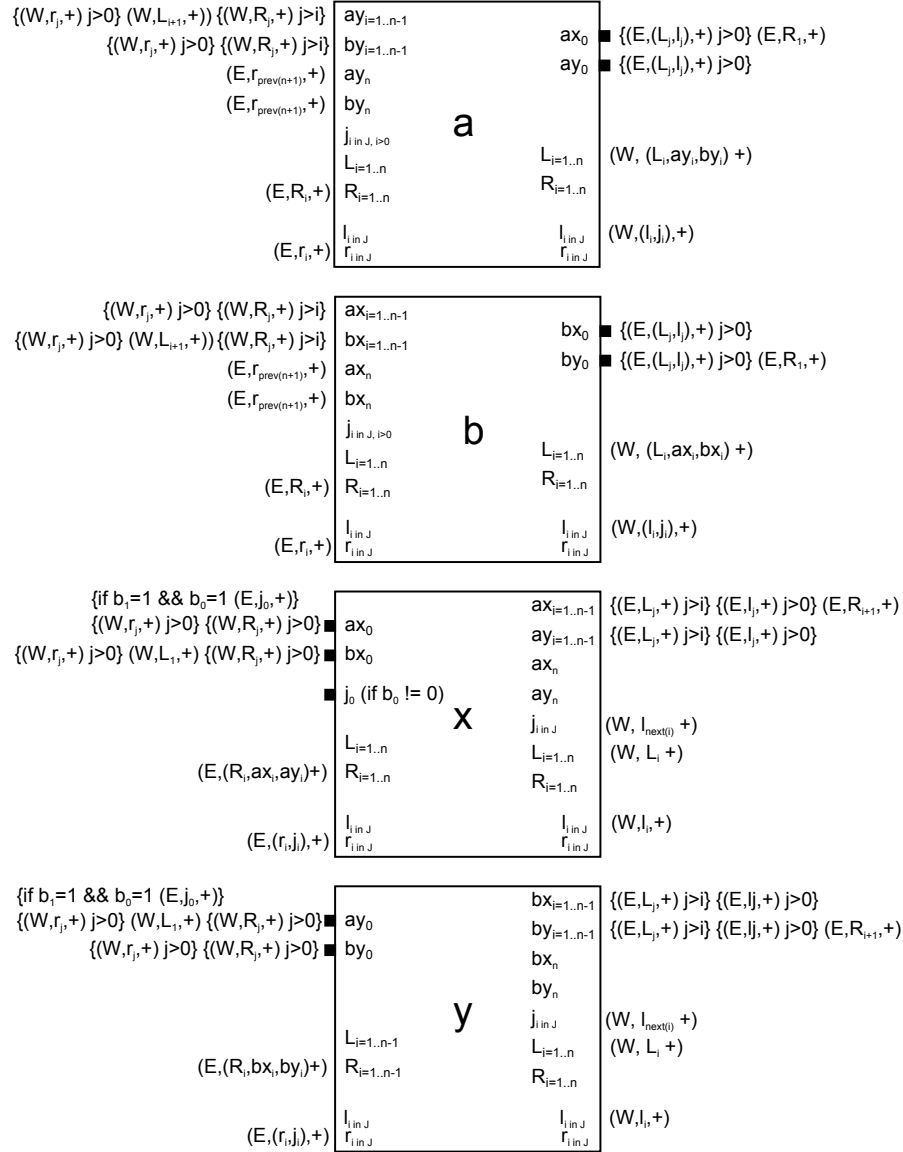
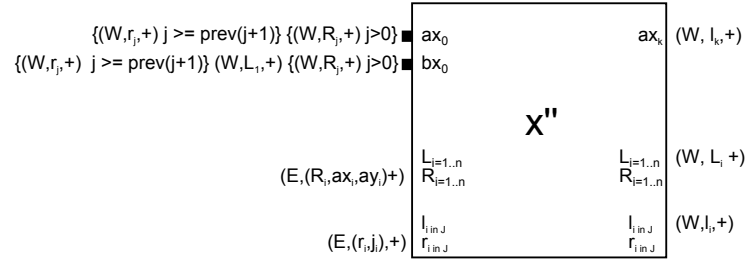
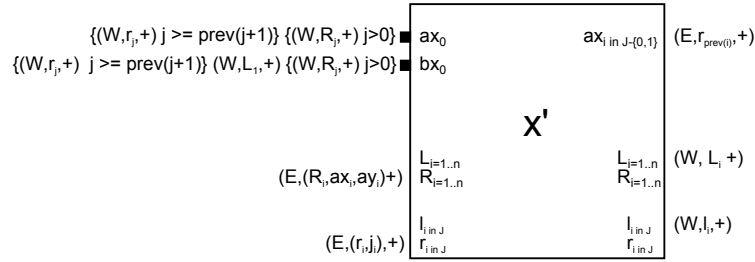


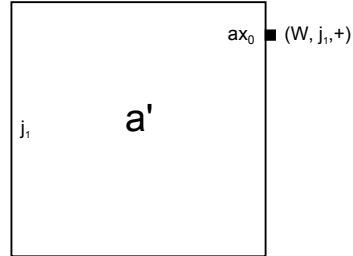
Figure 13: The tile set (part 1 of 2) for self-assembling an $n \times 1$ line for arbitrary n . Notation: $b = bin(k)$, i.e. k represented in binary, e.g. for $k = 19$, $bin(k) = 10011$. $b_i = i$ th bit of b (where b_0 is the least significant bit), e.g. for $k = 19$, $b_0 = 1$, $b_1 = 1$, $b_2 = 0$, $b_3 = 0$, and $b_4 = 1$. $n = \lfloor \log k \rfloor$, e.g. for $k = 19$, $n = 4$. $I = \{i | b_i = 1\}$ as an ordered set, e.g. for $k = 19$, $bin(k) = 10011$, $I = \{0, 1, 4\}$. $J = I - \max(I)$, namely I without the largest element, e.g. if $I = \{0, 1, 4\}$ then $J = \{0, 1\}$. $prev(i)$ = the element of J immediately less than i , e.g. $prev(4) = 1$. $next(i)$ = the element of J immediately greater than i , e.g. $next(0) = 1$.



x'' : used for $k \neq n$ if $k > 1$, $b_k = 1$, and $b_m = 0$ for all $m < k$



x' : used for all $k \neq n$ if $k > 1$, $b_k = 1$, and $b_m = 1$ for some $m < k$



a' : used if $b_1 = 1$ & $b_0 = 0$

Figure 14: The tile set (part 2 of 2) for self-assembling an $n \times 1$ line for arbitrary n . See Figure 13 for notation.

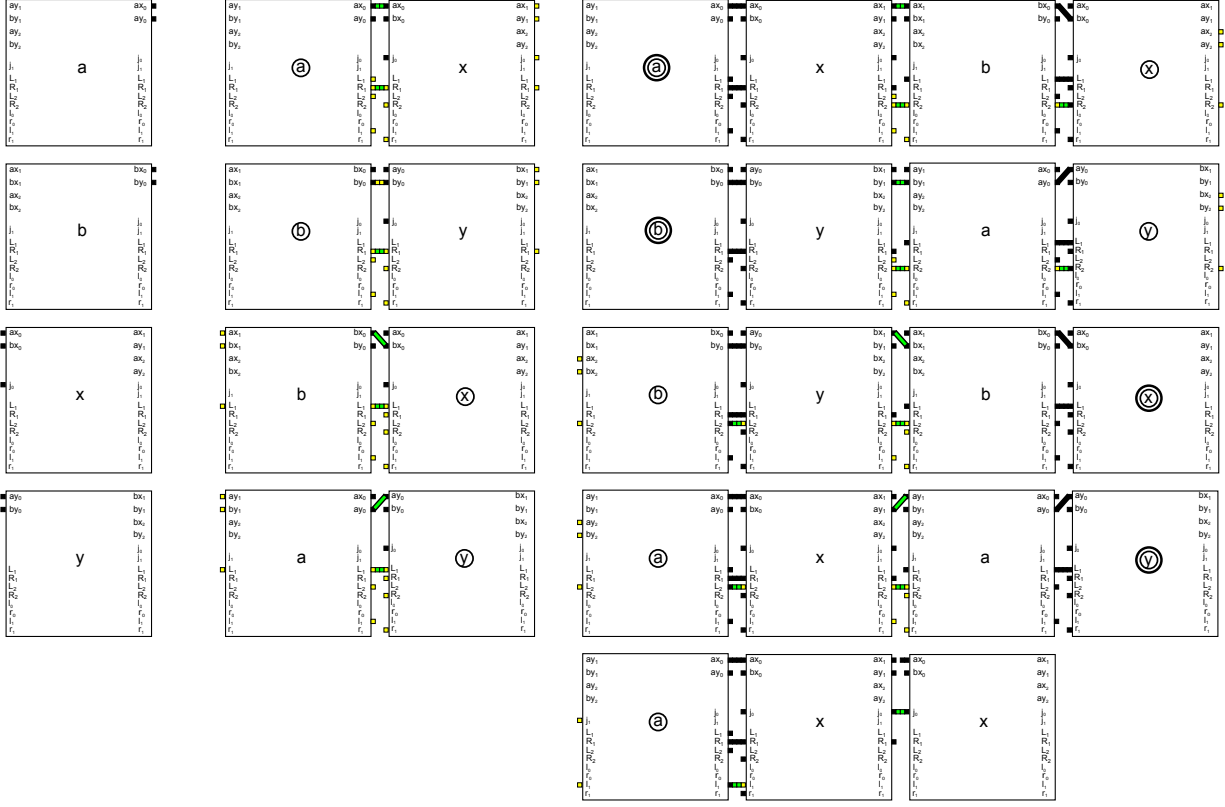


Figure 15: Example assembly of a line of length 11 (part 1 of 2).

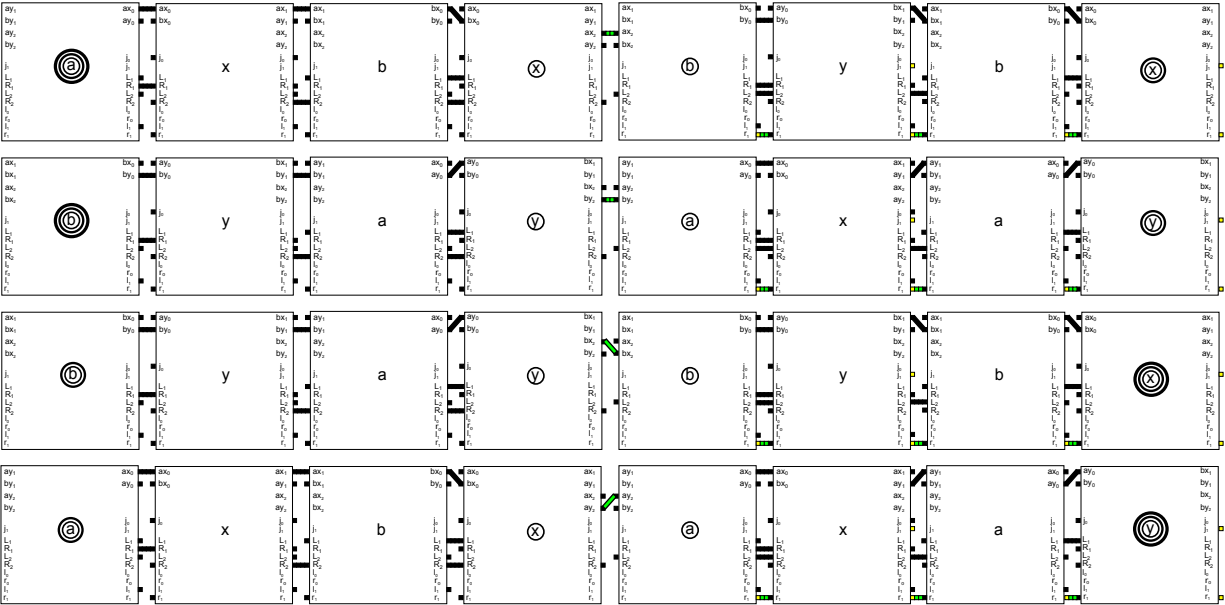


Figure 16: Example assembly of a line of length 11 (part 2 of 2). The final step of combinations, where each length 8 segment combines with the single assembly of length 3 from Figure 15 is not shown since there are no additional glue activations which occur, merely the binding of each assembly of length 8 with one of length 3 to form final line assemblies of the target length, 11.

In this section, we exhibit a construction which is capable of simulating a universal Turing machine in the STAM, but while doing so only requires a small constant number of tiles (never more than 7) as fuel for each computational step and maintains an assembly which consists of a number of tiles which is only twice the total number of tape cells used by the Turing machine up to that step. It is possible that with significantly fewer binding events in the STAM construction than in those of previous models (even taking into account those used for signaling), it may be the case that the number of errors which occur could decrease, assuming, of course, that the mechanism which carries out the transition function is sufficiently error-free.

Throughout this paper, and without loss of generality, we define Turing machines as follows. Let M be an arbitrary single-tape Turing machine, such that $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ with state set Q , input alphabet $\Sigma = \{0, 1\}$, tape alphabet $\Gamma = \{0, 1, _ \}$, transition function δ , start state $q_0 \in Q$, accept state $q_{\text{accept}} \in Q$, and reject state $q_{\text{reject}} \in Q$. Furthermore, M begins in state q_0 on the leftmost cell of the tape, expects a one-way infinite-to-the-right tape, and is guaranteed to never attempt to move left while on the leftmost tape cell.

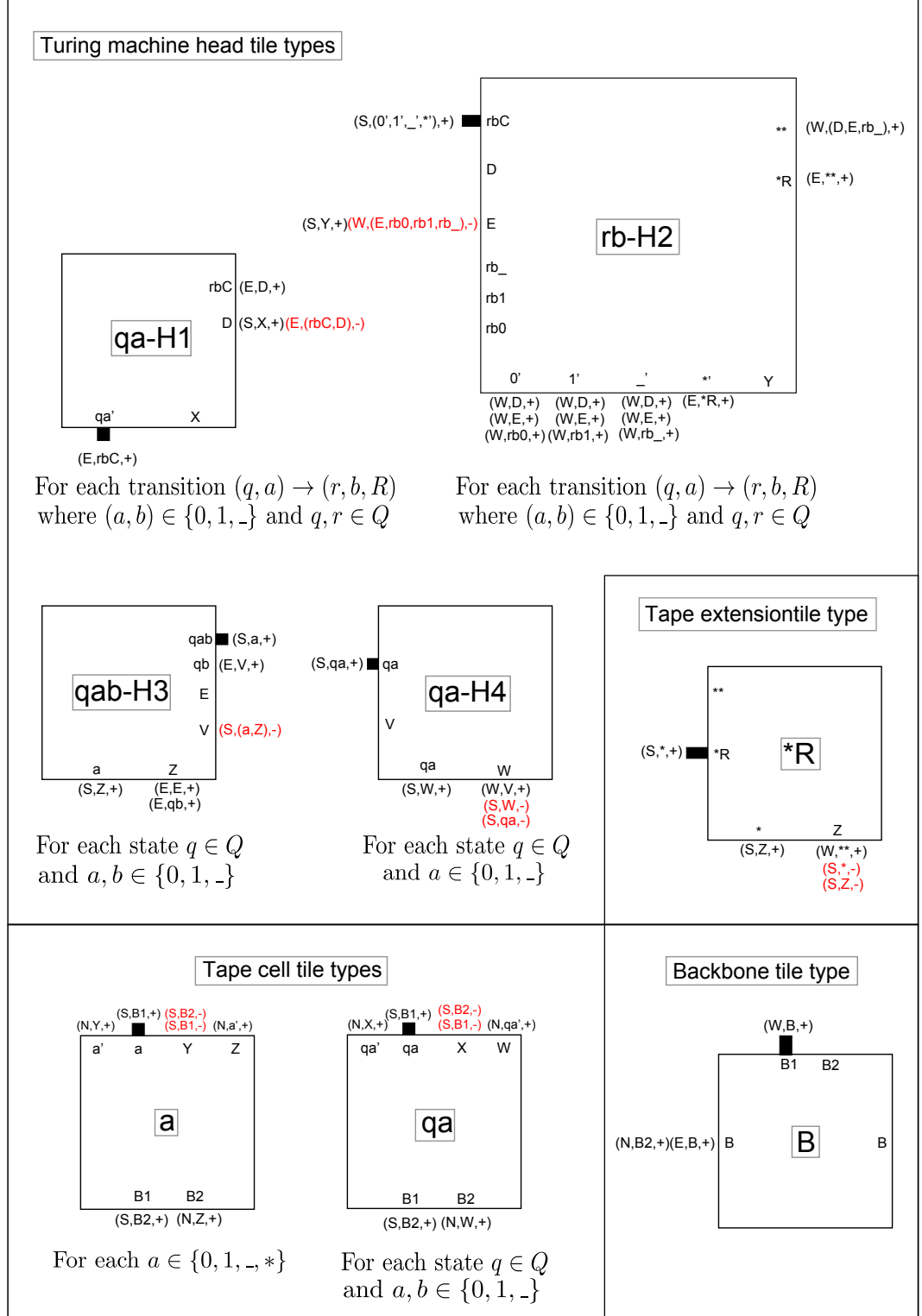
Theorem 5.1 (Fuel efficient Turing machines). For any Turing machine M with input $w \in \{0, 1\}^*$, there exists an STAM system $\mathcal{T}_{M(w)} = (T_{M(w)}, 1)$ with tile complexity $O(|Q|)$, signal complexity $O(1)$, and fuel efficiency $O(1)$, which simulates M on w in the following way:

1. $T_{M(w)}$ contains an active supertile consisting of $2|w| + 2$ active tiles representing w and M 's start state.
2. If M halts on w , then $\mathcal{A}_{\square}[T_{M(w)}, 1]$ contains exactly one supertile with > 3 tiles and that supertile contains exactly one *ACCEPT* (*REJECT*) tile if M accepts (rejects) w .
3. If M does not halt on w , then $\mathcal{A}_{\square}[T_{M(w)}, 1]$ contains exactly 0 (terminal) supertiles with > 3 tiles.

Our proof of Theorem 5.1 is by construction. Here, we provide a brief overview.

Our construction works by utilizing a set of tile type templates that, along with the definition of a Turing machine M , are used to generate the set of active tiles which are specific to M . The construction uses a pair of tiles to represent each tape cell, with one tile representing the value (0, 1, or $_$) of that cell and one tile providing a “backbone” which the other attaches to and which also attaches to the backbone tiles of the cells to its left and right. Additionally, there is a special tile for the tape cell representing the rightmost end of the tape, and also, at any given time, exactly one tape cell which also represents one state of M along with the tape cell value. The location of the tape cell with that information denotes the location of M 's tape head at that point, and the value of the state tells what state M is in. Transitions of M occur in a series of 4 main steps in which tiles bind to the north of the tape cell denoting the head location, then to the north of the tape cell to the immediate left or right (depending on whether or not M 's transition function specifies a left or right moving transition from the current state while reading the current tape cell value), and along the way cause the dissociation of the tiles representing the tape cell values in both locations and their replacement with tiles which represent the correct output tape cell value of the transition and correctly record the new state and head location at the tape cell immediately to the left or right. Due to the asynchronous nature of glue deactivations, and also the necessity that any “junk” assemblies produced (i.e. those assemblies which break off from the assembly representing the Turing machine tape and which don't contribute to the final “answer”) must not be able to attach to any portion of any supertile which represents any stage of the computation, junk assemblies are produced as size 2 or 3 so that any activated glues which would otherwise be able to bind to another supertile are hidden between the tiles composing the junk assembly. In such a way, $M(w)$ is correctly simulated while requiring only a constant number of new tiles per simulated transition step, and all junk assemblies remain inert and at size either 2 or 3. If $M(w)$ halts, there will be one unique, terminal supertile which represents the result of that computation and is of size > 3 . If $M(w)$ does not halt, only the junk assemblies will be terminal.

Our proof is by construction. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ be a Turing machine and $w \in \{0, 1\}^*$ be the input. We define the finite set of tile types $T_{M(w)}$ in relation to M as shown in Figure 17. Note that the tile types presented are generic templates applicable for generating the tile types specific to a given M (with the exception of the “tape extension tile type”, “backbone tile type”, and the left “tape cell tile type”



which are used for all M). Also note that the tile type templates provided for the “Turing machine head tile types” are specific to transitions which move the tape head to the right, and those for left-moving transitions are simply mirror images with a unique set of glues (e.g. every glue is prefixed with “ L ”) defined specifically for those transitions. The actual tile types that would be generated for a specific M can be determined by substituting each valid variable value (as specified below each tile type template) into glue labels, and potentially consist of a large number of actual tile types generated by each template definition.

The tiles in the “Tape cell tile types” and “Backbone tile type” sections of Figure 17 are used to compose the tape. The tape itself, as shown in Figure 18, consists of a row of east-west connected “backbone” tiles, each of which is connected on its north to a tile representing a tape cell. The backbone is used to keep the entire assembly connected during the process of tape cell replacement. The possible values for tape cells are: (1) “0”, (2) “1”, (3) “_” (a blank), (4) “*” (which represents the currently rightmost tape cell), or (5) $q \times \{0, 1, _ \}$ where $q \in Q$ (at any point there is exactly one such tape cell which represents the location of the head and M ’s current state along with the contents of the tape cell that the head is currently reading). The tiles in the “Turing machine head tile types” group are used to simulate the actions of the head, which can be understood as four high-level steps ($H1, H2, H3$, and $H4$ - to be explained shortly). The “Tape extension tile type” is used to extend the tape by one additional blank symbol (“_”) if and when the head reaches the rightmost end of the tape, thus allowing the assembly to simulate a one-way infinite-to-the-right tape.

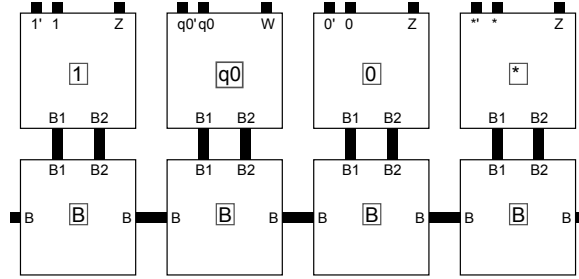


Figure 18: Start state of the Turing machine M simulation example assuming a tape of “100*” with M in state q and currently reading the leftmost 0.

The easiest method of explanation of this construction is to step through an example transition, which we now do. The specific example is depicted in the series of Figures 18, 19 and 20. Assume that M is currently in state “ q ” and reading symbol “0”, the output symbol is “1”, the next state should be “ r ”, and the head should move right (i.e. the simulated transition is $(q, 0) \rightarrow (r, 1, R)$). Glues which are currently on are shown as black, glues which have just been turned on or connected in the current step are shown as bright blue, glues which are queued to turn off are shown as red, and glues which are currently off or latent have been hidden. Triggered actions have been depicted by arrows.

The transition begins from the $q0'$ glue on the $q0$ tile. As depicted in Figure 19(a), the $q0$ - $H1$ tile encodes the information to read a 0 while in state q , change it to a 1, move the head to the right, and change to state r . The values r , 1 and “move to the right” are implicitly stored by $r1$ - $H2$ as shown in Figure 19(b). $r1$ - $H2$ then connects to the tape cell tile below to read the symbol on that cell (a 0 here), and sends a message back to $q0$ - $H1$ which causes $q0$ - $H1$ and the now obsolete tape cell beneath it to dissociate as a pair, as shown in Figure 19(c)-Figure 19(f). When $r1$ - $H2$ connected with $q0$ - $H1$, $r1$ - $H2$ didn’t know what the tape cell value beneath it was, so it had to activate all four possible glues, which are $0'$, $1'$, $_'$, and $*'$. Since only the $0'$ binds, $r1$ - $H2$ can pass along the value of 0 in its next glue $r10$ to a tile of type $r10$ - $H3$. Here the $r10$ means the next state will be r , the tape cell of the current head position will change to 1, and the symbol of next head position is 0. Then, in Figure 19(h)-Figure 19(l), the $r10$ - $H3$ tile causes a new tile representing a tape cell value of 1 to fully bind to the backbone and activate its $1'$ glue, which completes the transition of that tape cell. After that 1 tile is connected to the backbone, it sends a message to $r1$ - $H2$ that allows it to dissociate, as shown in Figure 19(l)-Figure 19(o). Finally, the $r0$ - $H4$ tile binds to $r10$ - $H3$ ’s $r0$ glue, allowing it to facilitate the binding of the $r0$ where the 0 previously was, and eventually dissociate along

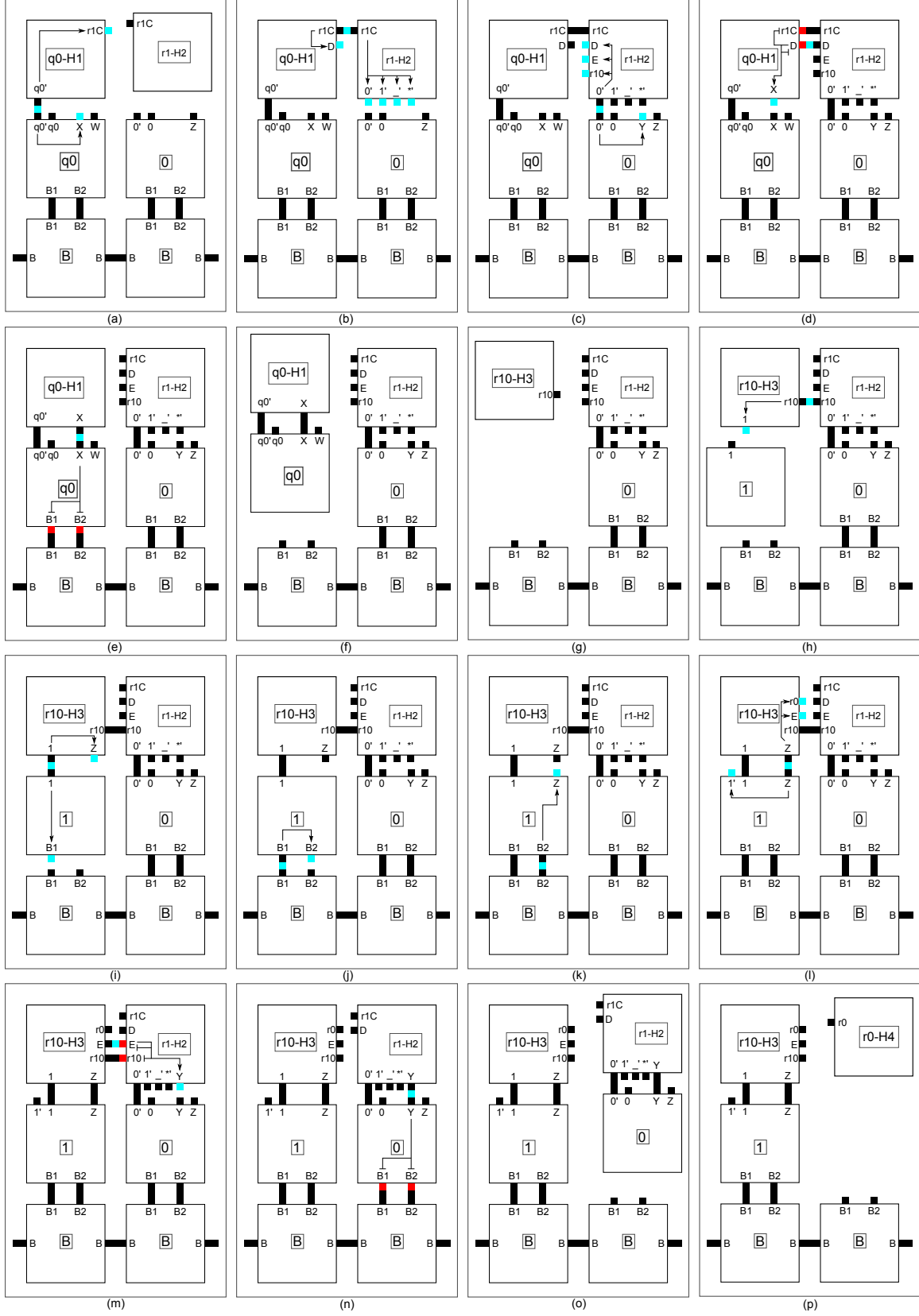


Figure 19: High-level sketch of the simulation of a transition, part 1 of 2

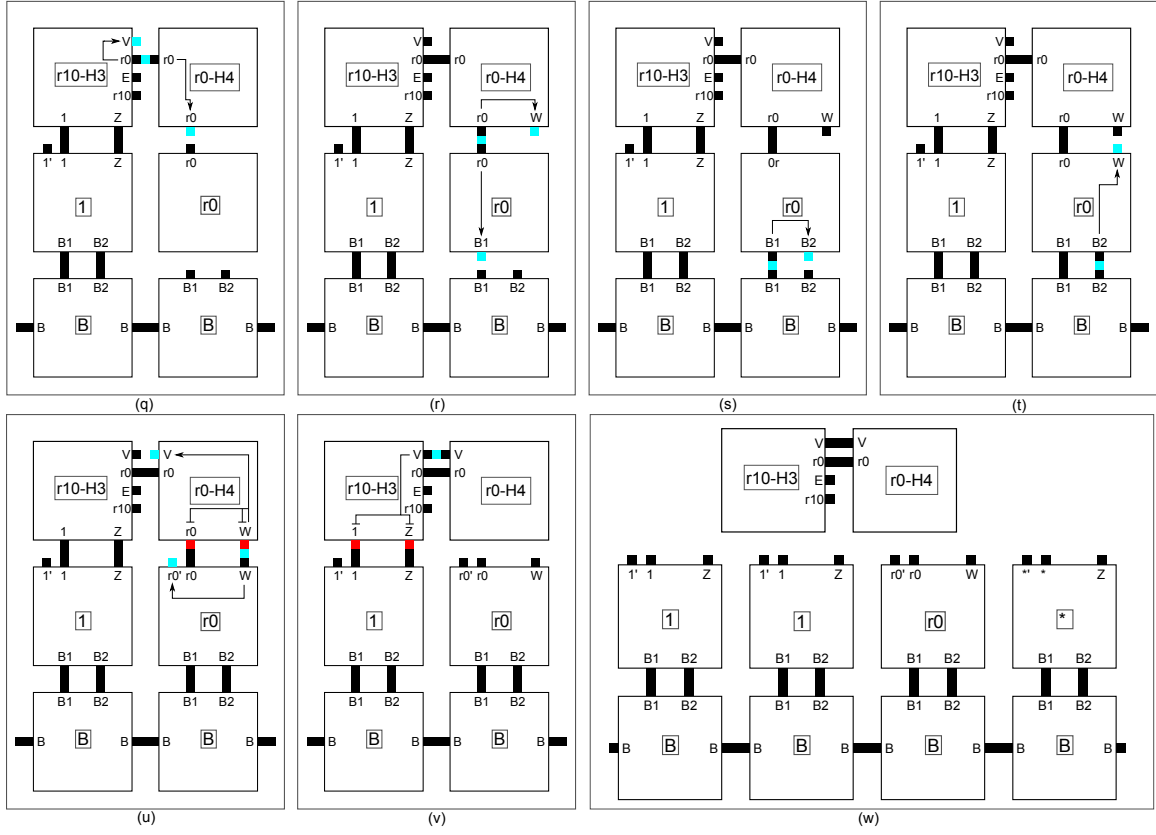


Figure 20: High-level sketch of the simulation of a transition, part 2 of 2

Following the above design techniques makes it necessary that for the leftmost and rightmost junk assemblies to detach, they must be completely inert and unable to bind to any other supertiles in the system. The middle supertile, with the $r1C$ and D glues still active, on the other hand, ensures that it cannot bind to any other supertile by utilizing a form of geometric hindrance. Essentially, the $r1-H2$ tile can only ever dissociate if it does so with a tape cell tile attached to its south. While the $r1C$ glue on its west would otherwise be able to attach to a $qa-H1$ tile that is currently bound to the tape using the eastern rbC (in this case $r1C$) glue, an $H1$ tile can only ever attach to a tape cell tile after an $H3$ tile dissociates from its north. The dissociation of the $H3$ can only happen if it does so while connected to an $H4$ tile, and the $H4$ tile could only dissociate along with the $H3$ tile after “filling in” the new tape cell tile. Thus, for the western facing rbC glue on an $H1$ tile to be available for binding, it can never be the case that a junk supertile like that pictured in the middle of Figure 21 would be able to bind: it is too tall (the necessary space for the southern tile would already be occupied by a tape cell tile).

Finally, in the case where the tape is grown one cell to the right there is a fourth type of junk assembly. This junk assembly is simply the middle junk assembly of Figure 21 with a $*R$ tile bound to the east of the $rb-H2$ tile by both $**$ and $*R$ glues (for a total size of 3 tiles). However, the southern glues of the $*R$ tile must be deactivated before dissociation of the junk assembly, and therefore no new glues in the **on** state are added. As this is the only other producible type of junk assembly, this means that all junk assemblies are fully inert and unable to bind to any other supertiles, and thus maintain the correctness of the simulation while also never growing larger than size 3.

5.3 Correctness of Theorem 5.1

It has been shown that the STAM system $\mathcal{T}_{M(w)}$ which simulates $M(w)$ correctly simulates the behavior of TM M on input w . If $M(w)$ halts, special $qa-H1$ tiles for $q \in \{q_{accept}, q_{reject}\}$ (previously undiscussed) will attach in the head location which halt further assembly (by containing no other **on** or activatable glues), thus creating a terminal assembly of size > 3 (since there must be at least one tape cell tile representing a 0, 1, or $_$ and the tape cell tile representing $*$, along with their backbone tiles). If $M(w)$ does not halt, the supertile representing the tape will never be terminal - only the junk assemblies will be terminal - and thus there will be no terminal supertiles of size > 3 . Furthermore, it is a temperature $\tau = 1$ system, all junk assemblies remain at size either 2 or 3, the signal complexity is $O(1)$ (specifically, 6 due to the maximum number of glues on a tile side being the 6 on the west side of $rb-H2$), the fuel efficiency is $O(1)$ as every transition produces exactly 3 junk assemblies, each with size ≤ 3 , and adds at most 4 new tiles to the tape assembly (the two new tiles for the tape cells which have swapped the head position, plus possibly two more tiles if the tape was grown by one symbol to the right), and tile complexity $O(|Q|)$ since the tape alphabet and all other glue values are a constant size set and each tile template can be used to generate at most a constant number of tiles for each transition in δ and there can be at most $O(|Q|)$ transitions in δ . \square

6 Self-Assembly of the Sierpinski Triangle

Discrete self-similar fractals are defined as sets of points in \mathbb{Z}^2 , and consist of infinite, aperiodic patterns. It is difficult, if not impossible, for them to strictly self-assemble in the aTAM, as is shown in [14, 21] where the impossibility of a class of discrete self-similar fractals, including the Sierpinski triangle, strictly self-assembling in the aTAM is proven. The impossibility of strictly self-assembling the Sierpinski triangle in the 2HAM was shown in [3]. Additionally, Doty [10] has shown a generalization of the impossibility proof from [21] which applies to, among other things, scaled versions of the Sierpinski triangle for any scaling factor. Thus, any method of strictly self-assembling the Sierpinski triangle, scaled or not, is of interest.

In this section, we show that weak self-assembly of the Sierpinski triangle is possible in the STAM with fewer tile types (4 versus 7) and lower temperature (1 versus 2) than existing TAM constructions, and we also show that strict self-assembly at scale factor 2 is possible in the STAM at temperature 1, a first for any model at any temperature.

6.1 The discrete Sierpinski triangle

Here we use the definition of [14]. Let $V = \{(1, 0), (0, 1)\}$. Define the sets $S_0, S_1, S_2, \dots \subset \mathbb{Z}^2$ by the recursion $S_0 = \{(0, 0)\}$, $S_{i+1} = S_i \cup (S_i + 2^i V)$, where $A + cB = \{\vec{m} + c\vec{n} \mid \vec{m} \in A \text{ and } \vec{n} \in B\}$. Then the (standard) discrete Sierpinski triangle is the set $S_\Delta = \cup_{i=0}^{\infty} S_i$. See Figure 22a for a depiction of the first five stages (i.e. S_0 through S_4).

Our Sierpinski triangle constructions are as stated in the following two theorems. Additionally, in the next section we provide a high-level sketch of the more technically challenging construction for Theorem 6.2.

Theorem 6.1. There exists an STAM system that weakly self-assembles the Sierpinski triangle. The system has 5 unique tiles, signal complexity = 4, assembles at temperature $\tau = 1$, and does not utilize glue deactivation.

Theorem 6.2. There exists an STAM system that strictly self-assembles the discrete Sierpinski triangle at temperature $\tau = 1$, with tile complexity = 19, scale factor = 2, signal complexity = 5, and which makes use of glue deactivation, producing terminal junk assemblies of size ≤ 6 .

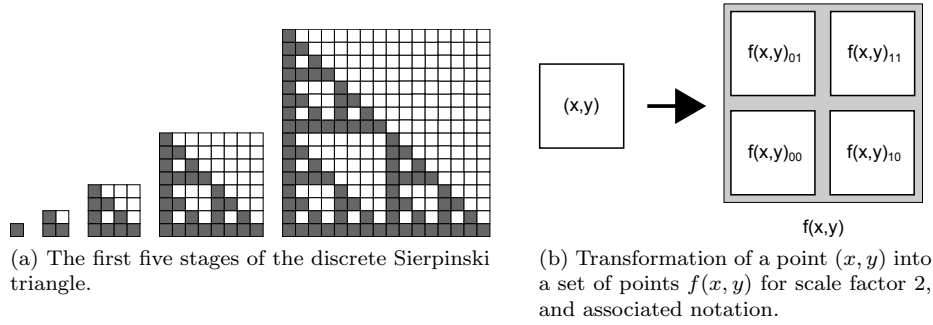


Figure 22: The Sierpinski triangle and a description of the mapping for the scaled version.

6.2 Weak Self-Assembly of the Sierpinski Triangle

Here we describe an STAM system that weakly self-assembles S_Δ to prove Theorem 6.1 by construction. Of note is that this construction works at temperature 1, thus using no cooperative binding, and does not utilize glue deactivation. Further, the tile complexity and signal complexity are small constants of 5 tiles and 4 signals per tile face. This construction is an example of a more general technique for simulating any $\tau = 2$ rectilinear aTAM system (rectilinear systems grow from south to north, west to east) with a temperature $\tau = 1$ STAM system. The general simulation of any $\tau = 2$ aTAM system with a temperature $\tau = 1$ system STAM system is direction for future work.

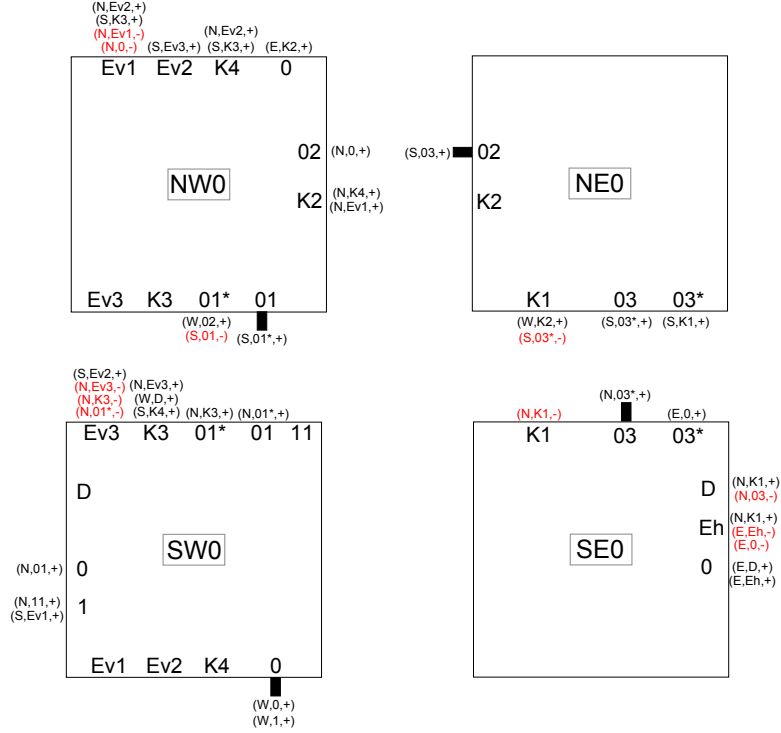
The weak Sierpinski STAM tile set is given in Figure 23. The construction is very similar to standard aTAM constructions for weakly self-assembling the Sierpinski triangle (e.g. [25]), with the main difference being that the aTAM construction works at temperature $\tau = 2$, by having each tile (which is not on an axis) attach with two input sides. Each input side receives as input either a 0 or a 1, and the value for both output sides is the **xor** of the input bits. Those tiles which output a 0 are colored white and considered outside of the Sierpinski triangle, and those which output a 1 are colored black and considered within it. Since our construction works at $\tau = 1$, one input direction - in this case the west - is chosen to be the first to bind. Then, signals cause glues for either possible value of the second input to be turned **on**, allowing the tile to query the tile to its south for its value of 0 or 1. Whichever of the glues answers that query then binds and activates the correct output glues as well as turning **on** the correct label value which identifies the tile as being either white or black. This general signaling mechanism of recruiting a new tile to the complex, activating query glues on edges that are adjacent to tiles already bound to the complex, and then responding to the results of the query is used throughout the temperature 1 STAM constructions given in this paper, and suggests the basis for simulating rectilinear systems.



Our proof of Theorem 6.2 is by construction. Here, we provide a brief overview.

Let $S_{2\Delta} = \{f(x, y) \mid (x, y) \in S_{\Delta}\}$ be the Sierpinski triangle at scale factor 2, i.e. where each point in the original Sierpinski triangle is replaced by a 2×2 square of points, which we will refer to as a *block*. To prove Theorem 6.2, we now present an STAM system, $\mathcal{T}_{2\Delta} = (T_{2\Delta}, 1)$ which strictly self-assembles $S_{2\Delta}$. At a high-level, it does so by weakly self-assembling $S_{2\Delta}$ by treating each block $f(x, y)$ as a single tile which receives one input each from the block to its south and the block to its west. Each input is either a 0 or 1, and the block performs the equivalent of an **xor** operation on those inputs and outputs the result to its north and east. A block $f(x, y)$ which outputs a 1 corresponds to a point $(x, y) \in S_{\Delta}$ and thus a location which must remain tiled in the final assembly (shown as grey locations in Figure 22a). A block $f(x, y)$ which outputs a 0 instead corresponds to a point $(x, y) \notin S_{\Delta}$ and must eventually be removed from the final assembly (shown as white locations in Figure 22a). Whenever a white region is completely tiled and completely surrounded by blocks corresponding to grey positions (note that all white regions in S_{Δ} are surrounded by grey positions), glue deactivation is used to “eject” the blocks of that white region as a set of “junk” supertiles. Those junk supertiles are then broken down into constant sized terminal supertiles (of sizes 3, 4, and 6) which are unable to attach to any portion of the infinitely growing assembly, and thus remain inert junk assemblies.

27



Proof sketch.

Figure 24: First group of tile types which strictly self-assemble the Sierpinski triangle

vertical blocks along the axes, with the exception of tile Ev-2 which only binds to junk assemblies after they are ejected from the construction, initiating the signals that further break down these junk assemblies into smaller pieces. The seed, horizontal and vertical blocks are shown as part of the assemblies in Figures 28-30. Initially, based on the glues that begin in the on state, only tiles of type S can bind to tiles of type $V1$ and $H1$, to begin building the seed block. These initial bindings initiate a series of glue activations which can, using tiles of the types shown in Figure 26, tile each of the blocks corresponding to the points of the positive x and y axes in S_Δ , namely the points $\{f(x, 0) \mid x \in \mathbb{N}\} \cup \{f(0, y) \mid y \in \mathbb{N}\}$. Arbitrarily large portions of the axes can form without the need for any tiles filling in the “interior” 0- and 1-blocks (in positions $f(x, y)$ where $x > 0$ and $y > 0$). However, no interior block $f(x, y)$ can be fully tiled until blocks $f(x, 0)$ and $f(0, y)$ have received tiles. In fact, the first tile placed in $f(x, y)$ is always $f(x, y)_{00}$, and for interior blocks this tile always attaches first to $f(x, y - 1)_{01}$ (that is, the north-west tile of the block immediately to its south). For the rest of the discussion, we will only be referring to interior blocks unless explicitly stated.

The first tile to attach in a 0- or 1-block is either $SW0$ or $SW1$ depending on whether or not the output from the block to the south is a 0 or 1, respectively (Figure 27). (An example of the assembly process can be seen in Figures 28-30.) Once it also binds to the block to its west to receive its second input, it is able to determine the correct output for that block and activate binding glues on its north which specify the identity of that entire block as either 0 (in a white position) or 1 (in a grey position). If it is a 0-block (1-block), the rest of the block fills in with the $NW0$, $NE0$, and $SE0$ ($NW1$, $NE1$, and $SE1$) tiles, in order. Note that either type of block can have either a $SW0$ or $SW1$ tile in its southwest position (the 0 and 1 merely corresponding to the identity of the first input), since the first input is not sufficient to determine the identity of the block. The 0- and 1- blocks produce the weak Sierpinski pattern at scale factor 2 until glue deactivations begin to occur.

Next, we describe the dissociation process that removes the 0-blocks from a region of the produced assembly once that region is fully enclosed by 1-blocks. After blocks determine their identity as 0- or 1-blocks, they presenting that output to their north and east, allowing further block assembly. 1-blocks display

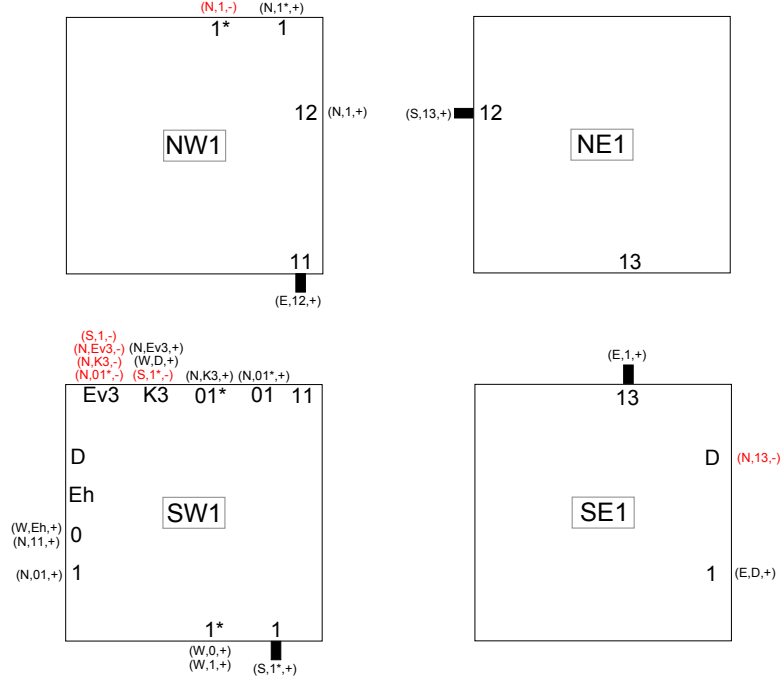


Figure 25: Second group of tile types which strictly self-assemble the Sierpinski triangle

glues to their south or east, while 0-blocks also activate glues on their output (north and east) sides which are used to detect that they are bordered by a 1-block and begin the dissociation process. When a 1-block forms in the block $f(x, y)$, it must be the case that one of the inputs to the tile in position $f(x, y)_{00}$ is a 0. If $f(x, y)_{00}$ is of type $SW0$, that input was to the south, and if it is $SW1$ then that input came from the west. Once both inputs have been determined, such a $SW0$ ($SW1$) tile will activate a glue which initiates a $Ev1$ (Eh) signal to the south (west). This signal will be received by the bordering 0-block and cause it to dissociate from the 1-block and pass the signal further into the block and the entire white region. The combination of Ev and Eh signals which pass through 0-blocks in a white region bounded by 1-blocks (note that they must be completely surrounded by 1-blocks since the order of block growth is strictly up and to the right and those signals are only initiated from the south and east sides of 1-blocks) cause them to separate into vertical chunks which can each dissociate as single supertiles. The design of the signal propagation and glue deactivation is such that the junk supertiles can only dissociate as supertiles which are bounded on both the north and south by 1-blocks (although they may only be 1 block wide and thus bounded on the east and west by 0-blocks). This fact is extremely important to the process which further breaks apart those junk supertiles into constant sized pieces but also ensures that they will always be broken apart and never interfere with further growth of the assembly representing the Sierpinski triangle.

The first phase of dissociation breaks the white regions into vertical columns which were previously bounded above and below by grey portions. (The process by which some example junk assemblies are broken down - including some partial re-growth with the addition of singleton $NW0$ and $SE0$ tiles - into latent, constant sized junk, is shown in Figure 31.) This means that the bonds between the white and grey regions are broken on the north and south, and also the bonds on the left and right of each column of blocks (both with white and grey blocks) are broken. The careful design of the signals which cause the dissociation of the white regions, always beginning from grey regions to the north and east and initially only causing dissociation with bordering grey blocks to the north and south (and not other white blocks to the north and south of blocks in potentially large white regions), ensures that when junk assemblies are able to dissociate that the glues on their borders which remain active do not allow them to re-attach to any other portion of

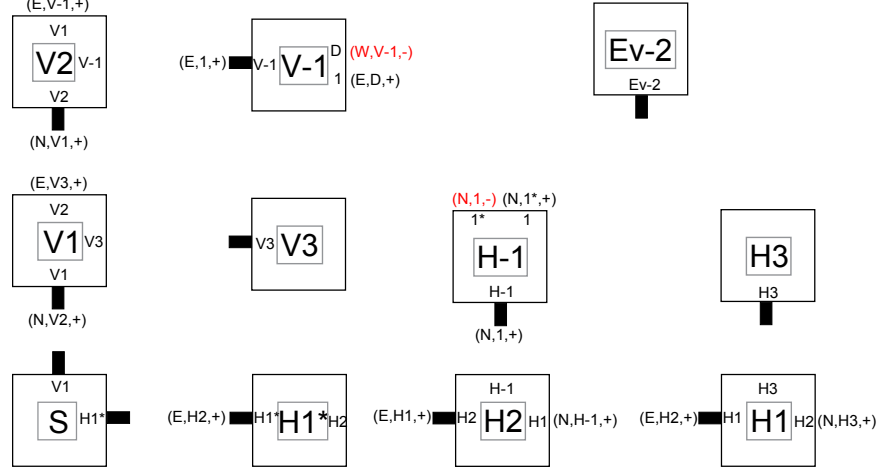


Figure 26: Third group of tile types which strictly self-assemble the Sierpinski triangle. With the exception of the tile type labeled ‘Ev-2’, these tile types tile the locations along the positive x and y axes, which are all included in S_{Δ} .

the growing assembly except for those which have no blocks formed to the north of them. Additionally, if they are larger than 6 tiles they are guaranteed to have an $NW0$ tile in the northwest corner of each block in their northernmost row, and that $NW0$ tile will have an active $Ev2$ glue on its north. Since any position in which these junk assemblies can re-attach cannot have blocks to their north, the $Ev2$ glue will allow an $Ev - 2$ tile to attach, and this attachment will initiate the second phase of dissociation in which a chain of signals cause the junk assembly to break up into fixed height portions. Any re-attachment of the junk assemblies to the growing structure will therefore be temporary and also not able to cause incorrect growth by initiating invalid signals. Thus the junk assemblies are broken into fixed width and height portions in two phases.

It is notable that junk assemblies can allow singleton $NW0$ and $SE0$ tiles to attach at some point during their break up into constant sized pieces, but that only these single attachments are possible and no additional signal activation is possible which would allow the junk assemblies to make further attachments. Another interesting aspect of the dissociation process is that white blocks which have grey blocks bordering them on their west side will cause a single tile of those grey blocks to dissociate with them. The newly formed hole in the grey block will be re-filled by either an $SE1$ or a $V-1$ tile. The reason for this is to “hide” the active 0 glue between the original $SE1$ tile and the $SW1$ tile to its east, which could allow incorrect binding of the junk assembly. In fact, the need to hide active bonds which cannot be guaranteed to be deactivated in this asynchronous model is the reason for the complexity of the dissociation process and the size of the final, terminal junk assemblies.

Through this process in which weak self-assembly of the Sierpinski triangle proceeds until white regions are surrounded by grey blocks, and then those white regions are forced to dissociate as arbitrarily large junk assemblies which are then further broken down into constant sized junk assemblies, all the while being guaranteed not to cause incorrect assembly by binding to the still growing structure, provides the correct strict self-assembly of the Sierpinski triangle at scale 2.

□

Acknowledgments

The authors would like to thank Nataša Jonoska and Daria Karpenko for fruitful discussions and comments on this work.

References

- [1] L.M. Adleman, Q. Cheng, A. Goel, M-D.A. Huang, D. Kempe, P. Moisset de Espanés, and P.W.K. Rothmund, *Combinatorial optimization problems in self-assembly*, Proceedings of the thirty-fourth annual ACM symposium on Theory of computing, 2002, pp. 23–32.
- [2] F. Becker, *Pictures worth a thousand tiles, a geometrical programming language for self-assembly*, Theoretical Computer Science **410** (2009), no. 16, 1495–1515.
- [3] 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)*, Tech. Report 1201.1650, Computing Research Repository, 2012.
- [4] Ho-Lin Chen and David Doty, *Parallelism and time in hierarchical self-assembly*, SODA 2012: Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2012, pp. 1163–1182.
- [5] 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** (2005), 1493–1515.
- [6] Matthew Cook, Yunhui Fu, and Robert Schweller, *Temperature 1 self-assembly: Deterministic assembly in 3d and probabilistic assembly in 2d*, Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, 2011.
- [7] 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** (2008), no. 3, 347–370.
- [8] Erik D. Demaine, Matthew J. Patitz, Robert T. Schweller, and Scott M. Summers, *Self-assembly of arbitrary shapes using rnaase enzymes: Meeting the kolmogorov bound with small scale factor (extended abstract)*, Proceedings of the Twenty Eighth International Symposium on Theoretical Aspects of Computer Science (STACS 2011) (Dortmund, Germany), 2011, to appear.
- [9] R.M. Dirks and N.A. Pierce, *Triggered amplification by hybridization chain reaction*, Proceedings of the National Academy of Sciences of the United States of America **101** (2004), no. 43, 15275.
- [10] David Doty, personal communication, 2012.
- [11] David Doty, Lila Kari, and Benoît Masson, *Negative interactions in irreversible self-assembly*, Algorithmica, to appear. Preliminary version appeared in DNA 2010.
- [12] David Doty, Matthew J. Patitz, and Scott M. Summers, *Limitations of self-assembly at temperature 1*, Theoretical Computer Science **412** (2011), 145–158.
- [13] M.Y. Kao and R. Schweller, *Randomized self-assembly for approximate shapes*, Automata, Languages and Programming (2008), 370–384.
- [14] James I. Lathrop, Jack H. Lutz, and Scott M. Summers, *Strict self-assembly of discrete Sierpinski triangles*, Theoretical Computer Science **410** (2009), 384–405.
- [15] F. Liu, R. Sha, and N.C. Seeman, *Modifying the surface features of two-dimensional DNA crystals*, J. Am. Chem. Soc **121** (1999), no. 5, 917–922.
- [16] W. Liu, H. Zhong, R. Wang, and N.C. Seeman, *Crystalline Two-Dimensional DNA-Origami arrays*, Angewandte Chemie International Edition **50** (2011), no. 1, 264–267.

- [17] K. Lund, A.J. Manzo, N. Dabby, N. Michelotti, A. Johnson-Buck, J. Nangreave, S. Taylor, R. Pei, M.N. Stojanovic, and N.G. Walter, *Molecular robots guided by prescriptive landscapes*, Nature **465** (2010), no. 7295, 206–210.
- [18] U. Majumder, T.H. LaBean, and J.H. Reif, *Activatable tiles: Compact, robust programmable assembly and other applications*, Proceedings of the 13th international conference on DNA computing, 2007, pp. 15–25.
- [19] T. Omabegho, R. Sha, and N.C. Seeman, *A bipedal DNA brownian motor with coordinated legs*, Science **324** (2009), no. 5923, 67.
- [20] J.E. Padilla, W. Liu, and N.C. Seeman, *Hierarchical self assembly of patterns from the Robinson tilings: DNA tile design in an enhanced tile assembly model*, Natural Computing **11** (2012), 323–338.
- [21] Matthew J. Patitz and Scott M. Summers, *Self-assembly of discrete self-similar fractals*, Natural Computing **9** (2010), 135–172.
- [22] ———, *Self-assembly of decidable sets*, Natural Computing **10** (2011), 853–877.
- [23] L. Qian and E. Winfree, *A simple dna gate motif for synthesizing large-scale circuits*, Journal of The Royal Society Interface **8** (2011), no. 62, 1281–1297.
- [24] Paul W. K. Rothmund and Erik Winfree, *The program-size complexity of self-assembled squares (extended abstract)*, STOC '00: Proceedings of the thirty-second annual ACM Symposium on Theory of Computing (Portland, Oregon, United States), ACM, 2000, pp. 459–468.
- [25] Paul W.K. Rothmund, Nick Papadakis, and Erik Winfree, *Algorithmic self-assembly of DNA Sierpinski triangles*, PLoS Biology **2** (2004), no. 12, 2041–2053.
- [26] P.W.K. Rothmund, *Folding DNA to create nanoscale shapes and patterns*, Nature **440** (2006), no. 7082, 297–302.
- [27] Robert Schweller and Michael Sherman, *Fuel efficient computation in passive self-assembly*, Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013) (New Orleans, Louisiana), 2013, to appear.
- [28] G. Seelig, D. Soloveichik, D.Y. Zhang, and E. Winfree, *Enzyme-free nucleic acid logic circuits*, science **314** (2006), no. 5805, 1585.
- [29] D. Soloveichik and E. Winfree, *Complexity of self-assembled shapes*, SIAM Journal on Computing **36** (2007), no. 6, 1544–1569.
- [30] H. Wang, *Proving theorems by pattern recognition II*, AT&T Bell Labs Tech. J **40** (1961), 1–41.
- [31] S.F.J. Wickham, M. Endo, Y. Katsuda, K. Hidaka, J. Bath, H. Sugiyama, and A.J. Turberfield, *Direct observation of stepwise movement of a synthetic molecular transporter*, Nature Nanotechnology **6** (2011), no. 3, 166–169.
- [32] E. Winfree, *Algorithmic self-assembly of DNA*, Ph.D. thesis, California Institute of Technology, June 1998.
- [33] E. Winfree, F. Liu, L.A. Wenzler, and N.C. Seeman, *Design and self-assembly of two-dimensional DNA crystals*, Nature **394** (1998), no. 6693, 539–544.
- [34] P. Yin, H.M.T. Choi, C.R. Calvert, and N.A. Pierce, *Programming biomolecular self-assembly pathways*, Nature **451** (2008), no. 7176, 318–322.

- [35] B. Yurke, A.J. Turberfield, A.P. Mills, F.C. Simmel, and J.L. Neumann, *A DNA-fuelled molecular machine made of DNA*, Nature **406** (2000), no. 6796, 605–608.
- [36] D.Y. Zhang and G. Seelig, *Dynamic DNA nanotechnology using strand-displacement reactions*, Nature Chemistry **3** (2011), no. 2, 103–113.

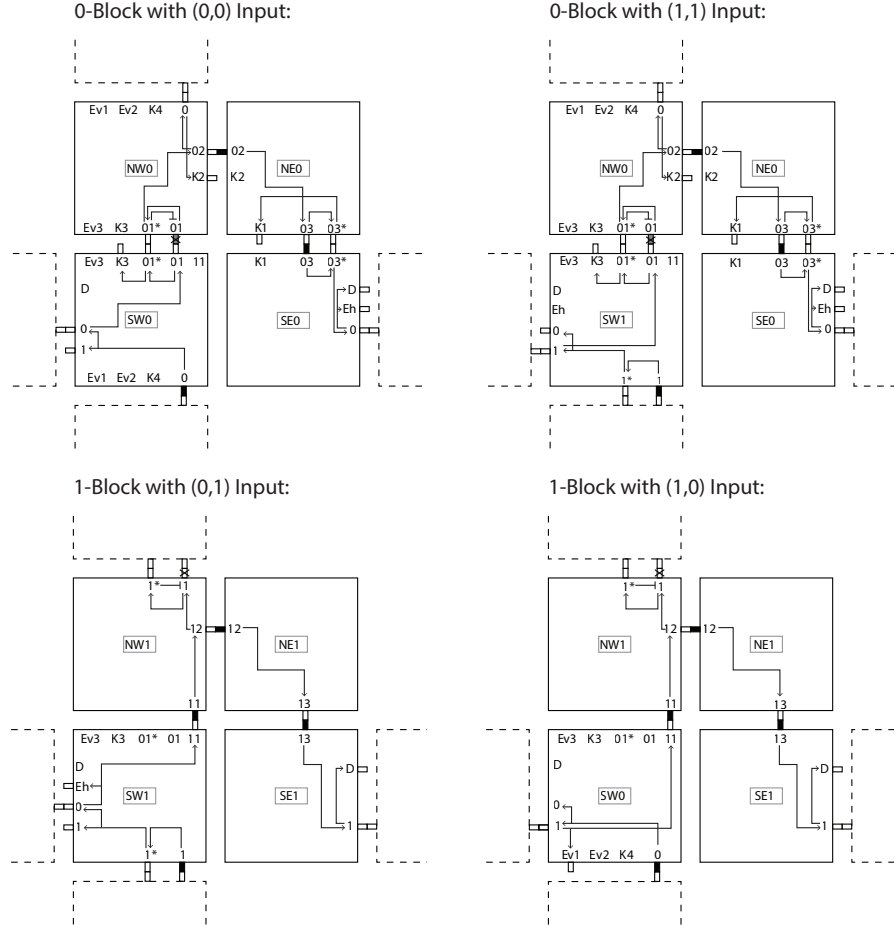
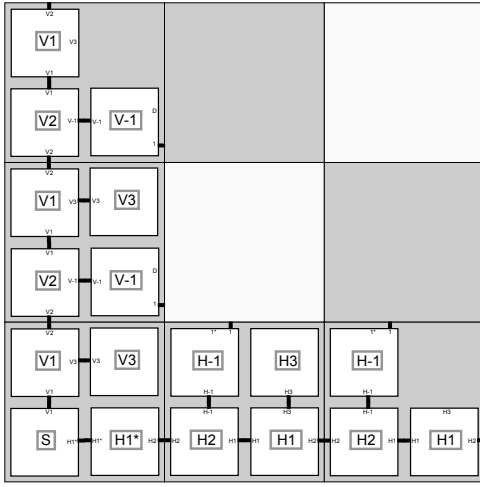
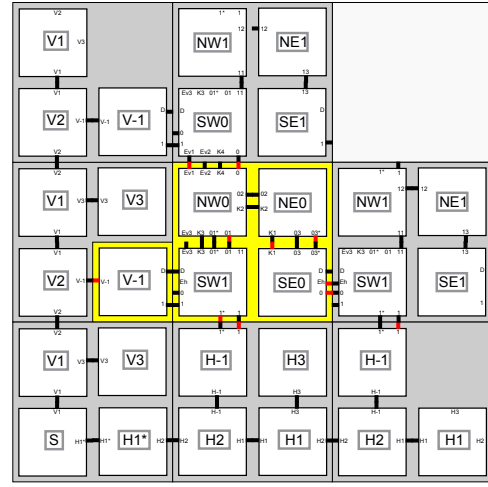


Figure 27: The interior 0- and 1-blocks formed during the strict Sierpinski construction. Signals that have been activated during assembly of the blocks are shown. Glues that are initially active in the tile set are colored black or grey. Glues that have been activated during the process of constructing the block are shown in outline, those that have been deactivated are crossed out, and dashed lines are used to indicate the positions of tiles outside the block that bind to the tiles inside the block.



(a) The beginning of the formation of the axes of the Sierpinski triangle (which can continue arbitrarily far without the non-axes locations being tiled). Since the scale factor is 2, the 2×2 squares which are included in the pattern are shaded with a darker grey.



(b) The Sierpinski triangle at the point where the first dissociation becomes possible. The tiles in the area with the yellow background will dissociate as one “junk” supertile.

Figure 28: The initial stages of the strict self-assembly of the Sierpinski triangle.

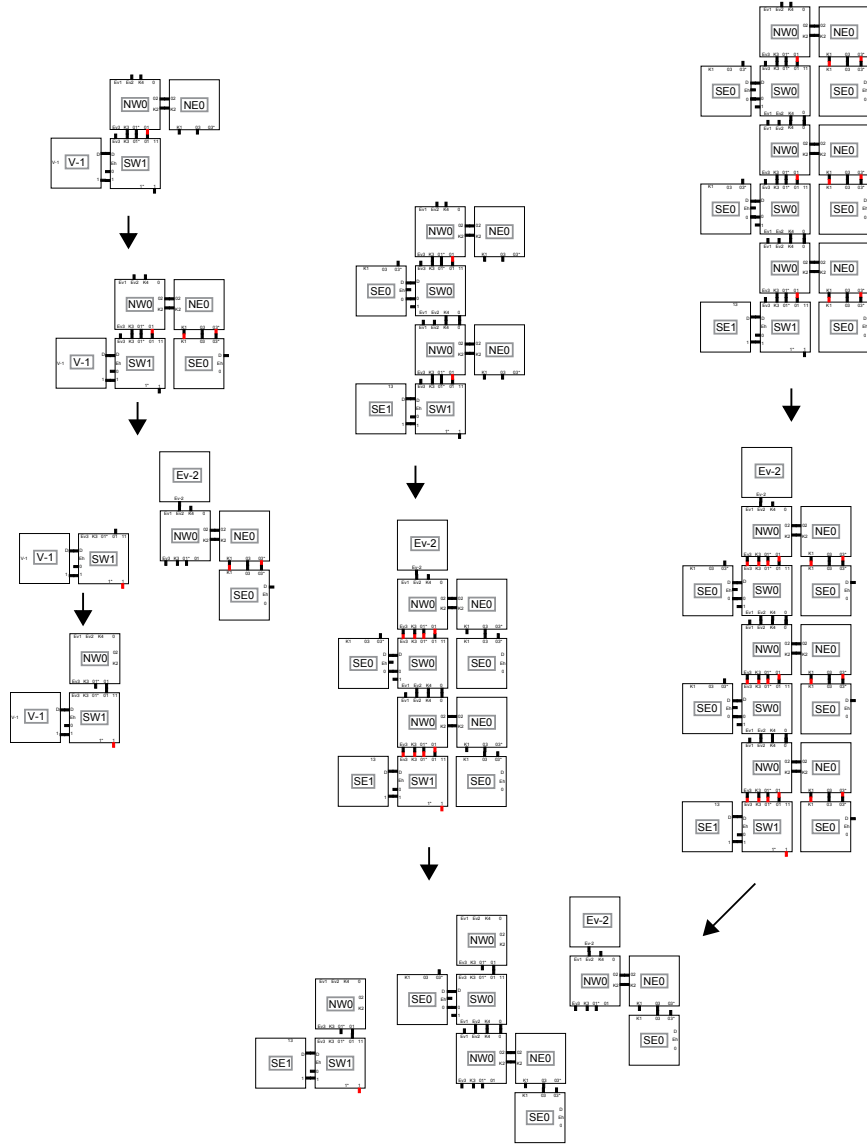


Figure 31: The “junk” ejected from the first three stages and the pathways along which it is broken down following the attachment of *Ev2* tiles. Note that at some edges, singleton tiles can attach to the junk assemblies. However, all junk assemblies are guaranteed to break down into sizes of 3, 4, or 6 and become terminal.