Self-replication via tile self-assembly

Andrew Alseth[1] · Daniel Hader[1] · Matthew J. Patitz[1]

Abstract

In this paper we present a model containing modifications to the Signal-passing Tile Assembly Model (STAM), a tile-based self-assembly model whose tiles are capable of activating and deactivating glues based on the binding of other glues. These modifications consist of an extension to 3D, the ability of tiles to form "flexible" bonds that allow bound tiles to rotate relative to each other, and allowing tiles of multiple shapes within the same system. We call this new model the STAM*, and we present a series of constructions within it that are capable of self-replicating behavior. Namely, the input seed assemblies to our STAM* systems can encode either "genomes" specifying the instructions for building a target shape, or can be copies of the target shape with instructions built in. A universal tile set exists for any target shape (at scale factor 2), and from a genome assembly creates infinite copies of the genome as well as the target shape. An input target structure, on the other hand, can be "deconstructed" by the universal tile set to form a genome encoding it, which will then replicate and also initiate the growth of copies of assemblies of the target shape. Since the lengths of the genomes for these constructions are proportional to the number of points in the target shape, we also present a replicator which utilizes hierarchical self-assembly to greatly reduce the size of the genomes required. The main goals of this work are to examine minimal requirements of self-assembling systems capable of self-replicating behavior, with the aim of better understanding self-replication in nature as well as understanding the complexity of mimicking it.

Keywords Self-assembly · Self-replication · Tile Assembly Model · Signal-passing tiles

Notes. A conference version of this paper was presented at DNA27 in September 2021 Alseth et al. (2021). This paper differs substantially from the conference version; in particular, this version includes significantly more details of the constructions and proofs of their correctness, as well as Theorem 4 which demonstrates the necessity of deconstruction during the process of self-replication for a class of shapes. Also, due to space limitations some figures and

details are ommited from this version but a full version can be found on online Alseth et al. (2021).

1 Introduction

1.1 Background and motivation

Research in tile based self-assembly is typically focused on modeling the computational and shape-building capabilities of biological nano-materials whose dynamics are rich enough to allow for interesting algorithmic behavior. Polymers such as DNA, RNA, and poly-peptide chains are of particular interest because of the complex ways in which they can fold and bind with both themselves and others. Even when only taking advantage of a small subset of the dynamics of these materials, with properties like binding and folding generally being restricted to very manageable cases, tile assembly models have been extremely successful in exhibiting vast arrays of interesting behavior (Rothemund and Winfree 2000; Soloveichik and Winfree 2005;

✉ Matthew J. Patitz
patitz@uark.edu

Andrew Alseth
awalseth@uark.edu

Daniel Hader
dhader@uark.edu

[1] Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, USA

Doty et al. 2012; Demaine et al. 2014, 2013; Lathrop et al. 2011; Summers 2012; Doty et al. 2013; Becker et al. 2006; Cheng et al. 2005; Doty 2009). Among other things, a typical question in the realm of algorithmic tile assembly asks what the minimal set of requirements is to achieve some desired property. Such questions can range from very concrete, such as "how many distinct tile types are necessary to construct specific shapes?", to more abstract such as "under what conditions is the construction of self-similar fractal-like structures possible?". Since the molecules inspiring many tile assembly models are used in nature largely for the purpose of self-replication of living organisms, a natural tile assembly question is thus whether or not such behavior is possible to model algorithmically.

In this paper we show that we can define a model of tile assembly in which the complexities of self-replication type behavior can be captured, and provide constructions in which such behavior occurs. We define our model with the intention of it (1) being hopefully physically implementable in the (near) future, and (2) using as few assumptions and constraints as possible. Our constructions therefore provide insight into understanding the basic rules under which the complex dynamics of life, particularly self-replication, may occur.

We chose to use the Signal-passing Tile Assembly Model (STAM) as a basis for our model, which we call the STAM*, because (1) there has been success in physically realizing such systems (Padilla et al. 2015) and potential exists for further, more complex, implementations using well-established technologies like DNA origami (Rothemund 2005; Liu et al. 2011; Wei et al. 2012; Andersen et al. 2009; Barish et al. 2009) and DNA strand displacement (Qian and Winfree 2011; Wang et al. 2018; Simmel et al. 2019; Zhang and Seelig 2011; Zhang et al. 2013; Bui et al. 2018), and (2) the STAM allows for behavior such as cooperative tile attachment as well as detachment of subassemblies. We modify the STAM by bringing it into 3 dimensions and making a few simplifying assumptions, such as allowing multiple tile shapes and tile rotation around flexible glues and removing the restriction that tiles have to remain on a fixed grid. Allowing flexibility of structures and multiple tile shapes provides powerful new dynamics that can mimic several aspects of biological systems and suffice to allow our constructions to model self-replicating behavior. Prior work, theoretical (Keenan et al. 2014) and experimental (Schulman et al. 2012), has focused on the replication of patterns of bits/letters on 2D surfaces, as well as the replication of 2D shapes in a model using staged assembly (Abel et al. 2010), or in the STAM (Hendricks et al. 2015). However, all of these are fundamentally 2D results and our 3D results, while strictly theoretical, are a superset with constructions capable of replicating all finite 2D and 3D patterns and shapes.

Biological self-replication requires three main categories of components: (1) instructions, (2) building blocks, and (3) molecular machinery to read the instructions and combine building blocks in the manner specified by the instructions. We can see the embodiment of these components as follows: (1) DNA/RNA sequences, (2) amino acids, and (3) RNA polymerase, transfer RNA, and ribosomes, among other things. With our intention to study the simplest systems capable of replication, we started by developing what we envisioned to be the simplest model that would provide the necessary dynamics, the STAM*, and then designed modular systems within the STAM* which each demonstrated one or more important behaviors related to replication. Quite interestingly, and unintentionally, our constructions resulted in components with strong similarities to biological counterparts. As our base encoding of the instructions for a target shape, we make use of a linear assembly which has some functional similarity to DNA. Similar to DNA, this structure also is capable of being replicated to form additional copies of the "genome". In our main construction, it is necessary for this linear sequence of instructions to be "transcribed" into a new assembly which also encodes the instructions but which is also functionally able to facilitate translation of those instructions into the target shape. Since this sequence is also degraded during the growth of the target structure, it shares some similarity with RNA and its role in replication. Our constructions don't have an analog to the molecular machinery of the ribosome, and can therefore "bootstrap" with only singleton copies of tiles from our universal set of tiles in solution. However, to balance the fact that we don't need preexisting machinery, our building blocks are more complicated than amino acids, instead being tiles capable of a constant number of signal operations each (turning glues on or off due to the binding of other glues).

1.2 Our results

Beyond the definition of the STAM* as a new model, we present a series of STAM* constructions. They are designed and presented in a modular fashion, and we discuss the ways in which they can be combined to create various (self-)replicating systems.

1.2.1 Genome-based replicator

We first develop an STAM* tileset which functions as a simple self-replicator (in Sect. 3) that begins from a seed assembly encoding information about a target structure, a.k.a. a *genome*, and grows arbitrarily many copies of the genome and target structure, a.k.a. the *phenotype*. This tileset is universal for all 3D shapes comprised of $1 \times 1 \times 1$ cubes when they are inflated to scale factor 2 (i.e. each $1 \times 1 \times 1$ block in

the shape is represented by a cube of $2 \times 2 \times 2$ tiles). This construction requires a genome whose length is proportional to the number of cube tiles in the phenotype; for non-trivial shapes the genome is a constant factor longer in order to follow a Hamiltonian path through an arbitrary 3D shape at scale factor 2. This is compared to the Soloveichik and Winfree universal (2D) constructor (Soloveichik and Winfree 2007) where a "genome" is optimally shortened, but the scale factor of blocks is much larger.

The process by which this occurs contains analogs to natural systems. We progress from a genome sequence (acting like DNA), which is translated into a messenger sequence (somewhat analogous to RNA), that is modified and consumed in the production of tertiary structures (analogous to proteins). We have a number of helper structures that fuel both the replication of the genome and the translation of the messenger sequence.

1.2.2 Deconstructive self-replicator

In Sect. 4, we construct an STAM* tileset that can be used in systems in which an arbitrarily shaped seed structure, or phenotype, is disassembled while simultaneously forming a genome that describes its structure. This genome can then be converted into a linear genome (of the form used for the first construction) to be replicated arbitrarily and can be used to construct a copy of the phenotype. We show that this can be done for any 3D shape at scale factor 2 which is sufficient, and in some cases necessary, to allow for a Hamiltonian path to pass through each point in the shape. This Hamiltonian path, among other information necessary for the disassembly and, later, reassembly processes, is encoded in the glues and signals of the tiles making up the phenotype. We then show how, using simple signal tile dynamics, the phenotype can be disassembled tile by tile to create a genome encoding that same information. Additionally, a reverse process exists so that once the genome has been constructed from a phenotype, a very similar process can be used to reconstruct the phenotype while disassembling the genome.

In sticking with the DNA, RNA, protein analogy, this disassembly process doesn't have a particular biological analog; however, this result is important because it shows that we can make our system robust to starting conditions. That is, we can begin the self-replication process at any stage be it from the linear genome, "kinky genome" (the messenger sequence from the first construction), or phenotype. Finally, since this construction requires the phenotype to encode information in its glues and signals, we show that this can be computed efficiently using a polynomial time algorithm given the target shape. This not only shows that the STAM* systems can be described efficiently for any target shape via a single universal tile set, but that results from intractable computations aren't built into our

phenotype (i.e. we're not "cheating" by doing complex pre-computations that couldn't be done efficiently by a typical computationally universal system). We also provide a result about the necessity for deconstruction in a universal replicator in Section 6.

1.2.3 Hierarchical assembly-based replicator

For our final construction, in Sect. 5, our aims were two-fold. First, we wanted to compress the genome so that its total length is much shorter than the number of tiles in the target shape. Second, we wanted to more closely mimic the biological process in which individual proteins are constructed via the molecular machinery, and then they are released to engage in a hierarchical self-assembly process in which proteins combine to form larger structures.

Biological genomes are many orders of magnitude smaller than the organisms which they encode, but for our previous constructions the genomes are essentially equivalent in size to the target structures. Our final construction is presented in a "simple" form in which the general scaling approximately results in a genome which is length $n^{\frac{1}{3}}$ for a target structure of size n. However, we discuss relatively simple modifications which could, for some target shapes, result in genome sizes of approximately $\log n$, and finally we discuss a more complicated extension (which also consumes a large amount of "fuel", as opposed to the base constructions which consume almost no fuel) that can achieve asymptotically optimal encoding.

1.2.4 Combinations and permutations of constructions

Due to length restrictions for this version of the paper, and our desire to present what we found to be the "simplest" systems capable of combining to perform self-replication, there are several additions to our results which we only briefly mention. For instance, to make our first construction (in Sect. 3) into a standalone self-replicator, and one which functions slightly more like biological systems, the input to the system, i.e. the seed assembly, could instead be a copy of the target structure with a genome "tail" attached to it. The system could function very similarly to the construction of Sect. 3 but instead of genome replication and structure building being separated, the genome could be replicated and then initiate the growth of a connected messenger structure so that once the target structure is completed, the genome is attached. Thus, the input assembly would be completed replicated, and be a self-replicator more closely mirroring biology where the DNA along with the structure cause the DNA to replicate itself and the structure. Attaching the genome to the structure is a technicality that could satisfy the need to have a single seed

assembly type, but clearly it doesn't meaningfully change the behavior. At the end of Sect. 5 we discuss how that construction could be combined with those from Sects. 3 and 4, as well as further optimized. The next section begins with a high-level overview of the STAM* and then gives a more detailed set of definitions.

2 Preliminaries

In this section we define the notation and models used throughout the paper.

We define a *3D shape* $S \subset \mathcal{Z}^3$ as a connected set of $1 \times 1 \times 1$ cubes (a.k.a. unit cubes) which define an arbitrary polycube, i.e. a shape composed of unit cubes connected face to face where each cube represents a voxel (3-D pixel) of S. For each shape S, we assume a canonical translation and rotation of S so that, without loss of generality, we can reference the coordinates of each of its voxels and directions of its surfaces, or faces. We say a unit cube is *scaled by factor c* if it is replaced by a $c \times c \times c$ cube composed of c^3 unit cubes. Given an arbitrary 3D shape S, we say S is *scaled by factor c* if every unit cube of S is scaled by factor c and those scaled cubes are arranged in the shape of S. We denote a shape S scaled by factor c as S^c.

2.1 Definition of the STAM*

The 3D Signal-passing Tile Assembly Model*

(3D-STAM*, or simply STAM*) is a generalization of the STAM (Padilla et al. 2014; Fochtman et al. 2015; Hendricks et al. 2019; Keenan et al. 2013) (that is similar to the model in Jonoska and Karpenko (2014a, 2014b)) in which (1) the natural extension from 2D to 3D is made (i.e. tiles become 3-dimensional shapes rather than 2-dimensional squares), (2) multiple tile shapes are allowed, (3) tiles are allowed to flip and rotate (Demaine et al. 2014; Hendricks et al. 2017a), and (4) glues are allowed to be rigid (as in the aTAM, 2HAM, STAM, etc., meaning that when two adjacent tiles bind to each other via a rigid glue, their relative orientations are fixed by that glue) or *flexible* (as in Durand-Lose et al. (2018)) so that even after being bound tiles and subassemblies are free rotate with respect to tiles and subassemblies to which they are bound by bending or twisting around a "joint" in the glue. (This would be analogous to rigid glues forming as DNA strands combine to form helices with no single-stranded gaps, while flexible glues would have one or more unpaired nucleotides leaving a portion of single-stranded DNA joining the two tiles, which would be flexible and rotatable.) See Fig. 1 for a simple example. These extensions
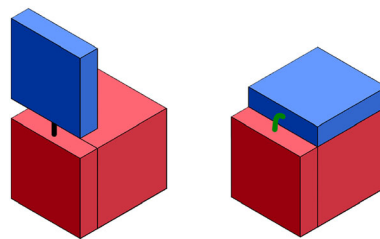

Fig. 1 Example showing flat and cubic tiles, and possible behavior of a flexible glue allowing the blue tile to fold upward, away from the red cubic tile, or down against it. In all constructions, we assume lengths for all flexible glues which make the folding and alignment in this figure possible, and length 0 for rigid glues between cubic and flat tiles (as though one tile's glue strand binds into a cavity)

make the STAM* a hybrid model of those in previous studies of hierarchical assembly (Cheng et al. 2005; Demaine et al. 2008, 2016; Patitz et al. 2016; Hendricks et al. 2017b), 3D tile-based self-assembly (Cook et al. 2011; Furcy et al. 2015; Becker et al. 2008; Hader et al. 2020), systems allowing various non-square/non-cubic tile types (Fekete et al. 2015; Gilbert et al. 2016; Demaine et al. 2014; Fu et al. 2012; Hader and Patitz 2019; Kari et al. 2012), and systems in which tiles can fold and rearrange (Durand-Lose et al. 2018; Jonoska and McColm 2006, 2005, 2009).

We now provide a high-level overview of several aspects of the STAM* model, and full definitions can be found in Sect. 2.2.

The basic components of the model are *tiles*. Tiles bind to each other via *glues*. Each glue has a *glue type* that specifies its domain (which is the string label of the glue), integer strength, *flexibility* (a boolean value with `true` meaning *flexible* and `false` meaning *rigid*), and length (representing the length of the physical glue component). A glue is an instance of a glue type and may be in one of three states at any given time, `latent`, `on`, `off`. A pair of adjacent glues are able to bind to each other if they have complementary domains and are both in the `on` state, and do so with strength equal to their shared strength values (which must be the same for all glues with the same label l or the complementary label l^*).

A *tile type* is defined by its 3D shape (and although arbitrary rotation and translation in $\mathbb{R}^3$ are allowed, each is assigned a canonical orientation for reference), its set of glues, and its set of *signals*. Its set of glues specify the types. locations, and initial states of its glues. Each signal in its set of signals is a triple (g_1, g_2, δ) where g_1 and g_2 specify the *source* and *target* glues (from the set of the tile type's glues) and $\delta \in \{\texttt{activate}, \texttt{deactivate}\}$. Such a signal denotes that when glue g_1 forms a bond, an action is initiated to turn glue g_2 either on (if $\delta == \texttt{activate}$) or off (otherwise). A *tile* is an instance of a tile type

represented by its type, location, rotation, set of glue states (i.e. latent, on or off for each), and set of *signal states*. Each signal can be in one of the signal states {pre, firing, post}. A signal which has never been activated (by its source glue forming a bond) is in the pre state. A signal which has activated but whose action has not yet completed is in the firing state, and if that action has completed it is in the post state. Each signal can "fire" only one time, and each glue which is the target of one or more signals is only allowed to make the following state transitions: (1) latent → on, (2) on → off, and (3) latent → off.

We use the terms *assembly* and *supertile*, interchangeably, to refer to the full set of rotations and translations of either a single tile (the base case) or a collection of tiles which are bound together by glues. A supertile is defined by the tiles it contains (which includes their glue and signal states) and the glue bonds between them. A supertile may be flexible (due to the existence of a cut consisting entirely of flexible glues that are co-linear and there being an unobstructed path for one subassembly to rotate relative to the other), and we call each valid positioning of it sets of subassemblies a *configuration* of the supertile. A supertile may also be translated and rotated while in any valid configuration. We call a supertile in a particular configuration, rotation, and translation a *positioned supertile*.

Each supertile induces a *binding graph*, a multigraph whose vertices are tiles, with an edge between two tiles for each glue which is bound between them. The supertile is τ-*stable* if every cut of its binding graph has strength at least τ, where the weight of an edge is the strength of the glue it represents. That is, the supertile is τ-stable if cutting bonds of at least summed strength of τ is required to separate the supertile into two parts.

For a supertile α, we use the notation $|\alpha|$ to represent the number of tiles contained in α. The *domain* of a positioned supertile α, written dom α, is the union of the points in $\mathbb{R}^3$ contained within the tiles composing α. Let α be a positioned supertile. Then, for $\vec{v} \in \mathbb{R}^3$, we define the partial function $\alpha(\vec{v}) = t$ where t is the tile containing $\vec{v}$ if $\vec{v} \in$ dom α, otherwise it is undefined. Given two positioned supertiles, α and β, we say that they are *equivalent*, and we write $\alpha \approx \beta$, if for all $\vec{v} \in \mathbb{R}^3$ $\alpha(\vec{v})$ and $\beta(\vec{v})$ both either return tiles of the same type, or are undefined. We say they're *equal*, and write $\alpha \equiv \beta$, if for all $\vec{v} \in \mathbb{R}^3$ $\alpha(\vec{v})$ and $\beta(\vec{v})$ either both return tiles of the same type having the same glue and signal states, or are undefined.

An STAM* tile assembly system, or TAS, is defined as $\mathcal{T} = (T, C, \tau)$ where T is a finite set of tile types, C is an initial configuration, and $\tau \in \mathbb{N}$ is the minimum binding threshold (a.k.a. temperature) specifying the minimum binding strength that must exist over the sum of binding glues between two supertiles in order for them to attach to each other. The initial configuration $C = \{(S, n) \mid S$ is a supertile over the tiles in T and $n \in \mathbb{N} \cup \infty$ is the number of copies of $S\}$. Note that for each $s \in S$, each tile $\alpha = (t, \vec{l}, S, \gamma) \in s$ has a set of glue states S and signal states γ. By default, it is assumed that every tile in every supertile of an initial configuration begins with all glues in the initial states for its tile type, and with all signal states as pre, unless otherwise specified. The initial configuration C of a system $\mathcal{T}$ is often simply given as a set of supertiles, which are also called *seed* supertiles, and it is assumed that there are infinite counts of each seed supertile as well as of all singleton tile types in T. If there is only one seed supertile σ, we will we often just use σ rather than C.

2.1.1 Overview of STAM* dynamics

An STAM* system $\mathcal{T} = (T, C, \tau)$ evolves nondeterministically in a series of (a possibly infinite number of) steps. Each step consists of randomly executing one of the following actions: (1) selecting two existing supertiles which have configurations allowing them to combine via a set of neighboring glues in the on state whose strengths sum to strength $\geq \tau$ and combining them via a random subset of those glues whose strengths sum to $\geq \tau$ (and changing any signals with those glues as sources to the state firing if they are in state pre), or (2) randomly select two adjacent unbound glues of a supertile which are able to bind, bind them and change attached signals in state pre to firing, or (3) randomly select a supertile which has a cut $< \tau$ (due to glue deactivations) and cause it to *break* into 2 supertiles along that cut, or (4) randomly select a signal on some tile of some supertile where that signal is in the firing state and change that signal's state to post, and as long as its action (activate or deactivate) is currently valid for the signal's target glue, change the target glue's state appropriately.[1] Although at each step the next choice is random, it must be the case that no possible selection is ever ignored infinitely often. (See Sect. 2.2 for more details.)

Given an STAM* TAS $\mathcal{T} = (T, C, \tau)$, a supertile is *producible*, written as $\alpha \in \mathcal{A}[\mathcal{T}]$, if either it is a single tile from T, or it is the result of a (possibly infinite) series of combinations of pairs of finite producible assemblies (which have each been positioned so that they do not overlap and can be τ-stably bonded), and/or breaks of producible assemblies. A supertile α is *terminal*, written as $\alpha \in \mathcal{A}_\square[\mathcal{T}]$, if (1) for every $\beta \in \mathcal{A}[\mathcal{T}]$, α and β cannot be τ-stably attached, (2) there is no configuration of α in which a

---

[1] The asynchronous nature of signal firing and execution is intended to model a signalling process which can be arbitrarily slow or fast. Please see Sect. 2.2 for more details.

pair of unbound complementary glues in the `on` state are able to bind, and (3) no signals of any tile in α are in the `firing` state.

In this paper, we define a shape as a connected subset of $\mathbb{Z}^3$ to both simplify the definition of a shape and to capture the notion that to build an arbitrary shape out of a set of tiles we will actually approximate it by "pixelating" it. Therefore, given a shape S, we say that assembly α has shape S if α has only one valid configuration (i.e. it is *rigid*) and there exist (1) a rotation of α and (2) a scaling of S, S', such that the rotated α and S' can be translated to overlap where there is a one-to-one and onto correspondence between the tiles of α and cubes of S' (i.e. there is exactly 1 tile of α in each cube of S', and none outside of S').[2]

Definition 1 We say a shape X *self-assembles* in $\mathcal{T}$ with *waste size* c, for $c \in \mathbb{N}$, if there exists terminal assembly $\alpha \in \mathcal{A}_\square[\mathcal{T}]$ such that α has shape X, and for every $\alpha \in \mathcal{A}_\square[\mathcal{T}]$, either α has shape X, or $|\alpha| \leq c$. If $c = 1$, we simply say X *self-assembles in* $\mathcal{T}$.

Definition 2 We call an STAM* system $\mathcal{R} = (T, C, \tau)$ a *shape self-replicator for shape* S if C consists exactly of infinite copies of each tile from T as well as of a single supertile σ of shape S, there exists $c \in \mathbb{N}$ such that S self-assembles in $\mathcal{R}$ with waste size c, and the count of assemblies of shape S increases infinitely.

Definition 3 We call an STAM* system $\mathcal{R} = (T, C, \tau)$ a *self-replicator for* σ *with waste size* c if C consists exactly of infinite copies of each tile from T as well as of a single supertile σ, there exists $c \in \mathbb{N}$ such that for every terminal assembly $\alpha \in \mathcal{A}_\square[\mathcal{T}]$ either (1) $\alpha \approx \sigma$, or (2) $|\alpha| \leq c$, and the count of assemblies $\approx \sigma$ increases infinitely.[3] If $c = 1$, we simply say $\mathcal{R}$ is a self-replicator for σ.

The multiple aspects of STAM* tiles and systems give rise to a variety of metrics with which to characterize and measure the complexity of STAM* systems, beyond metrics seen for models such as the aTAM or even STAM. For a brief discussion, please see the end of Sect. 2.2.

2.1.2 STAM* conventions used in this paper

Although the STAM* is a highly generalized model allowing for variety in tile shapes, glue lengths, etc., throughout this paper all constructions are restricted to the following conventions.

1. All tile types have one of two shapes (shown in Fig. 1):

    (a) A *cubic* tile is a tile whose shape is a $1 \times 1 \times 1$ cube.

    (b) A *flat* tile is a tile whose shape is a $1 \times 1 \times \epsilon$ rectangular prism, where $\epsilon < 1$ is a small constant.

    (c) We call a 1×1 face of a tile a *full* face, and a $1 \times \epsilon$ face is called a *thin* face.

2. Glue lengths are the following (and are shown in Fig. 2):

    (a) All rigid glues between cubic tiles, as well as between thin faces of flat tiles, are length 2ϵ.

    (b) All rigid glues between cubic and flat tiles are length 0. (Note that this could be implemented via the glue strand of one tile extending into the tile body of the other tile in order to bind, thus allowing the tile surfaces to be adjacent without spacing between the faces.)

    (c) All flexible glues are length $\frac{3}{2}\sqrt{2}\epsilon$.[4]

Given that rigidly bound cubic tiles cannot rotate relative to each other, for convenience we often refer to rigidly bound tiles as though they were on a fixed lattice. This is easily done by first choosing a rigidly bound cubic tile as our origin, then using the location $\vec{l}$, orientation matrix R, and rigid glue length g, put in one-to-one correspondence with each vector $\vec{v}$ in $\mathbb{Z}^3$, the vector $\vec{l} + gR\vec{v}$. Once we define an absolute coordinate system in this way, we refer to the directions in 3-dimensional space as North ($+y$), East ($+x$), South ($-y$), West ($-x$), Up ($+z$), and Down ($-z$), abbreviating them as $N, E, S, W, U,$ and D, respectively.

Figure 3 is an illustration of a tile with various signals. We use glues are represented as squares on the side of a tile with adjacent labels. If a glue begins in the on state the glue will be colored black whereas it will not be colored if the glue begins in the latent state. Glues on the front and back of the tile are drawn using a circle with a dot inside or a circle with an X inside respectively. Lines between glues

---

[2] In this paper we only consider completely rigid assemblies for target shapes, since the target shapes are static. We could also target "reconfigurable" shapes, i.e. sets of shapes, but don't do so in this paper. Also, it could be reasonable to allow multiple tiles in each pixel location as long as the correct overall shape is maintained, but we don't require that.

[3] We use $\approx$ rather than $\equiv$ since otherwise either both the seed assemblies and produced assemblies are terminal, meaning nothing can attach to a seed assembly and the system can't evolve, or neither are terminal and it becomes difficult to define the product of a system. However, our construction in Sect. 4 can be modified to produce assemblies satisfying either the $\approx$ or $\equiv$ relation with the seed assemblies.

[4] These glue lengths were chosen so that (1) rigidly bound cubic tiles could each have a flat tile bound to each of their sides if needed and (2) so that two flat tiles attached to diagonally adjacent rigid tiles could be attached via flexible glue.
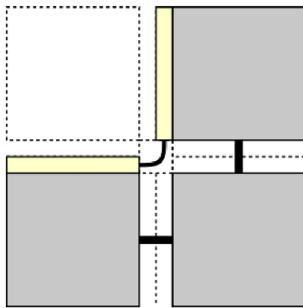
Fig. 2 The glue lengths used in our constructions: (1) length 2ϵ rigid bonds between cubic tiles (straight black lines connecting grey square tiles), (2) length 0 rigid bonds between flat and cubic tiles (invisible, between flat yellow tiles and grey square tiles), and (3) length $3\sqrt{2}\,\epsilon/2$ flexible glues between flat tiles (curved line between yellow tiles)
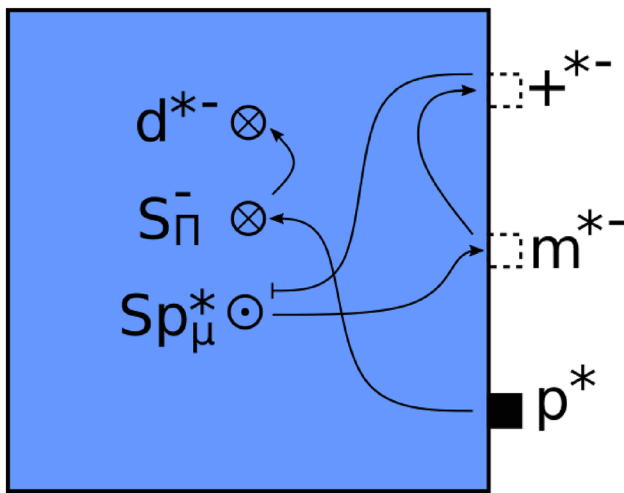

Fig. 3 An example tile signal diagram

indicate signals which end in an arrow if the signal turns on a glue or a serif if the signal turns off a glue.

2.2 Detailed STAM* dynamics

1. The binding of a glue causes any signals associated with that glue to change states, i.e. fire (if they haven't already fired due to a prior binding event).
2. A glue and its complementary pair which are bound overlap, causing the distance between their tiles to be the length of the glue (not two times the length).
3. The binding of a single rigid glue or two flexible glues on different surfaces lock a tile in place. Two flexible glues on the same surface prevent "flipping" (or "twisting") but allow "hinge-like" rotation.
4. The assembly process proceeds step by step by nondeterministically selecting one of the following types of moves to execute unless and until none is available. While the following set of choices for a next step are made randomly, no action which is valid can be postponed infinitely long.

(a) Randomly select any pair of supertiles, α and β, which can bind via a sum of $\geq \tau$ strength bonds if appropriately positioned (and binding only via glues in the on state). Position α and β to combine them to form a new supertile by binding a random subset of the glues which can bind between them whose strengths sum to $\geq \tau$. For each bound glue which has a signal associated with it, but that signal is still in the pre state, change the signal's state to firing. Note that rigid glues must form bonds which extend perpendicularly from their surfaces, but flexible glues are free to bend to form bonds.

(b) Randomly select any supertile which has a cut in its binding graph $< \tau$ (due to one or more glue deactivations), and split that supertile into two supertiles along that cut. We call this operation a *break*.

(c) Randomly select any pair of subassemblies (each of one or more tiles) in the same supertile but bound only by flexible glues so that the sub-assemblies are free to rotate relative to each other, and perform a valid rotation of one of those subassemblies.

(d) Randomly select a supertile and pair of unbound glues within it such that the supertile has a valid configuration in which those glues are able to bind (i.e. they are complementary, both in the on state, and the glues can reach each other), and bind them. For each which has a signal associated with it, but that signal is still in the pre state, change the signal's state to firing.

(e) Randomly select a signal whose state is firing from any tile and execute it. This entails, based on the signal's definition, that its target glue is either activated or deactivated if that is still a valid transition for that glue, and for the signal's state to change to post, marking it as completed and unable to fire again. The STAM* is based on the STAM and it preserves the design goal of modeling physical mechanisms that implement the signals on tiles but which are arbitrarily slower or faster than the average rates of (super)tile attachments and detachments. Therefore, rather than immediately enacting the actions of signals, each signal is put into a state of firing along with all signals initiated by the glue (since it is technically possible for more than one signal to have been initiated, but not yet enacted, for a particular glue). Any firing signal can be randomly selected from the set,

regardless of the order of arrival in the set, and the ordering of either selecting some signal from the set or the combination of two supertiles is also completely arbitrary. This provides fully asynchronous timing between the initiation, or firing, of signals and their execution (i.e. the changing of the state of the target glue), as an arbitrary number of supertile binding (or breaking) events may occur before any signal is executed from the firing set, and vice versa.

The multiple aspects of STAM* tiles and systems give rise to a variety of metrics with which to characterize and measure the complexity of STAM* systems. Following is a list of some such metrics.

1. Tile complexity: the number of unique tile types
2. Tile shape complexity: the number of unique tile shapes, or the maximum number of surfaces on a tile shape, or the maximum difference in sizes between tile shapes
3. Tile glue complexity: the maximum number of glues on any tile type
4. Seed complexity: the size of the seed assembly (and/or the number of unique seed assemblies.
5. Signal complexity: the maximum number of signals on any tile type
6. Junk complexity: the size of the largest terminal assembly which is not considered the "target assembly" (a.k.a. *junk assembly*), or the number of unique types of junk assemblies

3 A genome based replicator

We now present our first construction in the STAM*, in which a "universal" set of tiles will cause a pre-formed seed assembly encoding a Hamiltonian path through a target structure, which we call the *genome*, to replicate infinitely many copies of itself as well as build infinitely many copies of the target structure at temperature 2. We consider 4 unique structures which are generated/utilized as part of the self-replication process: σ, μ, μ', and π. The seed assembly, σ, is composed of a connected set of flat tiles considered to be the *genome*. Let π represent an assembly of the target shape encoded by σ. μ is an intermediate "messenger" structure directly copied from σ, which is modified into μ' to assemble π. We split T into subsets of tiles, $T = \{T_\sigma \cup T_\mu \cup T_\phi \cup T_\pi\}$. T_σ are the tiles used to replicate the genome, T_μ are the tiles used to create the messenger structure, T_π are the cubic tiles which comprise the phenotype π, and T_ϕ are the set of tiles which combine to make fuel structures used in both the genome replication process and conversion of μ to μ'.

The tile types which make up this replicator are carefully designed to prevent spurious structures and enforce two key properties for the self-replication process. First, a genome is never consumed during replication, allowing for exponential growth in the number of completed genome copies. Second, the replication process from messenger to phenotype strictly follows $\mu \to \mu' \to \pi$; each step in the assembly process occurs only after the prior structure is in its completed form. This prevents unexpected geometric hindrances which could block progression of any further step. Complete details of T are located in Sect. 3.4.

3.1 Replication of the genome

The minimal requirements to generate copies of σ in $\mathcal{R}$ are the following: (1) for all individual tile types $s \in \sigma, s \in T_\sigma$, (2) the last tile is the end tile E, and (3) the first tile in σ is a start tile in the set (S^+, S^-). However, for the shape-self replication of S one additional property must hold: (4) σ encodes a Hamiltonian path which ends on an exterior cubic tile. We define the genome to be 'read' from left to right; given requirements (2) and (3), the leftmost tile in a genome is a start tile and the rightmost is an end tile. (4) can be guaranteed by scaling S up to S^2 and utilizing the algorithm in Sect. 4.3.1, selecting a cubic tile on the exterior as a start for the Hamiltonian path and then reversing the result. This requirement ensures the possibility of cubic tile diffusion into necessary locations at all stages of assembly.

The replication process of σ begins with the attachment of tiles from the set T_σ to σ due to the two strength-1 glues on the north face of individual tiles comprising σ. We denote the incomplete copy of σ as σ'. Asynchronously, a fuel tile assembly φ comprised of two subtiles $\varphi_1, \varphi_2 \in T_\phi$ binds to the leftmost tile of σ. Upon the binding of a start tile to the north thin face of the start tile of σ', the signal provided by φ begins a chain reaction binding to the the active 'n' glue on the west thin face of the newly attached tile and the signal propagates through the chain of connected σ' tiles. Once the end tile E_σ is bound to the remainder of σ' by the active 'n' glue, it returns a signal through its newly activated west glue to fully connect it to the prior tile and then detach from the genome to the south. This signal cascades back through the remaining tiles of σ' until reaching φ, at which point φ deactivates its glues. allowing the newly replicated copy of σ to separate and begin the process of replicating itself and translating copies of μ.

3.2 Translation of σ to μ

Translation is defined as the process by which the Hamiltonian path encoded in σ is built into a new messenger assembly μ. Since the signals to attach and detach μ
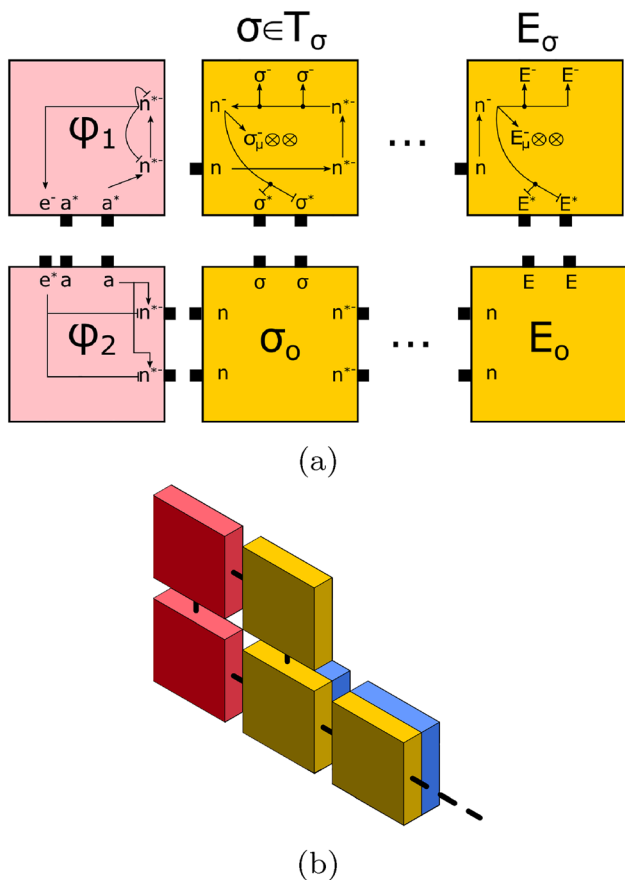
Fig. 4 (a) Initial genome replicator tiles. Note that $\otimes\otimes$ represents a two strength 1 glues which are on the full face of the seed tiles opposite from the reader. Lines between glues represent signals. Lines ending in an arrow turn the corresponding glue on while lines ending in a serif turn off a glue. (b) Illustration of an arbitrary translation process occurring at the same time as genome replication. Red tiles are representative of φ, gold tiles are representative of σ, and blue tiles are representative of μ

from σ are fully contained in the tiles of T_μ, translation continues as long as T_μ tiles remain in the system. We note that the translation process can occur at the same time as σ is replicating. This causes no unwanted geometric hindrances as demonstrated in Fig. 4b.

3.2.1 Placement of μ tiles

Messenger tiles from the set T_μ attach to σ as soon as complementary glues on the back flat face of σ are activated after the binding of φ to σ'. The process of building μ does not require a fuel structure to continue, as the messenger tiles have built-in signals to deactivate the glues on μ which attach μ to σ. This allows for a genome to replicate the messenger structure without itself being consumed in any manner.

Each genome tile contains two active strength-1 glues on its full face which are mapped to a single messenger tile type. Messenger tiles from the set T_μ attach to σ as soon as complementary glues on the back flat face of σ are activated after the binding of the fuel duple φ to σ'. The process of building μ does not require a fuel structure to continue, as the messenger tiles have built-in signals to deactivate the glues on μ which attach μ to σ. This allows for a genome to replicate the messenger structure without itself being consumed in any manner. Once a flat tile in μ is bound to its eastern neighbor, signals are fired from the eastern glues to deactivate the glue connecting μ to σ. This leaves μ as its own separate assembly when every tile has attached to its neighbor(s). The example of translation shown in Fig. 5b illustrates that the same information (i.e., sequence of tiles representing a Hamiltonian path) remains encoded in μ, but allows for new structural functionality that would otherwise not be possible by σ.

3.2.2 Modification of μ to μ'

The current shape of μ is such that it could only replicate a trivial 2D structure; μ must be modified to follow a Hamiltonian path in 3 dimensions as made possible by a set of *turning* tiles. Additionally, in the current state of μ no cubic tiles can be placed as all the glues which are complementary to cubic tiles are currently in the `latent` state. Once a glue of type 'p' is bound on the start tile, we then consider μ to have completed its modification into μ'. The 'p' glue on turning tiles can only be bound once they have been turned, and as such the turning tiles present in μ' must be turned before assembly of π begins.

Turning tiles modify the shape of μ by adding 'kinks' into the otherwise linear structure by the use of a fuel-like structure called a *kink-ase*. The kink-ase structure is generated from a set of 2 flat tiles and 2 cube tiles. These tiles must first fully bind to each other before connections can be made to a turning tile. The unique form of kink-ase allows for the orientation of two adjacent tiles to be modified without separating μ, shown in Fig. 6. The turning tiles are physically rotated such that the connection between a turning tile and its predecessor along the west thin edge of the turning tile is broken, and then reattached along either the up or down thin edge of the turning tile. Each turning tile requires the use of a single kink-ase, which turns into a junk assembly.

We now describe in detail how μ is converted to μ' utilizing the kink-ase structure, with the steps in this section matching up with the intermediate structures shown in Fig. 6.

A) Kink-ase attaches to a turning tile and the predecessor which will be re-oriented in μ. Simultaneously, glues are activated on the kink-ase cube structure attached to the turning tile to bind the turning tile

Fig. 5 (a) In step 0 (before replication begins) both fuel and tiles from T_σ bind to σ. Step 1 indicates the fuel tile binding with the leftmost S^+ tile in σ', propagating the binding of tiles from west to east indicated by blue arrow on the $++$ tile. Step 2 begins after all σ' glues are bound by strength-1, leading to the propagation of a second glue binding σ' from east to west. Additionally, glues on the north face of σ' tiles are activated and glues on the south face binding to σ are deactivated once they have a strength-2 connection to. Step 3 demonstrates the detachment - once the second glue binds to the fuel duple (φ_1, φ_2) signals propagate to detach from σ and σ'. (b) Process of translation: the information encoded in σ is copied to μ by a mapping of tiles via glue domains. Green glues on μ and μ' are flexible. One kink-ase (red) is used to convert μ to μ'
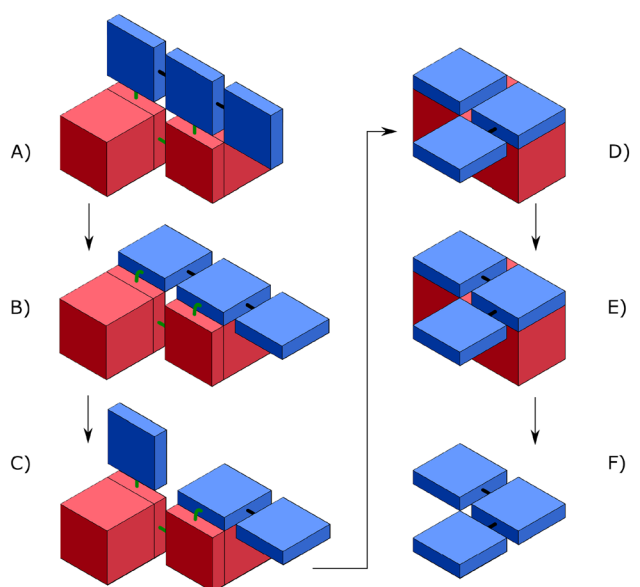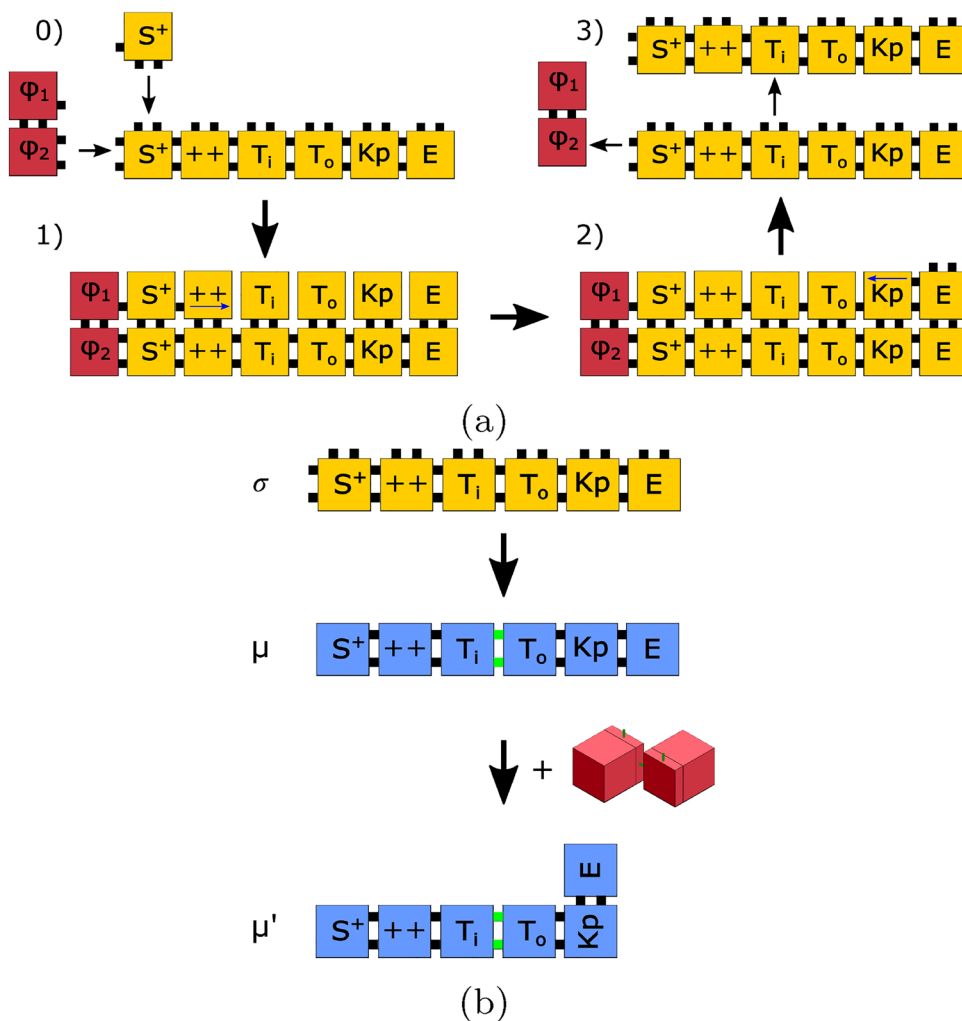
Fig. 6 Conversion of one turning tile. Blue tiles indicate μ, whereas the red indicate the kink-ase

face and to the kink-ase cube structure attached to the predecessor tile to enable the folding of the cube structure in step D). Note - glues connecting tiles in μ may be either rigid or flexible depending upon the Hamiltonian path generated for π. This does not effect any intermediate steps presented.

B) The turning tile's rear face binds to the kink-ase due to random movement allowed by the flexible glues which attach the kink-ase to the turning and predecessor tiles, i.e. the flexible bond allows the tile to rotate and randomly assume various relative positions. When it enters the correct configuration, the glues bind to "lock it in".

C) Upon connection of turning tile face to kink-ase cube, a signal deactivates the rigid glue attaching the predecessor tile to the turning tile. A signal activates glues on the exposed face of the kink-ase tile attached to cube and turning tile structure. The flexible connection between the predecessor tile and kink-ase ensures μ does not split into two pieces.

D) Kink-ase cube and kink-ase tile with activated glue bind on faces when they rotate into the correct configuration, bringing the turning tile into correct geometry with the predecessor tile. The kink-ase cube face adjacent to the predecessor tile activates its glue, allowing for binding with the face of the two. The flexible glue allows for random movement for the complementary glues to attach and bind. Concurrently, the flexible glue on the turning tile is deactivated and a rigid glue of similar type to the turning tile glue deactivated in step C) is activated.

E) A rigid glue between the turning tile and predecessor tile binds, leading to re-connection between both prior detached portions of μ. Activation of the final glue leads to the turning tile signaling to kink-ase to detach from μ.

F) This structure represents μ after one turning tile has been resolved. A completion signal is passed through glues attaching the turning tile and predecessor tile. This process continues for all turning tiles serially, working backwards from the termination tile. This is to prevent any interference between structures incurred by multiple adjacent turning tiles.

3.3 Assembly of π

At the end of translation, two strength-1 glues complementary to tiles in T_π are active on all tiles of μ'. The only cubic tile which starts with two complementary glues on is the start cubic tile. Once this cubic tile is bound to the start tile, a strength-1 glue of type 'c' is activated on the cube. This glue allows for the cooperative binding of the next cubic tile in the Hamiltonian path to the superstructure of both μ' and the first tile of π.

After this process continues and a cubic tile is bound to both its neighbors (or just one neighbor in the case of the start and end tiles) with strength 2, a 'd' glue is activated on the face of the cubic tile bound to μ'. This indicates to the flat tile of μ' that the cube tile is fully connected to its neighbors with strength 2. To prevent any hindrances to the placement of any cubic tiles in π, the flat tile jettisons itself from the remaining tiles of μ' by deactivating all active glues and becoming a junk tile.[5] This process is repeated, adding cube by cube until the end tile in μ' is reached. Once the end cube has been added to π, it has shape S^2 and μ' has been disassembled into junk tiles. An example process is shown in Fig. 5b, with a detailed step-by-step visualization of glue activation shown in Figs. 7, 8.

---

[5] Due to the asynchronous nature of signals, there may be instances which the addition of cubic tiles of π are temporarily blocked. These will be eventually resolved, allowing assembly to continue.

3.4 Tiles of T

We provide the enumerated sets of tiles in this section which provide for the dynamics as described in the prior sections.

3.4.1 T_σ

As shown in Fig. 4a, all tiles except for the end tile have the same structure of signals and glues, where the glues are a specific mapping to tiles in T_μ. Glues which bind between T_σ and T_μ have the μ subscript in the glue description. Glues without the μ subscript bind between the north and south glues of tiles in T_σ.

3.4.2 T_μ

The tiles presented in Fig. 9 represent the base tiles which make up a messenger sequence. Any glue which contains an 'f' subscript is a flexible glue. The tile denoted Ki is a placeholder for both Kp and Km tiles, where all glues which contain an 'i' can be replaced with p or m, respectively. All of the tiles aside from T_i, T_f, Kp_f or E can be a predecessor to a turning tile. This requires additional glues and signals in order to attach to a kink-ase structure. These modifications are shown in Fig. 10, and we note that these glues and signals overlay on top of the tiles in Fig. 9; glues not used in the turning process are omitted. The tiles to the right indicate the specific glues and signals for the Kp, Km tiles. The tiles to the left indicate the specific glues and signals which must be present on the predecessor tiles to Kp or Km. We note that Kp and Km can also be modified with the tiles on the left hand side. In the case of either two Kp or Km tiles in a row, it is required to leave the flexible glues f_f, g_f on instead of off when the 'p' glue on the east side of a tile is bound.

We note that the modifications require a mapping of a specific glue from T_σ to T_μ. This is accomplished by adding an additional 'm' or 'p' to the glue based upon the modification made. Glue which connect T_μ and T_π have the subscript π.

3.4.3 T_ϕ

The tiles presented in Fig. 11 are those that cause the replication of σ and form kink-ase. The kink-ase tiles first combine to form supertiles of size 4 as shown in Fig. 6. These supertiles are then able to perform the designated functions of the kink-ase. Similarly, the tiles φ_1 and φ_2 combine to a supertile in before replication of σ can begin.
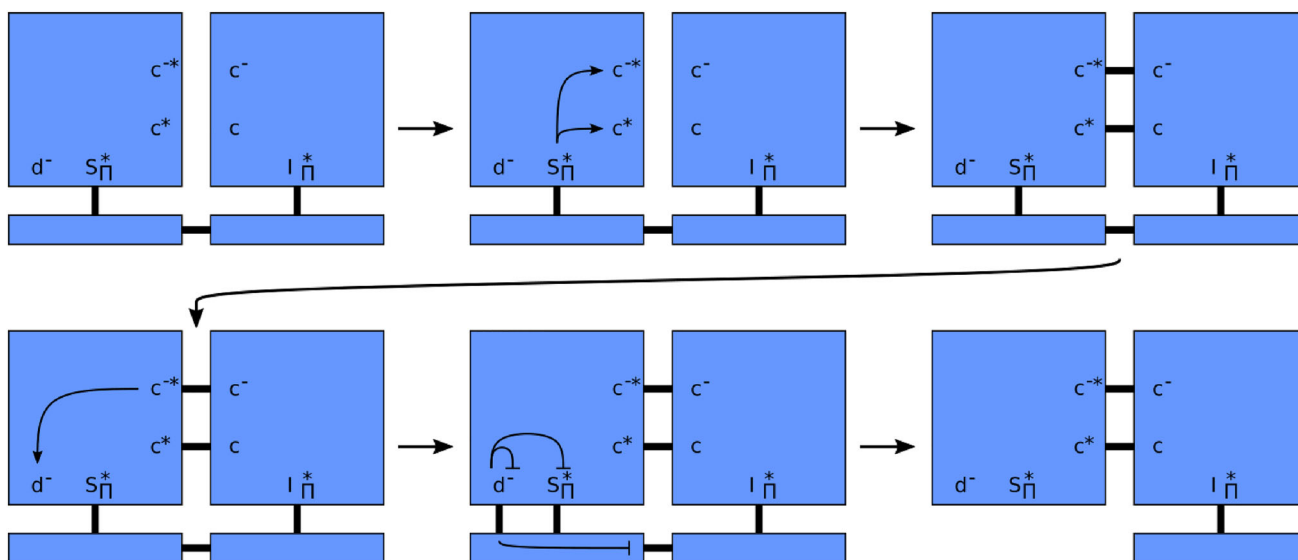
Fig. 7 The process of assembling π from μ'. Arrows within tiles represent the signals propagating through adjacent tiles to solidify connections between two successive cubic tiles in the Hamiltonian path of a phenotype. When the glue labeled d^- is activated, signals propagate which disable both d^- and S_{Π}^*
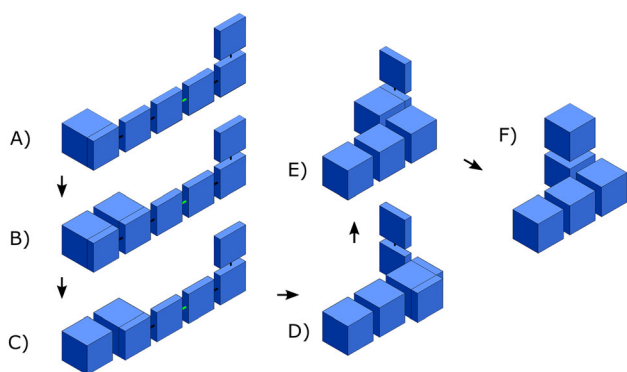

Fig. 8 Building π from μ' (same as in Fig. 5b). After the start cube binds to μ' in step A, the process of assembling π successively adds cubic tiles then detaches flat tiles from μ'. Step F is phenotype π originally encoded by σ

3.4.4 T_π

The tiles T_π are the structural blocks which recreate a desired shape given an input genome. These tiles are illustrated in Figs. 12 and 13. Two strength 1 glues of the type 'c' bind the final structure between cubic tiles in the Hamiltonian path dictated by σ.

3.5 Analysis of $\mathcal{R}$ and its correctness

Theorem 1 *There exists an STAM* tile set T such that, given an arbitrary shape S, there exists STAM* system $\mathcal{R} = (T, \sigma, 2)$ and S^2 self-assembles in $\mathcal{R}$ with waste size 4.*

We prove Theorem 1 via induction. Our base case is the start flat tile and its associated cube. Our inductive step is the addition of a cube and a direction associated with the next step of the Hamiltonian path within S^2. This direction is provided by the successor tile in μ', and all possible directions are enumerated in Fig. 14. At each step, we place a cubic tile in its associated direction based upon the flat tile in μ'. We analyze the possible direction of placement. Since μ is a translation of σ, x^- is not included as it is the location of the prior cubic tile. As a note, the directions provided in the proof reflect those indicated in Fig. 14, not necessarily the absolute reference of the entire system. Additionally, as our genome σ has a Hamiltonian path ending on an exterior face of S, we can guarantee that diffusion is possible for a tile at any stage of construction

x^+: This placement and output direction is carried out by the $++$ tile type - the cubic tile is placed in the existing direction of travel

y^+: This correlates to the T_i and T_o tile type.

y^-: This case is the most complex; we are changing the direction of travel in a direction which takes us through the tile of μ'. This requires the use of the following 4 tiles: Kpf, T_f, T_f, T_o. This could also be completed with a set of 3 tiles Kp, Km, Km, however this increases fuel usage per y^- from 1 to 3, and overall tile usage from 8 to 19 when including all the singleton tiles utilized to create the kink-ase structures consumed by the 3 turning tiles.

z^-: A single Km tile carries out this tile placement and path change. Note, the prior flat tile must additionally be modified to carry out the turning action by the kink-ase.
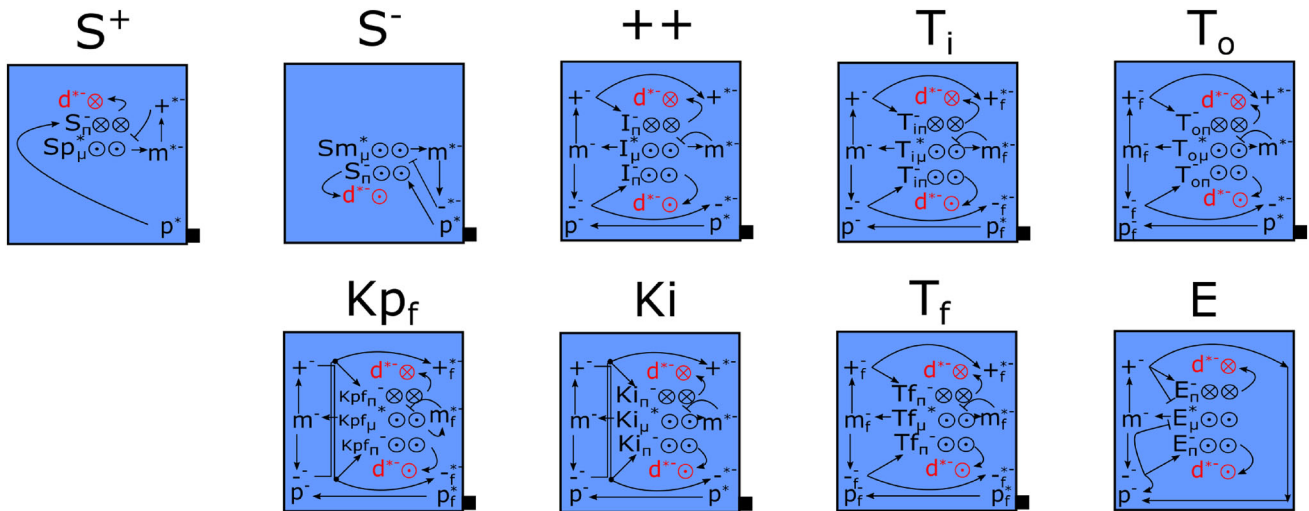
Fig. 9 Messenger tile types (non-kink). Note that the red 'd' glues have deactivation signals to all glues on the tile, but are omitted for visual clarity. This turns the messenger tile into a 'junk' product
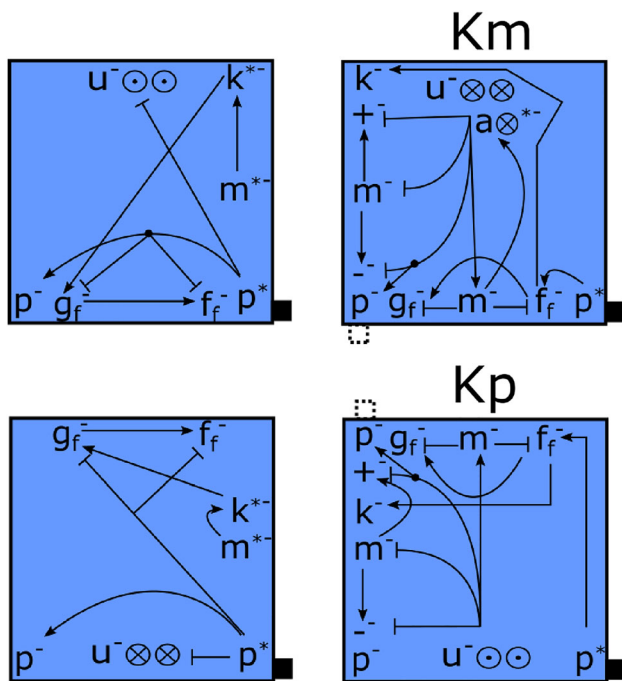

Fig. 10 Tile modifications for use with kink-ase. Note that the dashed square indicates the face that the 'p' glue is attached

z^+: A single *Kp* tile carries out this tile placement and path change. Note, the prior tile must additionally be modified to carry out the turning action by the kink-ase.

After the addition of a tile, we re-orient the frame of reference to align with that shown in Fig. 14. The last tile in the Hamiltonian path will not have a new direction - this is indicated by the end tile. We have then generated the structure S^2 utilizing R.

3.5.1 STAM* metrics of R

The STAM* metrics of R follow from the tileset found in Sect. 3.4:

- Tile complexity $= 57$

  - $|T_\sigma| = 22$
  - $|T_\mu| = 22$
  - $|T_\pi| = 7$
  - $|T_\phi| = 6$

- Tile shape complexity $= 2$
- Signal complexity $= 7$
- Seed complexity $= O(n)$; each cube in the phenotype must be placed by a tile, with some requiring multiple (e.g. turns). As described above, for any structure with greater than 2 tiles we end up with the following number of tiles in σ based upon the changes in directions which must occur: "start tile" + "end tile" $+|z^+| + |z^-| + 2|y^+| + 4|y^-| + |x^+|$.

4 A self-replicator that generates its own genome

In this section we outline our main result: a system which, given an arbitrary input shape, is capable of disassembling an assembly of that shape block-by-block to build a genome which encodes it. We describe the process by which this disassembly occurs and then show how, from our genome, we can reconstruct the original assembly. Here we describe the construction at a high level. We prove the following theorem by implicitly defining the system $\mathcal{R}$, describing the process by which an input assembly is
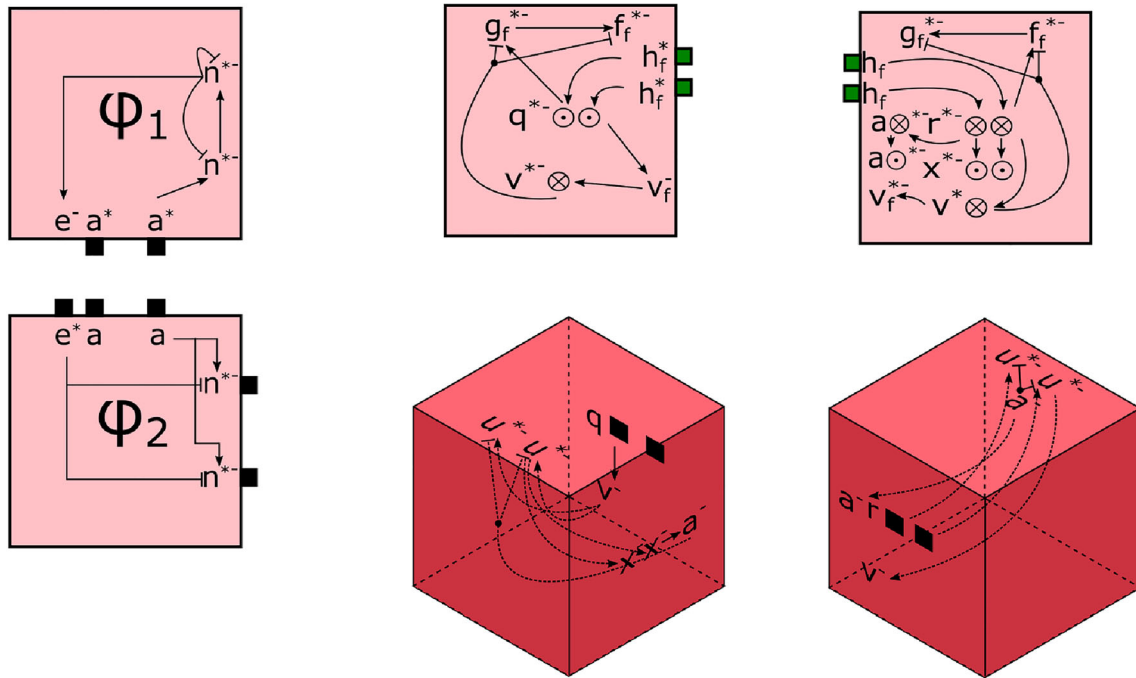
Fig. 11 Fuel tiles. Tiles on left utilized during replication of σ, tiles on right combine to form kink-ase structure
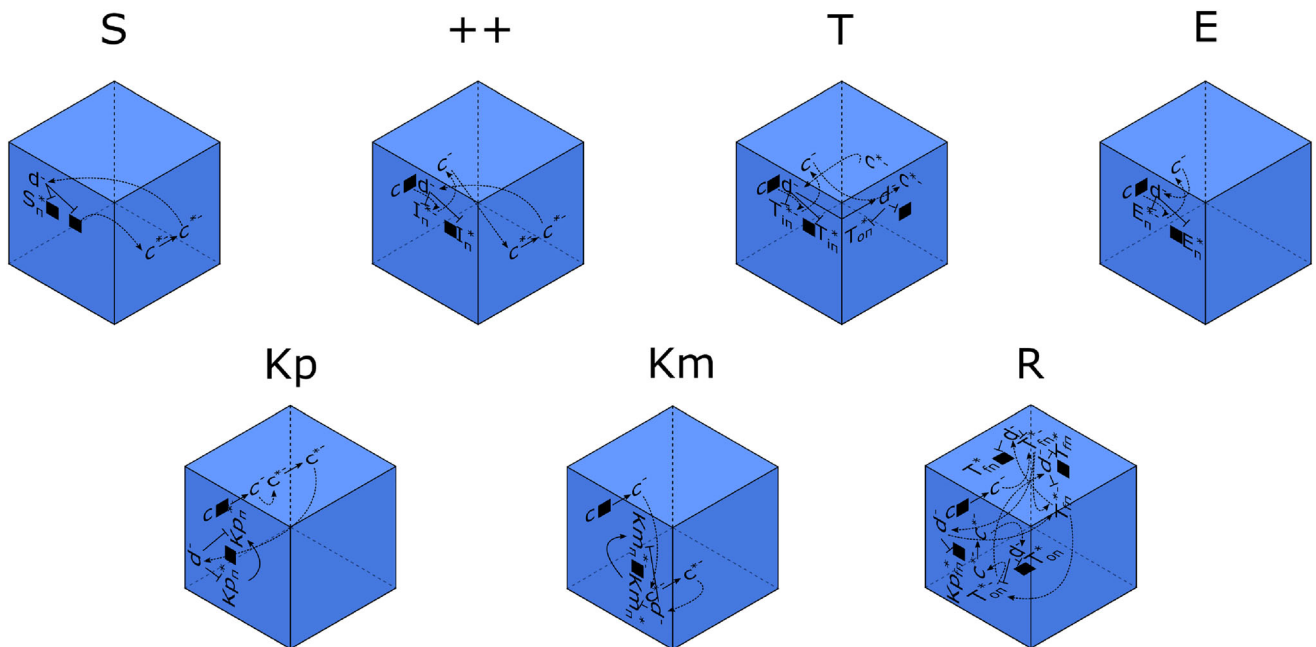

Fig. 12 Structural tiles which create the assembly π. Note that the 'R' tile has a second $Kp_{f\pi}^*$ glue activated, however is omitted for visual purposes
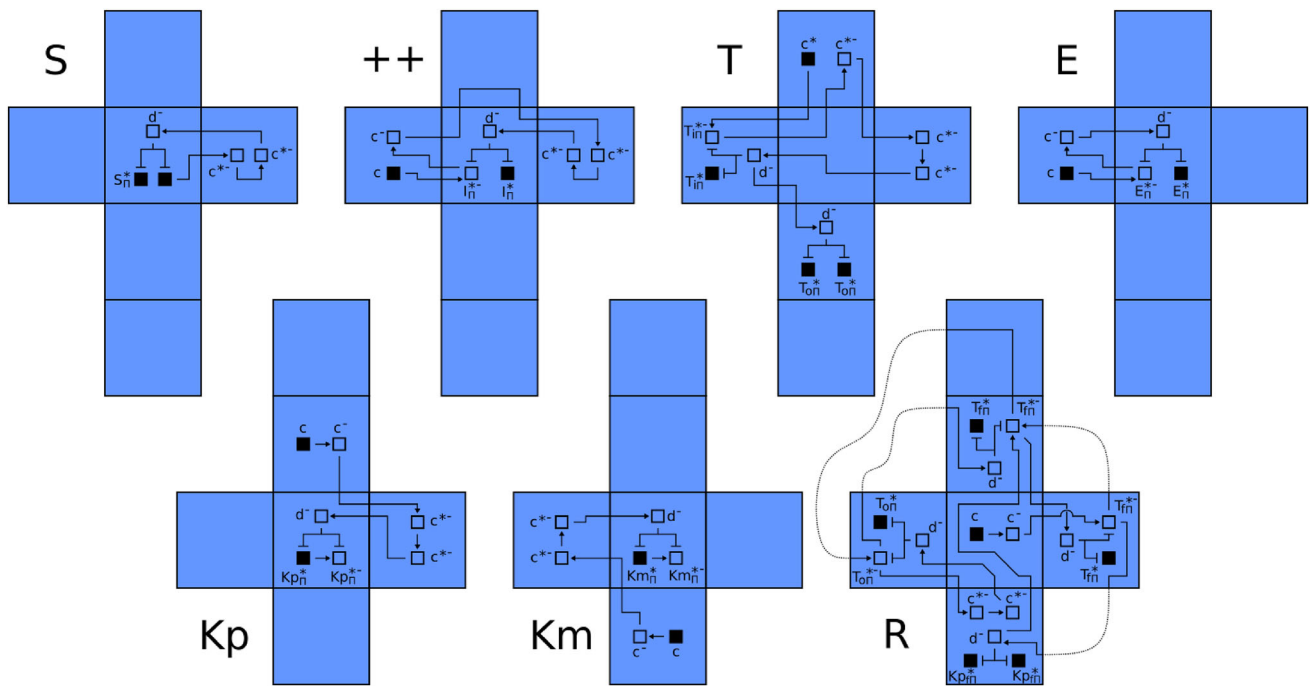
Fig. 13 Structural tiles which create the assembly π. This figure illustrates the same tiles as Fig. 12, but with the cubes unrolled into nets to make the signal paths more clear
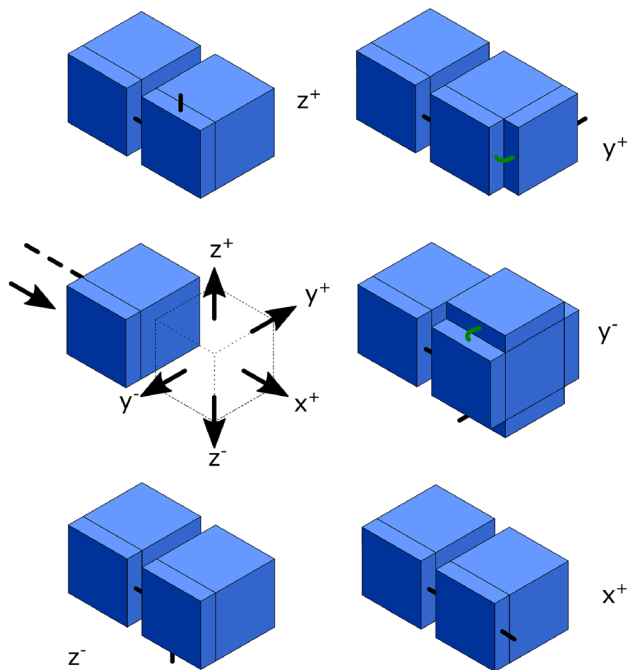

Fig. 14 The inductive steps required in the creation of π which follows a Hamiltonian path given by a σ. The arrow going into the flat tile is the direction taken by the Hamiltonian path in the prior tile addition step. The five arrows indicate possible directions for the direction of the Hamiltonian path after the placement of the transparent cubic tile

disassembled to form a "kinky" genome which is then used to make a copy of a linear genome (which replicates itself) and of the original input assembly.

Theorem 2 *There exists a universal tile set T such that for every shape S, there exists an STAM* system $\mathcal{R} = (T, \sigma_{S^2}, 2)$ where σ_{S^2} has shape S^2 and $\mathcal{R}$ is a self-replicator for σ_{S^2} with waste size 2.*

In this construction, there are two main components which here we call the *phenotype* and the *kinky genome*.

Given a shape S, the phenotype P will be a 2-scaled copy of the shape, so that each cube in S corresponds to a $2 \times 2 \times 2$ block of tiles in P. The shape of the phenotype will therefore be identical to S modulo our small, constant scale-factor. P will be made up of tiles from some fixed STAM* tile system $\mathcal{T}$ which we will define in more detail later.

Let H be a Hamiltonian path that goes through each tile in P exactly once. We will construct H later, but for now assume that it exists. Each tile in P will contain the following information encoded in its glues and signals.

– Which immediately adjacent tile locations belong to the phenotype
– Which immediately adjacent tile locations correspond to the next and previous points in the Hamiltonian path

– Any glues and signals necessary for allowing the deconstruction and reconstruction process to occur as described in Sects. 4.1 and 4.2

In our system, the genome will be constructed as the phenotype is deconstructed and then will be duplicated or used to make copies of the original phenotype. Throughout this section, we refer to the cubic tiles that make up the phenotype as structural tiles and the flat tiles that make up the genome as genome tiles. Additionally, the tiles used in this construction are part of a finite tile set T, making T a universal tile set. The genome is referred to as "kinky" due to the fact it must contain flexible glues, in contrast to the linear genome utilized in Sect. 3.

4.1 Disassembly

Given a phenotype P with embedded Hamiltonian path H, the disassembly process occurs iteratively by the detachment of at most 2 of tiles at at time. The process begins by the attachment of a special genome tile to the start of the Hamiltonian path. In each iteration, depending on the relative structure of the upcoming tiles in the Hamiltonian path, new genome tiles will attach to the existing genome encoding the local structure of H (to be used during the reassembly process) and, using signals from these incoming genome tiles, a fixed number of structural tiles belonging to nearby points in the Hamiltonian path will detach from P (Fig. 15). A property called the *safe disassembly criterion* will be preserved after each iteration assuring that disassembly can continue as described. This process will continue until we reach the last tile in the Hamiltonian path. Once the final genome tile binds to the existing genome and this final tile, signals will cause these final structural tiles to detach and leave the genome in its final state where it can be used to make linear DNA as described above or replicate that phenotype as described below.
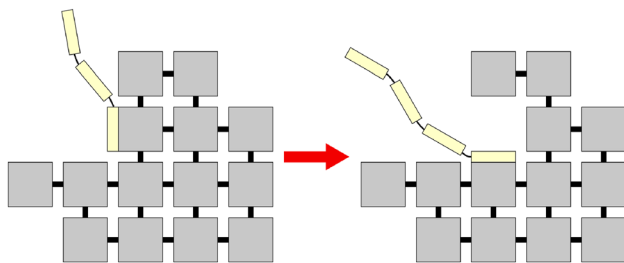

Fig. 15 During disassembly, the genome will be dangling off of a single structural tile in the phenotype. In each iteration, a new genome tile will attach and the old structural tile will detach along the Hamiltonian path embedded in the phenotype

4.1.1 Relevant tiles and directions

In each iteration of our disassembly procedure, indexed by i, we will label a few important directions and tiles which will be useful. Since our tiles in this model are not required to reside in a fixed lattice, we define our cardinal directions $\{N, E, S, W, U, D\}$ arbitrarily so that they are aligned with the faces of some arbitrarily chosen tile in our phenotype. These directions will only be used when referring to tiles bound rigidly to the phenotype so there will be no ambiguity in their use.

The first tile, which we will call the *previous structural tile* and write as S_i^{prev}, is the structural tile to which the genome is attached at the beginning of iteration i. This tile will detach from the rest of the phenotype by the end of iteration i. The *next structural tile*, written S_i^{next}, is the structural tile to which the genome will be attached at the end of iteration i. Note that in some cases, this may not be the tile corresponding to the next tile in the Hamiltonian path, since we may detach more than one tile in an iteration.

We will refer to the corresponding attached genome tiles accordingly and write G_i^{prev} and G_i^{next} respectively.

The first direction, which we will call the *next path direction* and write D_i^p, represents the direction from the previous structural tile to the next tile in the Hamiltonian path. Next, we will refer to the direction corresponding to the face of the previous structural tile upon which the previous genome tile is attached as the *genome direction* and write D_i^g.

We also define a direction called the *dangling genome direction*, written D_i^d, relative to the previous genome tile attached to the previous structural tile. At each iteration of the disassembly process new genome tiles will attach to the existing genome and the phenotype. By the end of in iteration, the previous genome tile will have detached from the structure and the next genome tile will be attached to the next structural tile. The dangling genome direction is defined to be the direction relative to the previous genome tile in which the rest of the genome is attached.

Figure 16 illustrates what these directions look like in a particularly simple case.

4.1.2 The safe disassembly criterion

To facilitate in showing that the disassembly process works without error, we define a criterion which is preserved through each iteration of the disassembly process effectively acting as an induction hypothesis. We call this criterion, the *safe disassembly criterion* or *SDC*. The SDC is met exactly when all of the following are met:
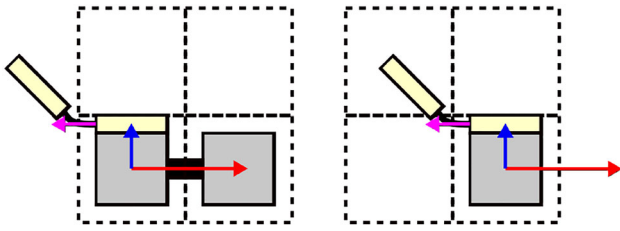
Fig. 16 The relevant directions before and after an iteration of the disassembly process. The red arrow represents the next path direction, the blue arrow represents the genome direction, and the magenta arrow indicates the dangling genome direction. In this simple case the directions do not change after an iteration, but this is not always the case

1. There is no phenotype tile in the location location in the direction D_i^g relative to the previous structural tile. This essentially means that there was room for the previous genome tile to attach to the previous structural tile.
2. At the current stage of disassembly, there is a path of empty tile locations that connects the previous tile location to a location outside the bounding box of the phenotype. This condition ensures that if our path digs into the phenotype during disassembly, there is a path by which detached tiles can escape and new genome tiles can enter to attach.
3. The dangling genome direction is not the same as the next path direction. This ensures that the existing genome is not dangling off of the previous genome tile in such a way that it would block the attachment of the next genome tile. This also ensures that our genome will never have to branch, though it may take turns.
4. Both the previous genome tile and some adjacent structural tile are presenting glues which allow for the attachment of another genome tile.

4.1.3 Disassembly cases

In each iteration of disassembly, there will be 6 effective possibilities regarding the local structure of the Hamiltonian path. Each of these possibilities will necessitate a different sequence of tile attachments and detachments for disassembly to occur. These cases are illustrated in figure 17 and described as follows.

Lemma 1 *The 6 cases illustrated in Fig. 17 are all of the possible cases for a disassembly iteration.*

First note that the next path direction can either be perpendicular to the previous genome direction or not. If it is, we consider two cases. Either the tile location in the next genome direction relative to the next structural tile in

the Hamiltonian path contains an attached structural tile or it doesn't. Case 1 is where it doesn't. If on the other hand it does, call the tile in that location the blocking tile; case 2 occurs when the blocking tile follows the next structural tile in the Hamiltonian path and case 3 occurs when it doesn't.

Supposing that the next path direction is not perpendicular to the previous genome direction, either it's the same direction or the opposite direction. By condition 1 of the SDC, it cannot be the same direction since there can be no structural tile attached in that location so all other cases must have the next path direction opposite the previous genome direction.

Now we define the working direction to be the direction opposite the dangling genome direction. This direction will be the direction in which genome tile attachments will occur during the remaining cases. Ultimately this choice is arbitrary, except that the working direction cannot be the dangling genome direction. Let location a be the tile location in the working direction of the previous structural tile and location b be the tile location in the opposite direction of the next path direction of location a. Case 4 is when neither location a nor b contains an attached structural tile, case 5 occurs when only location a has an attached tile, and case 6 occurs otherwise.

Notice that since we defined these cases by dividing the possibility space into pieces where either some condition is or isn't met, this enumeration of cases represents all possibilities, thus proving Lemma 1.

4.1.4 The disassembly process

Here we describe the disassembly process in enough detail that anyone familiar with basic tile assembly constructions should be able to derive the full details of the process without much difficulty.

Before any of the iterative disassembly cases can occur, the disassembly process begins with the attachment of the initial genome tile. The structural tile corresponding to the first point in the Hamiltonian path will be presenting a strength 2 glue to which this initial genome tile can attach. At this point in the process, this will be the only tile to which anything can attach with sufficient strength. This attachment activates a signal which turns off all glues in this initial structural tile except those holding it to the initial genome tile and the next structural tile in the Hamiltonian path. Also, now that this first genome tile has attached, the next genome tile can cooperatively attach initiating the disassembly process so that in the first iteration, the initial genome tile acts as the previous genome tile and the structural tile to which it's attached acts as the previous structural tile.
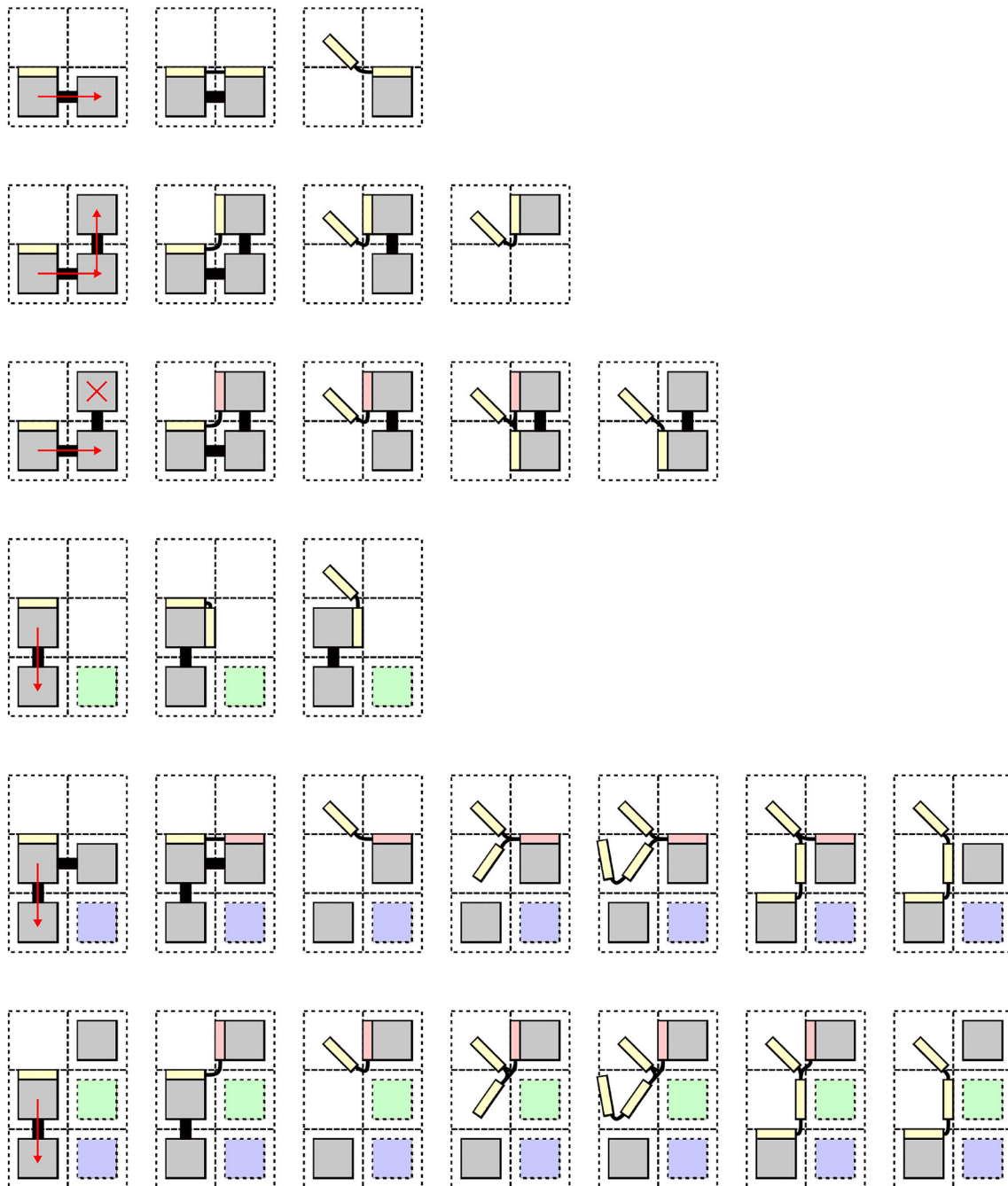
Fig. 17 A side view of the disassembly process for all 6 cases. Each row is a unique case, where the leftmost image is the starting condition. We orient these illustrations so that the previous genome direction is always up for convenience. Also note that we always illustrate the dangling genome direction to the left, but this need not be the case, this is just for making visualization easier. In reality, the dangling genome direction could be in any direction relative to the previous genome, so long as it satisfied the SDC condition that it is not the same as the next path direction. Gray squares represent attached structural tiles, green squares represent a location in which it does not matter if an attached structural tile exists, and empty squares represent locations in which no attached structural tile exists

In each following iteration, once complete, what used to be called the next structural tile and next genome tile become the previous structural tile and previous genome tile for the next iteration and any relevant directions in the next iteration are specified relative to these new previous tiles.

Each of the cases as described above makes use of a unique sequence of tile genome attachments and signals; however, much of the logic in each of the cases is the same. We will describe two of the cases in greater detail than the rest, specifically cases 1 and 3, since understanding the details of those cases will make understanding the others
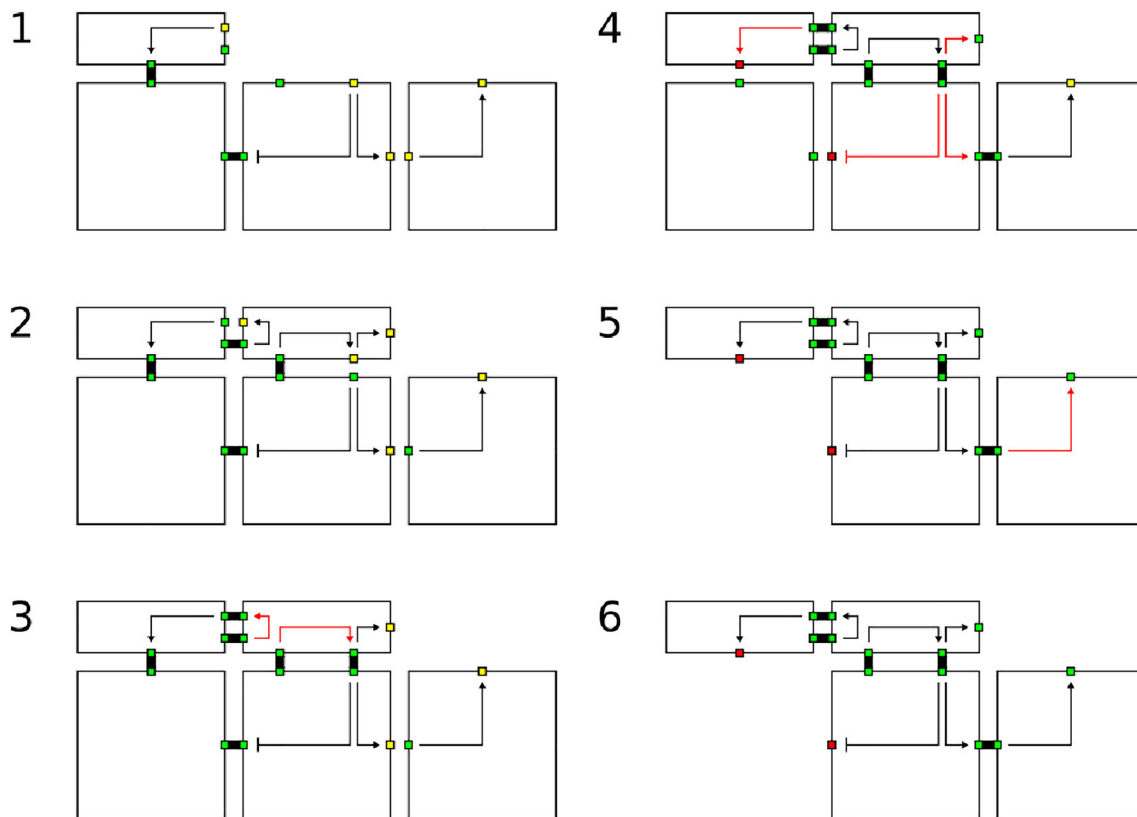
Fig. 18 A side view of some of the relevant glues and signals firing during the simplest disassembly case

much easier. Figure 17 illustrates the high level process of each case. It's important to keep in mind that the entire structure of the Hamiltonian path is encoded in the glues and signals of the phenotype tiles. This means that these cases can occur without issue since, for example, in an iteration where case 3 needs to occur, there will only be the glues and signals for case 3 present on the relevant tiles and none that would allow tiles for say case 5 to attach.

1. This case is the simplest case and is illustrated in Fig. 18. First, a genome tile G attaches cooperatively to the previous genome tile and the next structural tile. This attachment causes signals to fire in G that activate 2 glues from the latent state to the on state. The first of these glues is a rigid, strength 2 glue that allows G to bind rigidly and with more strength to the next structural tile. The other glue is a flexible, strength 2 glue that allows the genome to more strongly attach to the previous genome tile. The attachment of these glues activate signals which turn the old glues serving the same purpose into the off state. Additionally, signals are activated in the previous genome tile and the next structural tile disabling the glues in both that held onto the previous structural tile. Signals also deactivate any glues in the next structural tile that are attached to all other structural tiles except for the one following it in the Hamiltonian path.

At this point, there are no glues holding the previous structural tile to the genome nor the phenotype. This structural tile is now free to float away from what's left of the phenotype which is possible since the genome to which it was attached is now only bound with a flexible glue to the next genome tile and, by SDC condition 2, there is a path of empty tile locations along which it can escape.

In addition to all of the signals described previously, signals also activate a glue on the next genome tile which enables the attachment of the genome tile that will initiate the next iteration of the disassembly process.

By definition of case 1, SDC conditions 1 and 2 will be met after this process is done. Additionally, since the dangling genome direction now corresponds to the direction of the detached structural tile, condition 3 must also be satisfied. Condition 4 is also satisfied since glues were activated on the upcoming tile in the path to allow for cooperative binding of a new genome tile.

2. This case is largely similar to case 1 except that the next genome tile attaches to the structural tile following the next structural tile in the Hamiltonian path since the next is being blocked. In this case, it will be necessary for this tile to "know" that the next genome

tile will attach to it. To accomplish this, all of the necessary glues that allowed the disassembly process to occur in the first case exist on this tile instead of the one immediately following the previous structural tile in the Hamiltonian path.

3. In this case, we have to remove the previous structural tile before we can attach the genome to the next structural tile since it is being blocked. We do this by utilizing what we call *utility genome tiles*. These utility tiles are flat tiles that temporarily affix the genome to another part of the phenotype so that the previous tile can safely detach without the genome also detaching.

   At first, this case proceeds similar to case 2 (and is illustrated in Fig. 17), but with a utility tile attaching to the blocking structural tile instead of the next genome tile. This attachment activates signals which cause the previous structural tile to detach. Since the tile to which the utility tile attached is not immediately adjacent to the previous structural tile, this is done using a chain of signals (which is a common gadget in STAM systems). The detachment of the previous structural tile allows the next genome tile to cooperatively bind to the previous one and to the next structural tile. This attachment causes signals to deactivate glues holding the utility tile in place allowing it to detach.

4. This case is largely degenerate and doesn't involve detachment of any tiles. Instead, utilizing cooperation, the next genome tile attaches to another face of the previous structural tile which also plays the part of the next structural tile. Depending on the tile or lack thereof in the green tile location from Fig. 17, the next iteration will either be case 1, 2, or 3.

5. This case is largely similar to case 3 except that the utility tile attaches in a different location. Once this occurs, instead of a new tile attaching cooperatively to the next tile, which is impossible since the next tile is not adjacent to the previous genome tile, a filler genome tile attaches to glues that are now present after the attachment of the utility genome tile. This filler genome tile acts as a spacer and after signals activate its glues, the next genome tile can attach to it and the next genome tile.

   There is one consideration that needs to be made in this case. If the tile location illustrated in blue in case 5 of Fig. 17 is the tile in the Hamiltonian path immediately following the next structural tile, then condition 3 of the SDC will not be met. This is because the dangling genome direction at the start of the next step will be in the same direction as the next path direction. To handle this, we simply require that two filler genome tiles attach between the utility tile and the next genome tile in this case. Since the structure of the

Hamiltonian path is known in advance, this is possible, by requiring a different utility tile attach in the case where two filler tiles would be necessary than if only one was. Now, similar to case 3, the utility tile is free to detach following signals from the attachment of the next genome tile.

6. This case is identical to case 5 except that the utility tile attaches in a different location.

4.2 Reassembly

At each iteration of the disassembly process, tiles attached to the genome encoding which tiles were detached. In some stages multiple tiles were detached, but it shouldn't be hard to see how that could be encoded in a single genome tile. Recall that this genome is a "kinky" genome. At this point, we could have defined the disassembly process above so that this genome immediately reconstructs the phenotype, the process for which is defined below; however, the definition of self-replicator requires that we construct arbitrarily many copies of the phenotype. Because of this, we can instead define the genome here so that it has the glues and signals necessary to convert into a linear genome as described in Sect. 3.

We refer to the processes described in Sect. 3.2.2. There we use a gadget called kink-ase to convert a linear sequence of genome tiles into a "kinky" one which is capable of constructing a shape. This process is easily reversible using a similar gadget which follows the steps in Fig. 6 in reverse. This process converts the kinky genome made during the disassembly of our phenotype into a linear genome which can be replicated arbitrarily using the process described in Sect. 3.1. For our purposes, it's useful to modify this linear genome duplication process so that our linear genome is duplicated into two copies: one that can be further used for genome duplication and one that can be converted back to kinky form and used to reassemble the phenotype. This simply requires that we specify a second set of the corresponding glues and signals on the genome constructed from the disassembly process. This guarantees that we are generating arbitrarily many copies of the phenotype.

Once we have kinky genomes ready to reconstruct the phenotype, we can begin the reassembly process. This process behaves much like the disassembly process, but with the genome being disassembled and the structure being reassembled. Once a reassembly fuel tile attaches to the special tile at the end of the genome, signals will activate glues allowing a structural tile, identical to the last tile in the Hamiltonian path of the original phenotype, to attach. This initiates the reassembly process and each of the tiles in the Hamiltonian path will attach in reverse order as

the genome disassembles from the back. This process is in some ways more straightforward than disassembly because the only tiles that detach are genome tiles and they detach completely. In the assembly process, both structural tiles and genome tiles had to detach and the detachment of genome tiles had to happen in such a way that they were still attached by flexible glues to the rest of the genome.

The following is an outline of the reassembly processes for each of the cases. Figure 17 can still be used as a reference but be careful to keep in mind that the process is happening in the opposite direction, initiated by the attachment of what was called the *next* structural tile in the disassembly process. In this section we reverse the terminology so that in each iteration, what were the previous structural and genome tiles are now the next structural and genome tiles and vice-versa. In each iteration of this process, the attachment of the previous structural tile to our genome initiates the sequence of attachments, detachments, and signals that allow the next structural tile to attach and the previous genome tile to detach.

1. This is the most basic case, the attachment of the previous structural tile to the genome activates glues on the next genome tile. This enables the next structural tile to attach cooperatively which causes signals to deactivate glues so that the previous genome tile detaches.

2. The attachment of the previous structural tile in this iteration activates glues on it which immediately allows the next structural tile to attach. Again this attachment activates signals which turn on glues to allow another tile to attach forming the corner. Finally, the next genome tile can bind to this last structural tile which causes glues to deactivate so that the previous genome tile detaches.

3. The attachment of the structural tile to the genome in the previous iteration activates a glue on the genome tile and adjacent structural tile allowing a utility tile to attach. This causes signals to deactivate glues holding the previous genome tile and activating glues on the structural tile to which it was bound. This allows a new structural tile to attach and then the corresponding genome tile. These attachments create signal paths that deactivate glues on the utility tile and the structural tile to which it was attached, allowing it to fall off.

4. This stage just represents the genome tile turning a corner which causes the old genome tile to detach after signals deactivate its glues. This can only happen after case 1, 2, or 3 similar to the analogous case during disassembly.

5. The attachment of the structural tile activates glues which allow the utility tile to attach. This attachment initiates signals which do 3 things. the signals deactivate glues holding the previous genome to the

structural tile, the signals deactivate glues holding the utility tile to the old genome tiles, and the signals activate glues on the next genome tile. The next genome tile can then cooperate with the old structural tile to attach a new structural tile. Note that in this case the filler genome tiles from the disassembly will remain attached to the previous genome tile and they will detach as a short chain.

6. This case is almost identical to the previous case with a slightly different binding location for the utility tile.

Note that in each of the cases described above it's possible to reassemble the phenotype structure using the same tiles that were originally in the seed phenotype. As described here, we require that some of the signals in these reassembled phenotype tiles will be fired to facilitate in the reassembly process; however, with a more careful design it wouldn't be difficult to describe a process which reassembles the phenotype without using any signals on the structural tiles if this was a desired property. Additionally, during cases 5 and 6, pairs of filler tiles will detach depending on the next direction of the path in that iteration. This results in our waste size being 2, but again with a more careful design it would be easy to specify tiles which, say, bind to these waste pairs and break them down into single tiles if having waste size 1 was a desired property.

4.3 Phenotype generation algorithm

In this section, we describe an efficient algorithm for describing the $STAM^*$ system in which this process runs. Given that we require complex information to be encoded in the glues and signals of our components, particularly in the phenotype since it requires an encoded Hamiltonian path, it might seem like we are "cheating" by baking potentially intractable computations in these glues and signals. This however is not the case in the sense that, as we will show, all of the required tiles, glues, signals, paths, etc. (all from a fixed, finite set of types) can be described by a polynomial time algorithm given an arbitrary shape to self-replicate.

The algorithm described consists largely of two parts. First, we will determine a Hamiltonian path through our shape, and second we will use this path to determine which glues need to be placed where on our tiles.

4.3.1 Generating a hamiltonian path

Lemma 2 *Any scale factor 2 shape S^2 admits a Hamiltonian path and generating this path given a graph representing S^2 can be done in polynomial time.*

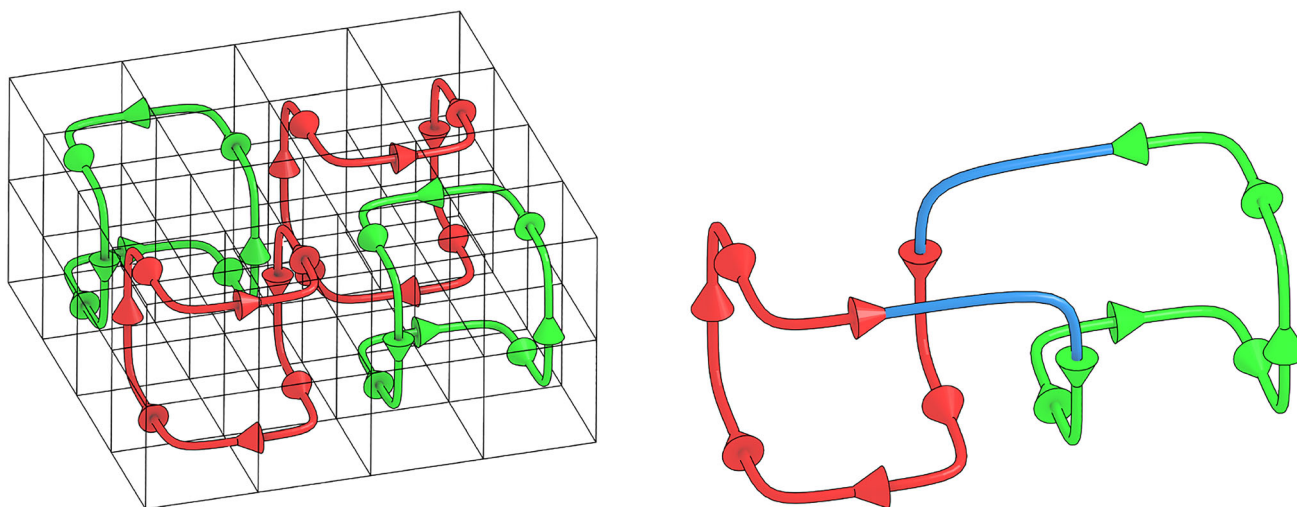In general, the problem of finding a Hamiltonian path through a graph is **NP**-complete and may be impossible for

Fig. 19 (Left) Each $2 \times 2 \times 2$ block of space is assigned an orientation graph which will be used to help generate the Hamiltonian path through our shape. Adjacent blocks are assigned opposite orientation graphs, the edges of which will help guide the Hamiltonian path around the shape. (Right) Orientation graphs of adjacent blocks are joined to form a continuous path

many shapes we may wish to use; however, if we scale our shape by a constant factor of 2, that is replace every voxel location with a $2 \times 2 \times 2$ block of tiles, then not only is there always a Hamiltonian path, but it can be computed efficiently. The algorithm for generating this Hamiltonian path is described in further detail in Cheung et al. (2011) and was inspired by Summers (2012), but we will describe the procedure at a high level here using terminology that is convenient for our purposes.

1. Given a shape S, we first find a spanning tree T through the graph whose vertices correspond to locations in S.
2. We embed this spanning tree in a space scaled by a factor of 2 so that each vertex corresponds to a $2 \times 2 \times 2$ block of locations.
3. To each $2 \times 2 \times 2$ block in this space, we assign one of two orientation graphs G_o^1 or G_o^2. These graphs each form a simple oriented cycle through all points. These graphs are assigned so that they form a checkerboard pattern such that no blocks assigned G_o^1 are adjacent to any blocks assigned G_o^2 and vice versa. Figure 19 illustrates what the orientation graphs look like for adjacent blocks.
4. For each edge in the spanning tree T, we join the orientation graphs corresponding to the vertices of the edge so that they form a single continuous cycle as illustrated in Fig. 19. This process is described in more detail in Cheung et al. (2011).
5. Once we do this for all edges in our spanning tree, the connected orientation graphs will form a Hamiltonian circuit through the $2 \times 2 \times 2$ blocks corresponding to the tiles in our shape. This is easy to see by analyzing a few cases corresponding to all possible vertex types in the spanning tree and noting that in none of them does

the path ever become disconnected. This is done in Cheung et al. (2011).

The resulting Hamiltonian path, which we will call H, passes through each tile in the 2-scaled version of our shape and only took a polynomial amount of time to compute since spanning trees can be found efficiently and only contain a polynomial number of edges. Given H, we can arbitrarily choose some vertex on the surface of our shape to represent the starting point of our path H_1 and label the rest of the path in order with respect to this one so that the next point is labeled H_2, then H_3, and so on. Additionally, we can also keep track of the location in space relative to some fixed origin to which each point in our path belongs and note that, using common data structures and basic arithmetic, determining the index of points in H given a location can be done efficiently.

4.3.2 Determining necessary information to encode in glues and signals

Recall that each case of the disassembly and reassembly processes sometimes required tiles nearby in space to have glues and signals to facilitate each step of the process. We define the following algorithm which is able to describe these glues and signals, showing that we can efficiently describe the tiles necessary for our construction.

Begin with tile H_1 and iterate over the entire Hamiltonian path performing the following operations with the current tile labelled T_i and keeping track of a counter t which starts at 0.

1. Determine which of the 6 disassembly cases would apply to this particular tile by looking at adjacent tile

locations and considering only those tiles not yet flagged with a detachment time.

2. At this point, we know exactly which case T_i will use during the detachment process. Assign any glues and signals necessary to this tile and adjacent tiles.

3. Flag T_i as being detached at time t.

4. If T_i used case 2, also mark the tile following T_i as being detached at time t and skip the next tile in the path for the next iteration.

5. increment t and i.

Our algorithm now knows which glues and signals are necessary for each tile that will make up the phenotype. We can now iterate over all tiles in the construction and make a set consisting of each unique tile in the phenotype. Additionally, the genome tiles necessary for the process are even simpler to define since there is only a small fixed number needed for each case. This shows that the system in which this process occurs can be described efficiently by an algorithm and that we are not doing an unreasonable amount of pre-computation by including the necessary information in our glues and signals.

4.3.3 Glues for converting to linear DNA

The disassembly process above results in arbitrarily many "kinky" genomes which are capable of being used to produce a replica of the original phenotype. In order for this process to be possible however, the kinky genome produced by the disassembly process needs glues and signals to indicate locations that should be "un-kinked" and replicated. This is no problem however since the only cases in the disassembly process that could induce a kink in our constructed genome are 1, 2, and 3. The kink induced in the genome in any of these cases solely depends on the dangling genome direction and next path direction. Since there are only a finite number of such cases and since our tileset will have a unique set of genome tiles that attach in each such case, we can easily specify the necessary glues and signals to the corresponding genome tiles. This guarantees that the conversion to linear DNA is possible for any genome constructed by the disassembly process.

4.4 Correctness of theorem 2

First, we restate Theorem 2 for convenience:

Theorem 2 *There exists a universal tile set T such that for every shape S, there exists an STAM* system $\mathcal{R} = (T, \sigma_{S^2}, 2)$ where σ_{S^2} has shape S^2 and $\mathcal{R}$ is a self-replicator for σ_{S^2} with waste size 2.*

We have shown how, given any shape S as input, we can scale it by factor 2 to S^2 and efficiently find a Hamiltonian

path through S^2. We can then compute the tile types and signals needed at each location to build a phenotype which can serve as a seed supertile for an STAM* system $\mathcal{R}$ using a universal tile set T. At temperature 2, $\mathcal{R}$ will deconstruct the input supertiles to create kinky genome assemblies. Each kinky genome assembly will then first create a copy of the linear genome, and then either continue to create copies of the linear genome, or initiate the growth of a new copy of the phenotype (which consumes the copy of the kinky genome). The new copies of the phenotype will become terminal assemblies, in the shape of S^2. The other terminal assemblies are junk assemblies of size ≤ 2 (during the reassembly process for cases 5 and 6, for certain next path directions, pairs of filler tiles will detach), and the linear genome assemblies are never terminal as each facilitates the growth of infinite new copies. Thus, $\mathcal{R}$ is a self-replicator for S^2 and since this works for arbitrary shapes at scale factor 2, T is a universal tile set for shape self-replication for the class of scale factor 2 shapes.

5 Shape building via hierarchical assembly

In this section we present details of a shape building construction which makes use of hierarchical self-assembly. The main goals of this construction are to (1) provide more compact genomes than the previous constructions, and (2) to attempt to more closely mimic the hierarchical assembly that occurs in the replication of biological systems, e.g. individual proteins are independently constructed and then they combine with other proteins to form cellular structures. First, we define a class of shapes for which our base construction works, then we formally state our result.

Let a *block-diffusable* shape be a shape S which can be divided into a set of rectangular prism shaped blocks[6] whose union is S (following the algorithm of Sect. 5.1) such that a connectivity tree T can be constructed through those blocks and if any prism is removed but T remains connected, that prism can be placed arbitrarily far away and move in an obstacle-free path back into its location in S.

Theorem 3 *There exists a tile set U such that, for any block-diffusable shape S, there exists a scale factor $c \geq 1$ and STAM* system $\mathcal{T}_S = (U, \sigma_{S^c}, 2)$ such that S^c self-assembles in $\mathcal{T}_S$ with waste size 1. Furthermore, $|\sigma_S| = O(|S|^{1/3})$.*

To prove Theorem 3, we present the algorithm which computes the encoding of S into seed assembly σ_S as well as the value of the scale factor c (which may simply be 1),

---

[6] A rectangular prism is simply a 3D shape that has 6 faces, all of which are rectangles.

and then explain the tiles that make up U so that $\mathcal{T}_S$ will produce components that hierarchically self-assemble to form a terminal assembly of shape S^c. At a high level, in this construction the seed assembly is the genome, which is a compressed linear encoding of the target shape that is logically divided into separate regions (called genes), and each gene independently initiates the growth a (potentially large) portion of the target shape called a block. Once sufficiently grown, each block detaches from the genome, completes its growth, and freely diffuses until binding with the other blocks, along carefully defined binding surfaces called interfaces, to form the target shape.

It is important to note that there are many potential refinements to the construction we present which could serve to further optimize various aspects such as genome length, scale factor, tile complexity, etc., especially for specific categories of target shapes. For ease of understanding, we present a relatively simple version of the construction, and in several places we point out where such optimizations and/or tradeoffs could be made. Throughout this section, we will refer to S as the target shape of our system. Note that for some shapes, it may be the case that a scale factor $c > 1$ is required for the input shape S (and the details of how that is computed are provided in Sect. 5.2) but for simplicity we'll refer to the target shape as S whether or not it is a scaled version. We will first describe how the shape S can be broken into a set of constituent blocks, then how the interfaces between blocks are designed, then how individual blocks self-assemble before being freed to hierarchically combine into an assembly of shape S.

5.1 Decomposition into blocks

Since S is a shape in $\mathbb{Z}^3$, it is possible to split it into a set of rectangular prisms whose union is S. We do so using a simple greedy algorithm which seeks to maximize the size of each rectangular prism, which we call a block, and we call the full set of blocks B.

After the application of a greedy algorithm to compute an initial set B, we refine it by splitting some of the blocks as needed to form a binding graph in the form of a tree T such that every block is connected to at least one adjacent block, but also so that each block has no more than one connected neighbor in each direction in T. This results in the final set of blocks that combine to define S, can join along the edges defined by T, and each block has at most 6 neighbors to which it combines. (Fig. 20 shows a simple example.)

Note that for our shape-replicating construction to work for S, it also requires that S, once divided into rectangular

prisms, is block-diffusable. Our algorithm does not ensure block-diffusability, and in fact, we conjecture that there exist shapes for which this is not possible without arbitrarily scaling the shapes. Below, we provide the algorithm which splits S into a set of blocks.

1. Define $S' = S$.
2. Initialize the set of blocks $B = \varnothing$.
3. Define the function P so that on input $v \in S'$ (i.e. v is a voxel in S'), $P(v)$ returns the largest (by volume) rectangular prism (as the set of coordinates contained within it) containing v within S'.
4. Let p_{max} be the largest rectangular prism (by volume) returned by P for any $v \in S'$.
5. Add p_{max} as a block to the set of blocks B, and remove the voxels of p_{max} from S'. (Note that this may make S' into a disconnected set of points, but that is okay.)
6. If $S' \neq \varnothing$, return to step 5.1.

We now have B as a preliminary set of blocks, which we will modify as necessary to ensure that each block has only one adjacent neighbor to which it will need to bind in each direction.

1. Define the graph G such that for each $b \in B$, G has a corresponding node, and there is an edge between each pair of nodes of G that correspond to blocks that are adjacent to each other in S.
2. Generate a tree T from graph G by removing edges from each cycle until no cycles remain.
3. For each $b \in B$, if there exist $b', b'' \in B$ where $b \neq b' \neq = b'' \neq b$ such that b is adjacent to both b' and b'' along the same plane in S, and there are edges in T (1) between the nodes representing b and b' and (2) the nodes representing b and b'', then split b into two new rectangular prisms, b_1 and b_2, such that each is adjacent to exactly one of b' and b'' (this is always possible since all of b, b', and b'' are rectangular prisms).
4. Remove b from B and add b_1 and b_2 to B.
5. If any block was split in step 3, loop back to step 1. The tree T is a graph whose edges connect the nodes representing blocks which must bind to each other in the final assembly. At this point, each $b \in B$ will have at most 1 adjacent $b' \in B$ on each side to which it must bind, and each $b \in B$ will have at least one other $b' \in B$ to which it must bind. We will refer to any pair of blocks which must bind to each other as *connected*.

5.2 Scale factor and interface design

The blocks self-assemble individually, then separate from the genome to freely diffuse until they combine

together via interfaces along the surfaces between which there were edges in the binding tree T. Each interface is assigned a unique length and number. The two blocks that join along a given interface are assigned complementary patterns of "bumps" and "dents" and a pair of complementary glues on either side of those patterns (to provide the necessary binding strength between the blocks).

We now describe the size and composition of the interface between connected blocks. Each interface will include two specially designated glues, one on each end of the interface, and assuming the length of the interface is n, an $n-2$ tile wide portion in between those glues which will eventually be mapped to a particular "geometry" of bumps and dents (i.e. tiles protruding from a surface, and openings for tiles in a surface). No interface can be shorter than 2. Also, since each interface must be unique, there is only one valid interface of length 2, and for each $n > 2$ there will be $2^{(n-2)/2}$ valid interfaces because each bit of the assigned number is represented by two bits in the geometry. For a 0-bit, the pattern 01 is used, and for a 1-bit the pattern 10 is used. This ensures that each geometry is compatible only with its complementary geometry (see Fu et al. 2012 for further examples.) Fig. 21 shows an example of interfaces which could be added to the blocks of the example shape from Fig. 20. Note, however, that for the sake of a more interesting example larger interfaces are shown than would be assigned by the algorithm presented, which would have created one interface of size 2, with only White and Black glues, and two of size 4, one with a "dent" then "bump" to represent 01 which maps to 0, and one with a "bump" then "dent" to represent 10 which maps to 1.

1. Define the function RECT such that, for each connected pair $b, b' \in B$, RECT(b, b') returns the rectangle along which b and b' are adjacent in S, and the function RECTMAX$(b, b') = \max(m, n)$ where m and n are the lengths of the sides of the rectangle returned by RECT(b, b') (i.e. it returns the length of the maximum dimension of the rectangle).

2. Initialize the mapping INTERFACE − LENGTH which maps a connected pair b and b' to an integer such that INTERFACE − LENGTH$(b, b') = 2$. (INTERFACE-LENGTH will eventually specify the length of the interface between blocks.)

3. Define the function COUNT such that, for each $k > 1$, COUNT(k) is equal to the number of connected pairs $b, b' \in B$ such that INTERFACE − LENGTH$(b, b') = k$. (That is, COUNT returns the number of pairs of blocks that are currently assigned interfaces of length k.)

4. While there exists $k > 1$ such that COUNT$(k) > 2^{(k-2)/2}$:

    (a) Select a connected pair b, b' where INTERFACE − LENGTH$(b, b') = k$ and update the mapping INTERFACE − LENGTH so that INTERFACE − LENGTH$(b, b') = k + 1$.

5. If there exists a connected pair $b, b' \in B$ such that INTERFACE − LENGTH$(b, b') >$ RECTMAX(b, b'), this (simplified) construction requires the shape S to be scaled because there are too many interfaces of one or more lengths for them all to be unique[7]. Therefore, replace S with S^2 (the scaling of S by 2) and restart the construction from shape decomposition, at the beginning of Sect. 5.1.

At this point, the mapping INTERFACE − LENGTH defines a valid mapping of lengths to each interface. We now assign a valid geometric pattern (i.e. a series of "bumps" and "dents") to each.

1. Let s equal the value of the maximum of the width, height, and depth of S (i.e. the length of its greatest dimension).

2. For each integer $1 < i \leq s$, let $I_i = \{(b, b') \mid$ where $b, b' \in B$ are connected and INTERFACE − LENGTH$(b, b') = i\}$. Thus, I_i is the set of connected pairs of blocks which have interfaces of length i.

3. For each I_i where $|I_i| > 0$, assign an arbitrary, fixed ordering to I_i and for $0 < |I_i| < j$, let I_{i_j} be the jth connected pair in I_i.

4. For each I_{i_j}:

    (a) Recall that i is the assigned interface length.

    (b) Assign j as the number assigned to the interface (after the number of bits is doubled so that each 0-bit is represented by 01 and each 1-bit by 10).

    (c) Let $(b, b') = I_{i_j}$ and $r =$ RECT(b, b')

    (d) As r is a rectangle, it is 2-dimensional and has only two of width (x dimension), height (y dimension), and depth (z dimension). If its width is $\geq i$, we call r an East–West (EW) rectangle. Else, if its height is $\geq i$, we call r a North–South (NS) rectangle. Otherwise, its depth must be $\geq i$ (by design of the algorithm determining the assigned value of i, it will fit in at least one dimension of r) and we call r an Up-Down (UD) rectangle.

---

[7] The number of unique interfaces for any length can easily be increased using methods discussed later.
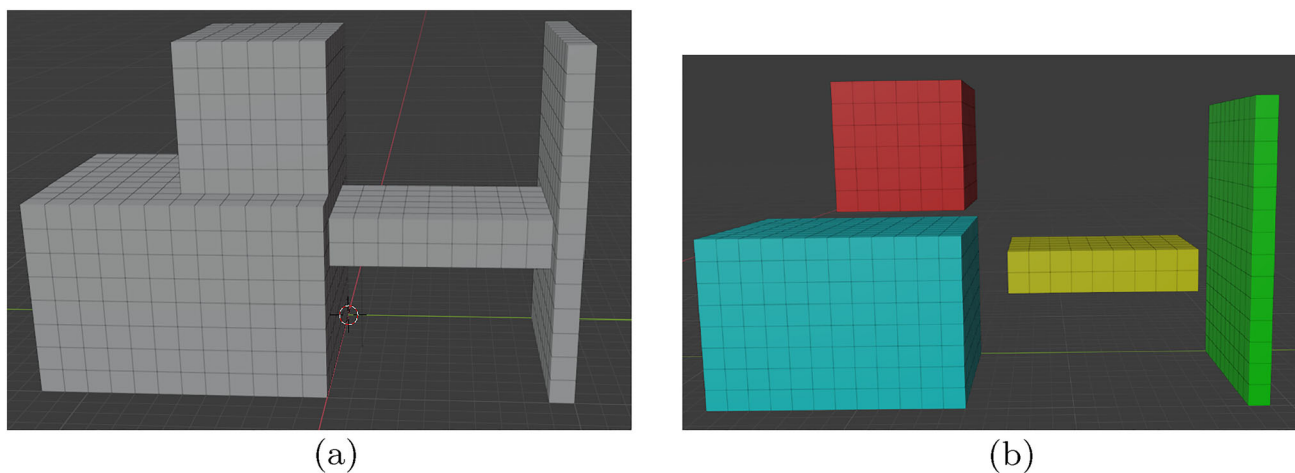
Fig. 20 (a) An example 3D shape *S*. (b) *S* split into 4 `blocks`, each of which can be grown from its own `gene`. Note that the surfaces which will be adjacent when the `blocks` combine will also be assigned `interfaces` to ensure correct assembly of *S*
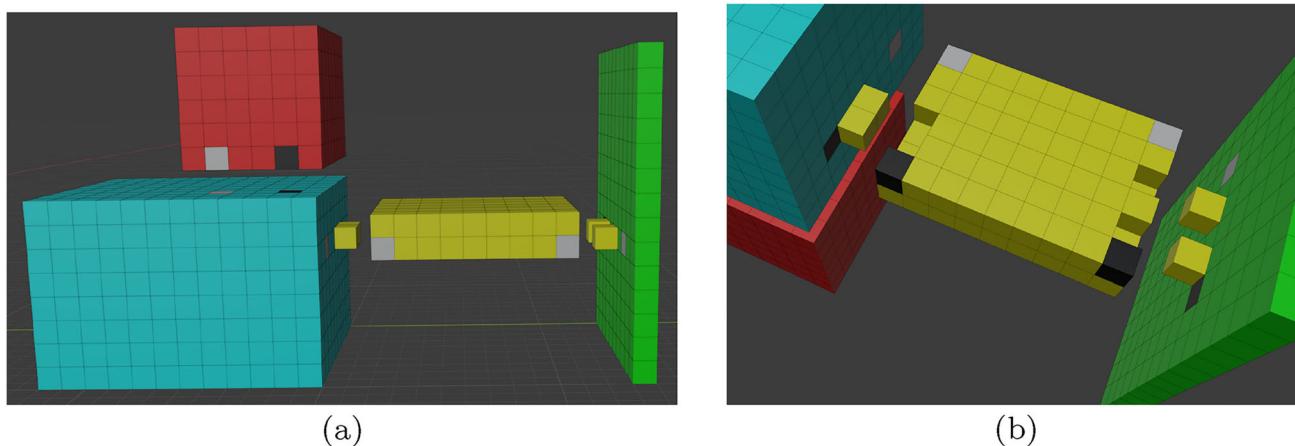

Fig. 21 (a) The `blocks` for the example shape *S* from Fig. 20 with example `interfaces` included. (b) View from underneath showing more of the `interfaces` between `blocks`. Note that the actual `interfaces` created by the algorithm would be shorter, but to make the example more interesting their sizes have been increased

(e) Define $\text{RECT} - \text{ROW}$ as a function such that on input $b, b' \in B$, $\text{RECT} - \text{ROW}(b, b')$ returns a single row of coordinates as follows. Rectangle r is either EW, NS, or UD and has one other non-zero dimension (x, y, or z) other than the dimension its type is named for. If that other non-zero dimension is x (resp. y, resp. z), set direction $d = E$ (resp. N, resp. U). If $\text{RECT}(b, b')$ returns EW (resp. NS, resp. UD) rectangle r, $\text{RECT} - \text{ROW}(b, b')$ returns the row furthest in direction d which runs EW (resp. NS, resp UD) in r.

(f) Let $r' = \text{RECT} - \text{ROW}(b, b')$. If r' is an EW (resp. NS, resp. UD) rectangle, we define the `inter-face` for I_{i_j} such that the easternmost (resp. northernmost, resp. uppermost) location in r' is assigned the Black glue, the adjacent $i - 2$ locations are assigned the $i - 2$ bits of the binary representation of the number j, in order, with the least significant bit in the easternmost (resp. northernmost, resp. uppermost) location, and the next location is assigned the White glue, making it the westernmost (resp. southernmost, resp. downwardmost) location containing a non-zero amount the `interface` information. The other locations of the row of r' are assigned "empty" values. Define the function $\text{INTERFACE}(b, b')$ such that it returns this `interface` definition for the entire row of r' for the `interface` between b and b'. (Recall that by our construction, any connected pair can have at most one `interface`.)
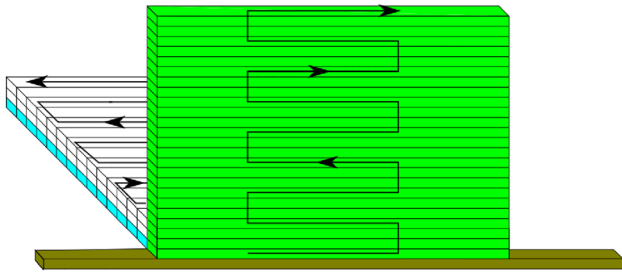
Fig. 22 Schematic representation of the order of block growth (without directions shown for every row). Starting from a gene section, the green surface grows upward in a zig-zag pattern. As each row of the green face completes, one plane can grow perpendicularly to it (the first is shown in blue, with the next two in white). Each of these also grows in a zig-zag pattern away from the green face

5.3 Growth of a block

Each block $b \in B$ making up shape S has at most 6 interfaces. Because of this constant bound, and the fact that each block is a rectangular prism, it is possible to encode all of the information needed to grow an entire block b within a sequence of glues, taken from a set of glues that is constant over any shape S, that is no longer than the longest dimension of b.[8] We call each such sequence a gene. In this section we show how a gene can be encoded and initiate growth of a block.

Each block grows so that one of its 6 faces grows directly upward off of the block's gene. The growth of this plane happens in a zig-zag manner, meaning that the first row grows completely from left to right (zigging), then the second from right to left (zagging), and the pattern continues until the growth terminates. (Shown schematically in green in Fig. 22.) The zig-zag pattern of growth allows for each row to transmit (and update) information it reads from the row below it (to be discussed shortly).

As each row of the first face completes, a plane growing perpendicular to the first face can begin its growth. (The first such plane is shown in light blue in Fig. 22, and the next two in white.) Every row of each such plane also grows in a zig-zag manner, which allows information to be transmitted from the green initiating rows throughout each plane.

To control the size of each plane, a pair of binary counters are used. The upward facing glues of the gene encode a series of bits (which we will call the *green* bits). As the face grows upward, every other row increments the value of the binary number represented by the bits, and every other row checks to see if all bits are equal to 1. If they are, upward growth terminates. (An example can be seen in Figure 21 of the full version Alseth et al. 2021.)

We will call the bits of the counter which control the length of the perpendicular planes (shown as blue and white in Fig. 22) the *blue* bits. These bits are also encoded in the upward facing glues of the gene (i.e. each glue can encode both a green and a blue bit by making 4 glues, one for each pair of bit values 00, 01, 10, and 11). However, as each row of the green face assembles, rather than using the blue bits to count, each row presents the blue bits on both its upward and backward facing glues. This allows them to be propagated up throughout the green face, unchanged, and to control the distance grown by each perpendicular plane, which uses them as the bits for its counter.

With the gene's length implicitly encoding the size of one dimension of the growing block, and the green and blue counter bits controlling the sizes of the other two dimensions, the block grows into a rectangular prism of the correct dimensions. (Note that growing counters, zig-zag growth, rotating bits, etc. are very standard techniques in tile assembly literature - see (Doty et al. 2012; Demaine et al. 2016; Cannon et al. 2013; Soloveichik and Winfree 2007; Rothemund and Winfree 2000) for just some examples - and issues like growing sides of odd length, despite the zig-zag pattern, are easily handled with a few extra glues that signal for one additional row to grow.)

Each block has a fixed orientation relative to the others when they are attached together to form the shape S, and since we (arbitrarily) assign each shape a canonical translation and rotation, each block has a canonical orientation which allows us to refer to its sides by the directions they face in that orientation. Throughout, we talk about blocks in term of this orientation, irrespective of that in which they grow.

This (simplified version of the) construction has each gene equal to the length of the longest dimension of the block it initiates. This could lead to the first surface to grow being any of at least 4 sides, so without lack of generality we fix a preferred ordering as: North, East, South, West, Up, Down. Therefore, of the multiple faces which share the longest dimension, that appearing first in the ordering grows "first" (i.e. as the green face, as shown in Fig. 22), and with the side attached to the gene being that whose coordinates are the smallest along the direction of upward growth of the first face.

5.4 interface growth

With the dimensions of each block correctly controlled, the next thing to ensure is correct growth of the block's interfaces. As previously mentioned, there are at most 6 of these (no more than one per side), and each interface consists of two outward facing glues (Black and White) with a possible series of "bumps" and "dents" between them, geometrically encoding the bits of the
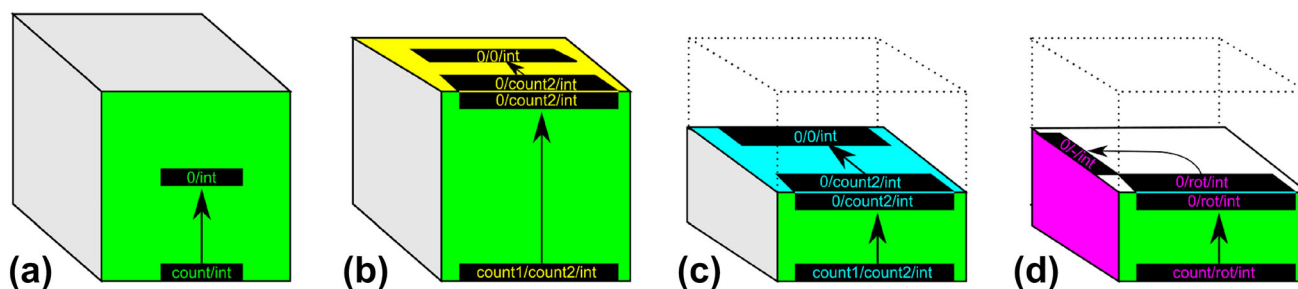
---

[8] Later we will also briefly mention ways in which the length can actually be as small as the log of the longest dimension.

Fig. 23 Schematic representation of the patterns by which `interface` information is propagated into the correct positions for "parallel" interfaces. (a) A counter is used to determine the correct height for the `interface` on the green side, (b) Two counters are used to position the `interface` on the yellow side. The first counts to the top of the green side, then the bits of the second and the `interface` are rotated onto the yellow colored plane and the second counts to the proper location for the `interface` on that side. (c) Two counters are used to position the `interface` on the back side of the `block`. The first counts to the correct height, then the bits of the second and the `interface` are rotated onto the blue colored plane and the second counter counts the distance to the back surface. (d) To position the `interface` on the pink side, a counter first counts to the correct height, then the bits are rotated to the pink face during the outward growth of the white plane. Note that the side opposite the pink `interface` is positioned analogously but with an opposite rotation, and the bottom `interface` is positioned similar to the top (yellow) but without the necessity of the first counter

number which is uniquely assigned to that `interface`. If the `interface` is on the North, East, or Up side, in the location of each bit $b = 1$ there is a tile which extends from the side as a "bump", and in the location of each bit $b = 0$, there is no such bump. If the `interface` is on the South, West, or Down side, in the location of each bit $b = 1$ there is an empty tile location (i.e. a "dent"), and in the location of each bit $b = 0$, there is no such dent. (See Fig. 21 for examples of `interfaces` with "bumps" and "dents".)

The information defining each `interface` can be encoded as a series of glues representing the locations of the Black and White `interface` glues plus each of the bits of the assigned `interface` number, as well as the information about whether the 1-bits are encoded as "bumps" or "dents" for the particular surface. Using the same technique as mentioned previously for adding information about an extra bit to the glues extending from the `gene`, we can similarly add the information which defines each of the (up to 6) `interfaces` of a `block`. Therefore, we individually discuss the patterns by which the information specifying each `interface` is propagated into the correct locations, and note that all of that information can be encoded in the outward facing glues of the `gene` and then distributed to the proper locations in the `block` during the growth process previously described. After explaining how the information about each `interface` arrives at the correct location, we discuss the tiles encoding it.

There are 6 sides, and for each side 2 orientations which must be considered for the possible `interface` on that side (note that on `block` sides which don't have `interfaces`, nothing needs to be done beyond the growth of the side to the correct dimensions as previously described). One orientation we will refer to as "parallel" to the `gene`, and the other as "perpendicular" (although these terms

aren't technically accurate for all cases). The parallel cases are depicted in Fig. 23, and the perpendicular cases are depicted in Fig. 24.

It is important to note that the patterns shown in Figs. 23 and 24 suffice when each `interface` is anywhere from the minimum allowed size (i.e. 2) up to the maximum size, which is the full length of the side on which it is located. This is because the construction is designed so that the length of the `gene`, and thus the green side, is the length of the longest dimension of the `block`. Thus, there is room for the information in a longest-possible `interface` to be correctly positioned, and shorter `interfaces` can also be correctly positioned by correctly shifting the locations of information in the `gene` so that the counters and rotations will propagate it correctly. Additionally, Figs. 23 and 24 depict the cases where each `interface` is in the center of its surface, but any position along each surface can be accommodated by simply adjusting initial information alignment along the `gene`, counter values, and/or the location of splits between rotations and counting.

Recall that the `blocks` on either side of an `interface` have complementary geometries, i.e. one has "bumps" in the 1-bit locations and the other has "dents". Once the information encoding an `interface` reaches the correct location on the correct surface, the locations assigned the Black and White glues of the `interface` receive tiles which have strength-1 glues of those types exposed on the exterior of the `block` for the `block` with a bump `interface`, and the `block` with the dent `interface` receives tiles which expose the complements of those glues (i.e. Black* and White*, respectively). Additionally, in 1-bit positions for a `block` with a bump `interface`, tiles attach which have strength-2 glues exposed, allowing the "bump" tiles to attach, and signals ensure that all "bump" tiles have attached before the Black
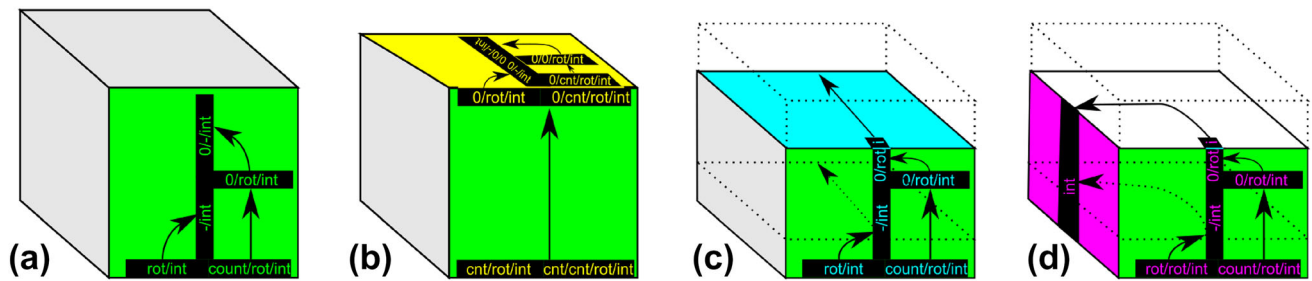
Fig. 24 Schematic representation of the patterns by which `interface` information is propagated into the correct positions for "perpendicular" `interfaces`. (a) For a perpendicular `interface` on the green side, the information is split into two halves (depending on the actual length and position of the `interface`). The left half is rotated upward immediately, and the second half has a counter which first moves it upward to the halfway point, and then it is rotated. Note that any offset from the center can be accommodated by shifting the location of the split and the height of the counter. If the `interface` needs to be completely to the left or right, only one rotation is needed, and no splitting of the information or counting is needed. (b) The positioning of the `interface` on the top is the same as for the green side, but a counter first propagates all information to the top, where it is rotated to the yellow surface. (And the same holds for the bottom surface but without the initial counter.) (c) To position the `interface` on the back surface, the same rotations and counting are used as for the green surface. However, then the information from each row is carried all the way to the back surface following the counter which dictates that distance. (d) To position the `interface` for the pink surface, again the same rotations and counting are used to align the information on the green surface, but then the information of each row is rotated to the pink surface as its plane grows away from the green surface. The surface opposite the pink is handled similarly, but with an opposite rotation

tile can attach and enable the `interface` to bind to its counterpart. The designs of the tile types and signals necessary to grow these `interfaces`, and also to allow for the detachment of `blocks` from the `genome`, are relatively straightforward and omitted from this version of the paper due to space constraints. However, more details (including tile type and signal definitions can be found in Section 5.4.1 of the online version Alseth et al. 2021).

5.5 Combination of `blocks` to form the target shape

Once a `block` has detached from its `gene`, it is a freely floating supertile which may or may not require additional tile attachments to complete its own growth. However, only `interfaces` that have completed are able to bind with strength 2 to the complementary `interfaces` of other `blocks`. Additionally, we now discuss a set of signals that allow for a `block` to determine when all tiles have attached. The growth of each plane in a `block` follows the same zig-zag pattern so that the final tile placed in each plane (other than possibly "bump" tiles of `interfaces`) falls into a single vertical column. These tiles are augmented with signals such that when the final tile of the bottommost plane attaches, it activates a glue that allows it to bind to the tile above it (whose complementary glue will be activated when it attaches). The tile above it in turn passes this signal upward, with each in the column doing the same until the final tile of the top plane is reached. Once that tile (which is of a special type) is placed, it is guaranteed that all tiles of all planes (other than possibly "bump" tiles of `interfaces`) have attached since each plane signals its completion in order from bottom to top.

Upon receiving the "completion" signal, the final tile of the top plane then sends that signal outward, spreading across all tiles on all 6 surfaces of the `block`. These "surface" tiles are all equipped with signals that allow them to receive and pass on this completion signal (and during the growth of the `block` it is always known which tiles will be on a surface since they are at an edge of their plane of growth). The previous description of the signals which activate the Black and White glues (and their complements) on `interfaces` was slightly simplified to omit this final detail: the previously described signals which activate those glues actually activate glues facing neighboring tiles so that only at that point they are able to receive the completion signal. It is the reception of this signal which actually activates the Black and White (and Black* and White*) glues on the `interfaces`.

The addition of the extra layer of "completion" signals ensures that only a `block` that has received all of the tiles of its body can have active `interfaces`. Once an `interface` is active and able to bind to the complementary `interface` of another `block`, the `block` combines to a growing supertile consisting of the `blocks` forming an assembly of shape S. Furthermore, by the definition of a block-diffusable shape and the fact that S is such a shape, it is always possible for a free `block` to attach as needed in any such growing supertile. Thus, the `blocks` will eventually form completed, and terminal, assemblies of shape S.

5.6 Overview of the hierarchical construction

We have described how we can begin with an arbitrary block-diffusable 3D shape S, decompose it into rectangular prisms called `blocks` with complementary `interfaces`

between them, encode the information needed to make each block into a `gene` subassembly of a `genome` seed assembly, and how the `blocks` can independently grow, detach from the `genome`, and attach to each other to form an assembly of the target shape S (or a scaled version if needed). By the design of the `interfaces`, the `blocks` can only combine in the correct manner. Once a `block` is freely diffusing and complete, it can combine along its `interfaces` with the `blocks` that have complementary `interfaces` since, due to the fact that S is a block-diffusable shape, free `blocks` can always diffuse into the proper locations to form the complete shape. We've described a tile set U that can be used to (1) form the linear seed assembly σ_S, and (2) to self-assemble the `blocks` which correctly combine to form the target assembly. The STAM* system $\mathcal{T}_S = (U, \sigma_S, 2)$ will produce an infinite number of copies of terminal assemblies of shape S (properly scaled if necessary). The only fuel (a.k.a. consumed, junk assemblies) will be singleton Dent tiles that attached during `block` growth then detached. Note that this construction can be combined with the previous constructions as well, to create a version of a shape self-replicator.

5.7 Enhancements to the hierarchical construction

There are many ways in which this construction could be easily modified to further optimize tile complexity and other parameters. For example, to shrink the length of the `genome`, `genes` could be compressed so that they are no longer required to be as long as the largest dimension of a `block`. Instead, in cases where `interfaces` are shorter than `block` side lengths and appropriately positioned, it is possible to shrink the `gene` encoding a block to as small as log-width. This can be done by incorporating counters that also grow out the width of a `block`. Additional, even asymptotically optimal, compression could be achieved by instead encoding the shortest program that outputs the `gene` necessary to grow a `block` and then a "fuel efficient" Turing machine (Padilla et al. 2014) can be simulated with signal tiles which grow from the `genome` until that encoding is output, allowing `block` growth to proceed from there. Note that this option could greatly increase the the fuel consumed.

As another example, the necessity to scale certain shapes could be removed by only slightly increasing tile complexity, i.e. the size of U. For example, by adding a constant number m of tile types to also be candidates for the ends of `interfaces` (along with the White and Black tiles), the number of `interfaces` of each length (which is the limiting number potentially requiring scaling of a shape) can be increased by a factor on the order of m^2.

There are many other such variations that can be used to balance several factors of the construction to optimize trade-offs for desired goals. Also, for many variations on the specific algorithm which is used to determine the encoding of S into the `genome`, no changes are even required to U, so the algorithm can be modified to favor particular tradeoffs over others (e.g. scale factor over `genome` length) without any other modifications to the system.

Finally, it is easy to combine this construction with the previous constructions. For instance, tile types could be added to U from the construction in Sect. 3 that also create duplicate copies of σ_S. Additionally, an actual self-replicating system could be built by including the shape-deconstruction capabilities of the construction in Sect. 4. Let M be a Turing machine that performs the following computation. Given an input string consisting of the turns of a path through $\mathbb{Z}^3$ (i.e. the path encoded in a seed assembly genome of the construction in Sect. 3), it first computes the points of the shape S generated by that path. It then performs the computations for the hierarchical replicator of this section to compute a valid input `genome` for it. Simulation of an arbitrary Turing machine is straightforward even with static aTAM tiles (e.g. Patitz and Summers 2011; Lathrop et al. 2011; Soloveichik and Winfree 2007) and can additionally be made "fuel efficient" using signal tiles (Padilla et al. 2014). Therefore, there exists a system which can take as input an assembly as for the construction of Sect. 4 and use the components of that construction to deconstruct it into a linear genome. Tiles which simulate M then perform the generation of the input `genome` for the hierarchical replicator, which proceeds to make copies of assemblies of shape S. This is a more complicated self-replicator which consumes much more fuel (i.e. the TM computation tiles - but note that using techniques of Padilla et al. (2014) that amount is greatly reduced, and the junk assemblies can all be guaranteed to be of small, constant size) but after the `genome` is computed once it is infinitely replicated along with copies of the shape.

6 The requirement for deconstruction

Definition 4 Given a tile set T, a *porous assembly* α, over tiles in T, is one in which it is possible for unbound tiles of one or more types in T to pass freely through either (1) the body of one or more tiles in α, or (2) the gaps between tiles in α (which means between bound glues if the tiles are bound to each other), or (3) a combination of both. Conversely, a *non-porous assembly* is one in which no unbound tiles can pass through any of the tile bodies or gaps between tiles.

For theoretical results, we tend to consider all tile bodies to be solid, or at least solid enough to prevent the diffusion of other tiles through them. Whether or not an assembly is porous then depends upon factors such as the spacing between tiles, lengths of glues, and spacing of glues. For instance, the seed assemblies for the construction in Sect. 4 are non-porous assuming glues are spread evenly along the edges of tiles.

In this section we prove that in the STAM* there cannot be a universal shape self-replicator in systems with non-porous assemblies that does not use (an arbitrary amount of) deconstruction.

Theorem 4 *Let U be an STAM* tile set such that for an arbitrary 3D shape S, the STAM* system $\mathcal{T} = (U, \sigma_S, \tau)$ with* dom $\sigma = S$, $\mathcal{T}$ *is a shape self-replicator for S and σ is non-porous. Then, for any $r \in \mathbb{N}$, there exists a shape S such that $\mathcal{T}$ must remove at least r tiles from the seed assembly σ_S.*

Proof We prove Theorem 4 by contradiction. Therefore, assume that U is a tile set in the STAM* capable of shape replicating any shape S and that seed assembly σ_S is non-porous. Let $t = |U|$, g be the maximum number of glues on any tile type in U, and s be the maximum number of signals on any tile type in U. Note that for any position in an assembly over tiles in U, there is a maximum number of $\lambda = t(3^g)(3^s)$ possible tile types and tile states (accounting for all possible states of glues and signals).

We define a shape c which is an $n \times n \times n$ cube, for some $n \in \mathbb{N}$ to be defined, with every point on the exterior of the cube included in the shape. For every xy plane (i.e. horizontal plane) in the interior of the cube, the points contained within c follow the pattern shown in Fig. 25, where the grey locations are all included and a subset of the green locations are included. Note that only one plane has a connection to the exterior, and no other tiles of any plane in the interior are adjacent to a location of the exterior. Define the set C as the set of all such c where there is one for each possible pattern of green locations included and excluded.

To ensure that only a single location of a single xy plane in the interior of the cube is adjacent to the exterior (leaving a gap all around) the number of xy planes with occupied locations is $n - 4$. The width of each green row is $n - 5$. The number of green rows in each xy plane is $(n - 4)/2$. Therefore, the number of green interior positions is $(n - 4)(n - 5)(n - 4)/2$. The number of shapes which include every possible subset of those green positions is $2^{(n-4)(n-5)(n-4)/2}$, and this is the size of the set C. Conversely, the number of unit cube locations on the exterior of each $n \times n \times n$ cube is $6(n-1)^2$.

By our assumption, for every $c \in C$, there exists an STAM* system $\mathcal{T}_c = (U, \sigma_c, \tau)$ such that $\mathcal{T}_c$ shape self-
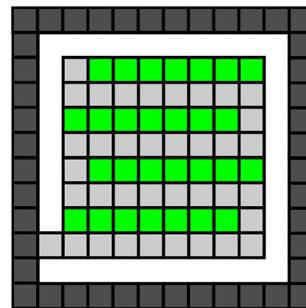

Fig. 25 An example interior xy plane within the cube c of the proof of Theorem 4. The plane in this example has the single connection to the exterior of the cube (dark grey), and all light grey locations are included, along with a subset of the green locations

replicates c. However, for each such σ_c, the total number of options for a tile in each exterior location (including states) is λ, and therefore the total number of unique subassemblies composing the exterior surfaces of the cube is $\lambda^{6(n-1)^2}$. Also, since s is the maximum number of signals on any tile type in T, $s!$ represents every possible ordering of completion of signals on the tile with the most signals. We can choose a value of n (for the side lengths of the cubes) such that $(s!)\lambda^{6(n-1)^2+1} < 2^{(n-4)(n-5)(n-4)/2}$, since the exponents of the left and right sides grow on the order of n^2 and n^3, respectively, and all other terms are constants with respect to n. Let n be such a sufficiently large value and then note that by the pigeonhole principle, for two $c_1, c_2 \in C$, the systems $\mathcal{T}_{c_1}$ and $\mathcal{T}_{c_2}$ must have identical subassemblies composing the exteriors of their seed assemblies as well as the single tile attaching each exterior to the interior planes. Additionally, there is an assembly sequence where the single tile of each exterior subassembly connected to the interior planes must experience the same ordering of completion of signals (since anything that could happen on their exteriors must be the same for both, and there were enough assemblies with the same subassemblies to guarantee the same order of completion of their signals for at least two of them). Since σ_{c_1} and σ_{c_2} are non-porous, there can be no other factors in $\mathcal{T}_{c_1}$ and $\mathcal{T}_{c_2}$ which influence the growth of assemblies, and so both systems must be able to yield the same terminal assemblies. This contradicts that they shape self-replicate c_1 and c_2 since these are different shapes. Finally, in order to achieve the arbitrary bound r for required tile removals, we can simply adapt our target shape to be a "chain" of r cubes (all of which can be made to be unique) connected by a single-tile-wide path of tiles and otherwise completely separated. The previous argument holds for each of the r cubes, and since none can be replicated without the removal of at least one tile, a lower bound of the removal of at least r tiles is established. $\square$

Declarations

References

Abel Z, Benbernou N, Damian M, Demaine ED, Demaine ML, Flatland R, Kominers SD, Schweller RT (2010) Shape replication through self-assembly and RNAse enzymes. In: SODA 2010: Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1045–1064. Society for Industrial and Applied Mathematics, Austin, Texas

Alseth A, Hader D, Patitz MJ (2021) Self-Replication via Tile Self-Assembly (Extended Abstract). In: Lakin MR, Šulc P (eds.) 27th International Conference on DNA Computing and Molecular Programming (DNA 27), Leibniz International Proceedings in Informatics (LIPIcs), vol. 205, pp. 3:1–3:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany. https://doi. org/10.4230/LIPIcs.DNA.27.3. https://drops.dagstuhl.de/opus/ volltexte/2021/14670

Alseth A, Hader D, Patitz MJ (2021) Self-replication via tile self-assembly (extended abstract). Tech. Rep. 2105.02914, Computing Research Repository. arxiv:2105.02914

Andersen ES, Dong M, Nielsen MM, Jahn K, Subramani R, Mamdouh W, Golas MM, Sander B, Stark H, Oliveira CLP, Pedersen JS, Birkedal V, Besenbacher F, Gothelf KV, Kjems J (2009) Self-assembly of a nanoscale DNA box with a controllable lid. Nature 459(7243):73–76. https://doi.org/10.1038/ nature07971

Barish RD, Schulman R, Rothemund PWK, Winfree E (2009) An information-bearing seed for nucleating algorithmic self-assembly. Proceedings of the National Academy of Sciences 106(15):6054–6059. https://doi.org/10.1073/pnas.0808736106

Becker F, Rapaport I, Rémila E (2006) Self-assembling classes of shapes with a minimum number of tiles, and in optimal time. In: Foundations of Software Technology and Theoretical Computer Science (FSTTCS), pp 45–56

Becker F, Rémila E, Schabanel N (2008)Time optimal self-assembly for 2d and 3d shapes: The case of squares and cubes. In: Goel A, Simmel FC, Sosík P (eds.) DNA, Lecture Notes in Computer Science, 5347: 144–155. Springer

Bui H, Shah S, Mokhtar R, Song T, Garg S, Reif J (2018) Localized DNA hybridization chain reactions on DNA origami. ACS Nano 12(2):1146–1155

Cannon S, Demaine ED, Demaine ML, Eisenstat S, Patitz MJ, Schweller RT, Summers SM, Winslow A (2013) Two hands are better than one (up to constant factors): Self-assembly in the 2HAM vs. aTAM. In: Portier N, Wilke T (eds.) STACS, LIPIcs,

20:172–184. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik

Cheng Q, Aggarwal G, Goldwasser MH, Kao MY, Schweller RT, de Espanés PM (2005) Complexities for generalized models of self-assembly. SIAM J Comput 34:1493–1515

Cheung KC, Demaine ED, Bachrach JR, Griffith S (2011) Programmable assembly with universally foldable strings (moteins). IEEE Trans Robot 27(4):718–729

Cook M, Fu Y, Schweller RT (2011) Temperature 1 self-assembly: Deterministic assembly in 3D and probabilistic assembly in 2D. In: SODA 2011: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM

Demaine ED, Demaine ML, Fekete SP, Ishaque M, Rafalin E, Schweller RT, Souvaine DL (2008) Staged self-assembly: nanomanufacture of arbitrary shapes with $O(1)$ glues. Nat Comput 7(3):347–370

Demaine ED, Demaine ML, Fekete SP, Patitz MJ, Schweller RT, Winslow A, Woods D (2014) One tile to rule them all: Simulating any tile assembly system with a single universal tile. In: Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP 2014), IT University of Copenhagen, Denmark, July 8-11, 2014, LNCS, 8572:368–379

Demaine ED, Patitz MJ, Rogers TA, Schweller RT, Summers SM, Woods D (2013) The two-handed assembly model is not intrinsically universal. In: 40th International Colloquium on Automata, Languages and Programming, ICALP 2013, Riga, Latvia, July 8-12, 2013, Lecture Notes in Computer Science. Springer

Demaine ED, Patitz MJ, Rogers TA, Schweller RT, Summers SM, Woods D (2016) The two-handed tile assembly model is not intrinsically universal. Algorithmica 74(2):812–850. https://doi. org/10.1007/s00453-015-9976-y

Doty D (2009) Randomized self-assembly for exact shapes. In: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp 85–94. IEEE

Doty D, Kari L, Masson B (2013) Negative interactions in irreversible self-assembly. Algorithmica 66(1):153–172

Doty D, Lutz JH, Patitz MJ, Schweller RT, Summers SM, Woods D (2012) The tile assembly model is intrinsically universal. In: Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, pp. 302–310

Durand-Lose J, Hendricks J, Patitz MJ, Perkins I, Sharp M (2018) Self-assembly of 3-D structures using 2-D folding tiles. In: D. Doty, H. Dietz (eds.) DNA Computing and Molecular Programming - 24th International Conference, DNA 24, Jinan, China, October 8-12, 2018, Proceedings, Lecture Notes in Computer Science, 11145:105–121. Springer

Fekete SP, Hendricks J, Patitz MJ, Rogers TA, Schweller RT (2015) Universal computation with arbitrary polyomino tiles in non-cooperative self-assembly. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015), San Diego, CA, USA January 4-6, 2015:148–167 . https://doi.org/10.1137/1.9781611973730.12. http://epubs.siam.org/doi/abs/10.1137/1.9781611973730.12

Fochtman T, Hendricks J, Padilla JE, Patitz MJ, Rogers TA (2015) Signal transmission across tile assemblies: 3D static tiles simulate active self-assembly by 2D signal-passing tiles. Nat Comput 14(2):251–264

Fu B, Patitz MJ, Schweller RT, Sheline R (2012) Self-assembly with geometric tiles. In: Czumaj A, Mehlhorn K, Pitts AM, Wattenhofer R (eds.) Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I, LNCS, 7391:714–725. Springer

Furcy D, Micka S, Summers SM (2015) Optimal program-size complexity for self-assembly at temperature 1 in 3D. In: DNA

Computing and Molecular Programming - 21st International Conference, DNA 21, Boston and Cambridge, MA, USA, August 17-21, 2015. Proceedings, pp 71–86. https://doi.org/10.1007/978-3-319-21999-8_5

Gilbert O, Hendricks J, Patitz MJ, Rogers TA (2016) Computing in continuous space with self-assembling polygonal tiles. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016), Arlington, VA, USA January 10-12, 2016, pp 937–956

Hader D, Koch A, Patitz MJ, Sharp M (2020) The impacts of dimensionality, diffusion, and directedness on intrinsic universality in the abstract tile assembly model. In: Chawla S (ed.) Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020, pp 2607–2624. SIAM

Hader D, Patitz MJ (2019) Geometric tiles and powers and limitations of geometric hindrance in self-assembly. In: Proceedings of the 18th Annual Conference on Unconventional Computation and Natural Computation (UCNC 2019), Tokyo, Japan June 3-7, 2019, pp 191–204

Hendricks J, Olsen M, Patitz MJ, Rogers TA, Thomas H (2019) Hierarchical self-assembly of fractals with signal-passing tiles. Submit Nat Comput

Hendricks J, Patitz MJ, Rogers TA (2015) Replication of arbitrary hole-free shapes via self-assembly with signal-passing tiles. In: Calude CS, Dinneen MJ (eds.) Unconventional Computation and Natural Computation - 14th International Conference, UCNC 2015, Auckland, New Zealand, August 30 - September 3, 2015, Proceedings, Lecture Notes in Computer Science, 9252:202–214. Springer. https://doi.org/10.1007/978-3-319-21819-9_15

Hendricks J, Patitz MJ, Rogers TA (2017) Reflections on tiles (in self-assembly). Nat Comput 16(2):295–316. https://doi.org/10.1007/s11047-017-9617-2

Hendricks J, Patitz MJ, Rogers TA (2017) The simulation powers and limitations of higher temperature hierarchical self-assembly systems. Fundam Inform 155(1–2):131–162. https://doi.org/10.3233/FI-2017-1579

Jonoska N, Karpenko D (2014) Active tile self-assembly, Part 1: universality at temperature 1. Int J Foundat Comput Sci 25(02):141–163. https://doi.org/10.1142/S0129054114500087

Jonoska N, Karpenko D (2014) Active tile self-assembly, Part 2: self-similar structures and structural recursion. Int J Foundat Comput Sci 25(02):165–194. https://doi.org/10.1142/S0129054114500099

Jonoska N, McColm G (2006) Flexible versus rigid tile assembly. In: Calude C, Dinneen M, Păun G, Rozenberg G, Stepney S (eds.) Unconventional Computation, Lecture Notes in Computer Science, 4135:139–151. Springer Berlin Heidelberg. https://doi.org/10.1007/11839132_12

Jonoska N, McColm GL (2005) A computational model for self-assembling flexible tiles. In: Proceedings of the 4th international conference on Unconventional Computation, UC'05, pp 142–156. Springer-Verlag, Berlin, Heidelberg. https://doi.org/10.1007/11560319_14

Jonoska N, McColm GL (2009) Complexity classes for self-assembling flexible tiles. Theoret Comput Sci 410(4–5):332–346. https://doi.org/10.1016/j.tcs.2008.09.054

Kari L, Seki S, Xu Z (2012) Triangular and hexagonal tile self-assembly systems. In: Proceedings of the 2012 international conference on Theoretical Computer Science: computation, physics and beyond, WTCS'12, pp 357–375. Springer-Verlag, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-27654-5_28

Keenan A, Schweller R, Zhong X (2014) Exponential replication of patterns in the signal tile assembly model. Nat Comput 14(2):265–278

Keenan A, Schweller RT, Zhong X (2013) Exponential replication of patterns in the signal tile assembly model. In: Soloveichik D, Yurke B (eds.) DNA, Lecture Notes in Computer Science, 8141:118–132. Springer

Lathrop JI, Lutz JH, Patitz MJ, Summers SM (2011) Computability and complexity in self-assembly. Theory Comput Syst 48(3):617–647

Liu W, Zhong H, Wang R, Seeman NC (2011) Crystalline two-dimensional dna-origami arrays. Angewandte Chemie Int Edit 50(1):264–267. https://doi.org/10.1002/anie.201005911

Padilla JE, Patitz MJ, Schweller RT, Seeman NC, Summers SM, Zhong X (2014) Asynchronous signal passing for tile self-assembly: fuel efficient computation and efficient assembly of shapes. Int J Foundat Comput Sci 25(4):459–488

Padilla JE, Sha R, Kristiansen M, Chen J, Jonoska N, Seeman NC (2015) A signal-passing DNA-strand-exchange mechanism for active self-assembly of DNA nanostructures. Angewandte Chemie Int Edit 54(20):5939–5942

Patitz MJ, Rogers TA, Schweller RT, Summers SM, Winslow A (2016) Resiliency to multiple nucleation in temperature-1 self-assembly. In: Proceedings of the 22nd International Conference on DNA Computing and Molecular Programming (DNA 22), Ludwig-Maximilians-Universität, Munich, Germany September 4-8, 2016, pp 98–113

Patitz MJ, Summers SM (2011) Self-assembly of decidable sets. Nat Comput 10(2):853–877

Qian L, Winfree E (2011) Scaling up digital circuit computation with DNA strand displacement cascades. Science 332(6034):1196–1201

Rothemund PWK (2005) Design of dna origami. In: ICCAD '05: Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design, pp 471–478. IEEE Computer Society, Washington, DC, USA

Rothemund PWK, Winfree E (2000) The program-size complexity of self-assembled squares (extended abstract). In: STOC '00: Proceedings of the thirty-second annual ACM Symposium on Theory of Computing, pp 459–468. ACM, Portland, Oregon, United States

Schulman R, Yurke B, Winfree E (2012) Robust self-replication of combinatorial information via crystal growth and scission. In: Proceedings of the National Academy of Sciences 109(17), 6405–10. http://www.biomedsearch.com/nih/Robust-self-replication-combinatorial-information/22493232.html

Simmel FC, Yurke B, Singh HR (2019) Principles and applications of nucleic acid strand displacement reactions. Chem Rev 119(10):6326–6369

Soloveichik D, Winfree E (2005) Complexity of compact proofreading for self-assembled patterns. In: Carbone A, Pierce NA (eds.) DNA Computing, 11th International Workshop on DNA Computing, DNA11, London, ON, Canada, June 6-9, 2005. Revised Selected Papers, Lecture Notes in Computer Science, 3892:305–324. Springer

Soloveichik D, Winfree E (2007) Complexity of self-assembled shapes. SIAM J Comput 36(6):1544–1569

Summers SM (2012) Reducing tile complexity for the self-assembly of scaled shapes through temperature programming. Algorithmica 63(1–2):117–136. https://doi.org/10.1007/s00453-011-9522-5

Wang B, Thachuk C, Ellington AD, Winfree E, Soloveichik D (2018) Effective design principles for leakless strand displacement systems. Proc Nat Acad Sci 115(52):E12182–E12191

Wei B, Dai M, Yin P (2012) Complex shapes self-assembled from single-stranded dna tiles. Nature 485(7400):623–626

Zhang DY, Hariadi RF, Choi HM, Winfree E (2013) Integrating DNA strand-displacement circuitry with DNA tile self-assembly. Nat Commun 4(1):1–10

Zhang DY, Seelig G (2011) Dynamic DNA nanotechnology using strand-displacement reactions. Nat Chem 3(2):103–113