Strict Self-Assembly of Discrete Self-Similar Fractal Shapes

Florent Becker

Strict Self-Assembly of Discrete Self-Similar Fractal Shapes

Florent Becker
LIFO — Université d'Orléans, rue Léonard de Vinci
45000 Orléans, France

Abstract

This paper gives a (polynomial time) algorithm to decide whether a given Discrete Self-Similar Fractal Shape can be assembled in the aTAM model.

In the positive case, the construction relies on a Self-Assembling System in the aTAM which strictly assembles a particular self-similar fractal shape, namely a variant K^∞ of the Sierpinski Carpet. We prove that the aTAM we propose is correct through a novel device, *self-describing circuits* which are generally useful for rigorous yet readable proofs of the behaviour of aTAMs.

We then discuss which self-similar fractals can or cannot be strictly self-assembled in the aTAM. It turns out that the ability of iterates of the generator to pass information is crucial: either this *bandwidth* is eventually sufficient in both cardinal directions and K^∞ appears within the fractal pattern after some finite number of iterations, or that bandwidth remains ever insufficient in one direction and any aTAM trying to self-assemble the shape will end up with an ultimately periodic pattern.

2012 ACM Subject Classification Theory of computation → Abstract machines; Theory of computation → Timed and hybrid models; Theory of computation → Computational geometry

Keywords and phrases Molecular Self-assembly, Discrete Fractals, Substitutions

1 Introduction

There is an area of particular interest at the intersection between dynamical systems, models of computation and automata, and discrete geometry. There, researchers try to determine the limits of various models of computation when faced with geometrical constraints, and to relate them with the dynamic properties of the models. This paper deals with Self-Assembling Tilings in the Abstract Tile Assembly Model (aTAM), as introduced by Winfree [21]. In addition to its *raison d'être* as an abstract model of DNA computing, this model enjoys an interest as such as an intermediary between one-dimensional and two-dimensional Cellular Automata (CAs), as well as a kind of asynchronous version of CAs and tilings. It can also be seen as a kind of branching Langton's Ant.

Two features distinguish this model from more classical models such as Cellular Automata: the fact that it consumes the space it computes in, forcing a steady growth of its output, as well as its asynchronicity. The aTAM is able to simulate classical models such as Turing Machines and Cellular Automata [21] by covering large rectangles in a synchronized manner. The synchronization can be wrought from the jaws of the environment's asynchronicity by using so called "Temperature 2" systems where the local attachement of a tile to a site can depend from the concommitent presence of two tiles on neighboring sites. This synchronization can also be substitued for by a richer local geometry, such as in a 3D space [2], or the presence of a variety of tiles shapes [12].

This hints that much of the subtelty of the asynchronous and write-only nature of the aTAM reflects in its ability to assemble fine geometrical features. Since 2010 [20], a particular focus has been set on so-called Discrete Self-Similar Fractal Shapes (DSSFS) as candidates for being tough to assemble in the aTAM and allowing to pinpoint the subtelty of its variants. In order to meaningfully constrain the aTAM using the geometry of the shape that is being

assembled, this line of research investigates its *strict* self-assembly. That is, the authors demand that in order to assemble a shape S, no tile is ever put outside of S.

This program has seen some progress, with two lines of results. The first line of results [6, 1, 15, 20, 13] looks at close variants of DSSFS, either slightly laxer and showing that they *can* be obtained by aTAM system, or slightly stricter and showing that the cannot be obtained. The second line of results [11, 10] looks at variants of the aTAM, generally more powerful, showing that they can assemble (some) DSSFS.

The big question since [20] has been whether the plain model (aTAM) can self-assemble some unadultered DSSFS. The conjecture has generally been that their geometry is too complicated for the aTAM, that is that *there was no aTAM system able to uniquely and strictly self-assemble a DSSFS*. In order to characterize these geometrical constraints, several notions have been proposed, such as ζ-dimension and sparsity [8, 9], and ease of disconnecting [11, 6, 1]. This paper answers the question differently: not only there *is* a DSFSS which can be strictly self-assembled, in fact most DSSFS can indeed be self-assembled, as long as a periodic tiling by their generator is well connected.

The idea for the positive construction of this paper relies on a family of ideas from the theory of Tilings. The hierarchical structure of discrete fractals is a lot alike that of many aperiodic tilings. In trying to assemble such a hierarchical shape, one can take inspiration from authors such as Mozes or Goodman-Strauss [7, 18] who use a system with several layers of information checking each other's work in order to impose from local rules the desired global hierarchy. Like theirs, the construction uses several layers carrying the same information at different scales. This part of the construction is somewhat simpler than in tilings because of the seeded nature of the assembly. In tilings again, Durand, Romashchenko and Shen show in [4] another way to impose such a structure, by obtaining their tiling through a Fixed-Point Theorem. The approach here through *self-describing circuits* is a bastardization of these two, with some tricks of its own to deal with the lack of global synchronisation in the assembly. The way it is presented through a circuit is a necessity here to deal with the proof of a big aTAM system acting in an environment with non-trivial geometric constraints, but it can certainly be useful as a tool to express and prove many constructions in the aTAM model.

The characterization of the fractal shapes which *cannot* be assembled in the aTAM relies on a Pumping Lemma in the tradition of [17]. This Pumping Lemma limits the amount of computation that can be done in the assembly by looking at its *treewidth*, or equivalently, the size of the largest square it encircles. This is a formalization of a new limit to computing power in the aTAM, and it should apply in a variety of settings, with a *je-ne-sais-quoi* of Complexity Theory.

The presentation will start in Section 2 with the definitions of the abstract Tile Assembly Model, as well as the statement of the Tree Pump Lemma (Lemma 8) and the difference between weak and strict assembly of shapes. The paper then covers the basics about discrete self-similar fractal shapes, and states the main positive result, Theorem 18. Then, Section 3 presents Self-Describing Embedded Circuits which form the main proof device for Theorem 18; they are put to use in Section 4, concluding the proof of the positive construction. Finally, Section 5 delimitates the fractal shapes which can be constructed through the aTAM. The dichotomy there is quite sharp: for a given shape S, either the technique of Theorem 18 can be straightforwardly adaptated to S or there is no way at all to obtain it in the aTAM. Lemma 56 gives a polynomial-time algorithm to distinguish between the two situations.

2 Definitions and statement of the main results

2.1 Notations

A variable with an arrow such as $\vec{a}$ denotes a vector of values indexed by some set S. Given $s \in S$, a_s is the element of $\vec{a}$ associated with s.

For any sets A, B and S, given $\vec{a} \in A^S$ and $\vec{b} \in B^S$, the standard notation $\vec{a} \otimes \vec{b}$ is the vector $\vec{v} \in (A \times B)^S$ defined by $\forall s \in S, v_s = (a_s, b_s)$. Likewise, given $f : A^I \mapsto A^O$ and $g : B^I \mapsto B^O$, the function $f \otimes g : (A \times B)^I \mapsto (A \times B)^O$ is defined by $(f \otimes g)(\vec{a} \otimes \vec{b}) = f(\vec{a}) \otimes g(\vec{b})$.

The unit vectors $(0,1), (1,0), (0,-1), (-1,0)$ of $\mathbb{Z}^2$ are noted N, E, S, W respectively. An oriented edge of $\mathbb{Z}^2$ is an *arc*, noted $(a \to b)$; its *direction* is the unit vector $b - a \in \{N, E, S, W\}$. Given an arc $e = (a \to b)$ and a vector $v \in \mathbb{Z}^2$, $e + v = (a + v \to b + v)$. The arc out of $a \in \mathbb{Z}^2$ in direction $d \in \{N, E, S, W\}$ is $(a \to a + d)$.

Let P be a finite subset of $\mathbb{N}^2$, with $w = \max(x | (x, y) \in P)$ and $h = max(y | (x, y) \in P)$. Then for $z = (x, y) \in \mathbb{Z}^2$, $\lfloor z/P \rfloor$ is the pair $(\lfloor x/w \rfloor, \lfloor y/h \rfloor)$ and $z \bmod P$ is the pair $(x \bmod w, y \bmod h)$.

The constructions in Section 3 and 4 make heavy use of sets built as cartesiaun products. In an attempt to make their exposition more readable and distinguish the function of each component of the product, they are given in a "record" or "object" like syntax using '{}' for record construction and "$x \cdot$ field" for field access. That is, given a base set X and a finite set of k labels such as $L = \{\text{zeroth}, \text{first}, \text{second}, \ldots, \text{not-quite-k-th}\}$, the element $x = (x_0, x_1, x_2, \ldots, x_{k-1})$ is written $x = \{\text{zeroth} = x_0, \text{first} = x_1, \ldots, \text{not-quite-k-th} = x_{k-1}\}$. Symetrically, x_0 can be extracted from x by writing $x \cdot \text{zeroth}$. A judicious set of labels adds greatly to the usefulness of that notation.

2.2 Self-assembly and the Abstract Tile-Assembly Model

2.2.1 Definitions

There follow a brief exposition of the basic definition of the aTAM. The survey by Patitz [19] as well as the classical article by Winfree [21] give a far less telegraphic exposition.

▶ **Definition 1** (Wang Tile). *Given an alphabet Σ, a* Wang Tile *is an element of $\Sigma^{\{N,E,S,W\}}$, i.e. a unit square with an element of Σ on each of its sides. Given $w \in \Sigma^{\{N,E,S,W\}}$ and $d \in \{N, E, S, W\}$, $w(d)$ is referred to as the* color *of the d* side *of w.*

▶ **Definition 2** (Assembly). *Given a set $\mathcal{W}$ of Wang Tiles, an* assembly *of $\mathcal{W}$ is a partial function $A : \mathbb{Z}^2 \to \mathcal{W}$. It is finite if its domain is. The notation $z \in A$ should be read as $z \in \mathrm{dom}(A)$, or equivalently "$A(z)$ is defined".*

The set of assemblies on $\mathcal{W}$ is noted $\mathcal{W}^{\subset \mathbb{Z}^2}$.

▶ **Definition 3** (aTAM). *Let Σ be an alphabet, an* unseeded Tile Assembly System *T is a triplet*

$$\left\{ \begin{array}{ccc} \text{tileset} & \in & \Sigma^{\{N,E,S,W\}} \\ \text{strength-function} & : & \Sigma \to \mathbb{Z} \\ \text{temperature} & \in & \mathbb{N} \end{array} \right\}.$$

A seeded Tile Assembly System *T' is a quadruplet*

$$\left\{ \begin{array}{ccc} \text{tileset} & = & t \in \Sigma^{\{N,E,S,W\}} \\ \text{strength-function} & = & s : \Sigma \to \mathbb{Z} \\ \text{temperature} & = & \tau \in \mathbb{N} \\ \text{seed} & = & \sigma \in (T' \cdot \text{tileset})^{\subset \mathbb{Z}^2} \end{array} \right\}.$$

In both cases, the notation $T^{\subset\mathbb{Z}^2}$ *implicitely refers to the set of assemblies* $(T\cdot\text{tileset})^{\subset\mathbb{Z}^2}$.

The crucial difference between a Tile Assembly System and a mere set of Wang Tiles are its strength function and its temperature. The strength function defines the *binding strength* of an edge of an assembly.

▶ **Definition 4** (Binding). *Let* $\mathcal{S}$ *be a Tile Assembly System, and* $A \in \mathcal{S}^{\subset\mathbb{Z}^2}$. *Let* e *be an edge between two positions* z, $z' = z + d$ *in* A. *The* binding $b(e)$ *of edge* e *in* A *is*

- 0 *if* $A(z)(d) \neq A(z')(-d)$, *and*
- $\mathcal{S}\cdot\text{strength}(g)$ *if* $A(z)(d) = A(z')(-d) = g$.

Given a set E *of edges of* A, *the binding strength of* E *is* $b(E) = \sum_{e\in E} b(e)$.

This binding strength corresponds to forces tying the assembly together in the face of thermal agitation. This thermal agitation is modeled by $\mathcal{S}\cdot$temperature. Whenever an assembly has a cut with a binding strength less than the temperature, it will tend to be torn along this cut by thermal agitation. Assemblies which do not have such cuts are *stable*.

▶ **Definition 5** (Stable assembly). *Given a Tile Assembly System* $\mathcal{S}$, *an assembly* $A \in \mathcal{S}^{\subset\mathbb{Z}^2}$ *is* stable *if for any cut* C *of* A, $b(C) \geq \mathcal{S}\cdot\text{temperature}$

Stability begets a definition for the dynamics of the process of self-assembly: tiles are added to an assembly as long as the resulting new assembly is stable.

▶ **Definition 6** (Attachment, Assembly Sequence). *Given a Tile Assembly System* $\mathcal{S}$ *and an assembly* $A \in \mathcal{S}^{\subset\mathbb{Z}^2}$, *an* attachment candidate *is a pair* $t@z$, *with* $t \in \mathcal{S}\cdot\text{tileset}$ *and* $z \in \mathbb{Z}^2$. *It is* valid *if* $z \notin \text{dom}\, A$ *and* stable *if* $A' = A \cup \{z \mapsto t\}$ *is stable. If it is both valid and stable, it is an attachment, noted* $A \xrightarrow{t@z} A'$

An assembly A' follows *from an assembly* A *if there is a sequence of valid and stable attachments* $A_0 = A \xrightarrow{t_1@z_1} A_1 \xrightarrow{t_2@z_2} \dots \xrightarrow{t_k@z_k} A_k = A'$. *Such a sequence is a* assembly sequence *from* A *to* A'. *It is noted* $\alpha = A \to_{\mathcal{S}} A'$, *and* A' *is noted* $\lim\alpha$. *If* α *is infinite, then* $\lim\alpha = \bigcup_i A_i$. *A* candidate assembly sequence *is a sequence of attachments which are not necessarily valid.*

The set of assembly sequences following from a given assembly A *is noted* $\mathcal{H}[\mathcal{S}, A]$, *and* $\mathcal{A}[\mathcal{S}, A] = \{\lim\alpha | \alpha \in \mathcal{H}[\mathcal{S}, A]\}$.

An assembly is terminal if $\mathcal{A}[\mathcal{S}, A] = \{A\}$. *The set of terminal assemblies following from* A *is* $\mathcal{A}_\square[\mathcal{S}, A]$.

Given a seeded Tile Assembly System, its *productions* are the assemblies following from its seed.

▶ **Definition 7** (Production, Terminal Production). *Given a seeded Tile Assembly System* $\mathcal{S}$, *a* production *is an assembly which follows from* $\mathcal{S}\cdot$seed.

The set of productions of $\mathcal{S}$ *is noted* $\mathcal{A}[\mathcal{S}]$, *and the set of terminal productions is noted* $\mathcal{A}_\square[\mathcal{S}]$. *Likewise,* $\mathcal{H}[\mathcal{S}] = \mathcal{H}[\mathcal{S}, \mathcal{S}\cdot\text{seed}]$.

Note that a production may be infinite but not terminal.

Given a subset $X \subset \mathbb{Z}^2$, the restriction of an assembly sequence α to X, noted $\alpha_{|X}$ is the sequence of attachemnts of α taking place within X. Likewise, for an assembly A, $A_{|X}$ is the restriction of A to the positions within X.

2.2.2 The Tree Pump Lemma

No paper about self-assembly would be complete without a "pump or die" lemma in the style of [17]. This one examines the case of skinny productions, that is those who do not encircle any square larger than N for some fixed N. The crucial property which makes them simple(-ish) is their bounded treewidth. The interest of this lemma —especially for eight year olds— is that it lets the user direct the flow of the firehose.

▶ **Lemma 8** (Tree Pump). *For any aTAM system $\mathcal{S}$ with $\mathcal{S}\cdot$seed finite and connected, define the following sets of assemblies:*

- *for any integer m, $C_m[\mathcal{S}]$ is the set of assemblies of $\mathcal{S}$ which encircle an $m \times m$ square;*
- *for any real k and vector $\vec{d}$, $B_{k,\vec{d}}[\mathcal{S}]$ is the set of assemblies of $\mathcal{S}$ which do not cover any position $\vec{p}$ such that $\vec{p}\cdot\vec{d} > k$*
- *for any vector $\vec{d}$, $P_{\vec{d}}[\mathcal{S}]$ is the set of* ultimately periodic *assemblies of $\mathcal{S}$ such that there is a vector $\vec{p}$ with $|\vec{p}\cdot\vec{d}| > 0$ and a non-empty sub-assembly $a \subseteq A$ such that $a + \vec{p} \subseteq a$.*

Then, there is a function $F : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ such that for aTAM system $\mathcal{S}$ with n tiles and a 1-tile seed, integer m and unit vector $\vec{d}$ of $\mathbb{R}^2$,

$$\mathcal{A}_{\square}[\mathcal{S}] \cap (C_m[\mathcal{S}] \cup B_{F(n,m),\vec{d}}[\mathcal{S}] \cup P_{\vec{d}}[\mathcal{S}]) \neq \emptyset.$$

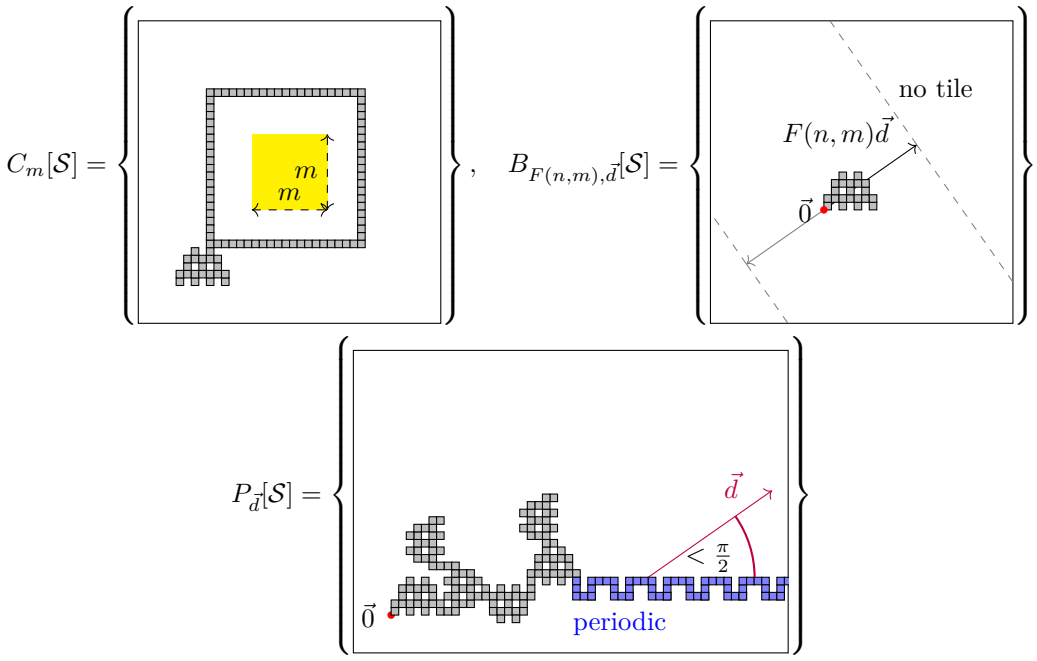


Figure 1 The three sets defined by Lemma 8: $C_m[\mathcal{S}]$ is the assemblies which encircle an $m \times m$ square, $B_{F(n,m),\vec{d}}[\mathcal{S}]$ is the set of assemblies which do not reach further than $F(n,m)$ in direction $\vec{d}$, and $P_{\vec{d}}[\mathcal{S}]$ is the assemblies which contain a periodic path with period $\vec{p}$ such that $\vec{p}\cdot\vec{d} > 0$.

The Tree Pump Lemma states that in order to do a meaningful amount of computation, a self-assembling system with n tiles needs to encircle large squares —say, of size m, thus hitting $C_m[\mathcal{S}]$. If it does not, then its productions look very much like trees drawn on $\mathbb{Z}^2$

with a m-cell wide brush. A branch of that tree then behaves like a finite automaton. If that automaton stops, the assembly goes no further than $F(n,m)$ in any given direction $\vec{d}$, which gives a final production in $B_{F(n,m),\vec{d}}[\mathcal{S}]$. If not, then the long branches must have an ultimately periodic behavior which gives a final production in $P_{\vec{d}}[\mathcal{S}]$.

The full proof is given in Appendix A, as it involves a fair few ancillary definitions.

2.3 Strict versus Weak Assembly of shapes

▶ **Definition 9** (Strict Assembly). *Lace-like*

▶ **Definition 10** (Weak Assembly). *Paint-like*

Weak and strict assembly are called thus because it is generally easier to weakly assemble a given shape than to strictly assemble it. Indeed, when weakly self-assembling a given shape S, it is possible to use positions in $\mathbb{Z}^2 \setminus S$ for computation. In strict self-assembly, doing so is impossible.

▶ **Lemma 11.** *Any shape that can be strictly assembled can also be weakly assembled.*

2.4 Discrete Self-Similar Fractal Shapes

Fractals are usually defined in some continuous space such as $\mathbb{R}^2$ or $\mathbb{C}^2$. In the context of self-assembly, one needs to work with *discrete* fractals, that is, subsets of $\mathbb{Z}^2$ which are self-similar. In this paper, discrete self-similar fractal shapes are defined as fixed points of some 2 dimensional substitution. In [20], self-similar fractal shapes are defined as the union of an infinite hierarchy of shapes. The following definition is identical, as long as the generator contains $(0,0)$. Explicitely defining a geometrical substitution will help, as that substitution can guide constructions happening on the fractal. If the generator does not contain $(0,0)$, the classical definition can be recovered by iterating the substitution associated with $G \cup (0,0)$ to reach the fix-point, then iterating once the substitution associated with G.

The substitution σ_G based on a finite pattern G replaces each pixel in a pattern X with a copy of G.

▶ **Definition 12** (Rectangular substitution). *Let G be a finite subset of $\mathbb{N}^2$. Let $w = \max(\{x|(x,y) \in G\})$ and $h = \max(\{y|(x,y) \in G\})$. The substitution $\sigma_G : \mathcal{P}(\mathbb{N}^2) \to \mathcal{P}(\mathbb{N}^2)$ associated with G is defined by:*

$$\forall X \subset \mathbb{N}^2, \sigma_G(X) = \{p \in \mathbb{N}^2 | \lfloor p/G \rfloor \in X \wedge p \bmod G \in G\}$$

Using substitution allows manipulating the recursive definition of DSSF with richer data than an in/out bit for each position.

▶ **Definition 13** (Colored substitution). *Let X be an alphabet and G a finite subset of $\mathbb{N}^2$; given for each $x \in X$ a coloring $C(x) \in X^G$, the substitution σ_C associated with C is defined for each partial coloring $Y : \mathbb{N}^2 \dashrightarrow X$ by:*

$$\begin{cases} \sigma_C(Y) : \mathbb{N}^2 \dashrightarrow X \\ \sigma_C(Y) : \vec{z} \mapsto C(Y(\lfloor z/G \rfloor))(z \bmod G), \end{cases}$$

Where $\sigma_C(Y)(\vec{z})$ is defined whenever $Y(\lfloor z/G \rfloor)$ is defined and $z \bmod G \in G$.

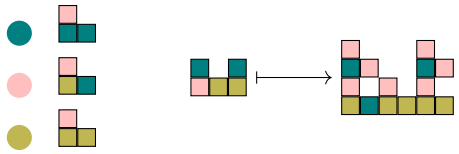


Figure 2 A colored substitution on a 3-colored alphabet $\Sigma = \{$●,●,●$\}$ is defined from a shape $G =$  and a coloring of G for each color of Σ. It can then be applied to any colored shape (right).

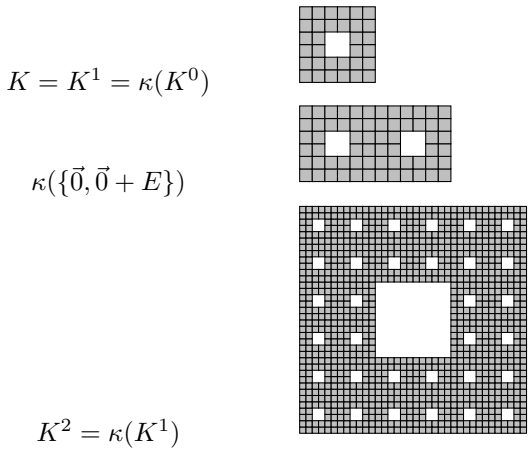


Figure 3 The substitution κ of the Sierpinski's Cacarpet.

▶ **Definition 14** (Self-Similar Fractal Shape). *Let G be a finite subset of $\mathbb{N}^2$ with $(0,0) \in G$. The discrete self-similar fractal shape with generator G is the following subset of $\mathbb{N}^2$:*

$$G^\infty = \bigcup_{i \in \mathbb{N}} \sigma_G^i(\{(0,0)\}).$$

The i-th step of the fractal is $G^i = \sigma_G^i(\{(0,0)\}$.

One self-similar fractal shape will be of particular interest to us in this paper, wich we name *Sierpinski's Cacarpet*. It is the self-similar fractal shape K_∞ with generator $K = \{0, \dots, 5\}^2 \setminus \{2,3\}^2$, represented on figure 3. We note $\kappa = \sigma_K$ the associated substitution.

Discrete Self-Similar Fractal Shapes have a lot of regularity. In particular, given a large enough "patch" of G^∞, it is possible to find copies of G^k for a given k around it. The first kind of large patches are squares centered on an element of G^∞.

▶ **Lemma 15.** *Let G be a finite subset of $\mathbb{N}^2$, of width w and height h, and $n = \min(w,h)$. Let $\vec{z} \in G^\infty$. Then the square of size $2n^k + 1$ centered on $\vec{z}$ contains a copy of G^k.*

Proof. For $\vec{z}$ to be in G^∞, it has to be in a copy of G^k. That copy is in a square of size n^k which contains $\vec{z}$. That square is included in the square of size $2n^k + 1$ centered on $\vec{z}$. ◀

The second kind of large patches which will turn out to be useful are sectors limited by two diverging lines.

▶ **Lemma 16.** *Let G be a finite subset of $\mathbb{N}^2$, let $\vec{u}, \vec{v}$ be two non-colinear vectors $\mathbb{N}^2$. Let $p \in \mathbb{N}^2$, and S be the sector spanning from p between directions $\vec{u}$ and $\vec{v}$. Assume $G^\infty \cap S \neq \emptyset$; then $G^\infty \cap S$ contains copies of G^k for all k.*

Proof. ◀

2.5 Patitz' conjecture

▶ **Conjecture 17.** *No DSSF strictly self-assemble in the aTAM.*

2.6 The Sierpinski Cacarpet can be Strictly Self-Assembled

Until this paper, in order to assemble a DSSF $\mathcal{S}$ in the aTAM, one would typically embed counters into $\mathcal{S}$, or rather some small superset of it [20, 16, 13]. Strict self-assembly of a DSSFS can be obtained in another way:

- first, choose a *suitable* DSSF to assemble, namely the Sierpinski Cacarpet K^∞
- observe that K^∞ is the fixed-point of a discrete substitution κ,
- in consequence, define a hierarchy of systems: a *self-describing circuit* $C_\square$, which entails an *Locally Deterministic Oriented Pattern* $P_\square$, whose tiles make a temperature 2 Tile Assembly System $S_\square$ in the abstract Tile Assembly Model which uniquely assembles $P_\square$.

In fine, the correctness of $S_\square$ follows thus:

- $C_\square$ is defined as a fixed-point of κ, and thus has K^∞ as its support,
- the structure of $C_\square$ allows it to be *evaluated*, which yields a pattern $P_\square$, with the same support,
- moreover, $C_\square$ is *self-describing* (Definition 24 and Theorem 32), which gives $P_\square$ its local determinism (Definition 26 and Lemma 27)
- the set of tiles in P_κ defines an aTAM System $S_\square$; since $P_\square$ is locally deterministic, the unique final production of $S_\square$ is $P_\square$ (Lemma 30).

▶ **Theorem 18.** *There is an aTAM system at temperature 2 which self-assembles the DSSF pattern K^∞.*

The proof of Theorem 18 spans sections 3 and 4. The constructivist reader will find the detailed construction of C_κ in section 4, this should enlighten them about most of the ideas of the construction. The formalist will find the tower of models and the links between them in section 3.

3 Abstract models: Self-Describing Embedded Circuits and Locally Deterministic Patterns

3.1 Embedded circuits

A circuit is made of gates. Each of these gates has a *function*, computing the outputs of the gates from the inputs. The gate is embedded on a unit square according to its *wiring*, which states where its inputs come from and where its outputs go to. The wiring and the function must be compatible: the wiring must have as many inputs as the function.

▶ **Definition 19** (Wire). *A wire is a pair of a list of distinct directions $\vec{D} \in \{N, E, S, W\}^*$ and a distinguished element $d \in D$. It is noted as a word on the alphabet $\{n, e, s, w\}$ containing the elements of D, with d capitalized. For instance, the wire $((N, E), E)$ is noted nE, while $((N, E), N)$ is noted Ne.*

Given a wire $w = (\vec{D}, d)$, its opposite $-w$ is $((-\vec{D}_i), -d)$, i.e. $-Ne = Sw$. The set of wires is noted Wires.

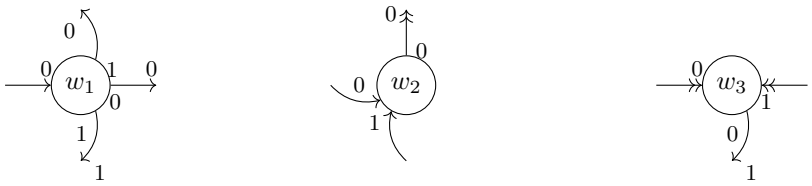


Figure 4 The graphical representation of three wirings: $w_1 = \{\text{Inputs} = (W), \text{Outputs} = \{N, S, E\}, \text{output-num} = \{N \to 1, E \to 0, S \to 1\}, \text{outwires} = \{Ne, E, eS\}\}$, $w_2 = \{\text{Inputs} = (W, S), \text{Outputs} = \{N\}, \text{output-num} = \{N \to 0\}, \text{outwires} = \{Ns\}\}$ and $w_3 = \{\text{Inputs} = (W, E), \text{Outputs} = \{S\}, \text{output-num} = \{S \to 0\}, \text{outwires} = \{eS\}\}$. Neighboring input wires bend to come together into the wiring. Double arrows indicate a pair of opposite input arrows. The output wire in direction d bends to mirror $w\cdot\text{outwires}(d)$.

▶ **Definition 20** (Wiring). *Given two integers i, o such that $i + o \leq 4$, a (i, o)-*wiring *w (i.e. a wiring with* in-degree *i and* out-degree *o) is given by*

- *a partition of $\{N, E, S, W\}$ into three sets, $w\cdot\text{Inputs}$, $w\cdot\text{Outputs}$ and $w\cdot\text{Inert}$, where $w\cdot\text{Inputs}$ is ordered*
- *a map $w\cdot\text{output-num} : w\cdot\text{Outputs} \to \{0, \dots, o-1\}$;*
- *a map $w\cdot\text{outwires} : w\cdot\text{Outputs} \to \text{Wires}$ such that d is the distinguished element of $w\cdot\text{outwires}(d)$.*

The set of all wirings is noted $\mathcal{W}$.

▶ **Definition 21** (Input / Output / Inert Sides, Degree). *Given a wiring $w \in \mathcal{W}$, the elements of $w\cdot\text{Inputs}$ are the* input sides *of w, the elements of $w\cdot\text{Outputs}$ are its* output sides*, and the elements of $w\cdot\text{Inert}$ are its* inert sides*. The in-degree of w is the number of its input sides, its out-degree of w is its number of output sides.*

Additionally, each output side also determines the *input* sides on the following gate and their ordering. That is, a wiring w with $nE \in w\cdot\text{outwires}$ can only have a wiring w' with $\text{Inputs}\, w' = (N, E)$ to its right.

The graphical representation of a wiring is given on Figure 4. Each direction in $w\cdot\text{Inputs}$ has an incoming arrow, and each direction in $w\cdot\text{Outputs}$ has an outgoing arrow. If $w\cdot\text{Inputs}$ has only one element, the corresponding arrow is straight and simple. Likewise for the outgoing arrow when $w\cdot\text{outwires}(d)$ is a singleton. When $w\cdot\text{outwires}(d)$ is made of two adjacent directions, the corresponding arrows bend to come near each other; likewise, modulo rotation, when $w\cdot\text{outwires}(N) \in \{Ne, eN\}$, the corresponding arrow bends left, and right if it is Nw or wN. For pairs of opposite directions, a double arrow is used. Triple and quadruple inputs are not used in this paper. Inputs are numbered according to the order of $w\cdot\text{Inputs}$ and output wires are numbered according to $w\cdot\text{output-num}$.

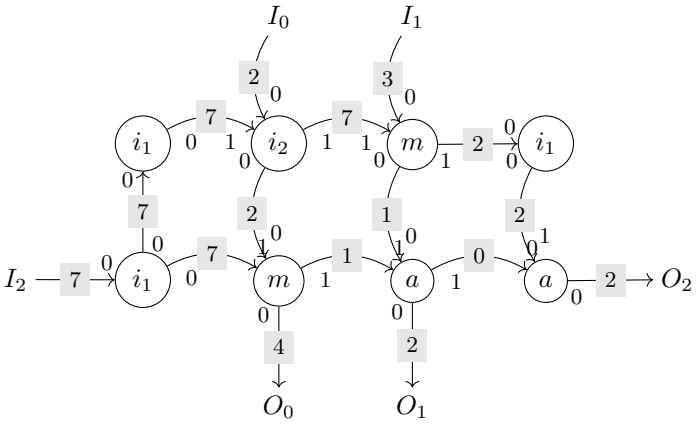
▶ **Definition 22** (Gate). *Given an alphabet Σ two integers i, o, a gate g is a pair $\{\text{func} : f, \text{wiring} : w\}$, with f a function $f : \Sigma^i \to \Sigma^o$ and w an (i, o)-wiring.*

The set of gates on alphabet Σ is noted $\text{Gates}(\Sigma)$.

These gates may be placed onto a subset of the grid $\mathbb{Z}^2$ to make *circuits*. An example of a circuit, a multiplier built from a handful of adders and auxiliary gates is given on Figure 5.

▶ **Definition 23** (Circuit). *Let Σ be an alphabet. Let D be an oriented subgraph of $\mathbb{Z}^2$, $\vec{I}$ be a sequence of arcs from $\mathbb{Z}^2 \setminus D$ to D and $\vec{O}$ a sequence of arcs from D to $\mathbb{Z}^2 \setminus D$.*

A circuit C with dependency graph D, input bus $\vec{I}$ and output bus $\vec{0}$ is a map $D \to \text{Gates}(\Sigma)$ such that:



$$\begin{cases} i_1 : \Sigma \to \Sigma \\ i_1(x) = x \\ i_2 : \Sigma^2 \to \Sigma^2 \\ i_2(x,y) = (x,y) \\ m : \Sigma^2 \to \Sigma^2 \\ m(x,y) = (xy \bmod 10, \lfloor xy/10 \rfloor) \\ a : \Sigma^2 \to \Sigma^2 \\ a(x,y) = ((x+y) \bmod 10, \lfloor (x+y)/10 \rfloor) \end{cases}$$

Figure 5 A multiplier circuit M_2^1 and the definition of the functions of its gates. The alphabet is the set of digits $\mathbb{Z}/10\mathbb{Z}$. The values in the grey squares are an example of evaluation: $32 \times 7 = 224$

- *for any arc E in direction $d \in \{N, E, S, W\}$ between two positions $a, b \in D$, let $w_a = C(a)\cdot\text{wiring}$ and $w_b = C(b)\cdot\text{wiring}$; then $d \in w_a\cdot\text{Outputs}$, $-d \in w_b\cdot\text{Inputs}$, and $w_a\cdot\text{Outputs}(d) = (d, w_b\cdot\text{Inputs})$,*
- *for any adjacent positions a, b in D with no arc between a and b, $d \in C(a)\cdot\text{wiring}\cdot\text{Inert}$ and $-d \in C(b)\cdot\text{wiring}\cdot\text{Inert}$*
- *and for any arc e in direction $d \in \{N, E, S, W\}$ between two positions $a \in D$ and $b \in \mathbb{Z}^2 \setminus D$, either:*
  - *$d \in C(a)\cdot\text{wiring}\cdot\text{Inert}$ and e is neither an element of $\vec{I}$ nor of $\vec{O}$,*
  - *$d \in C(a)\cdot\text{wiring}\cdot\text{Outputs}$ and e is an element of $\vec{O}$,*
  - *$-d \in C(a)\cdot\text{wiring}\cdot\text{Inputs}$ and $-e$ is an element of $\vec{I}$.*

The support of C is the vertex-set of D.

Let C be a circuit, and A the arc-set of its dependency graph D. A function $v : A \mapsto \Sigma$ *conforms* to C at a position $\vec{z}$ in the support of C, if for any $d \in C(\vec{z})\cdot\text{wiring}\cdot\text{Outputs}$ with $k = C(\vec{z})\cdot\text{wiring}\cdot\text{output-num}$, $v(o) = (C(v)\cdot\text{func})(v(i_0), \ldots, v(i_m))$ with $i_0, \ldots, i_m$ the input arcs of $C(\vec{z})$, o its output arc in direction d.

A circuit is *well-founded* when its dependency graph has no cycle or infinite backwards paths. For a well-founded circuit, given a vector of inputs $\vec{\imath}$ indexed by $\vec{I}$, there is a *unique* function $\widetilde{C}(\vec{\imath}, -)$ which is compatible with C and such that the values on $\vec{I}$ are $\vec{\imath}$. The circuit function $\bar{C} : \Sigma^{\vec{I}} \to \Sigma^{\vec{O}}$ is defined by $\bar{C}(\vec{\imath}) = (\widetilde{C}(\vec{\imath}, O_k))_k$.

A circuit is *finitely rooted* when its input bus is finite and its number of gates with in-degree 0 is finite. It is *evaluable* if it is well-founded and finitely rooted.

The circuit M_2^1 on figure 5 is a multiplier. Take two natural integers $a < 10, b < 100$, pose $b = b_0 + 10b_1$ with $\forall i, 0 \leq b_i < 10$. Let $\vec{i}_{a,b} = (a_0, b_0, b_1)$. Then $\bar{C}(\vec{i}_{a,b}) = \vec{o}$, with $ab = \sum o_i 10^i$ and $\forall i, 0 \leq o_i < 10$. Figure 5 gives an example of the evaluation of the circuit on inputs 32 and 7; the values of $\widetilde{C}(\vec{i}, -)$ on each arc are given on the figure.

When the alphabet Σ is structured in two layers, i.e. $\Sigma = A \times B$, the function f of a gate g may act independently on layer A and layer B, i.e. $f = f_A \otimes f_B$. Then, taking $g_A = \{\,\text{wiring} : w, \text{func} : f_A\} \in \text{Gates}(A)$ and $g_B = \{\,\text{wiring} : w, \text{func} : f_B\} \in \text{Gates}(B)$, g can be written as $g_A \otimes g_B$. In other words, the tensor product $\otimes$ applies to gates *with the same wiring*.

3.2 Self-description

The process of self-assembly in the aTAM and the evaluation of an evaluable embedded circuit are somewhat alike, in that information propagates from an initial region outwards. In both processes, there is no global synchronisation, but each step can take place as soon as its inputs are ready. In the aTAM, the initial region is the seed, while in a circuit, it is the input bus together with the gates of in-degree 0.

There are three differences between these processes. First, the input bus of a circuit does not have an analog in the aTAM. Hence, aTAM systems are akin to *closed* circuits. Secondly, there can be competition between attachments in the aTAM, as well as mismatches, both of which are ruled out in cricuits. This entails that an aTAM derived from a circuit by the compilation process detailed below will not exhibit mismatches or concurrent attachments. The last and most important difference is that in a circuit, the outputs of each gate depends not only on its input, but also on the function of the gate. For an aTAM system to simulate a circuit, this information needs to be derived from the inputs of the gate. Self-description captures the possibility to do so.

▶ **Definition 24** (Self-describing circuit). *A normal circuit C on alphabet Σ is* self-describing *on input vector $\vec{\imath}$ if there is a decoding function* dec-gate : $(\Sigma \times \{N, E, S, W\})^{<4} \to \text{Gates}(\Sigma)$ *such that for any position p, with incoming arcs $(e_0, \ldots, e_{k-1}) = p + (C(p) \cdot \text{wiring} \cdot \text{Inputs})$:*

$$\text{dec-gate}((\widetilde{C}(\vec{\imath}, e_0), d_0), \ldots, (\widetilde{C}(\vec{\imath}, e_{k-1}), d_{k-1})) = C(p),$$

where d_j is the direction of e_j.

A closed self-describing circuit really is a circuit with only one function of each in-degree. Having one function per in-degree is a harmless technicality: in a circuit, successive gates have compatible wirings, so the in-degree of each gate is known from the wiring of any of its predecessor gates, whether or not the circuit is self-describing.

▶ **Lemma 25.** *Let C be a closed self-describing circuit. There is a closed circuit C_1 with only only one function of each in-degree such that $\widetilde{C} = \widetilde{C_1}$.*

Proof. C_1 has the same wirings as C, and for each i, all of its gates of in-degree i have as function $f_i : x_0 \ldots x_{i-1} \mapsto (\delta_C(x_0, \ldots x_{i-1}) \cdot \text{func})(x_0, \ldots x_{i-1})$, where δ_C is the decoding function of C. ◀

3.3 Locally deterministic patterns

If C is a self-describing closed circuit, then from $\widetilde{C}$, one gets a coloring of the arcs of its dependency graph with the property of *local determinism*.

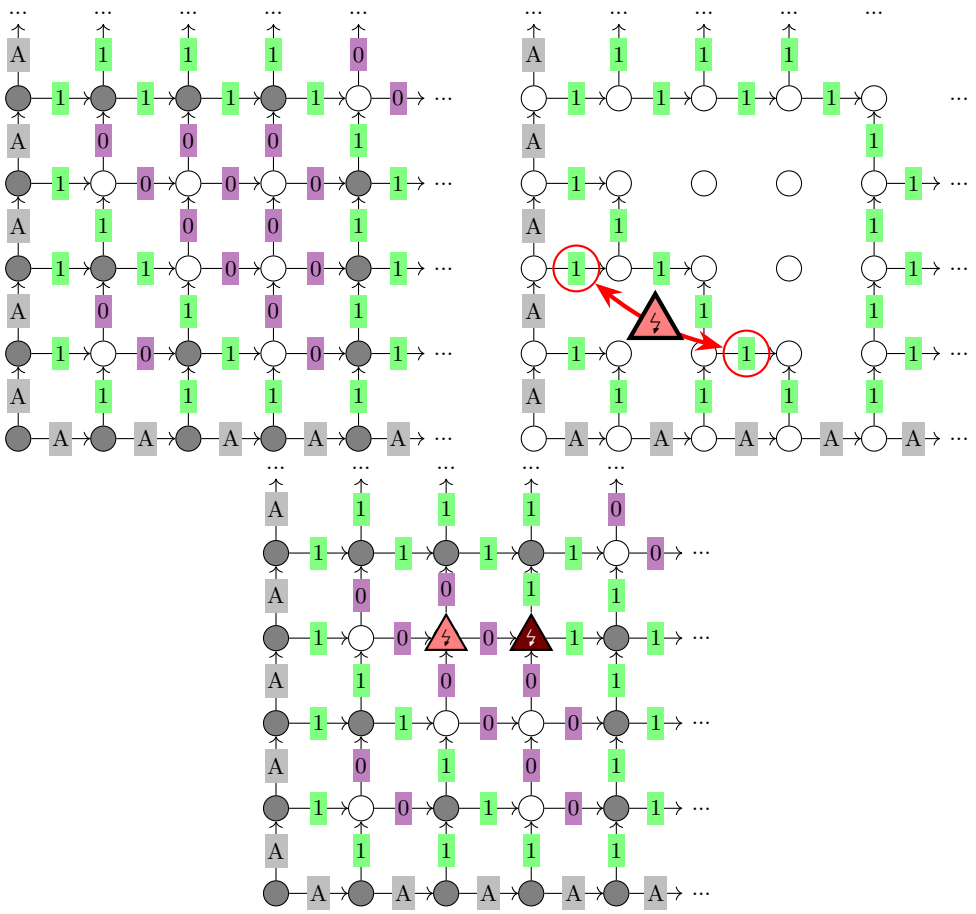


Figure 6 Three arc colorings on the alphabet $\Sigma = \{A, 0, 1\}$. The coloring C_1, in the upper left is locally deterministic. In the upper-right, C_2 is not because the two circled arcs have value 1 and direction E, but the vertices they point into have different incoming directions ($\{E\}$ versus $\{N, E\}$). At the bottom, C_3 is not locally deterministic, because the two triangle vertices get the same incoming values, but have different outputs: $(0, 0)$ versus $(1, 1)$.

▶ **Definition 26** (Locally deterministic arc coloring). *Let P be a coloring of the arcs of some acyclic oriented subgraph G of $\mathbb{Z}^2$. For each vertex $v \in G$ with in-degree δ, note $\vec{i}_v = (d_0, P(e_0)), \ldots, (d_{\delta-1}, P(e_{\delta-1}))$ for the input vector of v, where e_i is the i-th incoming edge of v and d_i its direction*

P is locally deterministic if there are two prediction functions pred-inputs : $\{N, E, S, W\} \times \Sigma \to \mathcal{P}(Dir)$ *and* pred-symbols : $(\{N, E, S, W\} \times \Sigma)^{<4} \times \{N, E, S, W\} \to \Sigma$ *such that for each arc e from vertex a to vertex b in direction $d \in \{N, E, S, W\}$,*

- $P(e) =$ pred-symbols$(\vec{i}_a, d)$
- pred-inputs$(d, P(e))$ *is the set of directions of the incoming arcs of b,*

▶ **Lemma 27.** *Let C be a closed self-describing circuit on alphabet Σ, with dependency graph $D_C = (V, A)$. Let C' be the self-describing circuit on alphabet $\Sigma \times \mathcal{W}$ where each gate outputs its wiring in addition to its normal output. Then the function $\widetilde{C'}$ is a locally deterministic coloring of D_C.*

Proof. A candidate function for pred-inputs takes as input the direction d of an arc $e : a \to b$,

and the value of $\widetilde{C'}(e)$, that is, an element of Σ corresponding to $\widetilde{C}(e)$ as well as the wiring w of the gate at a. Thus, the function $(d, x, w) \mapsto w \cdot \text{outwires}(d)$ must, by the constraints on neighboring gates in a circuit, output the set of directions of the incoming arcs at b.

Likewise, a candidate function for pred-symbols takes as input the direction of the arc $e : a \to b$, and a vector $\vec{m} \otimes \vec{d}$ of the inputs coming into a with their directions. Reorder $\vec{m} \otimes \vec{d}$ so that $\vec{d}$ is in the order of the inputs of $C(a)$. Let $\{\text{wiring} : w, \text{func} : f\} = \text{dec-gate}(\vec{m} \otimes \vec{d})$. Then taking pred-symbols$(e) = (f(\vec{m}), w)$ works. ◀

From locally deterministic patternts to aTAM systems

▶ **Definition 28** (vertex type, atlas)**.** *Let P be a coloring of the arcs of some acyclic oriented subgraph G of $\mathbb{Z}^2$. For a vertex v of G, its* vertex type *is the partial function $\bar{v} : d \in \{N, E, S, W\} \mapsto (o, x)$, where $o = -d$ if the edge e in direction d from v is an incoming edge, $o = d$ if it is an outgoing edge, and $x \in \Sigma$ is $P(e)$. If v has no edge in direction d, then $\bar{v}(d)$ is undefined.*

The atlas *of P is the set of its vertex types.*

A locally deterministic coloring can be reconstructed from its atlas and the positions of its in-degree 0 vertices.

▶ **Lemma 29.** *Let P_1, P_2 be locally deterministic colorings of the arcs of some oriented subgraphs of $\mathbb{Z}^2$, G_1 and G_2 respectively. If*
(1) *G_1 and G_2 are acyclic and have no infinite backwards paths,*
(2) *P_1 and P_2 have the same atlas,*
(3) *G_1 and G_2 have the same vertices with in-degree 0,*
then $G_1 = G_2$ and $P_1 = P_2$.

Proof. By induction on the longest path to each position in G_1. ◀

Finally, a locally deterministic pattern with a unique in-degree 0 vertex can be self-assembled in the aTAM.

▶ **Lemma 30.** *Let Σ be some finite alphabet, and let P be a locally deterministic coloring of the arcs of some acyclic graph G with no infinite backwards path. Assume the maximal in-degree of a vertex in G is δ and that G has only one vertex with in-degree 0.*

Then there is an aTAM system S_P with temperature δ with P as its only final production.

Proof. Let A_P be the atlas of P, pred-inputs and pred-symbols the prediction functions of P. By convention, suppose pred-inputs returns an input vectors which is ordered clockwise, starting from direction N.

First define the glues of S_p and their strength function st. Each glue is a pair $(s, d) \in \Sigma \times \{N, E, S, W\}$. For any symbol $s \in \Sigma$ and direction $d \in \{N, E, S, W\}$, let $\vec{i} = \text{pred-inputs}(s, d)$. If d is the first element of $\vec{i}$, then its strength is $\text{st}((s, d)) = (\delta + 1 - |\vec{i}|)$, otherwise it is 1. Hence, for any vertex type $v \in A_P$ with input vector $\vec{i}_v$, $\sum_{g \in \vec{i}_v} \text{st}(g) = \delta$.

Then define the set of tile types S_p to be A_p, with the glue on the d side of the tile type $\bar{v}$ being $\bar{v}(d)$ if defined, and the null glue otherwise. The seed tile of S_p is the only vertex type of A_p with in-degree 0.

Indeed, any production of S_P is an initial subset of P, as can be seen by induction on the attachments.

Then, consider a production p of S_P. If G contains some position not covered by p, then because G contains neither loops nor infinite paths, it must contain some v with all

its predecessors within $\mathrm{dom}(p)$. But then the total strength of glues into v is δ, and the tile corresponding to $P(v)$ must be attachable there. Hence p is not terminal.

Finally, S_P assembles P. ◀

Putting all this together, it is possible to compile a self-describing circuit into a self-assembling system.

▶ **Theorem 31.** *Let C be a self-describing normal circuit with only one gate without inputs. Then there is an aTAM system S_C which strictly self-assembles* $\mathrm{dom}(C)$.

Proof. By lemmas 27 and 30. ◀

4 A Self-Describing Embedded Circuit for the Sierpinski Cacarpet

The next step is to define a self-describing circuit on the Sierpinski Cacarpet.

▶ **Theorem 32.** *There is an evaluable circuit $C_\square$ with domain K_∞ and with an empty input bus which is self describing. Moreover, $C_\square$ has only one gate without inputs.*

The remainder of this section is the description of $C_\square$. This circuit is built from two sets of messages: layer 1 messages in Σ_1 and layer 2 messages in Σ_2, and three fractal structures: a wiring layer $W_\square : K^\infty \to \mathcal{W}$, and two function layers, F_1 operating on Σ_1 and F_2 operating on Σ_2.

The wiring layer $C_w : K^\infty \to \mathcal{W}$ is the fixed point of a substitution $\kappa_w : \mathcal{W} \to \mathcal{W}^K$ starting from a seed wiring s_w:

$$\begin{cases} C_w(\vec{0}) = s_w \\ C_w(\vec{z}) = \kappa_w(C_w\lfloor \frac{z}{K} \rfloor)(z \bmod K) \end{cases}$$

The definition of the function layers is a bit more indirect: there are two sets of labels L_1 and L_2 respectively, with two rules $\kappa_1 : \mathcal{W} \to L_1^K$ and $\kappa_2 : \mathcal{W} \times L_2 \to L_2^K$. The rule κ_2 takes the form of a substitution.

Their fixpoints, starting from two seed labels $\mathfrak{s}_1$ and $\mathfrak{s}_2$ define two tiling of K^∞ with labels of L_1 and L_2 respectively: Λ_1 and Λ_2. On layer 1, Λ_1 is defined for each position according to the wiring of its parent:

$$\Lambda_1(\vec{z}) = \kappa_1(C_w(\lfloor \frac{z}{K} \rfloor))(z \bmod K)$$

On layer 2, Λ_2 is defined for each position according to the wiring *and label* of its parent in a substitutive manner:

$$\begin{cases} \Lambda_2(\vec{0}) = \mathfrak{s}_2 \\ \Lambda_2(\vec{z}) = \kappa_2(C_w(\lfloor \frac{z}{K} \rfloor), C_2(\lfloor \frac{z}{K} \rfloor))(z \bmod K) \end{cases}.$$

Finally, the actual gate functions are defined from two functions, $\mathrm{instantiate}_1$ on layer 1 and $\mathrm{instantiate}_2$ on layer 2, which define a function on Σ_1 (respectively Σ_2) from four elements, two wirings and two labels in L_1 (respectively L_2), one each for the gate and its parent. Together with the substitutions κ_i, they define a circuit $\mu_i(w, l)$ with domain K known as the layer-i meta-gate obtained by w and l:

$$\begin{cases} \mu_1(w,l) : K \to \mathrm{Gates}(\Sigma_1) \\ \mu_1(w,l) : \vec{z} \mapsto \{gatewiring : \kappa_w(w)(\vec{z}),\ \mathrm{func} : \mathrm{instantiate}_1(w, l, \kappa_w(w)(\vec{z}), \kappa_1(w)(\vec{z}))\} \\ \mu_2(w,l) : K \to \mathrm{Gates}(\Sigma_2) \\ \mu_2(w,l) : \vec{z} \mapsto \{gatewiring : \kappa_w(w)(\vec{z}),\ \mathrm{func} : \mathrm{instantiate}_2(w, l, \kappa_w(w)(\vec{z}), \kappa_2(w,l)(\vec{z}))\}. \end{cases}$$

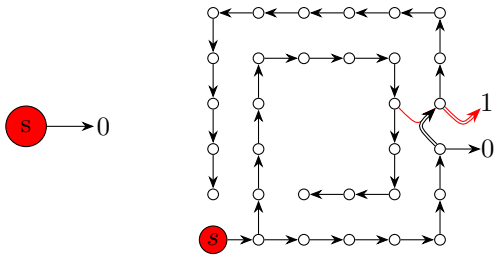


Figure 7 $\kappa_1(w_s, \mathfrak{seed})$, the first iteration of κ_1 on the seed gate. The wirings are given graphically; all gates have the $\mathfrak{normal}$ label, except the s at $(0,0)$ which has label $\mathfrak{seed}$. For the wiring with two inputs, the fat arrow represents the input number 0; all output wires have number 0: $w\cdot\text{output-num}(d) = 0$

In this circuit, the wiring of each gate is given by $\kappa_w(w)$, and its function is given by instantiating the label given by κ_i. When iterating this process, a gate g appearing at position $\vec{z}$ in some parent meta-gate $\mu_i(l, w)$, the child meta-gate $\mu_i(g)$ associated with g is defined as $\mu_i(g\cdot\text{wiring}, \kappa_i(l, w))$. The functions instantiate$_i$ are each injective in their last argument, ensuring this does not create any ambiguity.

Finally, these meta-gates beget the two layers of circuits C_1 and C_2 by:

$$\begin{cases} C_i(\vec{0}) = \{\,\text{wiring} : s_w,\ \text{func} : s_i\} \\ C_i(\vec{z}) = \mu_i(C_i(\lfloor \frac{\vec{z}}{K} \rfloor))(\vec{z} \bmod K) \end{cases}.$$

By this definition, the wirings of $C_1(\vec{z})$ and $C_2(\vec{z})$ are the same —they are given by C_w, so $C_\square : \vec{z} \mapsto C_1(\vec{z}) \otimes C_2(\vec{z})$ defines a circuit with alphabet $\Sigma_1 \times \Sigma_2$ and domain K^∞.

The next subsections are the description of C_w, followed by those of C_1, then C_2, and finally the proof of that $C_\square$ is self-describing.

4.1 The wiring layer

The seed wiring s_w is represented on the left of figure 7. On the right of the same figure is its image $\kappa_w(s_z)$.

For any wiring w, $\kappa_1(w)$ will only contain wirings with at most two inputs sides, and these input sides are adjacent. Thus, all gates in C_w have at most two input sides and they are adjacent. Hence, κ_w only needs to be defined on wirings with that property.

The definition of κ_w is isotropic: κ_w commutes with a rotation of $\pi/2$ and with reflections. This property will be upheld merely by giving the definition of κ_w *up to rotation and reflection.*

Lastly, for each input wire in w there are two wires in the input bus of $\kappa_w(w)$, and for each output wire in w there are two wires in the output bus of $\kappa_w(w)$. The position of these wires are as follows, up to rotation:

input of w		input i of $\kappa_w(w)$		
$w\cdot$ Inputs	direction	position	direction	$i\cdot$ Inputs
(W)	W	$(0,2)$	W	(W)
		$(0,3)$	W	(S,W)
(S,W)	W	$(0,0)$	W	(S,W)
		$(0,1)$	W	(S,W)
	S	$(0,0)$	S	(S,W)
		$(1,0)$	S	(S,W)

output of g		output o of $\kappa_w(g)$		
$w\cdot$ outwires(d)	direction d	position	direction d'	$i\cdot$ outwires(d')
(W)	E	$(5,2)$	E	W
		$(5,3)$	E	(S,W)
(S,W)	E	$(5,0)$	E	(S,W)
		$(5,1)$	E	(S,W)
(S,W)	N	$(0,5)$	N	(S,W)
		$(1,5)$	N	(S,W)

Thus, if w has $w\cdot$ Inputs $= (W)$, $w\cdot$ Outputs $= \{N, E, S\}$ with $w\cdot$ outwires$(N) = nW$, $w\cdot$ outwires$(E) = sE$ and $w\cdot$ outwires$(S) = N$, then $\kappa_w(w)$ has the corresponding input and output busses:

$$\begin{cases} \vec{I} = (W@(0,2), Ws@(0,3)) \\ \vec{O} = (Nw@(0,5), Nw@(1,5), Es@(5,0), Es@(5,1), S@(2,0), Sw@(3,0)) \end{cases}$$

Wirings with no inputs

All wirings without input are sent by κ_w to the array of wirings represented on figure 7. The seed wiring s_w is the only one to effectively appear when iterating κ_w.

Wirings with one input

A wiring with one input is cut according to figure 9.

The wirings in $\kappa_w(g)$ depend on the input and outputs of w. Since w has only one input, it is sufficient, up to rotation, to examine the case where $w\cdot$ Inputs $= \{W\}$. The black arrows of figure 9 depict the case where $w\cdot$ Outputs $= \emptyset$. For each $d \in w\cdot$ Outputs, some extra wires are needed. The additional wires for an output in the N direction depend on whether the directions appearing in $w\cdot$ outwires(N) are $\{S\}$, $\{S, E\}$ or $\{S, W\}$. The wires for the other output directions are derived from these by rotation. Figure 9 has one direction with each case, showing the complete range of possibilities for the extra wires; they are represented in dotted lines on the figure.

Wirings with two (adjacent) inputs

The image of such a wiring w by κ_w is defined on figure 10, which is rotated and reflected so that the S side of the figure is mapped to the 0 input of w, and the W side to the 1 input of w.

▶ **Lemma 33.** *For each gate w, $\kappa_w(w)$ is the wiring of a normal circuit. Moreover, C_w is the wiring of a normal, closed circuit.*

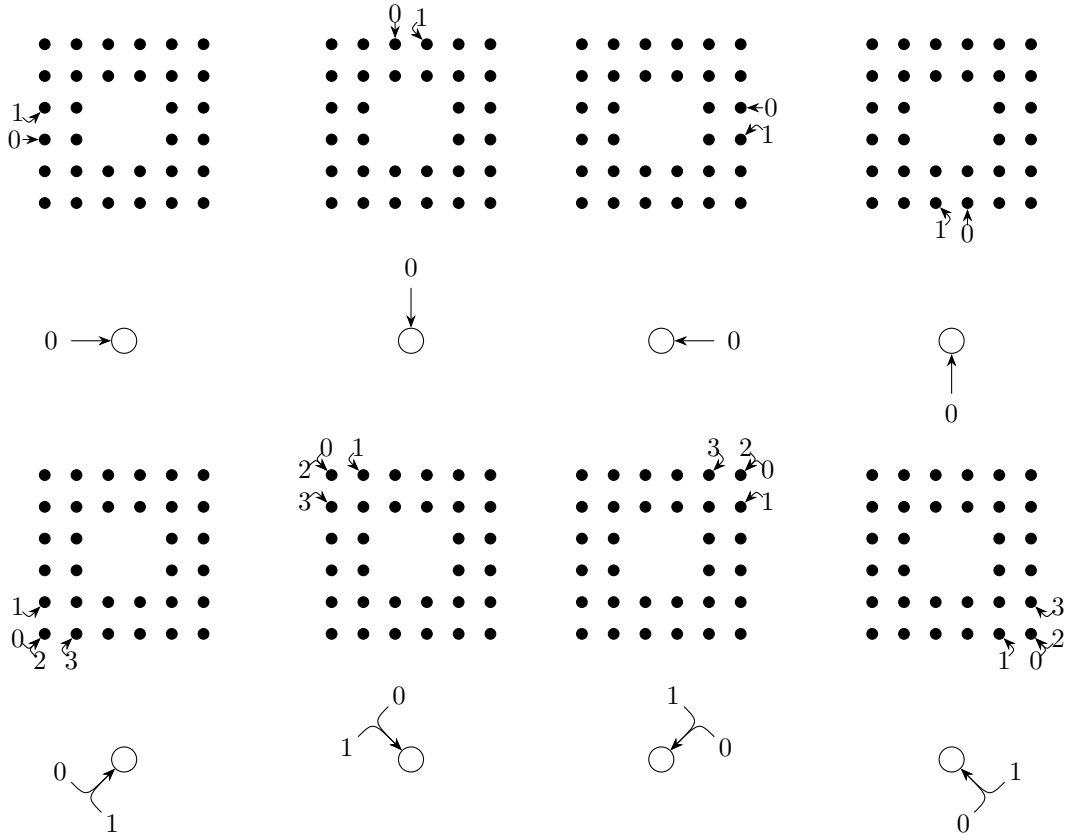


Figure 8 Location of the input wires in $\kappa_1(g)$ according to the input sides of $w =$ wiring g. The output wires mirror the input sides of the neighboring tiles thanks to $w\cdot$ outwires.

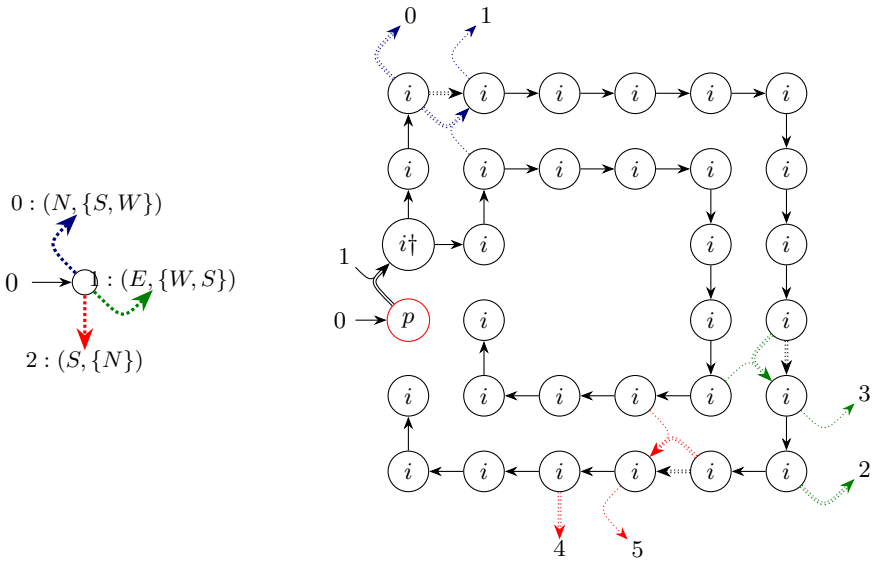


Figure 9 $\kappa_1(w, l)$ when $w \cdot \text{Inputs} = (W)$. The wirings are given graphically; on layer 1, all gates have the normal label (black), except for the gate at $(0, 3)$ which has label input (red). Dotted segments are parts of the wiring only if the corresponding element is in the outputs of w. For wirings with two inputs, the fat arrow represents the first input. The † marks the potential position for label $\mathfrak{dA}$ on layer 2.

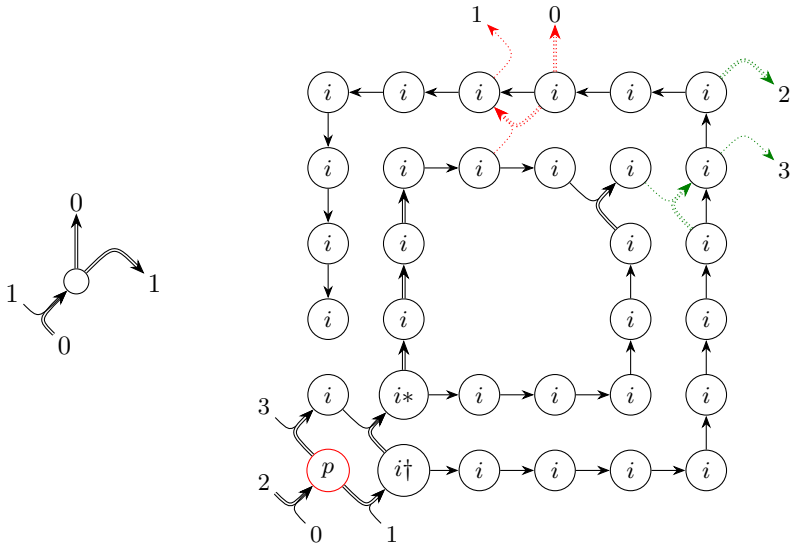


Figure 10 $\kappa_1(w, l)$ when $\text{Inputs}\, w = (S, W)$. The wirings are given graphically; all gates have the normal label except for the input gate at $(0, 0)$ which has label input. This configuration gets rotated and reflected according to the inputs of the gate, so that the arrows on the outer ring form a path from the side with input 0 to the side with input 1, "the long way around". The '*' marks the *special* position on layer 2, and the † marks the position where label $\mathfrak{dA}$ may appear.

Proof. This follows from observation of the schemata describing each $\kappa_w(w)$ and observing that for two wirings w, w', if an output side in direction d of w matches an input side in direction $-d$ of w', then the corresponding sides of $\kappa_w(w)$ and $\kappa_w(w')$ also match.

Additionally, each iteration of κ_w on s_w is without inputs, so C_w is closed. ◀

4.2 Layer 1

The components of layer 1 are an alphabet Σ_1, a finite set of labels L_1, a substitution $\kappa_1 : \mathcal{W} \mapsto L_1^K$, and for each label $l \in L_1$, and an instantiation function instantiate$_1$. Layer 1 can then be realized as described above into a circuit with the wirings of C_w and the functions given by instantiating the fixpoint of κ_1.

The set of labels is $L_1 = \{\mathfrak{s}_1, \mathfrak{normal}, \mathfrak{input}\}$.

The elements of Σ_1 are *layer 1 messages*.

▶ **Definition 34** (Layer 1 Message). *A layer 1 message is a triple* $\{\,\text{pos} = \vec{z} \in K_1,\ \text{parent-wiring} = w \in \mathcal{W}, \cdot\,\text{parent} = l \in L_1\}$.

The circuit C_1 is going to be defined by iterating κ_w and κ_1 starting from s_w and $\mathfrak{s}_1$, then instantiating the labels.

Labels

The label substitution $\kappa_1 : \mathcal{W} \times L_1$ assigns the label $\mathfrak{normal}$ at all positions except:

- $\kappa_1(s_w)(\vec{0}) = \mathfrak{s}_1$
- if w has at least one input, then $\kappa_1(w)(\vec{z}) = \mathfrak{input}$ for the position $\vec{z}$ which receives the input wire 0 in $\kappa_w(w)$.

The seed function

The starting label is $\mathfrak{s}_1$, it only ever appears at position $(0,0)$ in $\kappa_1(s_w, \mathfrak{s}_1)$.

Since s_1 has no inputs, its associated function $f_s = \text{instantiate}_1(s_w, \mathfrak{s}_1, s_w, \mathfrak{s}_1)$ is a constant function with value $f_s() = \{\,\text{pos} : (0,0),\ \text{parent-label} : \mathfrak{s}_1,\ \text{parent-wiring} : s_w\}$.

Gate functions for layer 1

There are two types of functions for the gates output by $\kappa_1(g)$ other than the seed. They have a either an *increment* function $\text{incr}[k, D]$ with $k \in \{1, 2\}$ and $D \in \{N, E, S, W\}$ or a *reparenting* function $\text{set-parent}[k, D, w, c]$, with $k \in \{1, 2\}$, $D \in \{N, E, S, W\}$, w a wiring and $c \in C$ a color. The versions with $k = 2$ take two inputs but ignore the second one. The functions incr and set-parent are defined as follows:

$$
\begin{aligned}
\text{incr}[1, D](m) \cdot \text{parent} &= m \cdot \text{parent}, \\
\text{incr}[1, D](m) \cdot \text{pos} &= \text{pos}\, m + D \\
\text{set-parent}[1, D, w, c](m) \cdot \text{parent} &= (w, c) \\
\text{set-parent}[1, D, w, c](m) \cdot \text{pos} &= m \cdot \text{pos} + D \\
\text{incr}[2, D](m, m') &= \text{incr}[1, D](m) \\
\text{set-parent}[2, D, w, c](m, m') &= \text{set-parent}[1, D, w, c](m).
\end{aligned}
$$

The function of each gate is fixed from its label and wiring, and those of its parent through instantiate$_1$ as follows:

$$\begin{cases} \text{instantiate}_1(w_p, l_p, w, \texttt{normal}) = \text{incr}[k, D] \\ \text{instantiate}_1(w_p, l_p, w, \texttt{input}) = \text{set-parent}[k, D, w_p, l_p], \end{cases}$$

where k is the number of inputs of w, and D is the direction of its first input.

Behavior and Self-Description of Layer 1

The circuit C_1 obtained from κ_1 is normal, by lemma 33. Let $e = \widetilde{C_1} : K^\infty \times \{N, E, S, W\} \to \Sigma_l$ be the evaluation function of C_1. That function e enjoys a simple description, which reflects the fact that in C_1, the different meta-gates do not actually communicate. On layer 2 however, there will be some communication between meta-gates, as described in the next section.

▶ **Lemma 35.** *Let $w \in \mathcal{W}$, $l \in L_1$ and $a : p \mapsto p'$ be an internal or outgoing arc in $\kappa_w(w)$. If w has no inputs, assume $l = \mathfrak{s}_1$.*

For any input $\vec{\imath}$ of $\mu_1(w, l)$, let $m = \widetilde{\mu_1(w, l)}(\vec{\imath}, a)$ be the value of arc a on input $\vec{\imath}$. Then $m{\cdot}\,\text{pos} = p$, $m{\cdot}\,\text{parent-wiring} = w$ and $\text{parent-label} = l$.

Proof. By induction on the non-incoming arcs of the (acyclic) dependency graph D of $\kappa_w(w)$.

If w is the seed wiring s_w, then the root of D is also s_1 and its outputs satisfy $\text{pos}\, e_g(a) = (0, 0)$, $\text{parent-wiring}\, e_g(a) = s_1{\cdot}\,\text{wiring}$ and $\text{parent-label}\, e_g(a) = \mathfrak{s}_1$.

Otherwise, $\kappa_1(w)$ has a unique position $\vec{z_i}$ with label $\texttt{input}$, corresponding to a gate in $\mu_1(l, w)$ with function set-parent$[k, D, w, l]$, for some k and D. By definition of set-parent, its outputs satisfy $e_g(a){\cdot}\,\text{pos} = \vec{z_i}$, $e_g(a){\cdot}\,\text{parent-wiring} = w$ and $e_g(a){\cdot}\,\text{parent-label} = l$.

In both cases, each other gate g' at position $\vec{z}$ of $\mu_1(l, w)$ has function incr$[k, D]$, where k is the number of inputs of g', and D is the direction of its first input arc $i = \vec{z} - D \overset{D}{\mapsto} z$. By induction, $e_g(i){\cdot}\,\text{pos} = z - D$ and $e_g(i) \cdot \text{parent} = (g{\cdot}\,\text{wiring}, \text{label}(g))$. By definition of incr$[k, D]$, each of the output arcs o of g' verify $e_g(o){\cdot}\,\text{pos} = (\vec{z} - D) + D = \vec{z}$ and $e_g(a){\cdot}\,\text{parent-wiring} = w$ and $e_g(a){\cdot}\,\text{parent-label} = l$. ◀

Additionally, C_1 is "mostly self-describing": the first input of a gate g is enough to recover g, except for the value of w and c in gates with function set-parent$[k, D, w, c]$.

▶ **Lemma 36.** *For a position $p \in K^\infty$, let $\vec{e} = \text{Inputs}\, C_1(p){\cdot}\,\text{wiring}$ be the input arcs of $C_1(p)$; let d_i be the direction of e_i.*

There is a function dec-gate $: \Sigma_1 \times \{N, E, S, W\} \to \text{Gates}(\Sigma_1) \cup \bot$ *such that for all $p \in K^\infty$,*

- *if $C_1(p){\cdot}\,\text{func}$ is $\text{incr}[k, d_0]$, then $\text{dec-gate}(\widetilde{C_1}(e_0), d_0) = C_1(p)$*
- *if $C_1(p){\cdot}\,\text{func}$ is $\text{set-parent}[k, d_0, -, -]$ then $\text{dec-gate}(\widetilde{C_1}(e_0), d_0) = \bot$.*

Proof. The function dec-gate is defined as follows: let $m \in \Sigma_1$ be a local message, and $d \in \{N, E, S, W\}$. Let $p = m{\cdot}\,\text{pos}$, if $p - d \notin \{0, \dots, 5\}^2$, then $\text{dec-gate}(m, d) = \bot$. Otherwise, $\text{dec-gate}(m, d)$ is the gate at position p in $\mu_1(m{\cdot}\,\text{parent-wiring}, m \cdot \text{parent})$.

By lemma 35, dec-gate satisfies the lemma. ◀

Moreover, when $\text{dec-gate}(e(e_O), d_0) = \bot$, g itself cannot be determined, but its label and wiring can, as well as $\mu_1(g)$.

▶ **Corollary 37.** *For a position $p \in K^\infty$, let $\vec{e} = C_1(p)\cdot\text{Inputs}\cdot\text{wiring}$ be the input arcs of $C_1(p)$; let d_i be the direction of e_i,*

- *there is a function* $\text{dec-gate}_w : \Sigma_1 \times \{N, E, S, W\} \to \mathcal{W}$ *such that for each position* $p \in K^\infty$, $\text{dec-gate}_w(\widetilde{C_1}(e_0), d_0) = C_1(p)\cdot\text{wiring}$
- *there is a function* $\text{dec-gate}_l : \Sigma_1 \times \{N, E, S, W\} \to \mathcal{W}$ *such that for each position* $p \in K^\infty$, $\text{dec-gate}_l(\widetilde{C_1}(e_0), d_0) = \Lambda_1(p)$

Proof. For dec-gate_w and dec-gate_l, it suffices to observe that whenever $\text{dec-gate}(\widetilde{C_1}(e_0), d_0) = \bot$ at some position p, $\widetilde{C_1}(p)$ has label $\mathtt{input}$, and its wiring only depends on its position within its metagate. ◀

4.3 Layer 2

A second layer is needed in order to get full self-description of the circuit on K^∞. This layer routes global information between the meta-gates so that the input gate of each meta-gate can be indentified by its incoming global message. The construction needs to "tie the knot", so it is not only needed to recover the identity of the input gate on the layer 1, but also on layer 2 itself.

This layer is defined by a set L_2 of labels, an alphabet Σ_2, the label substitution κ_2 and its instantiation function. In contrast with layer 1, κ_2 takes as input a wiring, as well as a label. This makes the definition of Λ_2 recursive. In contrast with layer 1, on layer 2, the two-step definition of gates using labels and the function instantiate is actually *needed* because of a subtelty related to the tying of the knot. The functions of some gates make use of κ_2. Thus κ_2 shall not directly manipulate the gates or their function, lest the definition of layer 2 becomes cyclical and possibly ill-founded.

The set of layer 2 labels is $L_2 = \{\mathfrak{s}, \mathfrak{i}_1, \mathfrak{i}_2, \mathfrak{dL}, \mathfrak{dR}\}$.

Layer 2 messages

A message on layer 2 is an element of Σ_2; it is built from:

- $m\cdot\text{parent-label}_2 \in L_2$,
- $m\cdot\text{global}$, itself consisting of:
  - $m\cdot\text{global}\cdot\text{label}_2 \in L_2$.
  - $m\cdot\text{global}\cdot\text{ancestor-msg} \in \Sigma_1$

These messages make C_2 self-describing by completing the information available in layer 1 and used in lemma 36. A message m_2 output by a gate g_2 in $\kappa_2(g_2')$ identifies g_2' through $m_2\cdot\text{parent-label}_2$ if g_2 is not the input gate of $\kappa_2(g_2')$, as in lemma 36. The global part $m_2\cdot\text{global}$ identifies *some ancestor* of g_2, on layer 1 through $m\cdot\text{global}\cdot\text{ancestor-msg}$ and on layer 2 through $m\cdot\text{global}\cdot\text{label}_2$. This global information will enable the determination of the entry gate of each meta-gate, on both layers. From two messages $m_l, m_g \in \Sigma_2$, two messages $(m_1, m_2) = \text{extract}(m_l, m_g) \in \Sigma_1 \times \Sigma_2$ can be extracted by reading the local information from m_l, and the global information from m_g, as follows:

$$\begin{cases} m_1 = m_l\cdot\text{ancestor-msg} \\ m_2\cdot\text{parent-label}_2 = m_l\cdot\text{global}\cdot\text{label}_2 \\ m_2\cdot\text{global} = m_g\cdot\text{global}\,. \end{cases}$$

The converse operation, embedding, takes as input three messages, a payload $(p_1, p_2) \in (\Sigma_1 \times \Sigma_2)$ and a context $c \in \Sigma_2$, and yields two messages m_l and m_g

$$\begin{cases} m_l \cdot \text{parent-label}_2 = m_g \cdot \text{parent-label}_2 = c \cdot \text{parent-label}_2 \\ m_l \cdot \text{global} \cdot \text{label}_2 = p \cdot \text{parent-label}_2 \,. \\ m_l \cdot \text{global} \cdot \text{ancestor-msg} = p_1 \\ m_g \cdot \text{global} = p \cdot \text{global} \,. \end{cases}$$

Extracting and embedding are dual operations, in the sense that

$$\forall c, p_1, p_2, \text{extract} \circ \text{embed}(c, p_1, p_2) = (p_1, p_2).$$

The substitution κ_2

Given a wiring w and a label $l \in L_2$, the substitution κ_2 yields a label for each position in K;

- $\mathfrak{s}_2$ for the seed gate, i.e. for position $(0, 0)$ if w has no inputs;
- $\mathfrak{dL}$ for the gate receiving the input number 0 of the meta-gate;
- $\mathfrak{dA}$ for the gate receiving the input number 1 of the meta-gate whenever $l = \mathfrak{dL}$;
- when w has two inputs, there is a *special* position within the meta-gate where the value depends on l as follows:
  - $\mathfrak{i}_2$ if $l \in \{\mathfrak{i}_2, \mathfrak{dL}\}$,
  - $\mathfrak{dA}$ if $l = \mathfrak{dA}$;
- $\mathfrak{i}_1$ for gates with one input and $\mathfrak{i}_2$ for gates with two inputs otherwise.

The special position is marked on figures 9 and 10 by a star.

The layer-2 seed gate

The function $f_s^2 = \text{instantiate}_2(s_w, \mathfrak{s}_2, s_w, \mathfrak{s}_2)$ of the seed gate is a constant function with value $f_s^2() = \{\text{parent-label}_2 : \mathfrak{s}_2, \text{global} : \{\text{label}_2 : \mathfrak{s}_2, \text{ancestor-msg} : f_s^1()\}\}$. Recall that $f_s^1()$ is the message output by the seed gate on layer 1.

Gate functions for layer 2

The functions of the gates depend on their label and on the direction D of their first input, as dictated by the function instantiate:

$$\begin{cases} \text{instantiate}(D, \mathfrak{s}) = f_s^2 \\ \text{instantiate}(D, \mathfrak{i}_1) : x \mapsto x \\ \text{instantiate}(D, \mathfrak{i}_2) : (x, y) \mapsto (x, y) \\ \text{instantiate}(D, \mathfrak{dL}) = \text{decode}_L[D] \circ \text{incr}_G[D] \\ \text{instantiate}(D, \mathfrak{dA}) = (m_l, m_g) \mapsto \text{decode}_A[D](\text{incr}_G[D](m_l), m_g), \end{cases}$$

where D is the direction of the first input of w_p.

The gate with labels $\mathfrak{i}_1$ or $\mathfrak{i}_2$ are wires; their function is the identity function of arity 1 or 2 respectively.

Given $m \in \Sigma_2$, the increment functions $\text{incr}_G[D]$ increments ancestor-msg global $m \cdot$ pos by the unit vector of direction D.

The two decoding functions decode_L and decode_A are based on the function decode : $\{N, E, S, W\} \times \Sigma_1 \times \Sigma_2 \to \Sigma_1 \times \Sigma_2$ defined as follows: let $m_1 \in \Sigma_1, m_2 \in \Sigma_2$, pose

$z = m_1 \cdot \text{pos}$, $z_a = m_2 \cdot \text{global} \cdot \text{ancestor-msg} \cdot \text{pos}$, $a = m_2 \cdot \text{global} \cdot \text{ancestor-msg}$. Let $l_p = \kappa_2(a \cdot \text{parent-wiring}, a \cdot \text{parent-label}, m_2 \cdot \text{global} \cdot \text{label}_2, p')$ if $z \in K_1$, and $\mathfrak{d}\mathfrak{L}$ otherwise. Then $\text{decode}[D](m_1, m_2)$ is the pair (m'_1, m'_2) with:

$$m'_1 = \left\{ \begin{array}{ccc} \text{pos} & : & z \bmod K \\ \text{parent-wiring} & : & \text{dec-gate}_w(a, D) \\ \text{parent-label} & : & \text{dec-gate}_l(a, D) \} \end{array} \right\}$$

$$m'_2 = \left\{ \begin{array}{lcl} \text{parent-label}_2 & : & p \\ \text{global} \cdot \text{label}_2 & : & \text{global}\, m \cdot \text{label}_2 \\ \text{global} \cdot \text{ancestor-msg} & : & \left\{ \begin{array}{ccc} \text{parent-wiring} & : & m \cdot \text{global} \cdot \text{ancestor-msg} \cdot \text{parent-wiring} \\ \text{parent-label} & : & m \cdot \text{global} \cdot \text{ancestor-msg} \cdot \text{parent-label} \\ \text{pos} & : & z_a \bmod K_1 \end{array} \right\} \end{array} \right\}$$

For a direction D and a message $m_2 \in \Sigma_2$, take an arbitrary $m_1 \in \Sigma_1$ and let $(m'_1, m'_2) = \text{decode}[D](m_1, m_2)$; the value of m'_2 does not depend on m_1, so $\text{decode}_L[D](m_2)$ is defined to be the $m'_2 \in \Sigma$ returned by $\text{decode}(m_1, m_2)$ for any m_1.

For a direction D and $m_l, m_g \in \Sigma_2$, $\text{decode}_A[D](m_l, m_g)$ is defined as follows. Let $(m_1, m_2) = \text{extract}(m_l, m_g)$, $(m'_1, m'_2) = \text{decode}[D](m_1, m_2)$. Then $\text{decode}_A[D](m_l, m_g)$ is the pair m'_l, m'_g with:

$$\begin{cases} m'_l \cdot \text{parent-label}_2 = m'_g \cdot \text{parent-label}_2 = m_l \cdot \text{parent-label}_2 \\ m'_l \cdot \text{global} \cdot \text{ancestor-msg} = m'_1 \\ m'_l \cdot \text{global} \cdot \text{label}_2 = m'_2 \cdot \text{parent-label}_2 \\ \text{global}\, m'_g = m' \cdot \text{global} \end{cases}$$

The function decode_A is engineered in order to enjoy the following property, a kind of commutation between decode and extract.

▶ **Lemma 38.** *For any direction $D \in \{N, E, S, W\}$,*

$$\text{decode}[D] \circ \text{extract} = \text{extract} \circ \text{decode}_A[D]$$

Proof. By computation. ◀

Properties of C_2

The messages passing through the circuit C_2 built above hold all the necessary information for $C_\square$ to be self-describing: in other words, C_2 carries all the information needed to determine the entry gate of each meta-gate in C_1, as well as the information needed to determine each of its gates.

The $\text{global} \cdot \text{ancestor-msg}$ part of the messages on input 0 each meta-gate simulate the gates of layer 1, as long as each meta-gate simulating a gate with label `input` receives on input 1 the message of its parent meta-gate.

▶ **Lemma 39.** *let $w \in \mathcal{W}$ with k inputs, $l_2 \in L_2$, and M be the circuit $\mu_2(w, l_2)$. Note that M has $2k$ inputs. Let D be the direction of the first input of w.*

Let $\vec{\imath} \in \Sigma^{2k}$, and let $\vec{o}$ be the output of M on input $\vec{\imath}$. Pose $i_a = i_0 \cdot \text{ancestor-msg} \cdot \text{global}$ and $i_p = i_1 \cdot \text{ancestor-msg} \cdot \text{global}$.

Then, if $g_1 = \text{dec-gate}(i_a, D) \neq \perp$, then o_0 is the output of g_1 on input i_a; else, if $\text{dec-gate}(i_a, D) = \perp$, then o_0 is the output of $\text{dec-gate}_c(i_p, D)(i_a \cdot \text{pos})$ on input i_a.

Proof. By computation. ◀

The rest of the global part of the messages allows C_2 to simulate itself.

▶ **Lemma 40.** *Let $D \in \{N, E, S, W\}$, $w \in \mathcal{W}$ with one input in direction D, $l_2 \in L_2$. Let $f = \text{instantiate}(D, l_2)$ and $C = \mu_2(w, l_2)$.*

Let $c \in \Sigma_1, i_1 \in \Sigma_1, i_2 \in \Sigma_2$ and $(m_l, m_g) = \text{embed}(c, i_1, i_2)$. Let (o_0, o_1) be the outputs of C on input (m_l, m_g). Then $\text{extract}(o_0, o_1) = f(i_2)$.

Proof. The proof proceeds by case on l_2, which can be either $\mathfrak{i}_1$ or $\mathfrak{dL}$. If $l_2 = \mathfrak{i}_1$, then f is the identity function, and it suffices to follow the wirings to check the result.

If $l_2 = \mathfrak{dL}$, then following the wirings reduces the desired equality to the definition of $\text{decode}_L[D]$.

◀

▶ **Lemma 41.** *Let $w \in \mathcal{W}$ with two inputs, $l_2 \in L_2$. Let $f = \text{instantiate}(l_2)$ and $C = \mu_2(w, l_2)$.*

For $k \in \{0, 1\}$, let $c^k \in \Sigma_1, i_1^k \in \Sigma_1, i_2^k \in \Sigma_2$ and $(m_l^k, m_g) = \text{embed}(c^k, i_1^k, i_2^k)$. Let $(o_0^0, o_1^0, o_0^1, o_1^1)$ be the outputs of C on input $(m_l^0, m_g^0, m_l^1, m_g^1)$. Then for $k \in \{0, 1\}$ $\text{extract}(o_0^k, o_1^k)$ is the k-th component of $f(i_2^0, i_2^1)$.

Proof. The proof proceeds by case on l_2. If $l_2 \in \{\mathfrak{s}, \mathfrak{i2}, \mathfrak{d}\mathfrak{A}\}$, following the wirings in $\kappa_2(w, l_1, l_2)$ confirms that the lemma holds.

If $l_2 = \mathfrak{dL}$, the lemma follows from lemma 38 by again following the wirings. ◀

Together, these properties entail a substitutive structure of the messages in $C_\square$. Going up the hierarchy, the message between two gates of $C_\square$ can be extracted from the messages between the corresponding meta-gates.

▶ **Lemma 42.** *Let $a = p \to p'$ be a wire between two positions p, p' of K^∞ in direction D. Let a_l, a_g be the two wires crossing the edges between pK and $p'K$ in clockwise order looking in direction D (i.e., if D is E, a_l is the northernmost of the two; if D is S, the westernmost...).*

Let $m = \widetilde{C_\square}(a)$, $l = (l_1, l_2) = \widetilde{C_\square}(a_l)$ and $g = (g_1, g_2) = \widetilde{C_\square}(a_g)$.

Then $\text{extract}(l_2, g_2) = m$.

Proof. By induction on p, following the wires of $C_\square$. ◀

▶ **Lemma 43.** *Let $\vec{p} \in K^\infty$. Let $g_1 = C_1(\vec{p})$, $g_2 = C_2(\vec{p})$, $g_1' = C_1(\lfloor \frac{\vec{p}}{K} \rfloor)$, $g_2' = C_2(\lfloor \frac{\vec{p}}{K} \rfloor)$. Let $g = g_1 \otimes g_2 = C_\square(\vec{p})$, and $g' = g_1' \otimes g_2' = C_\square(\lfloor \frac{\vec{p}}{K} \rfloor)$. Let e be an output wire of g, and $m_1 \times m_2 = \widetilde{C_\square}(e)$ its value in $C_\square$.*

Then $m_2 \cdot \text{parent-label}_2$ is the label of g_2', $m_1 \cdot \text{parent-label}$ is the label of g_1', $m_1 \cdot \text{parent-wiring}$ is $g' \cdot \text{wiring}$, and $m_1 \cdot \text{pos}$ is $\vec{p} \bmod K$.

Proof. By the previous lemma, input 0 of each meta-gate $\mu_1(l_1, w) \otimes \mu_2(l_2, w)$ encodes l_1, l_2 and w. The gate after that input has label $\mathfrak{dL}$, so by definition of its function decode_L, its output satisfies the lemma. The other gates in the meta-gate preserve the local part of the messages. ◀

This local information is just what is needed to reconstruct each gate from its *output*, which is just short of self-description.

▶ **Corollary 44.** *There are is a function $\text{dec-gate}' : \Sigma_1 \times \Sigma_2 \to \mathcal{W} \times L_1 \times L_2 \times K$ such that for any position $p \in K^\infty$ and any wire $a : p \to p'$ of $C_\square$,*

$$\text{dec-gate}'(\widetilde{C_\square}(a)) = (\Lambda_1(\lfloor p/K \rfloor), \Lambda_2(\lfloor p/K \rfloor), C_w(\lfloor p/K \rfloor), p \bmod K).$$

With a tiny bit of extra-work, each gate g can be reconstructed from its input number 0, making $C_\square$ self-descriptive.

▶ **Theorem 45.** *The circuit $C_\square$ is self-descriptive.*

Proof. Each gate g in position p can be reconstructed from its set of input directions and the message on its input number 0.

If g has no inputs, then $g = s_1 \otimes s_2$.

Otherwise, let D be the direction of its first input wire, and $m = (m_1, m_2)$ the message coming into its first wire.

Let $(w, l_1, l_2, p') = \text{dec-gate}'(m)$. Note that $p' = (p - D) \bmod K$. If $p' + D$ belongs to K, then $\lfloor \frac{p}{K} \rfloor = \left\lfloor \frac{p-D}{K} \right\rfloor$, so p belongs to the same meta-gate as its predecessor $p - D$, hence $C_\square(p)$ is $\mu_1(w, l_1)(p' + D) \otimes \mu_2(w, l_2)(p' + D)$.

Otherwise, since $p' + D \notin K$, it must be the case that $\lfloor \frac{p}{K} \rfloor = \left\lfloor \frac{p-D}{K} \right\rfloor + D$: p and its predecessor $p - D$ are in neighboring meta-gates. Let $a = \lfloor \frac{p}{K} \rfloor$ be the position of the parent gate. The position a mod k of a within its meta-gate is the position where input 0 enters that meta-gate.

A close examination of the values of $\mu_1(w, l_1)$ and $\mu_2(w, l_2)$ for all w, l_1, l_2 reveals that in each (non-seed) meta-gate the label and wiring of the gate in the position where input 0 comes only depends on the input directions of w. By lemma 43, those directions are exactly those of $m_2 \cdot \text{global} \cdot \text{ancestor-msg} \cdot \text{parent-wiring} \cdot \text{outwires}(D)$. ◀

This concludes the proof of Theorem 18. By theorem 31, this means that there is an aTAM system $S_\square$ which strictly self-assembles K^∞.

5 A Characterization of Admissible Generators

The previous construction can be adaptated to any self-similar discrete fractal within which the communication pattern used by $C_\square$ can be embedded. Whether a particular fractal is amenable to hosting such a communication pattern depends on the ability of its generator G to transport information to copies of itself around it.

▶ **Definition 46.** *Let G be a finite subset of $\mathbb{N}^2$, the grid $G^\#$ is the subset of $\mathbb{Z}^2$ defined by:*

$$G^\# = \{\ p \in \mathbb{Z}^2 | p \bmod G \in G\}$$

The grid neighborhood graph G^+ of G is the subgraph of $G^\#$ induced by the distance 1 neighborhood of G:

$$G^+ = G \cup \{p \in G^\# | \exists d \in \{N, E, S, W\}, p + d \in G\}.$$

For a direction $d \in \{N, E, S, W\}$, the d-port in G^+ is $G^{+d} = \{p \in G^\# | p - d \in G\}$.

For $d, d' \in \{N, E, S, W\}$, the (d, d')-bandwidth of G is the number $G[d \leftrightarrow d']$ of vertex-disjoint paths from G^{+d} to $G^{+d'}$ in G^+.

In order to compare generators, the classical notions of *subgraph* and *graph subdivision* is useful, accounting for marked vertices.

▶ **Definition 47** (Pointed subgraph)**.** *Let G, H be two graph, each with marked vertices. The graph H is a* pointed subgraph *of G if H is a subgraph of G in such a way that any marked vertex of H is mapped to a marked vertex of G. The graph G may have extra marked vertices.*

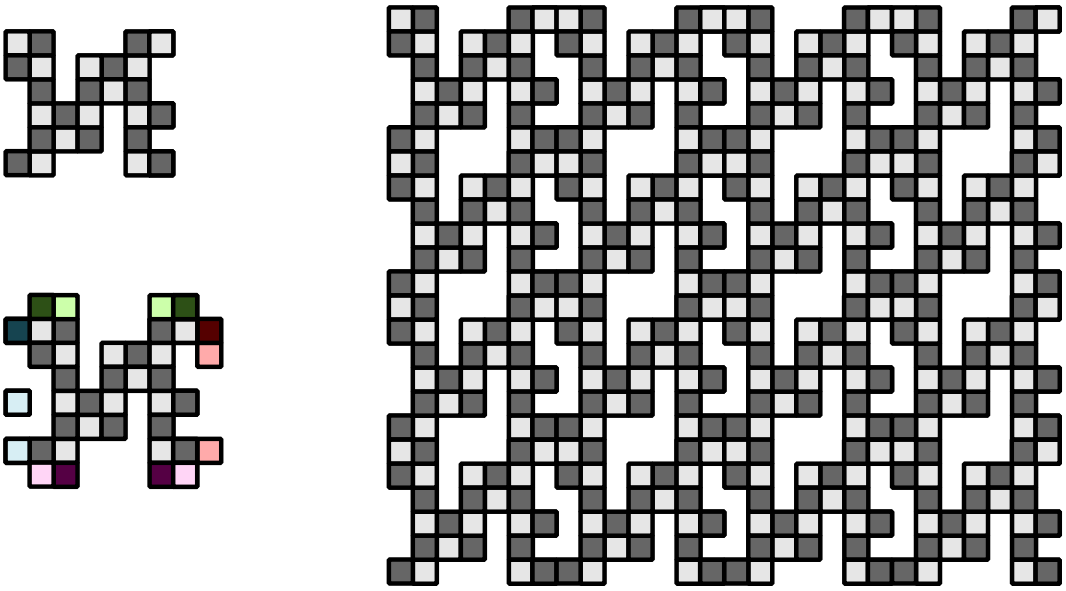


Figure 11 A finite shape G, its grid neighborhood G^+ and its grid graph $G^\#$.

▶ **Definition 48** (Edge subdivision). *Let $G = (V, E)$ be a graph, with some marked vertices $(v_0, \dots, v_{k-1}) \in V^k$. The edge subdivision operation for an edge $e = \{u, v\} \in E$ is the deletion of e from G and the addition of a new vertex $w \notin V$ and of the edges $\{u, w\}$ and $\{w, v\}$.*

This operation generates a new graph H, where the same vertices are marked as in G.

$$H = (V \cup \{w\}, (E \setminus \{u, v\}) \cap \{\{u, w\}, \{w, v\}\})$$

▶ **Definition 49** (Graph Subdivision). *A graph with marked vertices which has been derived from G by a sequence of edge subdivision operations is called a pointed subdivision of G.*

▶ **Definition 50** (Subconnector). *Let G, H be finite, connected shapes of $\mathbb{N}^2$, with $(0, 0) \in G \cap H$.*

Then H is a subconnector of G, written $H \preceq G$ if G^+ has a (pointed) subgraph which is a (pointed) subdivision of H^+.

The notion of subconnector is well-suited to the study of substitutions given the following properties.

▶ Remark 51. Let G, H, I be finite, connected shapes of $\mathbb{N}^2$, with $G \preceq H$, then:

$\sigma_I(G) \preceq \sigma_I(H)$
$\sigma_G(I) \preceq \sigma_H(I)$

▶ **Lemma 52.** *Let $G \ni (0, 0)$ a finite, connected subgraph of $\mathbb{N}^2$ such that $K \preceq G$. Then there is a self-descriptive circuit with domain G^∞.*

Proof. First, notice that κ_w can be completed to cover the case where w has two opposite input directions, as represented on figure 12 (modulo rotation and reflection).

Fix the vertices of G^+ which represent the vertices of K^+, the ones that sit in the middle of its edges, and the ones which are pending leaves.

TODO

Figure 12 The value of $\kappa_w(w)$ when $w\cdot\text{Inputs} = \{N, S\}$. This case is not needed in section 4, but it is necessary to generalize κ_w to G when K^+ is a subdivision of G^+.

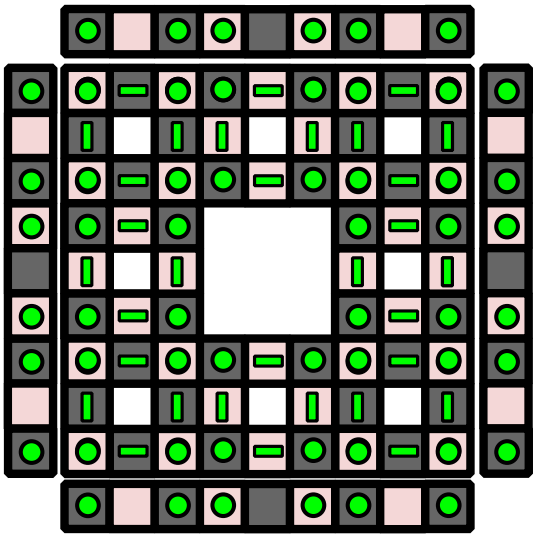


Figure 13 The second iteration S of the Sierpinski Carpet (in the center) is a subconnector of K: S^+ contains a subdivision of K^+, in which the cells with a circle correspond to vertices of K^+, and those with a bar are obtained by dividing the edge of K^+ the bar represents. Not all edges are subdivided: adjacent circles in G^+ are indeed adjacent in K^+.

Then it is possible to define a substitution $\gamma_w : \mathcal{W} \to \mathcal{W}^G$ by using κ_w to fix a subdivision of G^+, then adding the extra vertices to $\gamma_w(w)$. This process may have create vertices in $\gamma_w(w)$ with two opposite inputs, for which the extra cases of figure 12 are necessary.

The label substitutions κ_1 and κ_2 can also be adaptated to G^+, defining γ_1 and γ_2. In γ_1, all vertices which do not represent a vertex of K have label $\mathfrak{normal}$. For layer 2, each vertex of G^+ representing a vertex of K keeps its label, each new vertex gets a label $\mathfrak{i}_1$.

Set $\Sigma_1^G = \Sigma_1$, except that pos takes values in G rather than K. Then Σ_2^G is Σ_2, except that ancestor-msg takes values in Σ_1^G rather than Σ_1.

Then a circuit C_G on $\Sigma_1^G \times \Sigma_2^G$ can be defined from γ_w, γ_1 and γ_2 like $C_\square$. That circuit C_G has the same properties as $C_\square$ and is also self-descriptive. When accounting for the local part of the messages, gates which are upstream from all the K-vertices show the local message of the corresponding input meta-gate.

Thus C_G is self-descriptive. ◀

The question is now which G are such that G^+ has a subdivision of K^+ as a subgraph. This condition seems constraining since K is a rather large graph —32 vertices. Yet, one can make any G larger by iterating the substitution generated by G before trying to self-assemble G^∞.

▶ Remark 53. Let $G \ni (0,0)$ be a finite shape of $\mathbb{N}^2$, and σ_G the associated substitution. For any $k > 0$, $G^\infty = (G^k)^\infty$.

By iterating σ_G, it is quite easy to get a K as a subconnector, as long as one starts with sufficient vertical and horizontal bandwidth.

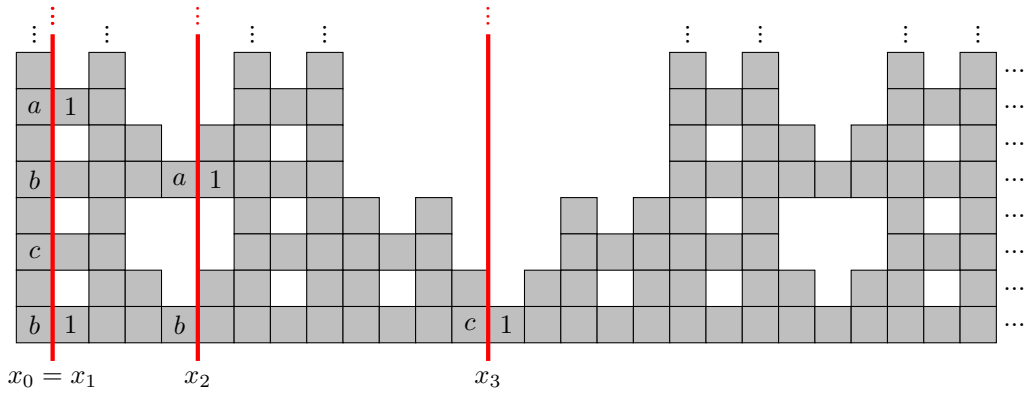


Figure 14 The set C_k is the set of all glues which attach across the vertical lines at coordinate x_k. Ignoring any tiles which are not visible in the picture, $C_0 = C_1 = \{a \cdot E, b \cdot E, c \cdot E\}$, $C_2 = \{a \cdot E, b \cdot E\}$ and $C_3 = \{c \cdot E\}$. The tiles marked t^1 are the ones which appear in their column in some production without tiles right of the red line; the subset F_k of C_k contains their east glues: , $F_0 = F_1 = \{a \cdot E, b \cdot E\}$, $C_2 = \{a \cdot E\}$ and $C_3 = \{c \cdot E\}$

▶ **Lemma 54.** *Let $G \ni (0,0)$ be a finite shape of $\mathbb{N}^2$. If $G[N \leftrightarrow S] \geq 2$ and $G[E \rightarrow W] \geq 2$, then $K \preceq G^3$*

Proof. Let $H = \{0,1\}^2$. If $G[N \leftrightarrow S] \geq 2$ and $G[E \rightarrow W] \geq 2$, then $H \preceq G$: pick two disjoint North-South paths, two disjoint East-West paths, their four intersections can act as the vertices of H.

By remark 51, since $H \preceq G$, $H^3 \preceq G^3$. But H^3 is a 8×8 grid, so $K \preceq H^3$.

Hence, $K \preceq G^3$. ◀

Finally, this characterization is essentially tight, by a generalization of the impossibility result of Hendricks et al. [11].

▶ **Theorem 55.** *Let $G \ni (0,0)$ a finite shape of $\mathbb{N}$.*

If for all $k > 0$, there is an x_k such that there is only one y with $(x_k, y) \in (G^k)^+$ and $(x_k, y+1) \in (G^k)^+$, then G^∞ cannot be strictly self-assembled in the aTAM model unless $G = \{\vec{0}\}$.

Proof. Let w and h be the width and height of G. Assume that there is an aTAM $\mathcal{G}$ which self-assembles G^∞ at temperature τ.

Let C_k be the set of all glues which appear on the eastern edge of position (x_k, y) for some value of y. Since $\mathcal{G}$ self-assembles G^∞, all the C_k are non-empty, so let $C_\infty = \bigcap_k (\bigcup_{k'>k} C_{k'})$ be the set of glues appearing in infinitely many C_k; C_∞ is also non-empty. Let $F_k \subseteq C_k$ be the set of glues g such that there is a position $\vec{z} = (x_k, y)$ and a production $\Pi_{k,t}$ of $\mathcal{G}$ where:

- the eastern glue of $\Pi_{k,t}(\vec{z})$ is g
- $\Pi_{k,t}$ contains no tile right of x_k

Like C_∞, $F_\infty = \bigcap_k (\bigcup_{k'>k} F_{k'})$ is non-empty. The sets C_k and F_k are illustrated on Figure 14.

Now let t be a glue of $\mathcal{G}$, and s_t be a (new) tile with t on its eastern side, and ϵ on all other sides. Let $\mathcal{G}_t[x \geq 1]$ be the set of productions of $\mathcal{G}$, starting from the seed configuration with s_t at $\vec{0}$ and attaching no tiles at positions $x < 1$. Let $u_t = \min_{p \in \mathcal{G}_t[x>1]} \max\{y | (x,y) \in p\}$ and $d_t = \min_{p \in \mathcal{G}_t[x>1]} \min\{y | (x,y) \in p\}$. These two quantities are illustrated on Figure 15.

Since $G^\infty \subset \mathbb{N}^2$, it does not contain any infinite southward path. Therefore, for any k and $t \in F_k$, $d_t > -\infty$, otherwise from $\Pi_{k,t}$ it would be possible to produce paths going

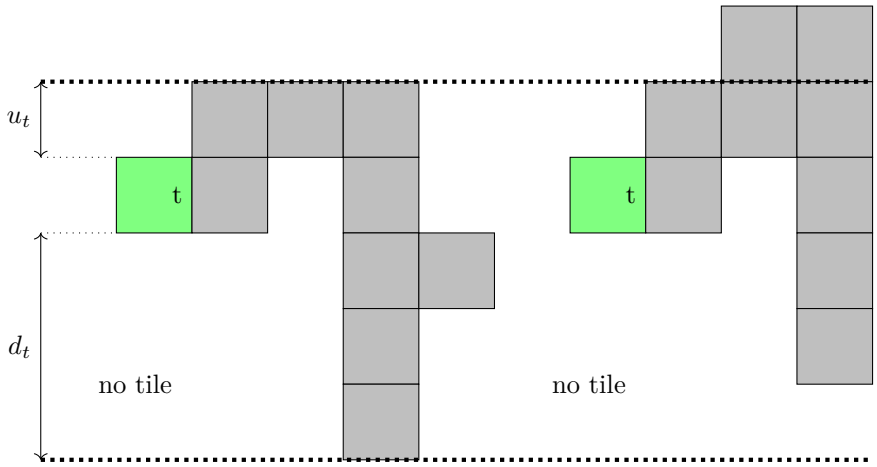


Figure 15 The definition of u_t and d_t given the two productions reachable from the glue t, without going left of the seed.

arbitrarily far down, out of $\mathbb{N}^2$. Let $d_\infty = \max_{t \in F_\infty}\{d_t | t \in \mathbb{G}\}$, d_∞ is finite. On the other hand, $u_\infty = \max\{u_t | t \in F_\infty\}$ must be infinite. Indeed, if it is finite, let $s = u_\infty - d_\infty + 1$. Consider a maximal production Π^* obtained by only placing tiles left of the vertical line L^* at x-coordinate x_s. Starting from Π^*, the only attachable positions are isolated points on L^*, separated by a distance at least w^s. From each of them, it is possible to grow an arm which reaches no higher than s upwards, whence, as illustruted on Figure 16, the part right of L^* of that production is contained within a union of $(u_\infty - d_\infty)$-width horizontal bands, so its domain cannot be G^∞.

For t such that $u_t = +\infty$, all terminal productions on the right half-plane reach infinitely high. Thus, by applying Lemma 8 with increasing values of m, either one of these terminal productions contains a periodic path going up, or there are productions P with arbitrarily large squares in their fill-in $P^\bullet$. In both cases, for each k, there is a production F_k for which some $k \times k$ square is inaccessible from any point $(0, y)$ with $y > 0$ without crossing F_k, as illustrated on Figure 17.

For a large enough level of substitution, any glue t for which u_t is finite does not even grow one "meta-cell" up. Let $F_! = \{g \in F_\infty | u_g = +\infty\}$, and $s_!$ be such that $\max\{u_g | g \notin F_!\} < h^{s_!}$ and $d_\infty > -h^{s_!}$.

Consider a vertical line L_x at some position x between copies of $G^{s_!}$ in G^∞. Let P_y be a maximal subproduction of P which can be assembled without attaching any tile right of L_y. In P_y, let $\vec{z}_x = (x, y)$ be the lowest position on L_x containing a tile T of $F_{s!}$.

By the pigeonhole principle, there must be two vertical lines L_n and L_m between copies of $G^{s_!}$ such that $P_n(\vec{z}_n) = P_m(\vec{z}_m)$. Let $\vec{v}$ be the vector $\vec{z}_m - \vec{z}_n$. By construction of P_n and P_m, for any production $P_n^!$ obtained from P_n, there is a production $P_m^!$ obtained from P_m such that the part of $P_n^!$ above and to the right of $\vec{z}_n$ and the part of $P_m^!$ above and to the right of $\vec{z}_m$ are identically filled. Hence, there is a vector which preserves either arbitrarily large cycles of G^∞ or its intersection with a half-plane, thus G must be trivial. ◀

This leaves open the case where in all the iterates $(G^k)^+$, there is a vertex which disconnects either the north and south or the east and west, but no straight line which crosses $(G^k)^+$ only once. A more precise variant of the previous proof takes care of that case.

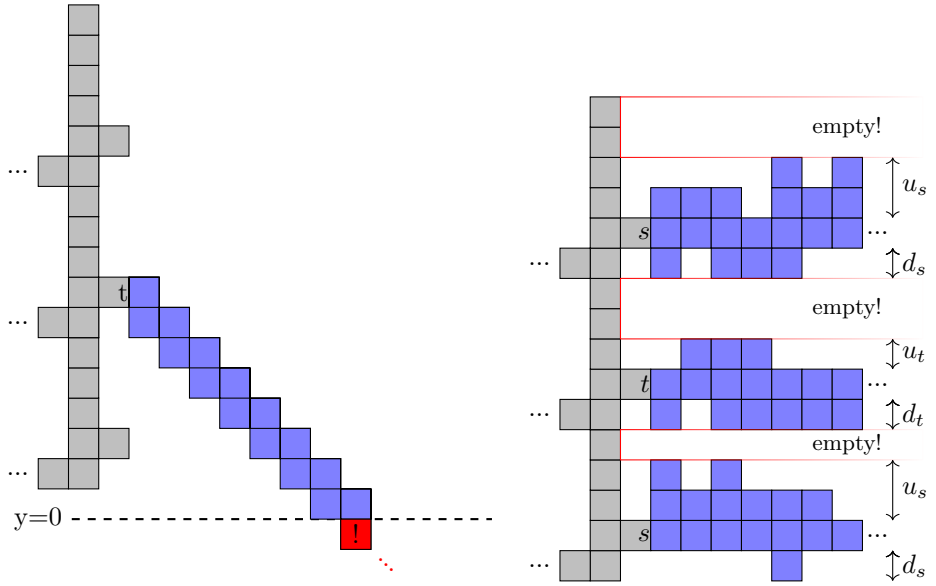


Figure 16 Faulty terminal productions which can be reached, left if some d_t is infinite, right if all u_t are finite: $\min_t(d_t) = -1$, $u_t = 1$, $u_s = 2$.

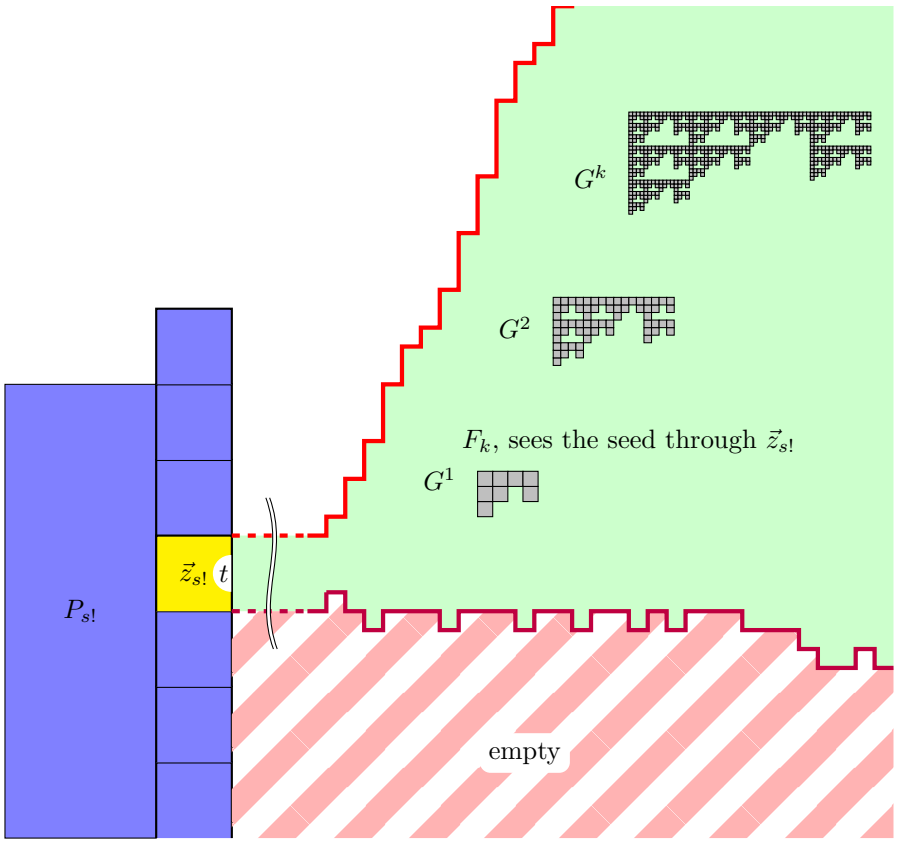


Figure 17 The part of the production that sees the seed only through $\vec{z}_!$ contains cycles of G^∞ of arbitrary size.

▶ **Lemma 56.** *Let $G \ni (0,0)$ a finite shape of $\mathbb{N}$, and for $k \in \mathbb{N}$, $G^k = \sigma_G^k(\{(0,0)\})$.*

If $\sup_k(G^k[N \leftrightarrow S]) = 1$ or $\sup(G^k[E \leftrightarrow W]) = 1$, then G_∞ cannot be strictly self-assembled in the aTAM model unless it is trivial.

Proof. Omitted due to laziness [14]. In the previous proof, take a cut per k which passes through the first occurence of each level of east-west bridges. Note that these cuts are not necessarily straight, but they remain within a bounded band around the bridges. The proof concludes as the previous one. ◀

Lemma 56 gives a dichotomy between generators for which the associated fractal can be strictly self-assembled and those for which it cannot. There is a polynomial time algorithm for determining which of Lemma 52 or Lemma 56 applies.

▶ **Theorem 57.** *There is a polynomial time algorithm which, on input G decides whether:*

- *there is a k such that $G^k[N \leftrightarrow S] \geq 2$ and $G^k[E \leftrightarrow W] \geq 2$ and thus G^∞ can be strictly self-assembled in the aTAM, or*
- *for all k, $G^k[N \leftrightarrow S] < 2$ and thus G^∞ cannot be strictly self-assembled in the aTAM, or*
- *for all k, $G^k[E \leftrightarrow W] < 2$ and thus G^∞ cannot be strictly self-assembled in the aTAM.*

Notice that the second and third case are not *mutually exclusive.*

Proof. Without loss of generality, assume G is connected.

First, use a max-flow algorithm to compute $G[d \leftrightarrow d']$ for all directions d, d'. If both $G[E \leftrightarrow W]$ and $G[N \leftrightarrow S]$ are 2 or more, then the first case applies.

Let D be a set of pairs of directions, v is a D-disconnector if v disconnects G^{+d} from $G^{+d'}$ for any $(d, d') \in D$. If $G[E \leftrightarrow W] = 1$, then there is at least one $\{EW\}$-disconnector. For any D-disconnector, define $\text{Cause}(v, D)$ as the set of pair of directions (δ, δ') such that there are $(d, d') \in D$ and a path from G^{+d} to $G^{+d'}$ which enters v from direction δ and leaves it through direction δ'.

Compute the disconnect causation graph Δ. It is a directed graph whose vertices are couples (v, D) where v is a D-disconnector, and there is an arc from (v, D) to each vertex $(v', \text{Cause}(v, D))$. The causation graph Δ has size at most $64|G|$ and the existence of each of its arcs can be tested in polynomial time.

The graph Δ contains a cycle reachable from $(v, \{EW\})$, if and only if for all k, $G^k[E \leftrightarrow W] < 2$, likewise for N and S. Indeed, for $k > 0$, a vertex v is a D-disconnector in G^k if and only if:

- The vertex of $\lfloor v/G^{k-1} \rfloor$ of G corresponding to the copy of G^{k-1} containing v is a D-disconnector, and
- the vertex $(v \bmod G^{k-1})$ of G^{k-1} corresponding to the position of v within that copy is a $\text{Cause}(v, D)$-disconnector.

Hence, from Δ, it is possible to distinguish between the three cases. ◀

A Proof of the Tree Pump Lemma

A.1 Free Assemblies

The proof of Lemma 8 needs some ancillary definitions. In particular, one needs to define what it means to do self-assembly with the ability to escape the confines of the plane $\mathbb{Z}^2$. In this context, assemblies sit on a graph called an *assembly support*.

▶ **Definition 58** (Assembly support). *An* assembly support *is a directed graph G with labels in $\{N, E, S, W\}$ on its arcs, such that:*

- *each vertex has at most one incoming arc with each label,*
- *each vertex has at most one outgoing arc with each label,*
- *for any cycle c of G, the sum of the labels of the arcs of c is equal to $\vec{0}$ as a vector of $\mathbb{Z}^2$,*
- *for any path π of 1 or 3 arcs from u to v, if the sum in $\mathbb{Z}^2$ of the labels of π is $\vec{d} \in \{N, E, S, W\}$, then there is an arc from v to u with label $-\vec{d}$.*

An example of a correct assembly support is given on Figure 19, while Figure 18 shows some counter-examples of labeled graphs which are not assembly support. Because of the conditions on the incoming and outgoing arcs of each vertex, it makes sense to represent the vertices as tiles, and arcs as sides the tiles share. This hopefully helps build the intuition that assembly support are "like $\mathbb{Z}^2$, but long separated paths which should come to the same position may avoid each other". On Figure 18, the direction corresponding to each side of the tile is given explicitely. With correct Assembly Supports, like the ones on Figure 19, the convention is that the direction of each side corresponds roughly to its direction on the page, and small adjustment allow tiles which would otherwise be adjacent on the page not to be. The appearance of tiles avoiding each other by going in the third dimension is deliberate as a way to build intuition.

An *embedding* from an assembly support G to another assembly support G' is a graph embedding which preserves the label of the arcs. An *isomorphism* between G and G' is a graph isomorphism preserving the arc labels.

▶ **Definition 59** (Perspective difference, Translation). *For any two vertices (z, z') of a connected component of an assembly support G, their* perspective difference $z' \mathbin{\mathring{-}} z$ *is $\sum_{a \in p} a$, where $p \in \{N, E, S, W\}^*$ is the sequence of labels of a path from z to z'. For any embedding $e : G \to \mathbb{Z}^2$, $e(z') - e(z) = z' \mathbin{\mathring{-}} z$. By convention, z, z' are in different connected components, $z' \mathbin{\mathring{-}} z = \infty$.*

A translation between two subgraphs $A \subset G$ and $B \subset G'$ of two assembly support is an isomorphism from A to B which preserves perspective differences. Any isomorphism between connected subgraphs is a translation.

For any Assembly Support G and $z \in G$, $\text{squash}_{z,G} : G \to \mathbb{Z}^2$ is the embedding $z' \mapsto (z' \mathbin{\mathring{-}} z)$. Since changing z in $\text{squash}_{z,G}$ only amounts to a translation, it will generally be omitted. The index G will be omitted as well when this causes no ambiguity.

The analysis in the Tree Pump Lemma relies on an examination of the *holes* of the productions of $\mathcal{S}$. These holes need to be suitably defined now that the assemblies are no longer necessarily planar.

▶ **Definition 60** (Exterior Arc). *An* exterior arc *of an assembly support G is a pair of a vertex v of G and a direction d such that v has no arc labelled d in G.*

▶ **Definition 61** (Converging Exterior Arcs, Growth). *Two exterior arcs (v, d), (v', d') are* converging *when $v' \mathbin{\mathring{-}} v = d' - d$ –they virtually point to the same empty position.*

Given an assembly support G and a set Z of converging arcs, the assembly support $G + Z$ (G grown by Z) is $G \cup \{\zeta\}$, where there is an arc $z \to \zeta$ (for $z \in G$) in direction d whenever $(z, d) \in Z$.

▶ **Definition 62** (Step, Exterior Path, Hole). *There is a* step *between two arcs or exterior arcs (v, d) and (v', d') if:*

- *$G = G'$ and d, d' form a $\pm\frac{\pi}{2}$ angle,*

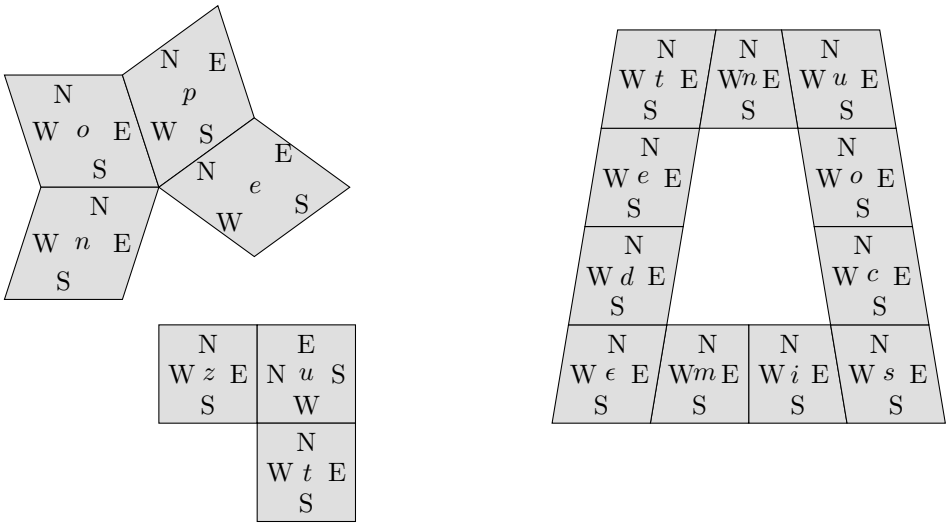


Figure 18 Ceci n'est pas un Assembly Support: three labelled graphs which fail to be assembly supports. Each vertex is represented by a tile, its labels are the labels of its outgoing arcs. A label on a side with no neighbor corresponds to an exterior arc. The bottom-left one has a path of length 1 with label E, from z to u, but the $-E = W$ neighbor of u is t. Also, the N neighbor of t is u, which has no S neighbor. The one in the top right has a 3 arc path "*nope*" whose labels sum to E, but n is not the W neighbor of e (actually, there is none). The one on the right has a cycle "ϵ *miscounted*" for which the sum of the labels is $3E + 3N + 2W + 3S = E$, which is not $\vec{0}$.

- *$d = d'$ and there is an (non-exterior) arc from G to G' with label d'', with d, d'' forming a $\pm\frac{\pi}{2}$ angle*
- *d, d' form a $\pm\frac{\pi}{2}$ angle and there is a cell G'' with $G'' + d = G$ and $G'' + d' = G'$.*

A (simple) exterior path *is a sequence $P = (p_0, p_1, \ldots, p_k)$ of arcs, such that:*

- *for each i, there is a step between p_i and p_{i+1}, and*
- *for $0 < i < k$, p_i is an exterior arc.*

A hole *is a simple exterior path which loops back to its starting exterior arc.*

The perimeter *of a hole $H = (h_0, \ldots h_k = h_0)$ is the set of vertices appearing in the exterior vertices h_i.*

These definitions are illustrated on Figure 19, which features an assembly support with two holes of perimeter 20. These holes are "morally infinite", so they resist attempts at a definition of area —at least the author's naive ones. For an assembly which embeds into $\mathbb{Z}^2$, the free definition of hole corresponds to the intuition of a hole in an assembly, up to the detail that the exterior of the assembly forms a hole, so that all finite assemblies have at least one hole. Note that exterior paths are oriented: they turn in the positive direction around each tile.

▶ **Definition 63** (Free Assembly). *A free assembly A of the TAS $\mathcal{S}$ is composed of an assembly support $A{\cdot}\,\text{support}$ and a total function $A{\cdot}\,\text{tile} : A{\cdot}\,\text{support} \to \mathcal{S}{\cdot}\,\text{tileset}$.*

For a TAS $\mathcal{S}$, the set of free assemblies of $\mathcal{S}$ is written $\mathcal{S}^{\text{Free}}$.

When there is an injective embedding $e : A{\cdot}\,\text{support} \to G$, it naturally defines a free assembly $e(A)$ with $e(A){\cdot}\,\text{support} = e(A{\cdot}\,\text{support})$ and for $\in e(A{\cdot}\,\text{support})$, $e(A)(e(z)) = A(z)$. An assembly A *embeds* into $\mathbb{Z}^2$ if squash is a one-to-one embedding $A{\cdot}\,\text{support} \to \mathbb{Z}^2$; as above, this embedding naturally defines an assembly $\text{squash}(A)$.

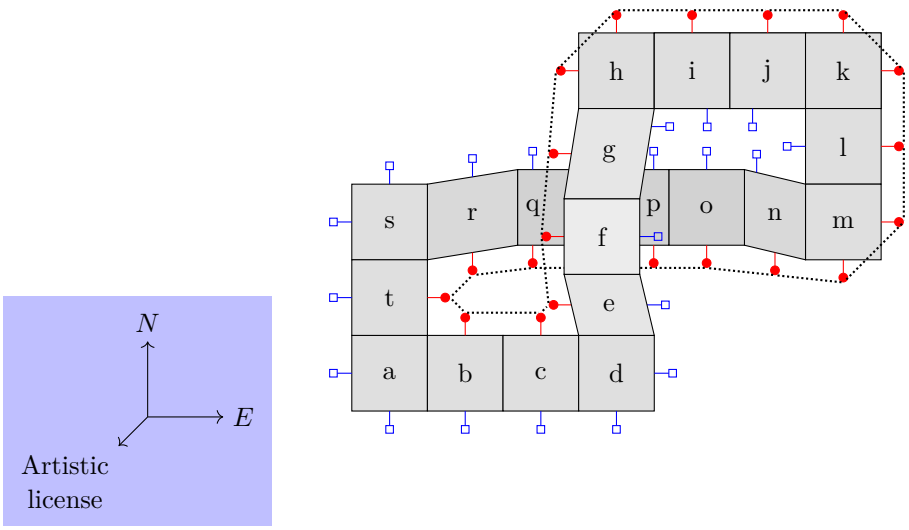


Figure 19 An example of an assembly support, the shifts in the 3rd dimension between e and g as well as between r and n are artistic license to show the absence of adjacencies between those two branches. The sum of the vectors between neighbors on the cycle $abcdefghijklmnopqrsta$ is $3E + 4N + 3E + 2S + 6E + 2S = \vec{0}$. The sum of vectors between neighbors on the path $fedcbatsrqp$ is $2S + 3W + 2N + 3E = \vec{0}$, yet the vertices f and p are distinct. The small pending vertices next to the tiles are the exterior arcs; they form two holes, one for the square exterior arcs, and one for the circles. The dotted lines shows the steps between the exterior arcs of the "circle" hole. The step between the exterior arcs (m, S) and (m, E) stems from the first case of the definition, the one between (b, N) and (c, N) from the second case, and the one between (t, E) and (r, S) because of the third. In this last case, s acts as the pivot between r and s. Both holes have perimeter 20.

▶ **Definition 64** (Free Attachment, Free Assembly Sequence). *Let A, A' be free assemblies of some TAS system $\mathcal{S}$, and $\eta : A \to A'$ an embedding. The tuple (A, A', η) is an assembly candidate if:*

- *there is a $z \in A' \cdot \text{support}$ such that $\eta : A \cdot \text{support} \to A' \cdot \text{support} \setminus \{z\}$ is a translation,*
- *for any $z' \in A \cdot \text{support}$, $A \cdot \text{tile}(z') = A \cdot \text{tile}(\eta(z'))$.*

The definitions of attachment, assembly sequence, production and terminal production extend to free assemblies. The $\mathbb{Z}^2$ version of each definition is simply the particular case where every assembly embeds into $\mathbb{Z}^2$.

For a TAS $\mathcal{S}$, the set of free assembly sequences of $\mathcal{S}$ is written $\mathcal{H}^{\text{Free}}[\mathcal{S}]$.

The definition of free attachment is illustrated on Figure 20. The fact that η is an isomorohpism ensures that parts of the assembly which do not touch the position z are unaffected by the attachment. This preserves the locality of the aTAM process, in contrast to what happens in the FTAM [5] for instance. Hence, while the above definition of attachment is given from the point of view of A', "after the fact", from the point of view of A, an attachment candidate can be defined from a tile type t and a set Z of convergent exterior arcs: $t@Z$ is the attachment candidate (A, A', η) where:

- $A' \cdot \text{support} = A \cdot \text{support} + Z$, and ζ is the vertex of $A' \cdot \text{support}$ which is not in $A \cdot \text{support}$,
- η is the identity on $A \cdot \text{support}$,
- $A' \cdot \text{tile}(\zeta) = t$,
- $A' \cdot \text{tile}(z) = A \cdot \text{tile}(z)$ whenever $z \neq \zeta$.

For clarity, a "normal" assembly will be referred to as a $\mathbb{Z}^2$-assembly, likewise for the other concepts which were just endowed with a "free" variant.

Embeddings act not only on assembly supports and assemblies, but also on assembly sequences. In doing so, they may break the correctness of attachments if they are not one-to-one; when that happens, the sequence is cut short.

▶ **Definition 65** (Assembly Sequence Embedding). *Let $\alpha = (A_0 \to A_1 \to \ldots) \in \mathcal{H}^{\text{Free}}[\mathcal{S}]$, let $G = (\lim \alpha) \cdot \text{support}$ and G' an assembly support. Let $e : G \to G'$ be an embedding; since up to translation, $A_0 \cdot \text{support} \subset A_1 \cdot \text{support} \subset \ldots \subset (\lim \alpha) \cdot \text{support}$, for each i, e induces an embedding from $A_i \cdot \text{support}$ into G'.*

Let k be the largest i such that e is an isomorphism on $A_i \cdot \text{support}$, then $e(\alpha)$ is the free candidate assembly sequence $(e(A_0), \ldots, e(A_k))$.

A.2 Ordinal Assembly Sequences

It is often practical to consider what happens in an assembly sequence α after it has placed an infinity of tiles. For this, *ordinal* assembly sequences come useful.

▶ **Definition 66** (Ordinal Assembly Sequence). *Let $o \in \omega_1$ be a countable ordinal, an o-Assembly Sequence starting from an assembly A_0 is a sequence of length o of free assemblies such that for any $i < j \leq o$, there is an injective embedding $\eta_{i \to j} : A_i \to A_j$ with:*

- *for $i < o$, $(A_i, A_{i+1}, \eta_{i \to i+1})$ is a free attachment,*
- *for $i < j < k \leq o$, $\eta_{j \to k} \circ \eta_{i \to j} = \eta_{i \to k}$*

The i-th production of α is A_i, and its i-th attachment is $\eta_{i \to i+1}$.

For an o-Assembly Sequence α, the notation $\lim \alpha = A_o$ is consistent with the case of usual Assembly Sequences, i.e. ω-Assembly Sequences.

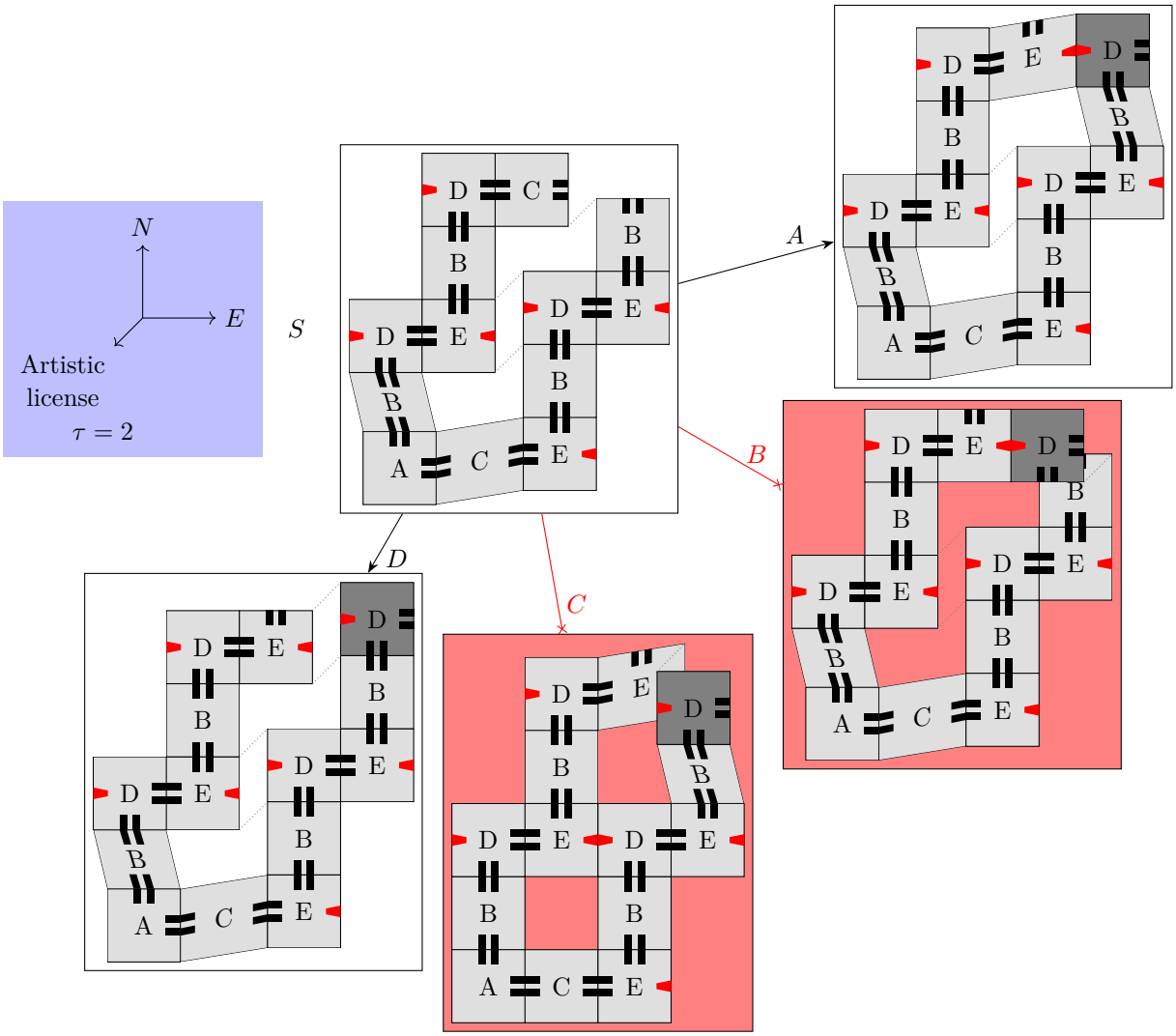


Figure 20 Some possible and impossible free attachments from a starting free assembly S, at temperature τ. Again, the third coordinate is a visual aid to show non-adjacencies; small dashed lines link points which embed in the same position in $\mathbb{Z}^2$. Assemblies A and D are reachable through one attachment $t_D@z$, where in each case, z is the position in dark gray. The assembly B is an attachment candidate, but it is not stable, since the new tile t_D only binds through a strength-1 glue. The assembly C can not even form an attachment candidate from S, as C·support does not contain an isomorphic copy of S·support

These sequences correspond to the intuitive generalization of attachment sequences to "times larger than infinity". In particular, any tile attached through an ordinal attachment sequence reaches the starting assembly of the sequence through a finite number of attachments. Indeed, for a tile to have been added through an ordinal assembly sequence α at time t, the neighbors to which it attaches must have been attached at some time $t' < t$. Since these times are ordinals, such an decreasing sequence of times reaches 0 in a finite number of steps. In particular, these ordinal assembly sequences produce the same productions as the usual ω-assembly sequences.

This remark formalizes thus:

▶ Remark 67. Let $\mathcal{S}$ be an assembly system, o a countable ordinal, α an o-Assembly Sequence, $k < o$ and $t@Z$ the k-th attachment of α. There is a sequence α' of length o' and a bijection $\iota : o \to o'$ such that:

- for all $t < o'$, the attachments α_t and $\alpha'_{\iota(t)}$ are the same,
- $\iota(k)$ is finite.

▶ **Lemma 68.** *Let $\mathcal{S}$ be an assembly system, o a countable ordinal, and α an o-Assembly Sequence. Then if o is infinite, there is a ω-Assembly Sequence α' such that $\lim \alpha' = \lim \alpha$.*

Proof. The proof goes by transfinite induction.

If $o = \omega$, then α itself satisfies the conclusion of the lemma.

If $o = p + 1$ is a successor ordinal, by induction hypothesis, there is an ω-assembly sequence α'' such that $\lim \alpha'' = A_p$, where A_p is the p-th production of α. Let $t@Z$ be the p-th attachment of α; there is an integer k such that at time k, all the origin vertices of the arcs in Z are attached in α''. The attachment sequence $\alpha' = (\alpha''_0, \dots, \alpha''_k, t@Z, \alpha''_{k+1}, \dots)$ satisfies the conclusion of the lemma.

If o is a limit ordinal, there is a sequence $(\alpha'^i)_{i<o}$ of ω-assembly sequences such that for each i, $\lim \alpha'^i = A_i$. Let A'^i_j be the j-th production of α'^i. Since o is countable, it has cofinality ω; pick an increasing sequence $\epsilon : \omega \to o$ such that $\sup_{i<\omega} \epsilon(i) = o$. Let α' be the enumeration of the set $\{A'^{\epsilon(i)}_j | j \leq i < \omega\}$ ordered lexicographically according to (i, j). This sequence α' is an attachment sequence, and it satisfies $\lim \alpha' = \bigcup_{i<o}(A'^{\epsilon(i)}_i) = \lim \alpha$. ◀

A.3 Holes and Fizziness

The Tree Pump lemma is about tree-like assemblies. Since trees are acyclic graphs, the *holes* of the assemblies play an important part in characterizing how these tree-like assemblies behave. Fizziness is the tendency of assembly sequences to create a lot of holes.

▶ **Definition 69** (Fizziness). *Let A, A' be two free assemblies with $A \cdot \text{support} \subset A' \cdot \text{support}$, the* fizziness $\text{fz}(A, A')$ *is the number of holes of A' whose perimeter is not contained in A.*

Let $\alpha = (A_0 \to A_1 \to \dots)$ be an Assembly Sequence in $\mathcal{H}^{\text{Free}}[\mathcal{S}]$. The fizziness of α, noted $\text{fz}(\alpha)$ *is the sequence $i \mapsto \text{fz}(A_i, A_i + 1)$.*

A sequence α is more fizzy than β, written $\alpha >_{\text{fz}} \beta$ if $\text{fz}(\alpha) > \text{fz}(\beta)$ lexicographically.

▶ **Lemma 70** (Maximal Fizziness). *Let $\mathcal{S}$ be a seeded assembly system. Assume $\mathcal{S} \cdot \text{seed}$ has a finite number of non-null glues on its external edges. Then there is an ω-Assembly Sequence $\alpha_{\max} \in \mathcal{H}^{\text{Free}}[\mathcal{S}]$ with maximal fizziness among ω-Assembly Sequences.*

Proof. Using Kőnig's Lemma. ◀

Note that this lemma does not hold for o-assembly sequences with $o > \omega$. Indeed, after time ω, there might be an infinity of possible attachments, thwarting Kőnig's Lemma.

▶ **Lemma 71** (Fizziness-Increase of Embeddings). *Let $\mathcal{S}$ be a seeded TAS, and $\alpha \in \mathcal{H}^{\text{Free}}[\mathcal{S}]$. Let $G = \text{support}(\lim \alpha)$, G' an assembly support, and $e : G \to G'$ an embedding.*

Then:

1. *$e(\alpha)$ is a free assembly sequence,*
2. *$e(\alpha) \geq_{\text{fz}} \alpha$*
3. *$e(\alpha) >_{\text{fz}} \alpha$ if e is not injective on $\text{dom}(\lim \alpha)$.*

Proof. Let $\alpha = (t_i @ z_i)_i$ be an assembly sequence in $\mathcal{H}^{\text{Free}}[\mathcal{S}]$.

For Item 1, the attachments of $e(\alpha)$ are valid by definition of k. They are stable since for each attachment, the edges adjacent to z_i which make this attachment stable are preserved by e.

For Item 2, for each assembly A of α which is mapped injectively, each hole of α is mapped by e to a hole of $e(A)$ with the same labels on its border.

For Item 3, if e is not injective on $\text{dom}(\lim \alpha)$, then there are $i < k$ such that $e(z_i) = e(z_k)$. Since α is a valid assembly sequence, $z_i \neq z_k$, there are two different paths from the seed to $e(z_i)$. From these two paths, it is possible to construct a hole in $\text{dom}(e(\alpha))$ which is not a hole in $\text{dom}(\alpha)$. Hence, from the first attachment where this hole appears, the assemblies of $e(\alpha)$ are more fizzy than the corresponding ones in α. ◀

▶ **Lemma 72** (Maximal Sequences are Flat). *Let $\mathcal{S}$ be a seeded TAS, S an assembly of $\mathcal{S}$ and $X \subset \mathcal{H}^{\text{Free}}[\mathcal{S}, S]$ be a set of Free, Ordinal Assembly Sequences such that for any $\alpha \in X$, there is a $\alpha' \in X$ such that $\alpha' \geq_{\text{fz}} \text{squash}(\alpha)$.*

Assume α is a free assembly sequence with maximal fizziness within X. Then α embeds into $\mathbb{Z}^2$.

Proof. If α does not embed into $\mathbb{Z}^2$ by squash, then $\text{squash}(\alpha)$ is fizzier than α and thus α cannot be maximal in X. ◀

A.4 The Window Movie Lemma

The Window Movie Lemma [17] applies to free assembly sequences as well as to $\mathbb{Z}^2$ assembly sequences. In order to account for holes, movies need to record not only the glues appearing on either side of the window, but also the paths between edges of the window through either side.

▶ **Definition 73** (Window, Frame, Movie). *Let $\alpha = (A_0, A_1, \ldots)$ be an assembly sequence of $\mathcal{H}^{\text{Free}}[\mathcal{S}]$. A* window *$W$ of α is a cut of $(\lim \alpha) \cdot \text{support}$ separating it into two connected components $\text{Near}(W)$ and $\text{Far}(W)$.*

The width $w(W)$ of a window is the maximum perspective difference between two vertices along the window.

Let $\text{Arcs}(W)$ be the set of arcs (either normal or external) of $(\lim \alpha) \cdot \text{support}$ through W.

The frame *f on W at time k records for each arc $a = (v, d) \in \text{Arcs}(W)$:*

- *$f(a) \cdot \text{present}$: whether $v \in A_k \cdot \text{support}$, and if so,*
- *$f(a) \cdot \text{glue} = A_k \cdot \text{tile}(v)(d)$, as well as*
- *$f(a) \cdot \text{successor}$ the arc of $\text{Arcs}(W)$ which is reachable from a through an exterior path which does not cross W, if there is one.*

The movie *associated with W is the ordered set $\mathcal{MOVIE}(W)$ of distinct frames appearing on W.*

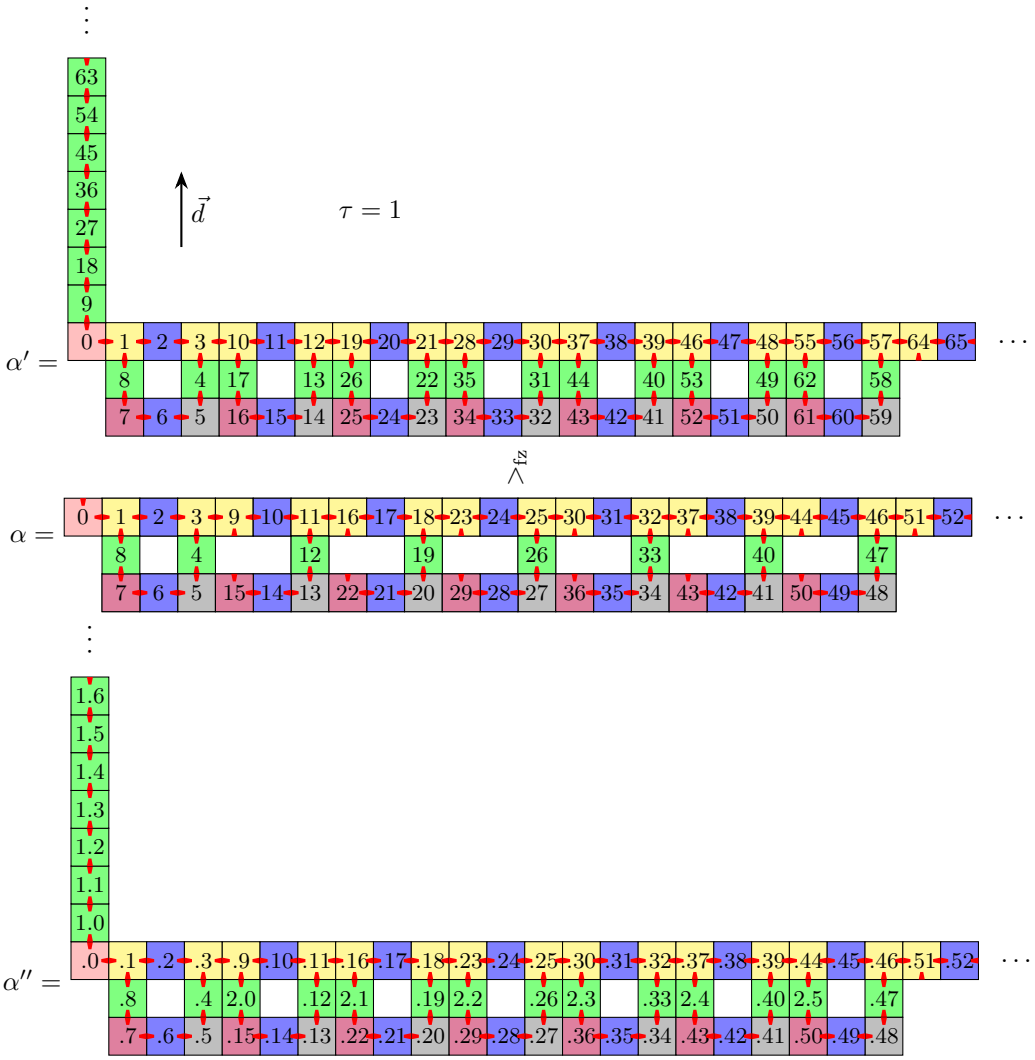


Figure 21 An assembly sequence α' of some temperature 1 aTAM, which yields a terminal production $F = \lim \alpha'$. Below a sequence $\alpha >_{\text{fz}} \alpha'$ which yields a smaller production P which is bounded in direction $\vec{d}$ (vertically). The assembly sequence α is fizzier since by step 50, it just closed its seventh hole while α' is lagging at only 5 holes. Because α skips any unprofitable attachment, $\lim \alpha$ is very much not terminal: its seed (tile number 0) as well as every tile attached at a time of the form $16 + 7k$ has an attachable yet unfilled position. It can thus be extended into an ordinal assembly sequence α'' which does reach $\lim \alpha$, but in time 3ω. The attachment times of the form $i.j$ in α'' should be read as $i\omega + j$. At time ω, α'' has assembled $\lim \alpha$; at time 2ω, it has added the upwards path, and at time 3ω, it has added the last tile to each hole of $\lim \alpha$.

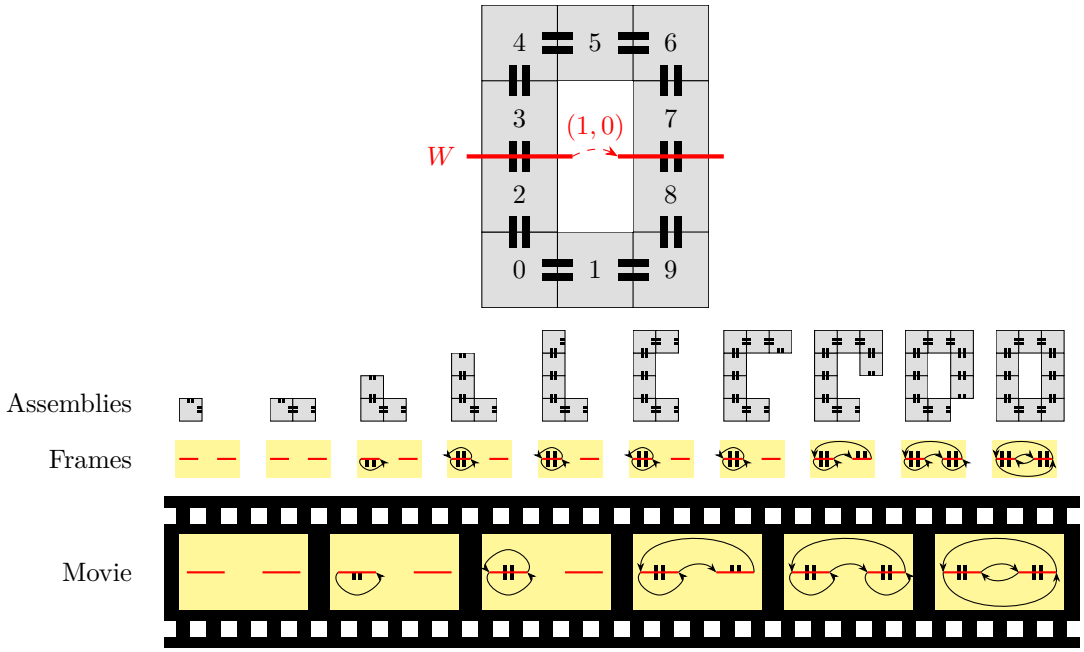


Figure 22 Example assembly sequence α and window W. On the main picture, the labels of the tiles of $\lim\alpha$ are the order in which they attach. The small pictures are the successive assemblies of α, their associated frames and the obtained movie. The movie is made up of the sequence of the unique frames, in order of apparition.

Figure 22 gives an example free assembly sequence α with a window W. The arrows between the edges on either side of W represent the relation " successor".

▶ **Lemma 74** (Window Movie Lemma). *Let $\alpha \in \mathcal{H}^{\text{Free}}[\mathcal{S}, \sigma_A]$ and $\beta \in \mathcal{H}^{\text{Free}}[\mathcal{S}, \sigma_B]$ two assembly sequences. Assume there are two windows A in α and B in β and a translation τ such that $\tau(A) = B$, $\mathcal{MOVIE}(A)$ is the same as $\mathcal{MOVIE}(B)$ up to translation by τ, and lastly τ can be extended to a translation from $\text{Far}(A)$ to $\text{Far}(B)$ which maps $\sigma_B \cap \text{Far}(B)$ to $\sigma_A \cap \text{Far}(A)$.*

Let $\alpha^\dagger = \alpha_{|\,\text{Far}(A)} \in \mathcal{H}^{\text{Free}}[\mathcal{S}, \lim\alpha \cap \text{Near}(A)]$ be the sequence of attachments of α within $\text{Far}(A)$, and likewise $\beta^\dagger = \alpha_{|\,\text{Far}(B)} \in \mathcal{H}^{\text{Free}}[\mathcal{S}, \lim\beta \cap \text{Near}(B)]$ be the sequence of attachments of β witihn $\text{Far}(B)$.

Let α' be the candidate assembly sequence consisting of α where for every k, the attachment $\alpha'^\dagger_k$ is replaced with $\beta^\dagger_k$.

Then there is a window A' on α' and two translations $\tau_N : \text{Near}(A') \to \text{Near}(A)$ and $\tau_F : \text{Far}(A') \to \text{Far}(B)$ such that:

$$\begin{cases} \forall z \in \text{Near}(A'), & (\lim\alpha')\cdot\text{tile}(z) = (\lim\alpha)\cdot\text{tile}(\tau_N(z)) \\ \forall z \in \text{Far}(A'), & (\lim\alpha')\cdot\text{tile}(z) = (\lim\beta)\cdot\text{tile}(\tau_F(z)) \end{cases} \tag{1}$$

Moreover, if $\beta^\dagger >_{\text{fz}} \alpha^\dagger$, then $\alpha' >_{\text{fz}} \alpha$.

Proof. Define α' as α where for each k, the k-th attachment $t@Z$ within $\text{Far}(A)$ has been replaced by the k-th attachment within $\text{Far}(B)$ of β. When doing this, for an attachment $t@Z$ within $\text{Far}(B)$, any arc $a = (v, d)$ in Z with $v \in \text{Near}(B)$ is replaced by the arc $\tau^{-1}(a)$.

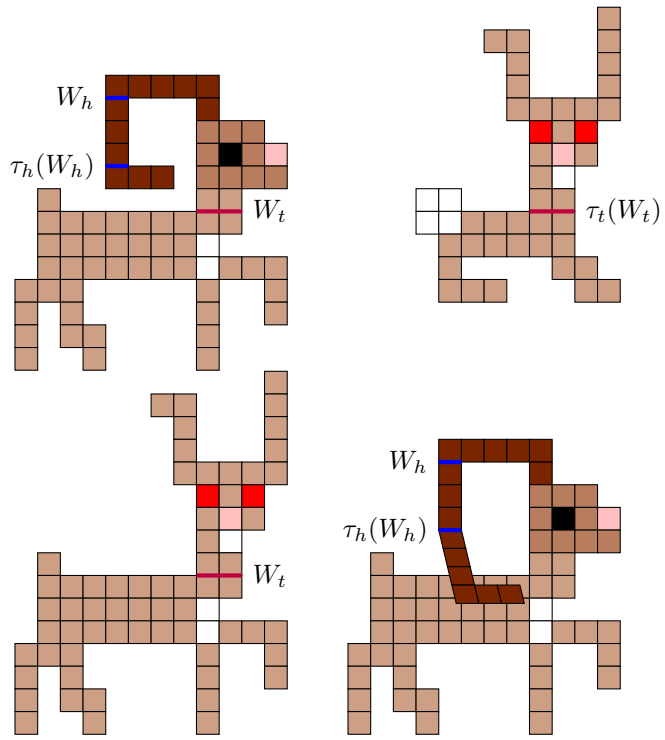


Figure 23 The Window Movie Lemma: consider two assembly sequences ι (the ibex) and β (the bunny), with their productions depicted in the top row. The sequence ι, has two windows W_t and W_h. In W_t, $\mathrm{Far}(W_t)$ is the head of the ibex, $\mathrm{Near}(W_t)$ its body. In W_h, $\mathrm{Far}(W_h)$ is the tip of the horn. Each of them is associated with a translation, τ_t and τ_h respectively, which sends the window to another window with the same movie, and satisfying the hypothesis of Lemma 74. Applying Lemma 74 in each case gives two new assembly sequences represented on the bottom row. Applying it with W_t yields a fearsome chimera, while its application to W_h yields an ibex with a horn so long it needs to bend in the third dimension to avoid piercing its spine, i.e. avoiding the conflict that would arise were our attachment sequences not free.

The translations τ_N and τ_F can be defined inductively on the attachments of α', such that Equation (1) holds.

The proof that α' is a valid free assembly sequence is the same as in the $\mathbb{Z}^2$ case. Note that since B is a cut of $(\lim \beta)\cdot$ support, any attachment of β in $\mathrm{Far}(B)$ can be replayed in α' in $\mathrm{Far}(A')$ without conflicts since they do not create any adjacency outside $\mathrm{Far}(\beta)$.

Observe that every attachment in $\mathrm{Far}(B)$ which affects holes in β affects the same number of holes in α': holes which are wholly within $\mathrm{Far}(B)$ have had all of their tiles attached in α', and holes which go through B become holes through A because the part within $\mathrm{Near}(B)$ has the same connections in $\mathrm{Near}(A)$ at that frame in the movie. ◀

The fact that the grafting process of Lemma 74 is increasing in the fizziness of the far part of the assembly sequences implies that when the original assembly sequences have maximal fizziness, so does the chimera sequence α'.

Lemma 74 will be most useful in this paper in the case where the cuts A and B are on the same branch of the assembly. In this case, by iterating Lemma 74, it is possible to get a production with a periodic subassembly.

▶ **Corollary 75.** *Let $\alpha = (A_i)_{i<o}$ be an assembly sequence with two windows A and B satisfying the hypothesis of Lemma 74 and such that $\mathrm{Far}(B) \subset \mathrm{Far}(A)$. Let τ be the translation between A and B. Then, there is an assembly sequence α^τ such that $F = \lim \alpha^\tau \cap \mathrm{Far}(A)$ verifies $\tau(F) \subset F$, and within $\mathrm{Near}(B)$, α^τ does the same attachments in the same order as α.*

Proof. Let τ be the translation sending A to B.

Show by induction that for any k, there is a sequence α^k such that the movie on the windows A and B within α^k are the same as in α, and for each $z \in \mathrm{Near}(B) \cap \mathrm{Far}(A)$ and $j \leq k$, $(\lim \alpha^k)(\tau^j(z)) = (\lim \alpha)(z)$, and the attachments done by α^k and α^j within $\bigcup_{i\leq j} \tau^i(\mathrm{Near}(B) \cap \mathrm{Far}(A))$ are the same.

For $k = 0$, it suffices to pick $\alpha^0 = \alpha$. Assume that α^k satisfies the induction hypothesis. Then Lemma 74 applies, and the assembly sequence it yields, α^{k+1}, satisfies the induction hypothesis at rank $k + 1$.

For a pair $(j, k) \in \mathbb{N}$, if for all $i \leq j$, $\alpha^k_i \in \bigcup_{l\leq k} \tau^k(\mathrm{Near}(B) \cap \mathrm{Far}(A))$, then the j first attachments are unchanging after rank k: for all $m \geq k$ and $i \leq j$, $\alpha^m_i = \alpha^k_i$. Let $\alpha^{\vec{p}}$ be the sequence of such unchanging attachments. Let m be the supremum of the $j \in \mathbb{N}$ such that the j first attachments are unchanging after some rank k. Define $\alpha^{\vec{p}}$ as the sequence of length m with $\alpha^{\vec{p}}_j = \alpha^k_j$, where k is such that the j first attachments are unchanging.

It is easy to check that indeed, $\lim \alpha^{\vec{p}} \cap \mathrm{Far}(A)$ has period $\vec{p}$: any attachment in $\alpha^{\vec{p}}$ within $\mathrm{Far}(A)$ gets picked up by subsequent applications of Lemma 74 which translate it by $\vec{p}$. ◀

In this situation with a branch and two cuts with the same movie, it is also possible to cut assembly sequences short, so that they can do their business in $\mathrm{Near}(A)$ without needing to mess with $\mathrm{Far}(B)$.

▶ **Corollary 76.** *Let α be an assembly sequence with two windows A and B satisfying the hypothesis of Lemma 74 and such that $\mathrm{Far}(B) \subset \mathrm{Far}(A)$. Then there is an assembly sequence α' which does the same attachments as α within $\mathrm{Near}(A)$, but has no attachment in $\mathrm{Far}(B)$.*

Proof. Let j be the last time that α does an attachment in $\mathrm{Near}(A)$ next to A, i.e. the date of the last frame in the movie of A where a tile is attached on the near side of the window. By Remark 67, up to a reordering of α, j is finite.

Let α^0 be the prefix of length j of α. For each k, if α^k has any attachment in $\mathrm{Far}(B)$, it is possible to use Lemma 74 between B and A to obtain a sequence α^{k+1} with fewer

attachments within $\mathrm{Far}(B)$. Hence, there is a finite k such that α^k has no attachments in $\mathrm{Far}(B)$, and $\lim \alpha^k \cap \mathrm{Near}(A) = \lim \alpha^0 \cap \mathrm{Near}(A)$. Since by time j, the movie on A is over, the rest of the attachments of α which take place in $\mathrm{Near}(A)$ can be replayed after α^k. ◀

A.5 The Tree Pump Lemma

After all these considerations about the fantastic properties of Ordinal Free Assembly Sequences, it is now time to come back to Earth, or rather $\mathbb{Z}^2$. The object is now to prove Lemma 8, for which an investigation of the properties systems whose $\mathbb{Z}^2$-assembly sequences do not circle large squares is in order.

The statement of Lemma 8 is given again, recall that it deals with the $\mathbb{Z}^2$-productions of the aTAM system $\mathcal{S}$, hence in its statement, $\mathcal{A}_\square[\mathcal{S}]$, $C_m[\mathcal{S}]$, $B_{F(n,m),\vec{d}}[\mathcal{S}]$ and $P_{\vec{d}}[\mathcal{S}]$ are sets of $\mathbb{Z}^2$-assemblies.

▶ **Lemma 8** (Tree Pump). *For any aTAM system $\mathcal{S}$ with $\mathcal{S}\cdot$seed finite and connected, define the following sets of assemblies:*

- *for any integer m, $C_m[\mathcal{S}]$ is the set of assemblies of $\mathcal{S}$ which encircle an $m \times m$ square;*
- *for any real k and vector $\vec{d}$, $B_{k,\vec{d}}[\mathcal{S}]$ is the set of assemblies of $\mathcal{S}$ which do not cover any position $\vec{p}$ such that $\vec{p} \cdot \vec{d} > k$*
- *for any vector $\vec{d}$, $P_{\vec{d}}[\mathcal{S}]$ is the set of* ultimately periodic *assemblies of $\mathcal{S}$ such that there is a vector $\vec{p}$ with $|\vec{p} \cdot \vec{d}| > 0$ and a non-empty sub-assembly $a \subseteq A$ such that $a + \vec{p} \subseteq a$.*

Then, there is a function $F : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for aTAM system $\mathcal{S}$ with n tiles and a 1-tile seed, integer m and unit vector $\vec{d}$ of $\mathbb{R}^2$,

$$\mathcal{A}_\square[\mathcal{S}] \cap (C_m[\mathcal{S}] \cup B_{F(n,m),\vec{d}}[\mathcal{S}] \cup P_{\vec{d}}[\mathcal{S}]) \neq \emptyset.$$

When considering the statement of Lemma 8, the holes of the $\mathbb{Z}^2$-productions of $\mathcal{S}$ might as well be filled in. Hence, the definition of their fill-in, illustrated on Figure 24.

▶ **Definition 77** (Fill-in). *For a subgraph $D \subset \mathbb{Z}^2$, define the* fill-in $D^\bullet$ *as the subgraph of $\mathbb{Z}^2$ induced by the positions p such that there is no infinite path from p in $\mathbb{Z}^2 \setminus D$.*

A $\mathbb{Z}^2$ assembly A which does not circle any square larger than $m \times m$ looks like a tree, as expressed by the notion of *Connected Treewidth* [3]. This tree is obtained by grouping the positions of $A\cdot$support in connected sets of size at most $2m$, known as *bags*, organized in a tree in such a way that:

- any arc of $A\cdot$support is between two positions which appear in the same bag, and
- for any position $z \in A\cdot$support, the set of bags which contain z forms a subtree.

This decomposition is represented on Figure 24.

▶ **Lemma 78.** *Let $\mathcal{S}$ an aTAM system, m an integer and A a $\mathbb{Z}^2$ assembly of $\mathcal{S}$ such that $A \notin C_m[\mathcal{S}]$.*

*Then $A\cdot$*support *has Connected Tree Width $2m$.*

Proof. Indeed, it has treewidth m: otherwise it would contain an $m \times m$ grid as a minor; that minor would have to be realized in $\mathbb{Z}^2$ as a subgraph which would encircle some $m \times m$ square. Moreover, since $P^\bullet$ does not encircle any empty position, it does not have any *geodesic cycle* of length more than 4. ◀

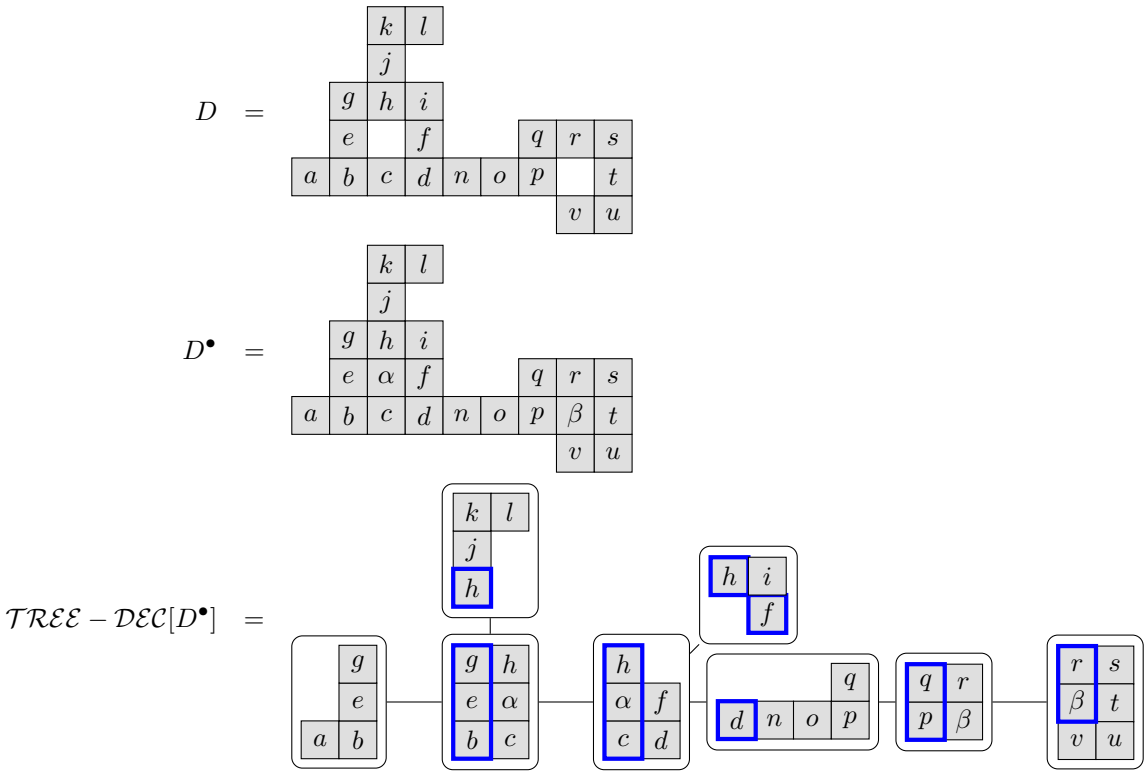


Figure 24 The domain D of a production P, its fill-in $D^\bullet$ and an associated connected tree decomposition of $\mathcal{TREE} - \mathcal{DEC}[D^\bullet]$. The blue part of each bag b of $\mathcal{TREE} - \mathcal{DEC}[D^\bullet]$ is its intersection W_b with its parent, which disconnects the vertices appearing in its subtree from the rest of $D^\bullet$. The choice of the root of $\mathcal{TREE} - \mathcal{DEC}[D^\bullet]$ is arbitrary and does not affect the sets W_b, up to renaming.

The point of using Connected Treewidth rather than the usual treewidth is that thanks to the connectivity of the bags of $\mathcal{TREE} - \mathcal{DEC}[P^\bullet]$, the distances in $\mathcal{TREE} - \mathcal{DEC}[P^\bullet]$ reflect those in $P^\bullet$:

- there is a function r such that for any vertex $v \in P^\bullet$, the subtree of the bags of $\mathcal{TREE} - \mathcal{DEC}[P^\bullet]$ containing v has a size at most $r(m)$,
- there are two increasing functions d_m, D_m such that for any vertices u, v at distance δ in $P^\bullet$, any bags $B_u \ni u$ and $B_v \ni v$ of $\mathcal{TREE} - \mathcal{DEC}[P^\bullet]$ are at distance Δ with $d_m(\delta) \leq \Delta \leq D_m(\delta)$.

As a consequence, if two tiles are far enough in an assembly A and the path between them does not go through the seed, there must be two windows cutting that path which satisfy the hypothesis of Corollary 75.

▶ **Lemma 79.** *Let $\mathcal{S}$ be an aTAM system with n tiles and m an integer.*
Let $\alpha = (A_i)_{i \leq o} \in \mathcal{H}^{\text{Free}}[\mathcal{S}, A_0]$ be such that for all $i \leq o$, $\text{squash}(A_i) \notin C_m[\mathcal{S}]$.
There is a constant $F(n,m)$ such that if $z, z' \in \text{squash}(\lim \alpha) \cdot \text{support} \setminus A_0 \cdot \text{support}$ are such that the distance between z' and z is greater than $F(n,m)$, then there are two windows A and B with $z \in \text{Near}(A)$ and $z' \in \text{Far}(B)$ which satisfy the hypothesis of Lemma 74.

Proof. let $W(n,m)$ be the number of movies with n glues on a window of width $2m$. Note that $W(n,m)$ counts the number of arrangements of the edges within the window, the perspective difference between the connected components of the window, as well as the events on the frames of the movie. Let $F(n,m) = d_m^{-1}(W(n,m))$.

Let $P = \lim \alpha \cdot \text{support}$. By Lemma 78, pick a tree decomposition $\mathcal{TREE} - \mathcal{DEC}[P^\bullet]$ with connected bags of size at most $2m$. Let δ be the distance between z and z'. If $\delta \leq F(n,m)$, by the pigeonhole principle, there must be two windows A and B between z and z' with the same movie. ◀

In the case where the assembly embeds into $\mathbb{Z}^2$, one actually controls the direction of the vector between the two windows.

▶ **Lemma 80.** *Let $\mathcal{S}$ be an aTAM system with n tiles, m an integer and $\vec{d}$ a unit vector.*
Let $\alpha = (A_i)_{i \leq o} \in \mathcal{H}^{\text{Free}}[\mathcal{S}, A_0]$ be such that for all $i \leq o$, A_i embeds into $\mathbb{Z}^2$.
There is a constant $F'(n,m)$ such that if $z, z' \in \lim \alpha \cdot \text{support} \setminus A_0 \cdot \text{support}$ are such that the distance between $(z' - z) \cdot \vec{d} > F(n,m)$, then there are two windows A and B with $z \in \text{Near}(A)$ and $z' \in \text{Far}(B)$ which satisfy the hypothesis of Lemma 74, and such that the vector $\vec{p}$ of the translation between A and B satisfies $\vec{p} \cdot \vec{d} > 1$.

Proof. Pick $F'(n,m) = 2F(n,m)+1$. If $(z'-z)\cdot\vec{d} > F'(n,m)$, then in $\mathcal{TREE} - \mathcal{DEC}[\lim \alpha \cdot \text{support}^\bullet]$ there there are two bags, one b containing z, the other $b' \ni z'$ such that on the path between b_z and $b_{z'}$, there are $2F(n,m)+1$ bags $(b_i)_i$ such that for all $x \in b_i, y \in b_j, (x-y)\cdot\vec{d} \geq i - j$. Thus, there are i, j such $j > i+1$ and two windows, A between b_i and b_{i+1}, and B between b_j and b_{j+1} which have the same movie. Because the windows A and B are separated by at least two elements of (b_i), the translation vector $\vec{p}$ between them satisfies $\vec{p} \cdot \vec{d} > 1$. ◀

Thus, there is a simple argument to prove Lemma 8: pick a sequence of maximal fizziness which produces a terminal assembly of $\mathcal{S}$ which is neither in $B_{F'(n,m),\vec{d}}[\mathcal{S}]$ nor in $C_m[\mathcal{S}]$. Lemma 80 yields two windows with the same movie; by applying Corollary 75 between them, a production in $P_{\vec{d}}[\mathcal{S}]$ appears. Alas, the sequence on which this argument is founded may simply fail to exist: the ω-sequences of maximal fizziness may fail to reach a terminal production. One could extend them in order to reach a terminal production, but may not be a sequence of maximal fizziness among these (ordinal) extensions.

Thus, the last ingredient of the proof is a set X of assembly sequences of $\mathcal{S}$ such that:

- any assembly sequence $\alpha \in X$ can be extended into a sequence $\alpha' \in X$ such that $\lim \alpha' \in \mathcal{A}_\square[\mathcal{S}]$,
- for any assembly sequence $\alpha \in X$, there is $\alpha' \in X$ such that $\alpha' \geq_{\text{fz}} \text{squash}(\alpha)X$,
- there is a sequence in $\alpha_{\max} \in X$ such that for any $\alpha \in X, \alpha_{\max} \geq_{\text{fz}} \alpha$.

For this, it is necessary to prevent the pesky tendency ordinal sequences have to try and buy time by spacing out their attachment so as to generate an infinite family of sequences with increasing fizziness. The mysterious set of assembly sequences X is the set of *straight* sequences.

▶ **Definition 81.** *Let $\mathcal{S}$ be an aTAM system, and $\prec$ an arbitrary total order on its sequences. A free, ordinal assembly sequence $\alpha = (A_i)_{i<o} \in \mathcal{H}^{\text{Free}}[\mathcal{S}]$ is* $\prec$-straight *if for every pair of window N, F such that N and F have the same movie and $A_0 \cdot \text{support} \subset \text{Near}(N)$, α is the smallest sequence for $\prec$ obtained by applying Corollary 75 to N and F in α.*

The order $\prec$ is henceforth fixed and left implicit.

▶ **Lemma 82.** *Let $\mathcal{S}$ be an aTAM system and w an integer.*

Let $X \subset \mathcal{H}^{\text{Free}}[\mathcal{S}]$ be a set of assembly sequences, and d an integer. If for any sequence $\alpha \in X$ and any position $z \in \lim \alpha \cdot \text{support}$ at distance more than d from $\mathcal{S} \cdot \text{seed}$, there are two windows A and B such that $\mathcal{S} \cdot \text{seed} \cdot \text{support} \subset \text{Near}(A)$ and $z \in \text{Far}(B)$, then there is a finite number of straight sequences in X.

Proof. In this case, each straight sequence in X is determined by what it does in a radius d around $\mathcal{S} \cdot \text{seed}$. ◀

▶ **Lemma 83.** *let $\mathcal{S}$ be an aTAM system and α a straight assembly sequence of $\mathcal{S}$. There is a straight assembly sequence α' which extends α such that $\lim \alpha' \in \mathcal{A}_\square[\mathcal{S}]$.*

Proof. Let $P = \lim \alpha$. If $P \notin \mathcal{A}_\square[\mathcal{S}]$, then there is an attachment $t@Z$ which is possible in P.

Let α' be α with $t@Z$ appended at its end, and z the position of that last attachment in $\lim \alpha' \cdot \text{support}$. Assume that Z is such that the distance between $\text{seed}\,\mathcal{S} \cdot \text{support}$ and z is minimal.

If there are no pairs of windows (N, F) in α' such that $\text{Far}(F) \subset \text{Far}(N)$, $\mathcal{MOVIE}(N) = \mathcal{MOVIE}(F)$, $\text{seed}\,\mathcal{S} \cdot \text{support} \subset \text{Near}(N)$ and $z \in \text{Far}(N)$, then α' is straight.

If there is a pair of windows (N, F) in α' such that $\text{Far}(F) \subset \text{Far}(N)$, $\mathcal{MOVIE}(N) = \mathcal{MOVIE}(F)$, $\text{seed}\,\mathcal{S} \cdot \text{support} \subset \text{Near}(N)$ and $z \in \text{Far}(F)$, then by Corollary 76, there is a place closer to $\text{seed}\,\mathcal{S}$ than z where an attachment was possible.

Lastly, if there is a pair of windows (N, F) in α' such that $\text{Far}(F) \subset \text{Far}(N)$, $\mathcal{MOVIE}(N) = \mathcal{MOVIE}(F)$, $\text{seed}\,\mathcal{S} \cdot \text{support} \subset \text{Near}(N)$ and $z \in \text{Far}(N) \cap \text{Near}(F)$, then the sequence α'' obtained by applying 75 in α' is straight and has α as a prefix, since the new attachments (the repetitions of $t@Z$) are done last in each repetition of the movie in $\text{Far}(N) \cap \text{Near}(F)$. ◀

▶ **Lemma 84.** *Let $\mathcal{S}$ be an aTAM system and α a straight assembly sequence of $\mathcal{S}$. Then* $\text{squash}(\alpha)$ *is a prefix of a straight sequence α'. Moreover, the connected treewidth of* $\lim \alpha' \cdot \text{support}$ *is no greater than that of* $(\lim \text{squash}(\alpha)) \cdot \text{support}$.

Proof. If α embeds cleanly in $\mathbb{Z}^2$, there is nothing to prove since $\text{squash}(\alpha) = \alpha$.

Otherwise, $\text{squash}(\alpha)$ stops because one of its attachment $t@z$ creates a hole which does not exist in α. Because α is straight, there cannot be a pair of windows with the same

movie separating $\mathcal{S}\cdot$ seed and z. Applying Corollary 75 in $\mathrm{squash}(\alpha)$ on each pair of windows with the same movie which are the closest to $\mathcal{S}\cdot$ seed yields a straight sequence α' extending $\mathrm{squash}(\alpha)$.

Fix a tree decomposition of $(\lim \mathrm{squash}(\alpha))\cdot$ support. There is a tree decomposition of $\lim \alpha'\cdot$ support where each bag which lies in the far part of a pair of windows with the same movie is a copy of the corresponding bag close to the seed. This tree decomposition has the same width as the decomposition of $(\lim \mathrm{squash}(\alpha))\cdot$ support. ◀

At last, all the ingredients are there to prove Lemma 8.

Proof of Lemma 8. Define a sequence (α_i) of straight assembly sequences as follows:

- Fix α_0 to be the assembly sequence with zero attachments: $\alpha_0 = (\mathcal{S}\cdot \text{seed})$;
- for i even, α_{i+1} is an extension of α_i which reaches a terminal production;
- for i odd, α_{i+1} is obtained from α_i by Lemma 84.

For every i, either $\alpha_{i+1} = \alpha_i$ or $\alpha_{i+1} >_{\mathrm{fz}} \alpha_i$: at the even steps, if $\alpha_{i+1} \neq \alpha_i$, then it is a proper extension and is thus more fizzy; at the odd steps, if α_i does not embed into $\mathbb{Z}^2$, then $\mathrm{squash}(\alpha_i)$ is more fizzy, and so is its extension α_{i+1}.

If $\mathcal{A}_\square[\mathcal{S}] \cap C_m[\mathcal{S}] = \emptyset$, then for each i odd, the Connected Treewidth of $\mathrm{squash}(\alpha_i)$ is at most $2m$, hence the Connected Treewidth of α_{i+1} is at most $2m$. But then, by Lemma 82, α_i can only take a finite number of different values for i even.

Thus, the sequence (α_i) is eventually stationnary, and its fixpoint β is a straight sequence which embeds into $\mathbb{Z}^2$ and reaches a terminal production. If $\mathcal{A}_\square[\mathcal{S}] \cap B_{F(n,m),\vec{d}} \neq \emptyset$, this terminal production must reach at least $F(n,m)$ in direction $\vec{d}$. Thus $\lim \beta\cdot$ support has a branch with two windows A, B such that $\mathcal{MOVIE}(A) = \mathcal{MOVIE}(B)$ and the vector sending A to B verifies $\vec{p}\cdot\vec{d} > 1$ (by Lemma 80). Because β is straight, that branch is periodic, thus $\lim \beta \in \mathcal{A}_\square[\mathcal{S}] \cap P_{\vec{d}}$. ◀

References

1 Kimberly Barth, David Furcy, Scott M. Summers, and Paul Totzke. Scaled tree fractals do not strictly self-assemble. In Oscar H. Ibarra, Lila Kari, and Steffen Kopecki, editors, *Unconventional Computation and Natural Computation - 13th International Conference, UCNC 2014, London, ON, Canada, July 14-18, 2014, Proceedings*, volume 8553 of *Lecture Notes in Computer Science*, pages 27–39. Springer. URL: https://doi.org/10.1007/978-3-319-08123-6_3, doi:10.1007/978-3-319-08123-6_3.

2 Matthew Cook, Yunhui Fu, and Robert Schweller. Temperature 1 self-assembly: Deterministic assembly in 3d and probabilistic assembly in 2d. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 570–589. Society for Industrial and Applied Mathematics. URL: https://epubs.siam.org/doi/10.1137/1.9781611973082.45, doi:10.1137/1.9781611973082.45.

3 Reinhard Diestel and Malte Müller. Connected tree-width. *Comb.*, 38(2):381–398, 2018. URL: https://doi.org/10.1007/s00493-016-3516-5, doi:10.1007/S00493-016-3516-5.

4 Bruno Durand, Andrei E. Romashchenko, and Alexander Shen. Fixed-point tile sets and their applications. *J. Comput. Syst. Sci.*, 78(3):731–764, 2012. URL: https://doi.org/10.1016/j.jcss.2011.11.001, doi:10.1016/J.JCSS.2011.11.001.

5 Jérôme Durand-Lose, Jacob Hendricks, Matthew J. Patitz, Ian Perkins, and Michael Sharp. Self-assembly of 3-d structures using 2-d folding tiles. *Nat. Comput.*, 19(2):337–355, 2020. URL: https://doi.org/10.1007/s11047-019-09751-9, doi:10.1007/S11047-019-09751-9.

6 David Furcy and Scott M. Summers. Scaled pier fractals do not strictly self-assemble. 16(2):317–338. doi:10.1007/s11047-015-9528-z.

7 Chaim Goodman-Strauss. Matching rules and substitution tilings. *Annals of Mathematics*, pages 181–223, 1998.

8 Daniel Hader, Matthew Patitz, and Scott Summers. Fractal dimension of assemblies in the abstract tile assembly model. *Natural Computing*, pages 1–16, 04 2023. doi:10.1007/s11047-023-09942-5.

9 Daniel Hader, Matthew J. Patitz, and Scott M. Summers. Fractal dimension of assemblies in the abstract tile assembly model. In Irina Kostitsyna and Pekka Orponen, editors, *Unconventional Computation and Natural Computation*, Lecture Notes in Computer Science, pages 116–130. Springer International Publishing. doi:10.1007/978-3-030-87993-8_8.

10 Jacob Hendricks, Meagan Olsen, Matthew J. Patitz, Trent A. Rogers, and Hadley Thomas. Hierarchical self-assembly of fractals with signal-passing tiles. 17(1):47–65. URL: https://doi.org/10.1007/s11047-017-9663-9, doi:10.1007/S11047-017-9663-9.

11 Jacob Hendricks, Joseph Opseth, Matthew J. Patitz, and Scott M. Summers. Hierarchical growth is necessary and (sometimes) sufficient to self-assemble discrete self-similar fractals. 19(2):357–374. doi:10.1007/s11047-019-09777-z.

12 Jacob Hendricks, Matthew J. Patitz, Trent A. Rogers, and Scott M. Summers. The power of duples (in self-assembly): It's not so hip to be square. 743:148–166. URL: https://doi.org/10.1016/j.tcs.2015.12.008, doi:10.1016/J.TCS.2015.12.008.

13 Steven M. Kautz and Brad Shutters. Self-assembling rulers for approximating generalized sierpinski carpets. 67(2):207–233. URL: https://doi.org/10.1007/s00453-012-9691-x, doi:10.1007/S00453-012-9691-X.

14 Paul Lafargue. *Le droit à la paresse*. La découverte, la découverte edition.

15 James I. Lathrop, Jack H. Lutz, and Scott M. Summers. Strict self-assembly of discrete sierpinski triangles. *Theor. Comput. Sci.*, 410(4-5):384–405, 2009. URL: https://doi.org/10.1016/j.tcs.2008.09.062, doi:10.1016/J.TCS.2008.09.062.

16 Jack H. Lutz and Brad Shutters. Approximate self-assembly of the sierpinski triangle. 51(3):372–400. URL: https://doi.org/10.1007/s00224-011-9345-4, doi:10.1007/S00224-011-9345-4.

17 Pierre-Etienne Meunier, Matthew J. Patitz, Scott M. Summers, Guillaume Theyssier, Andrew Winslow, and Damien Woods. Intrinsic universality in tile self-assembly requires cooperation. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, SODA '14, pages 752–771. Society for Industrial and Applied Mathematics.

18 Shahar Mozes. Tilings, substitution systems and dynamical systems generated by them. *Journal d'analyse mathématique*, 53(1):139–186, 1989.

19 Matthew J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Nat. Comput.*, 13(2):195–224, 2014. URL: https://doi.org/10.1007/s11047-013-9379-4, doi:10.1007/S11047-013-9379-4.

20 Matthew J. Patitz and Scott M. Summers. Self-assembly of discrete self-similar fractals. 9(1):135–172. URL: https://doi.org/10.1007/s11047-009-9147-7, doi:10.1007/S11047-009-9147-7.

21 Erik Winfree, Furong Liu, Lisa A. Wenzler, and Nadrian C. Seeman. Design and self-assembly of two-dimensional DNA crystals. 394(6693):539–544. Publisher: Nature Publishing Group UK London. URL: https://www.nature.com/articles/28998.