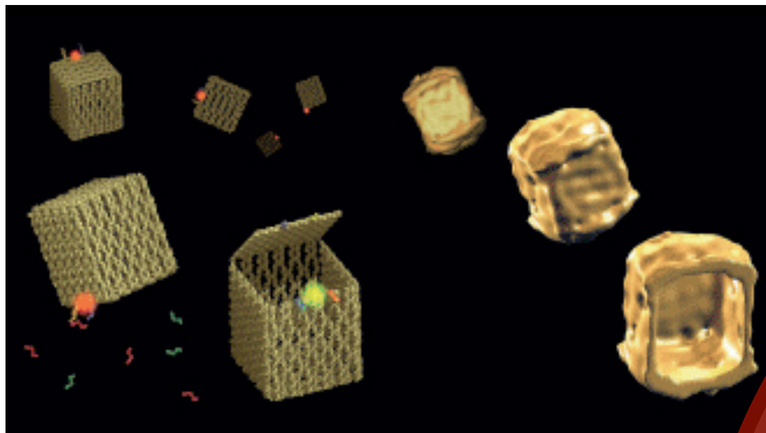
Oritatami:
A computational model for cotranscriptional folding

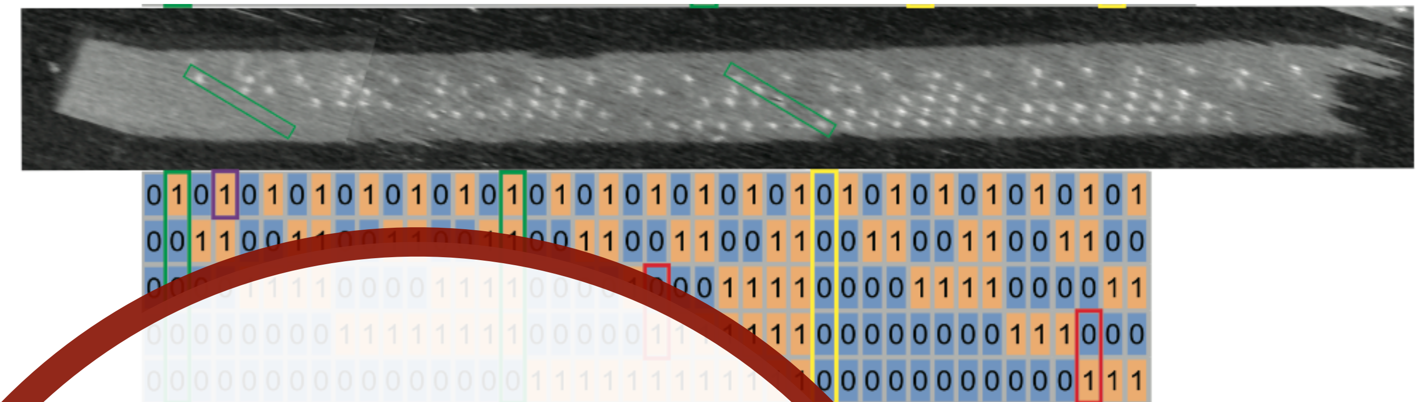
Nicolas **Schabanel**
CNRS - LIP, ENS Lyon & IXXI - France

Context: Biomolecular Computing & Engineering

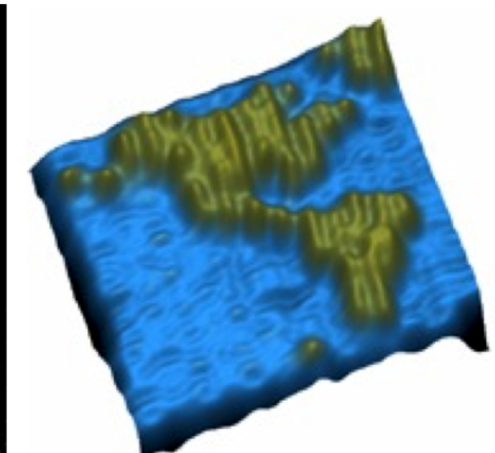
~100 nm



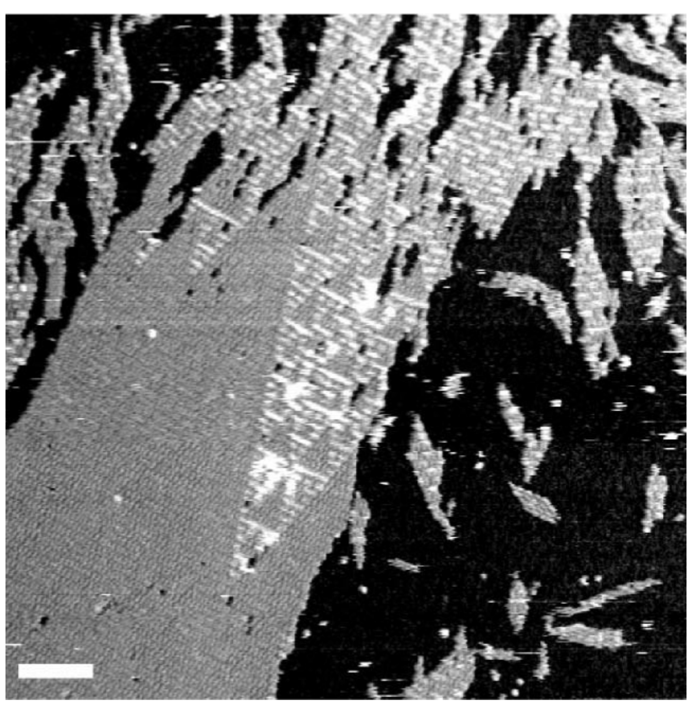
Andersen et al, 200



Constantine Evans, PhD Thesis, Caltech 2014

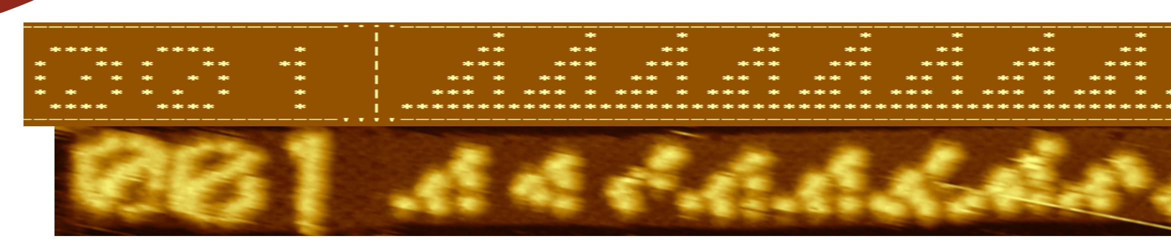


Rothemund, Nature 2006

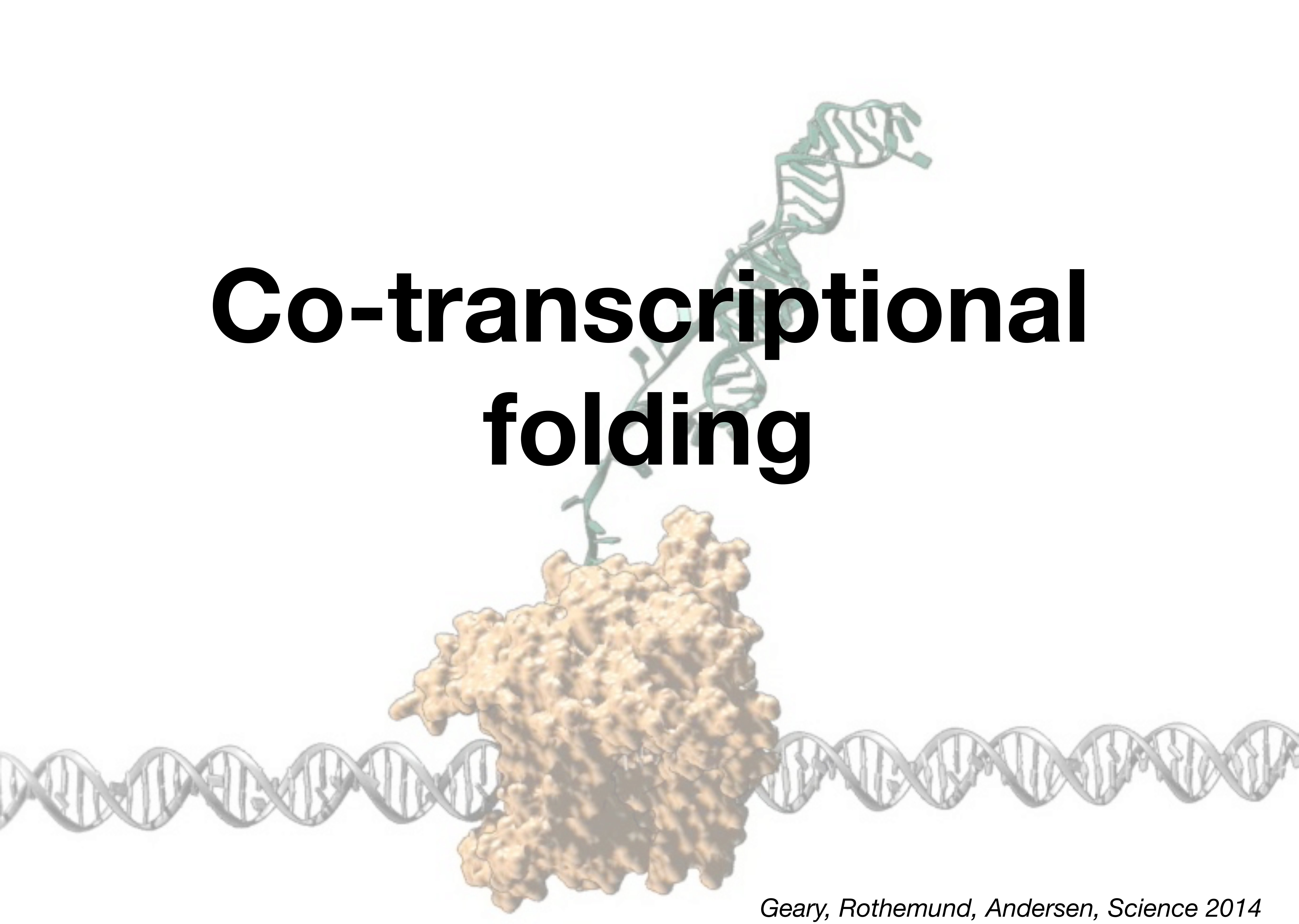


Fujibayashi et al, 2007

Han et al, Science 2011



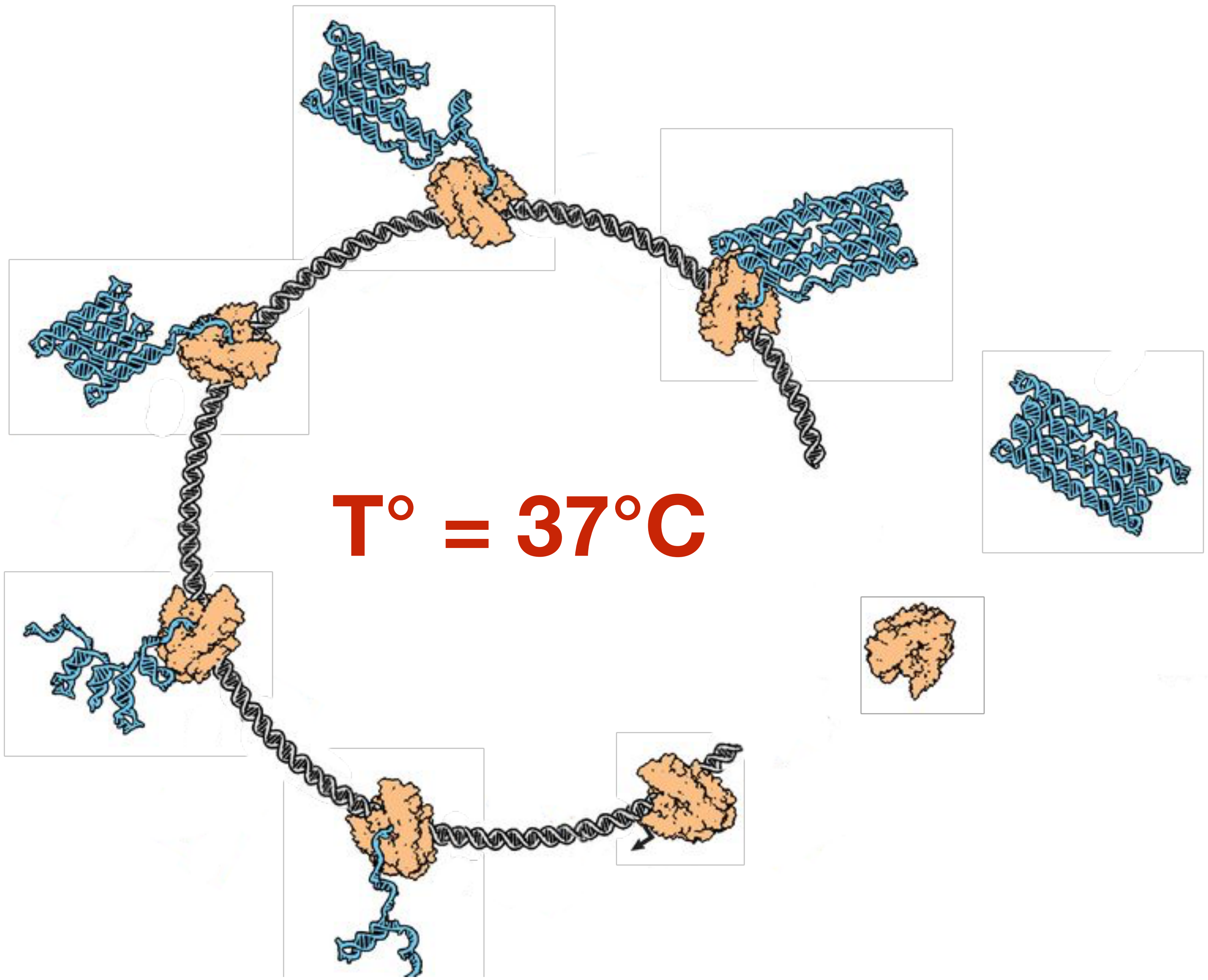
Rule 110 on input 001 - Woods et al, Nature 2019

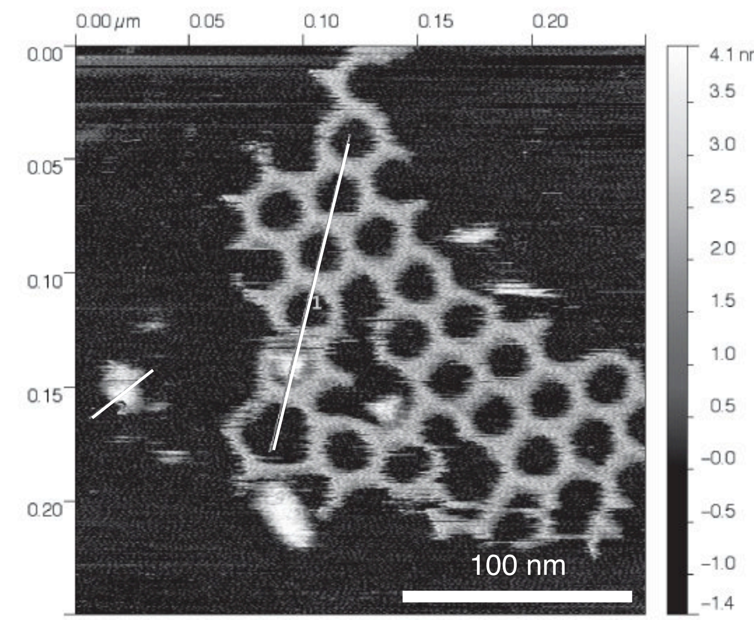


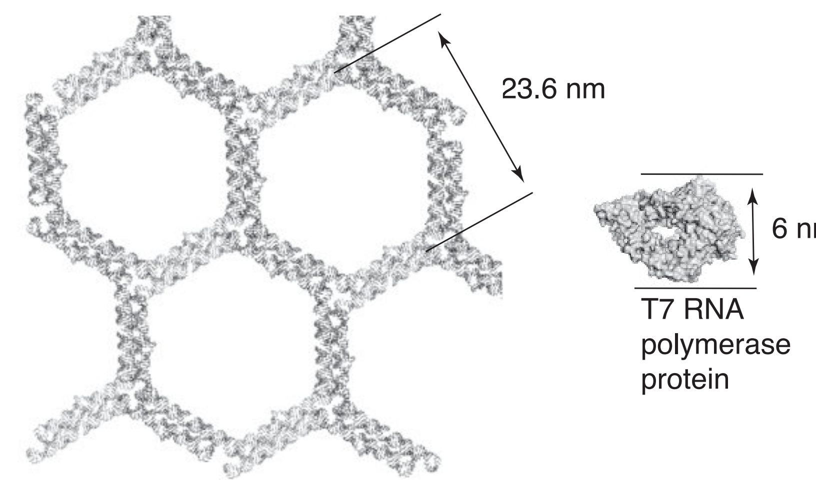
Co-transcriptional folding

Geary, Rothemund, Andersen, Science 2014

RNA
co-transcriptional folding

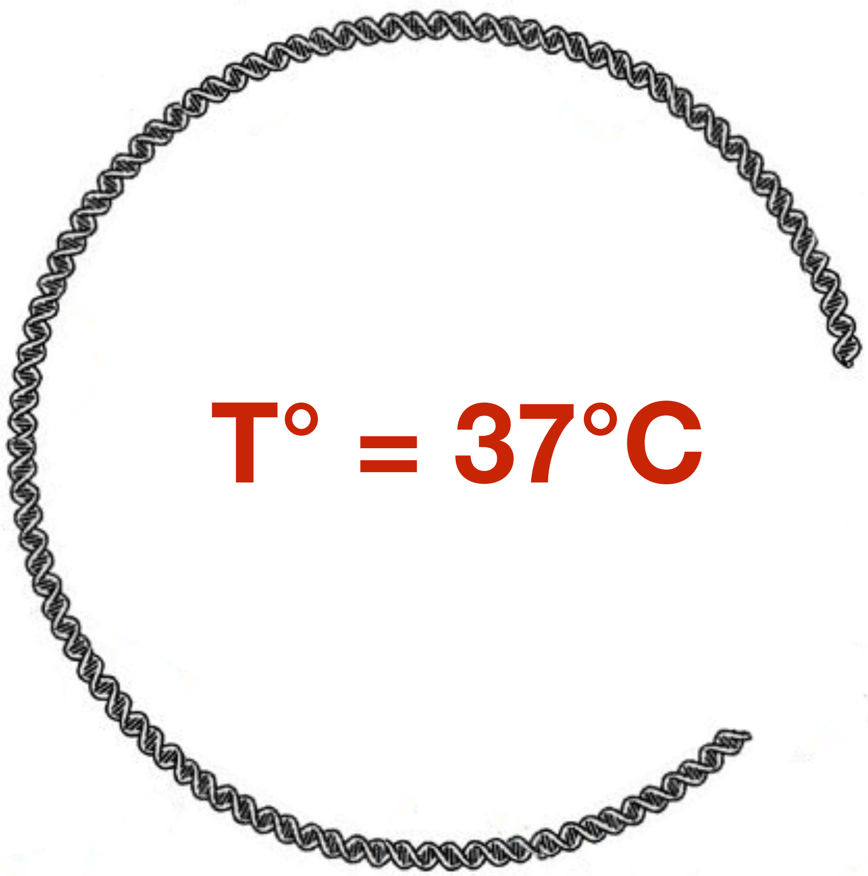




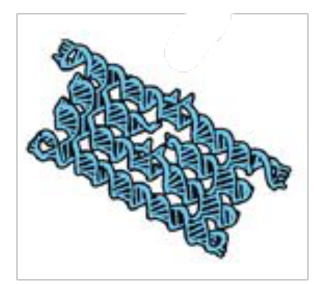


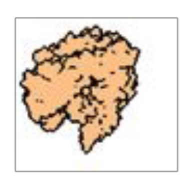
Geary, Rothemund, Andersen, Science 2014

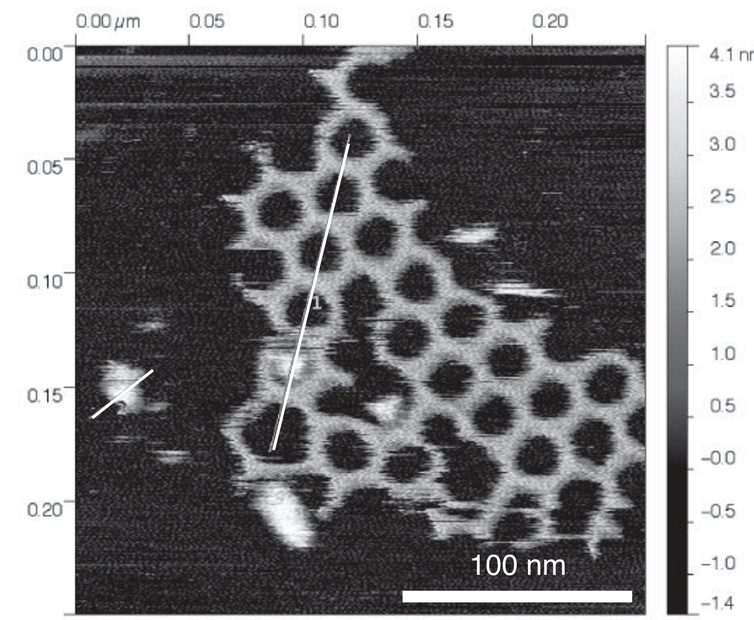
RNA
co-transcriptional folding

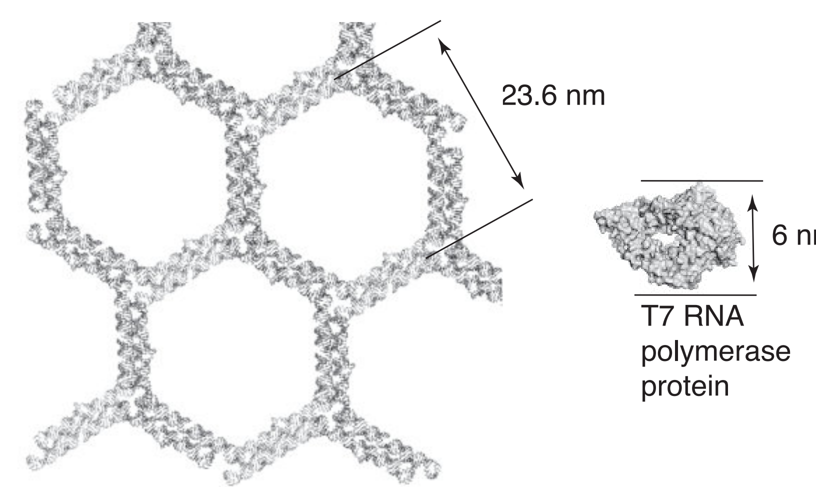


T° = 37°C



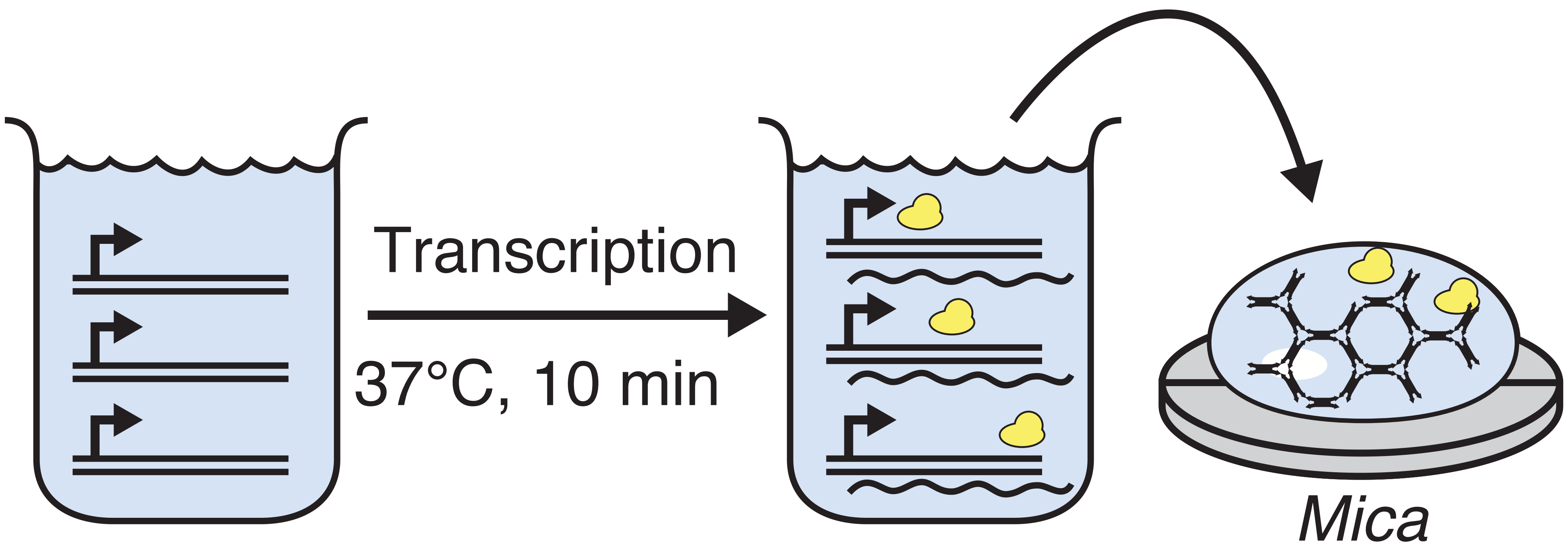




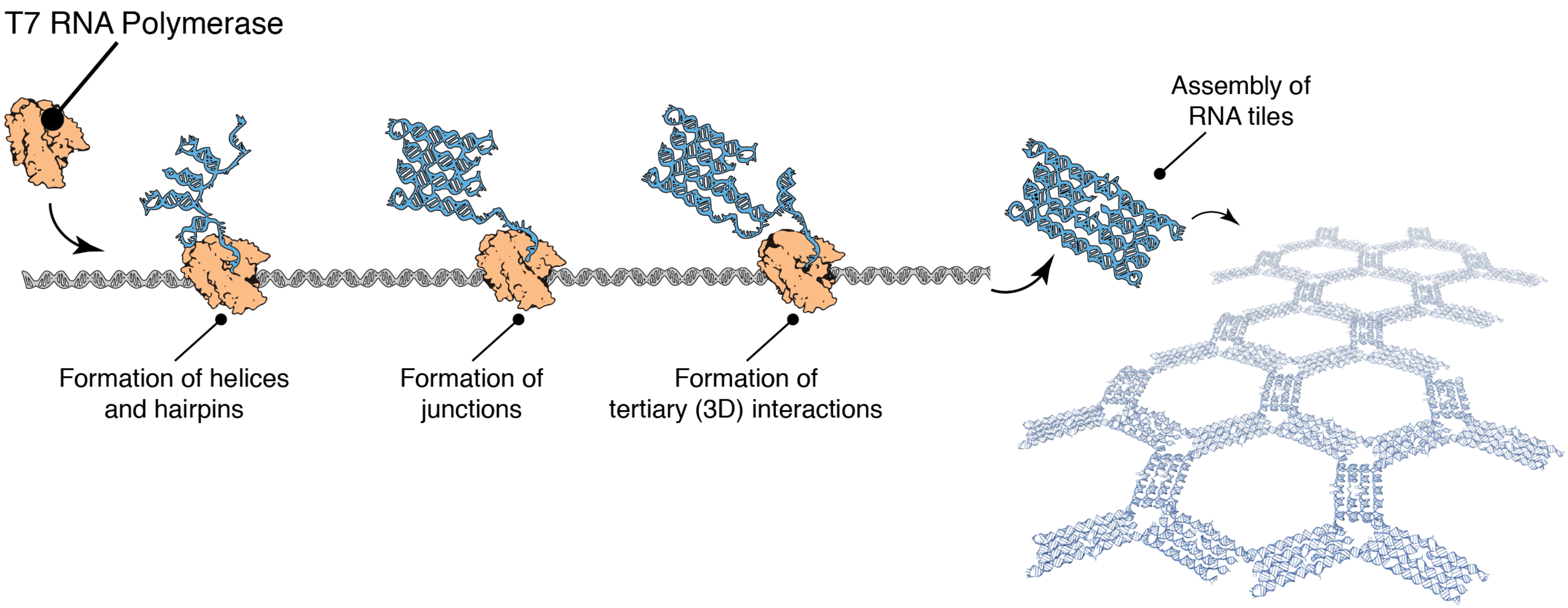


Geary, Rothemund, Andersen, Science 2014

Protocol



RNA Origami in Real Time



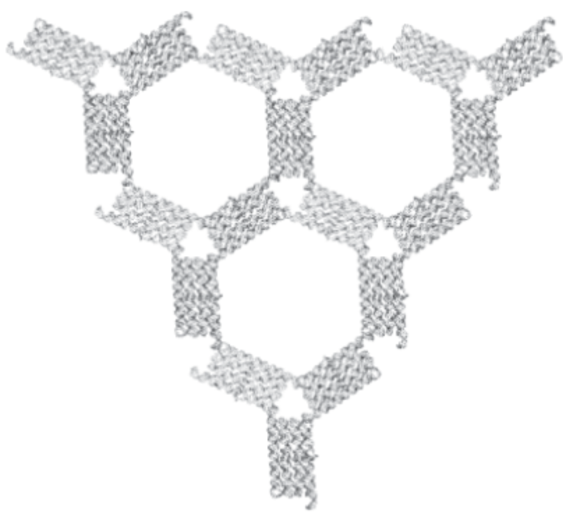
T7 RNA polymerase produces RNA directionally from 5’ to 3’, **at a rate much slower than the RNA folds up (few microseconds).**

The polymerase reads the DNA gene, and becomes an RNA origami production factory, **synthesizing a new RNA origami roughly every 1 second.**

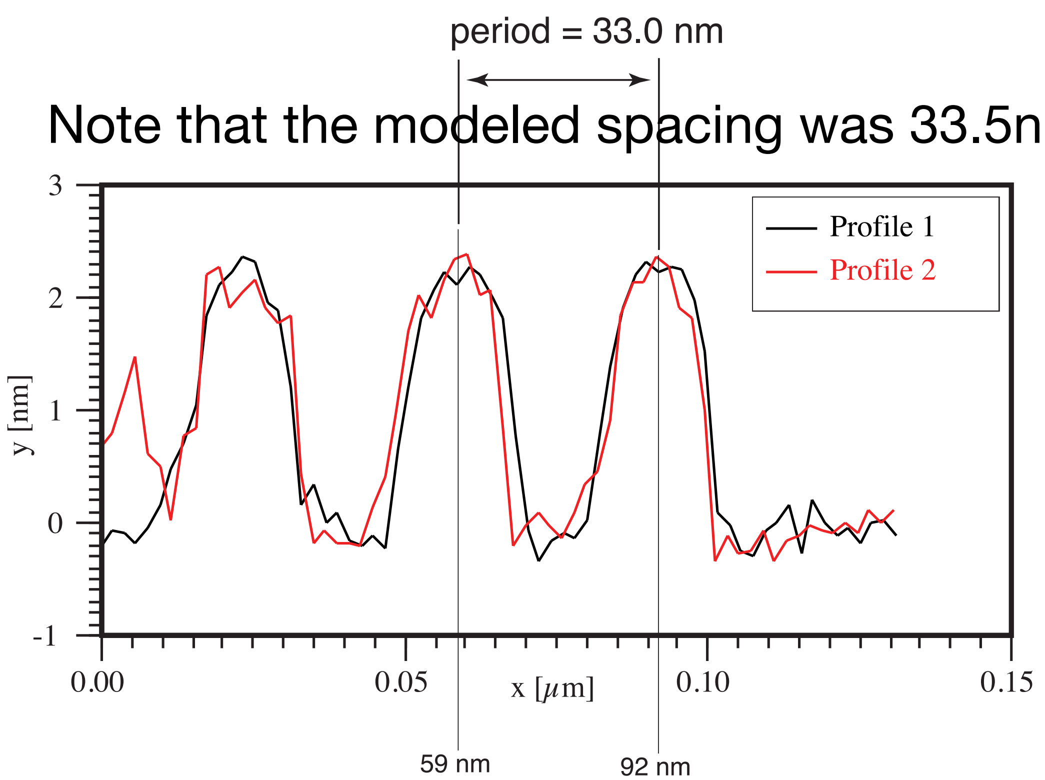
Slide by Cody Geary

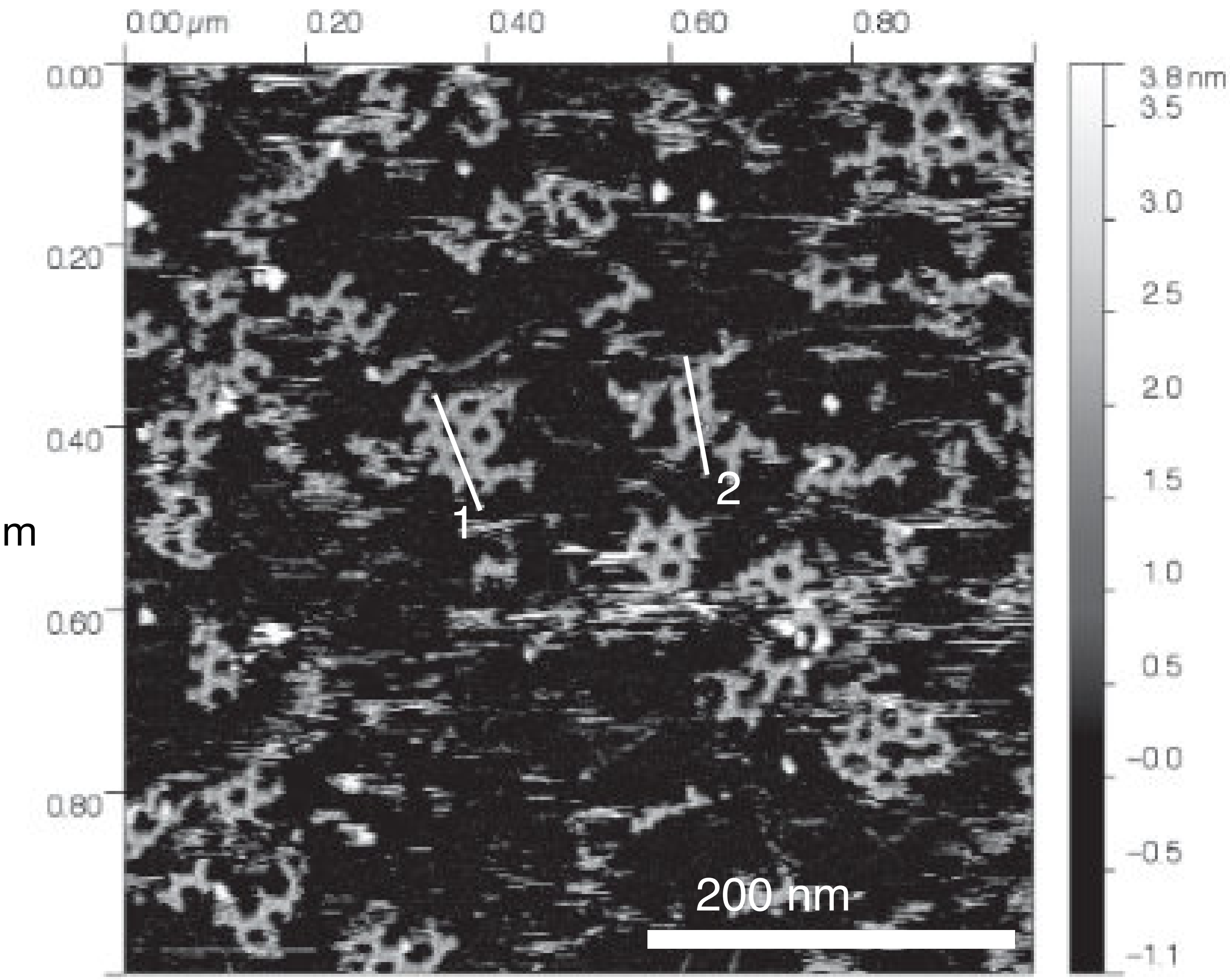
Westhof and Leontis (Science, 2014)

AFM imaging of 4H-AE co-transcriptional assembly



Note that the modeled spacing was 33.5nm

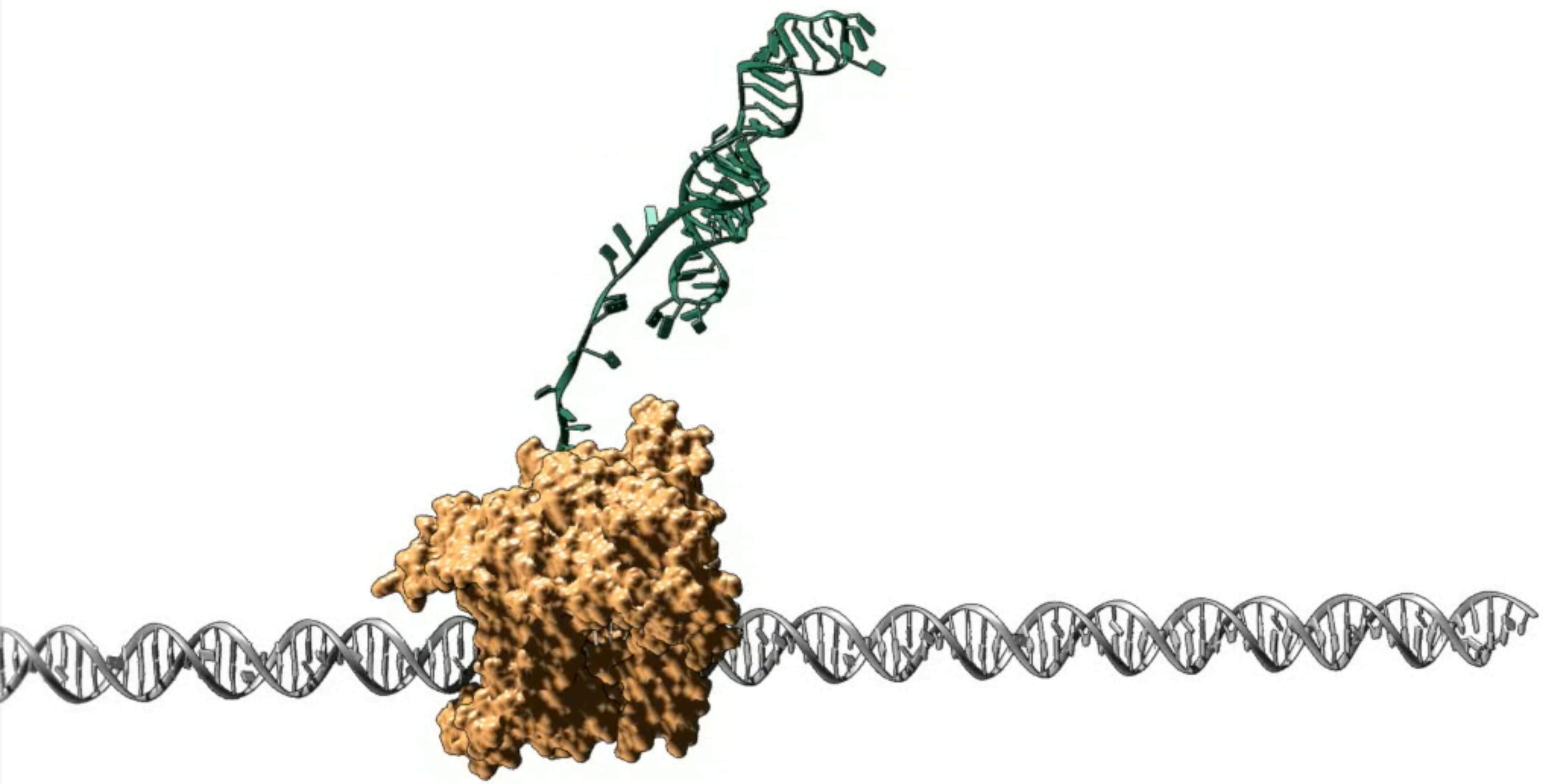




Geary et al (2014) Science

RNA Folding

(Real time: ~1 second)



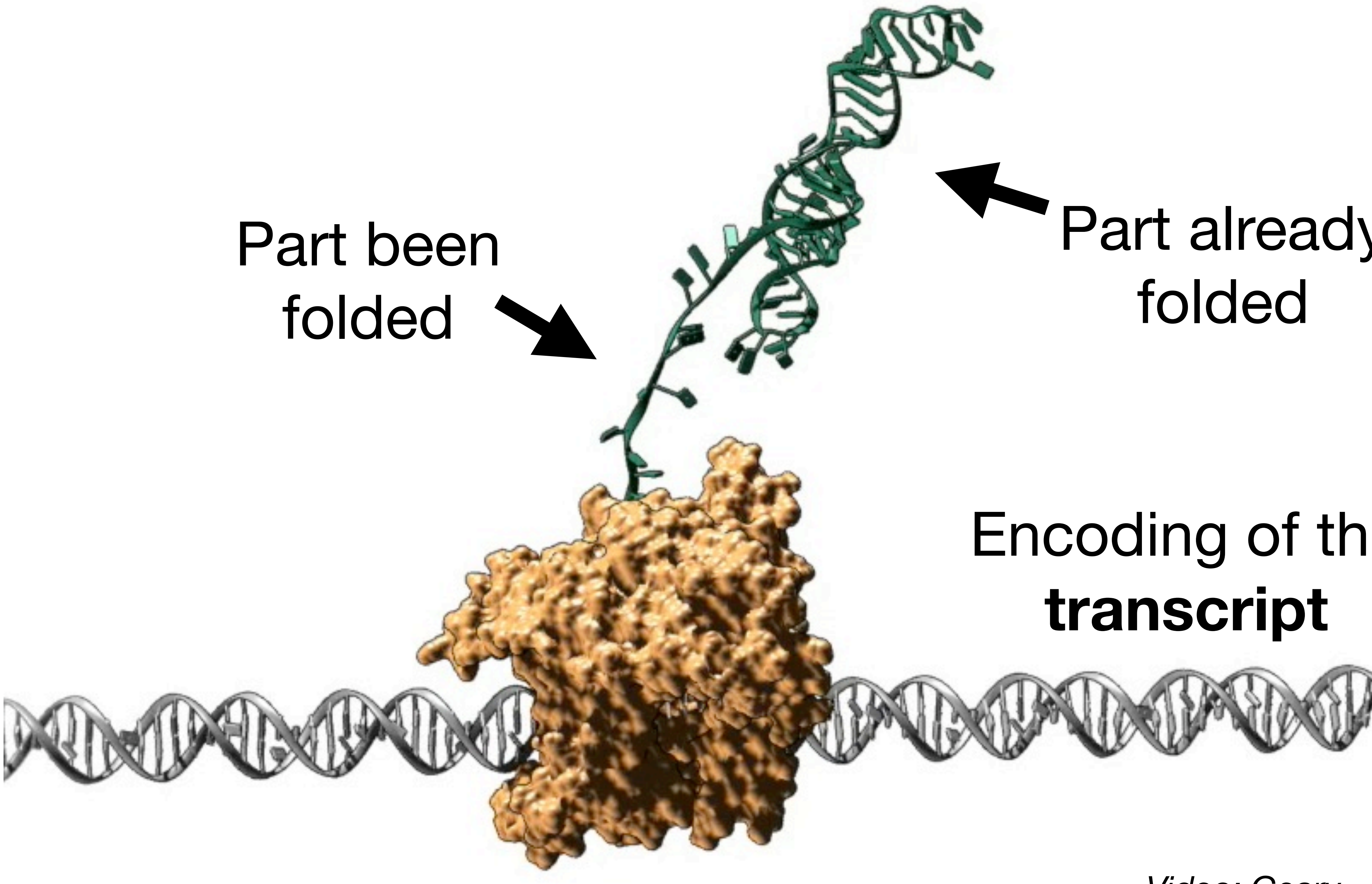
Video: Geary

Oritatami:
A computational model for co-transcriptional folding

Geary, Meunier, Schabanel, Seki MFCS 2016

RNA Folding

(Real time: ~1 second)



Video: Geary

Oritatami:
A model for co-transcriptional folding

The program:

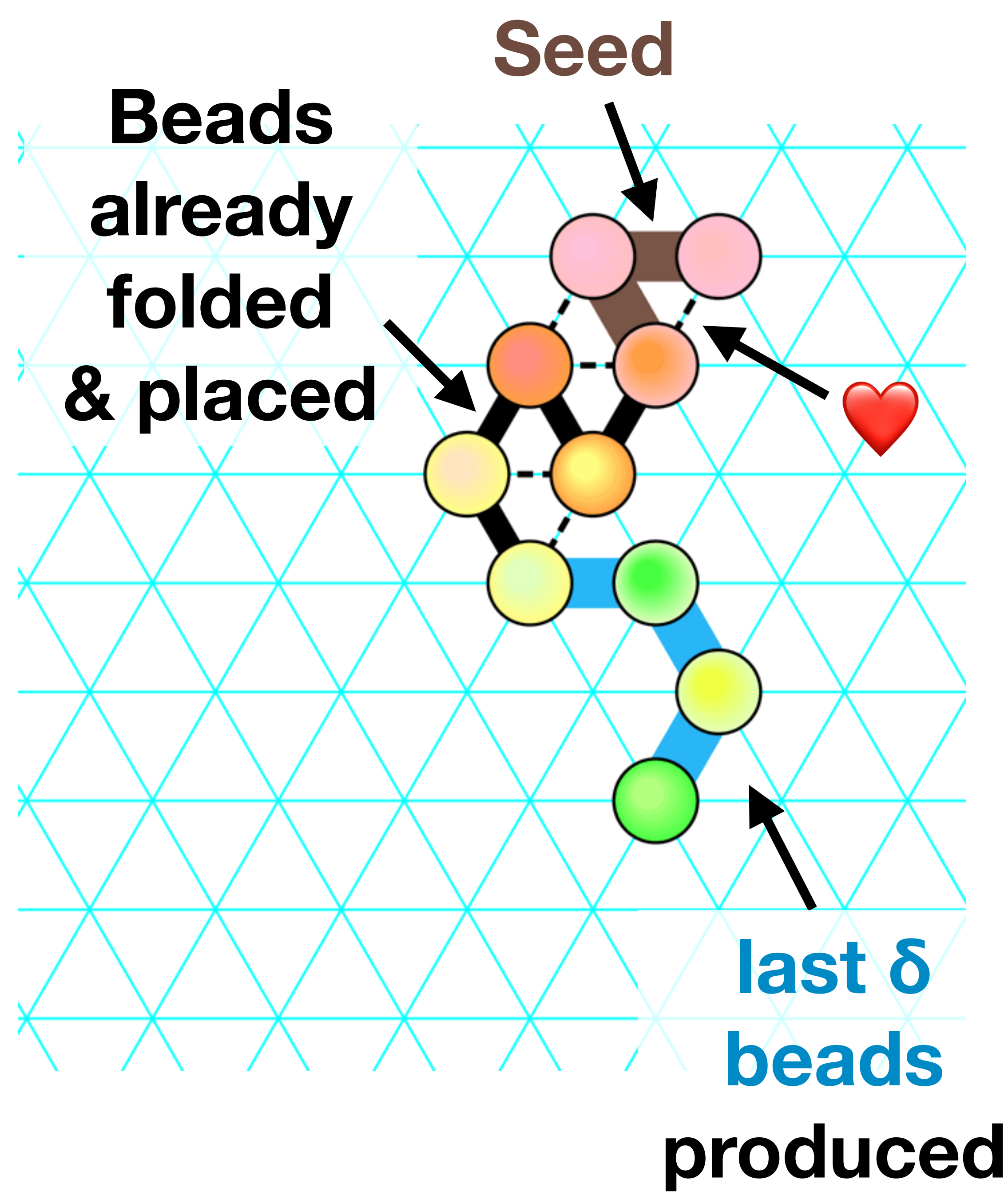
- a **sequence** of **bead types** (the **transcript**)

The instructions:

- the **rule** **a**❤️**b** if bead types **a** and **b** attract each other

The input configuration:

- Some beads placed beforehand (the **seed**)



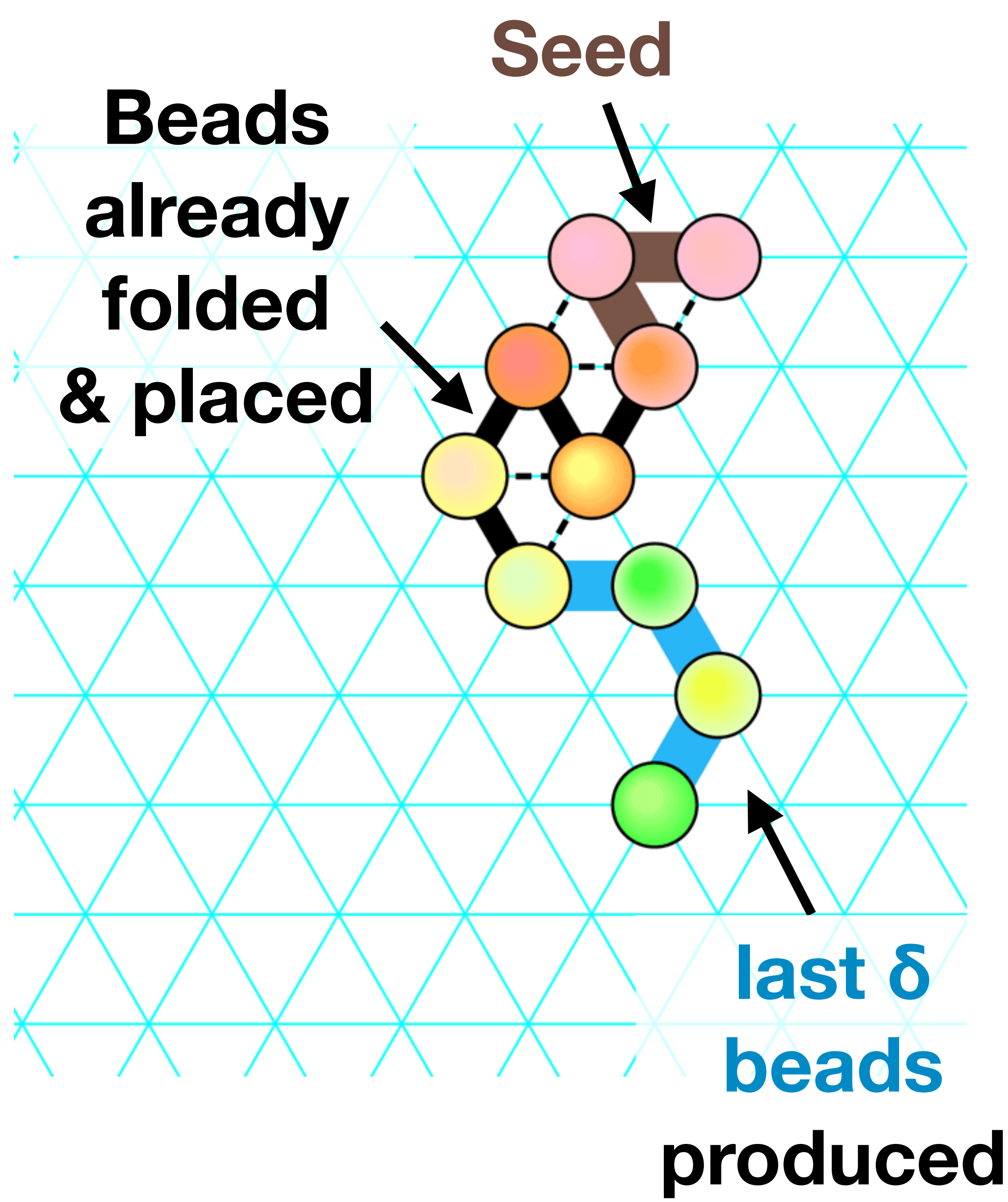
Geary, Meunier, Schabanel, Seki MFCS 2016

Oritatami: A model for co-transcriptional folding

The dynamics

- Starting from the seed, the sequence is *produced one bead at a time*
- **Only the δ last produced beads** are free to move and explore the accessible positions to settle in the ones **maximizing the number of bonds**
- All other beads remain in their last locations

here, delay **δ = 3**

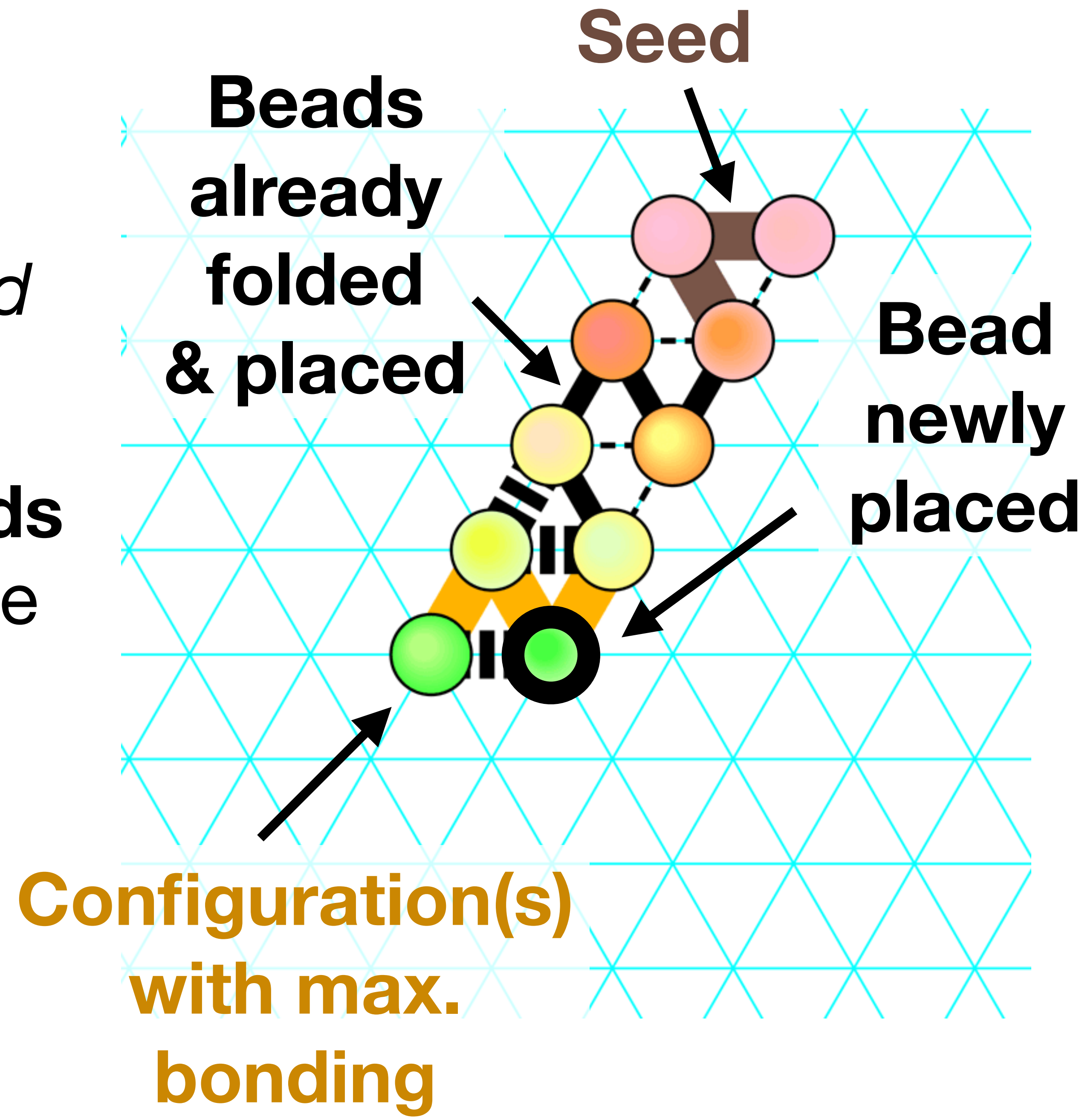


Geary, Meunier, Schabanel, Seki MFCS 2016

Oritatami:
A model for co-transcriptional folding

The dynamics.

- Starting from the seed, the sequence is *produced one bead at a time*
- **Only the δ last produced beads** are free to move and explore the accessible positions to settle in the ones **maximizing the number of bonds**
- All other beads remain in their last locations

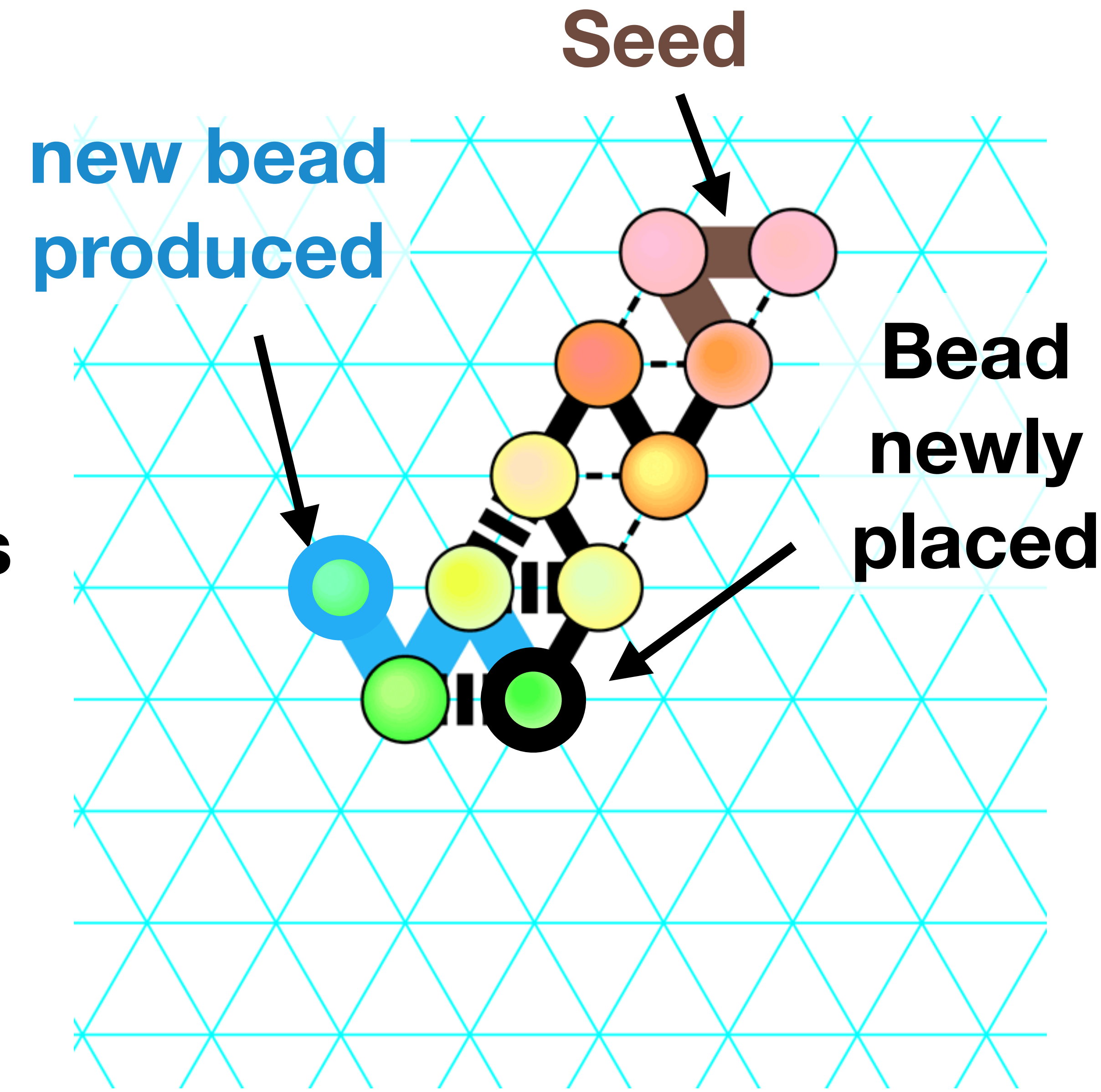


Geary, Meunier, Schabanel, Seki MFCS 2016

Oritatami:
A model for co-transcriptional folding

The dynamics.

- Starting from the seed, the sequence is *produced one bead at a time*
- **Only the δ last produced beads** are free to move and explore the accessible positions to settle in the ones **maximizing the number of bonds**
- All other beads remain in their last locations

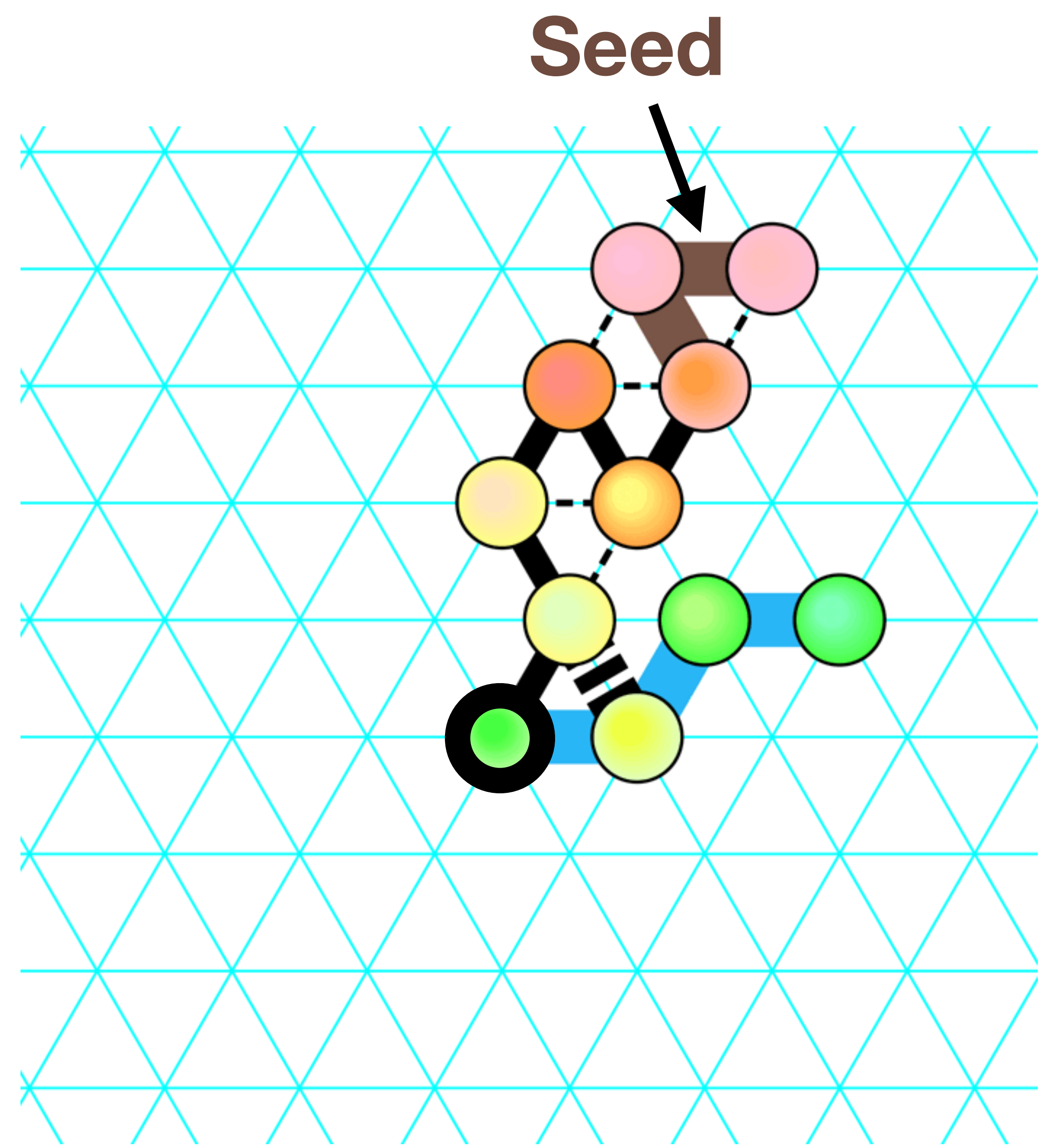


Geary, Meunier, Schabanel, Seki MFCS 2016

Oritatami:
A model for co-transcriptional folding

The dynamics.

- Starting from the seed, the sequence is *produced one bead at a time*
- **Only the δ last produced beads** are free to move and explore the accessible positions to settle in the ones **maximizing the number of bonds**
- All other beads remain in their last locations

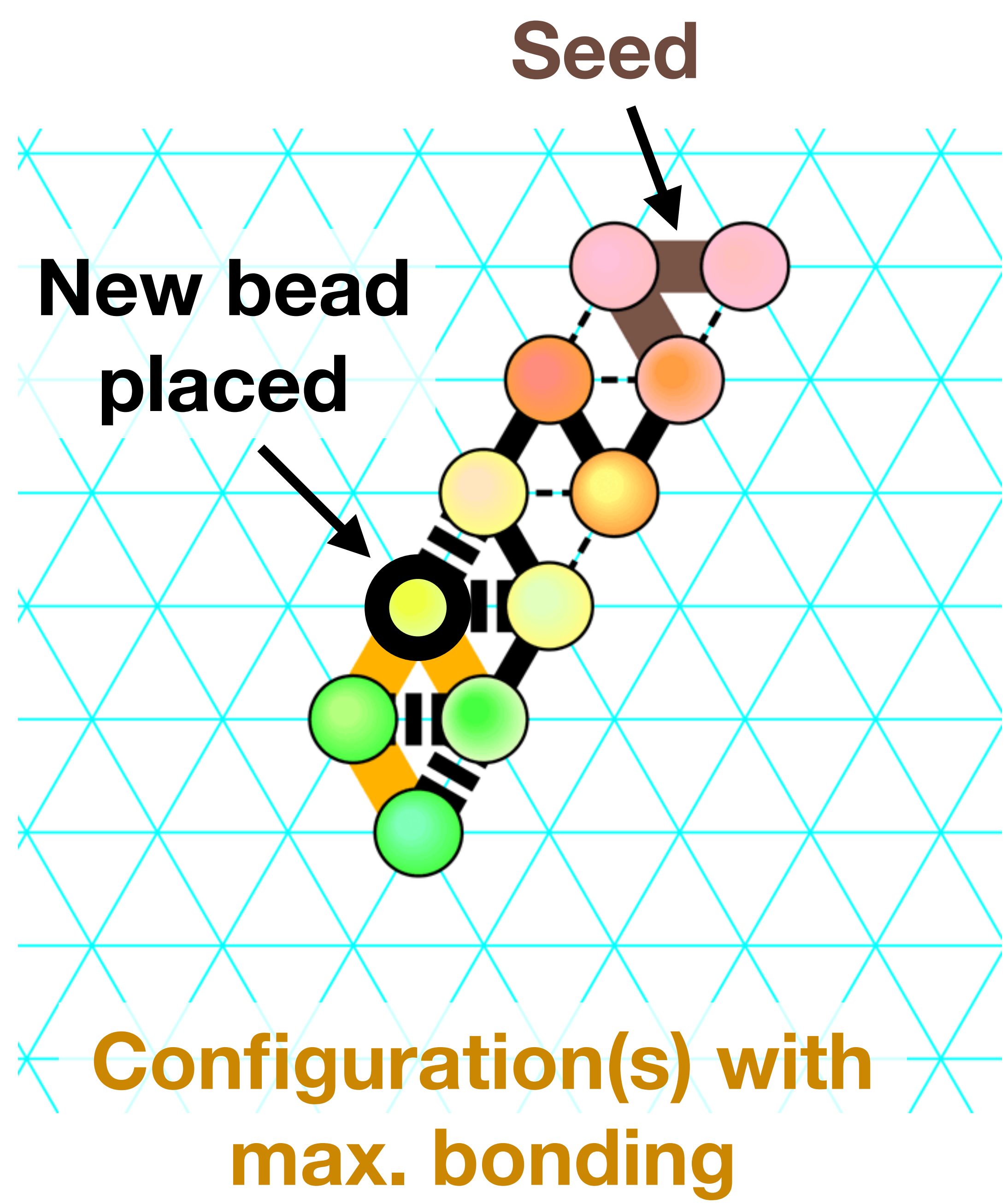


Geary, Meunier, Schabanel, Seki MFCS 2016

Oritatami:
A model for co-transcriptional folding

The dynamics.

- Starting from the seed, the sequence is *produced one bead at a time*
- **Only the δ last produced beads** are free to move and explore the accessible positions to settle in the ones **maximizing the number of bonds**
- All other beads remain in their last locations



Geary, Meunier, Schabanel, Seki MFCS 2016

Oritatami:
A model for co-transcriptional folding

The dynamics.

- Starting from the seed, the sequence is *produced one bead at a time*
- **Only the δ last produced beads** are free to move and explore the accessible positions to settle in the ones **maximizing the number of bonds**
- All other beads remain in their last locations

There might be several configurations with max. bonding

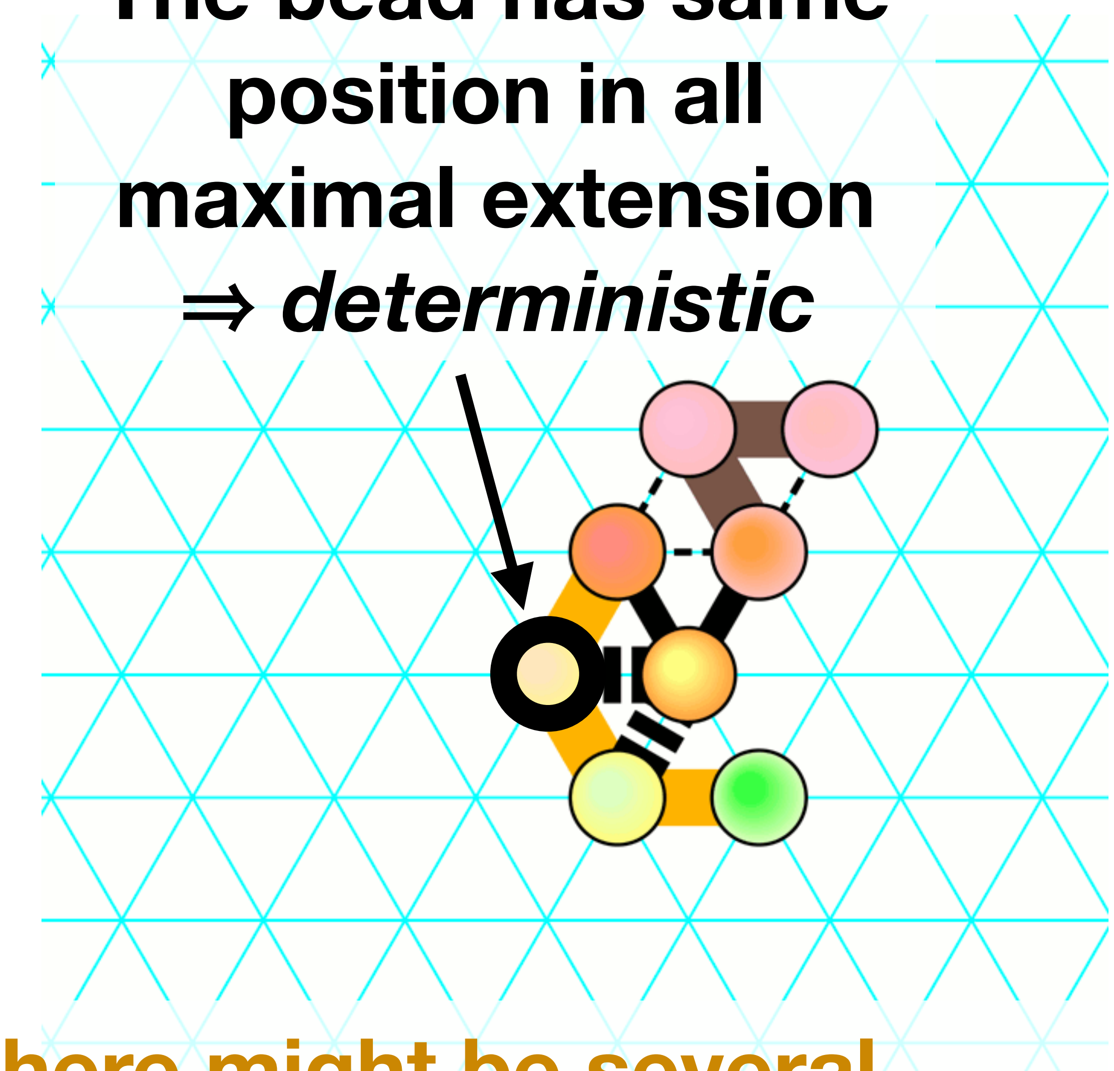
Geary, Meunier, Schabanel, Seki MFCS 2016

Oritatami:
A model for co-transcriptional folding

The dynamics.

- Starting from the seed, the sequence is *produced one bead at a time*
- **Only the δ last produced beads** are free to move and explore the accessible positions to settle in the ones **maximizing the number of bonds**
- All other beads remain in their last locations

The bead has same position in all maximal extension ⇒ *deterministic*

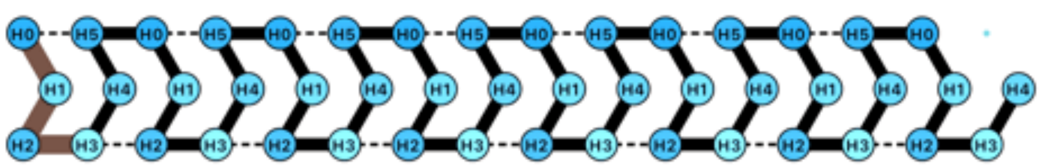


There might be several configurations with max. bonding

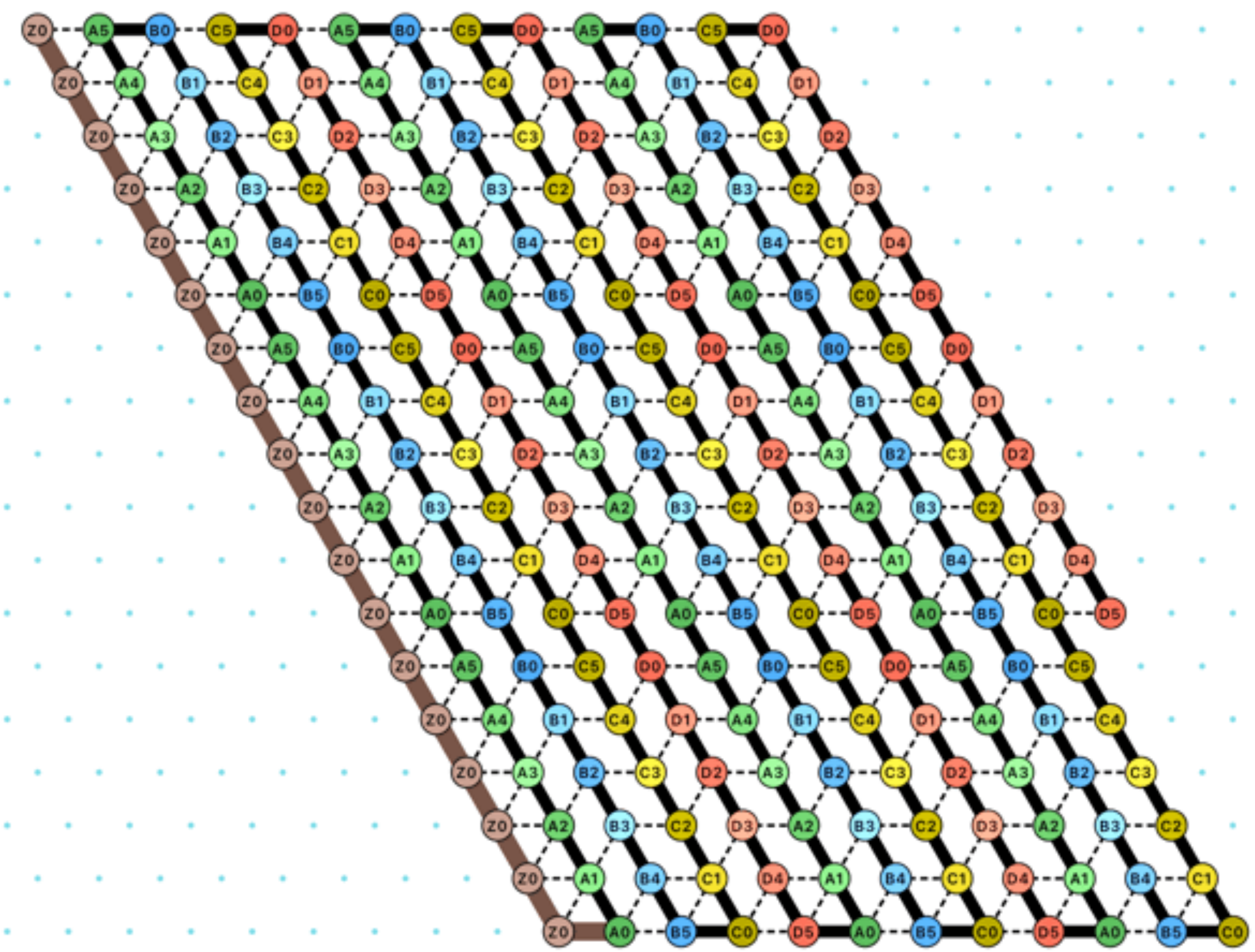
Geary, Meunier, Schabanel, Seki MFCS 2016

Oritatami: a first construction

here, delay **δ = 3**



A glider

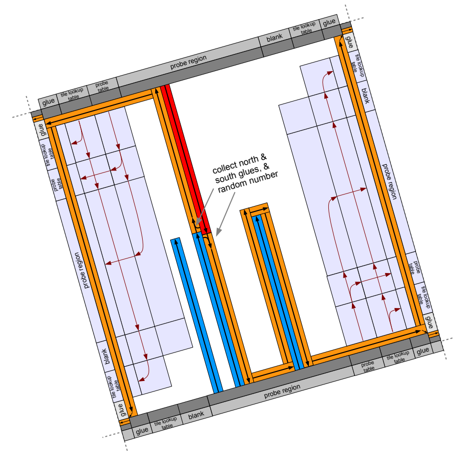


A switchback

Both can be combined together

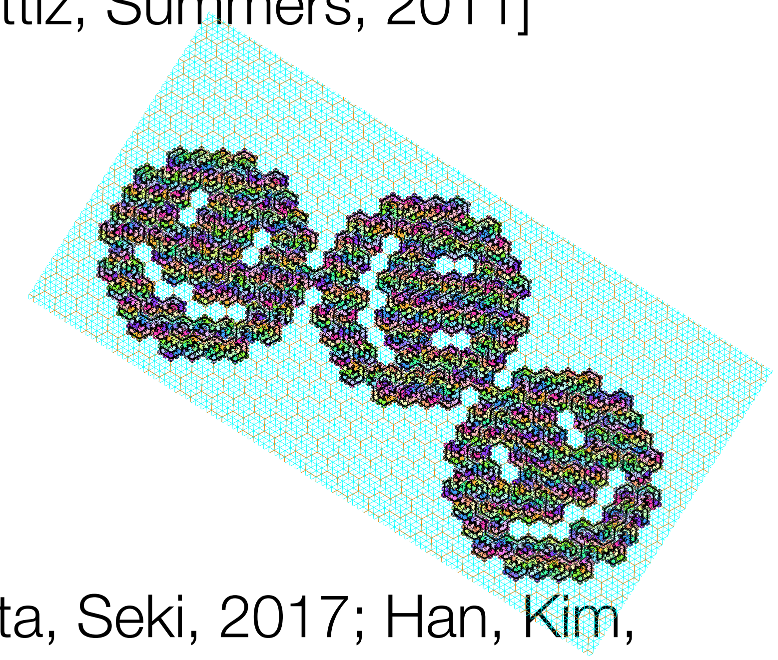
Geary, Meunier, Schabanel, Seki MFCS 2016

Oritatami *vs* aTAM



Some self-assembly seminal work (mostly aTAM)

- Tile assembly systems are **Turing universal** [Winfree, 1998]
- **Arbitrary shape assembly** with optimal tile set size [Soloveichik, Winfree, 2007]
- **Uncomputable limit configurations** [Lathrop, Lutz, Pattiz, Summers, 2011]
- **Intrinsic universality** [Doty et al, 2012]

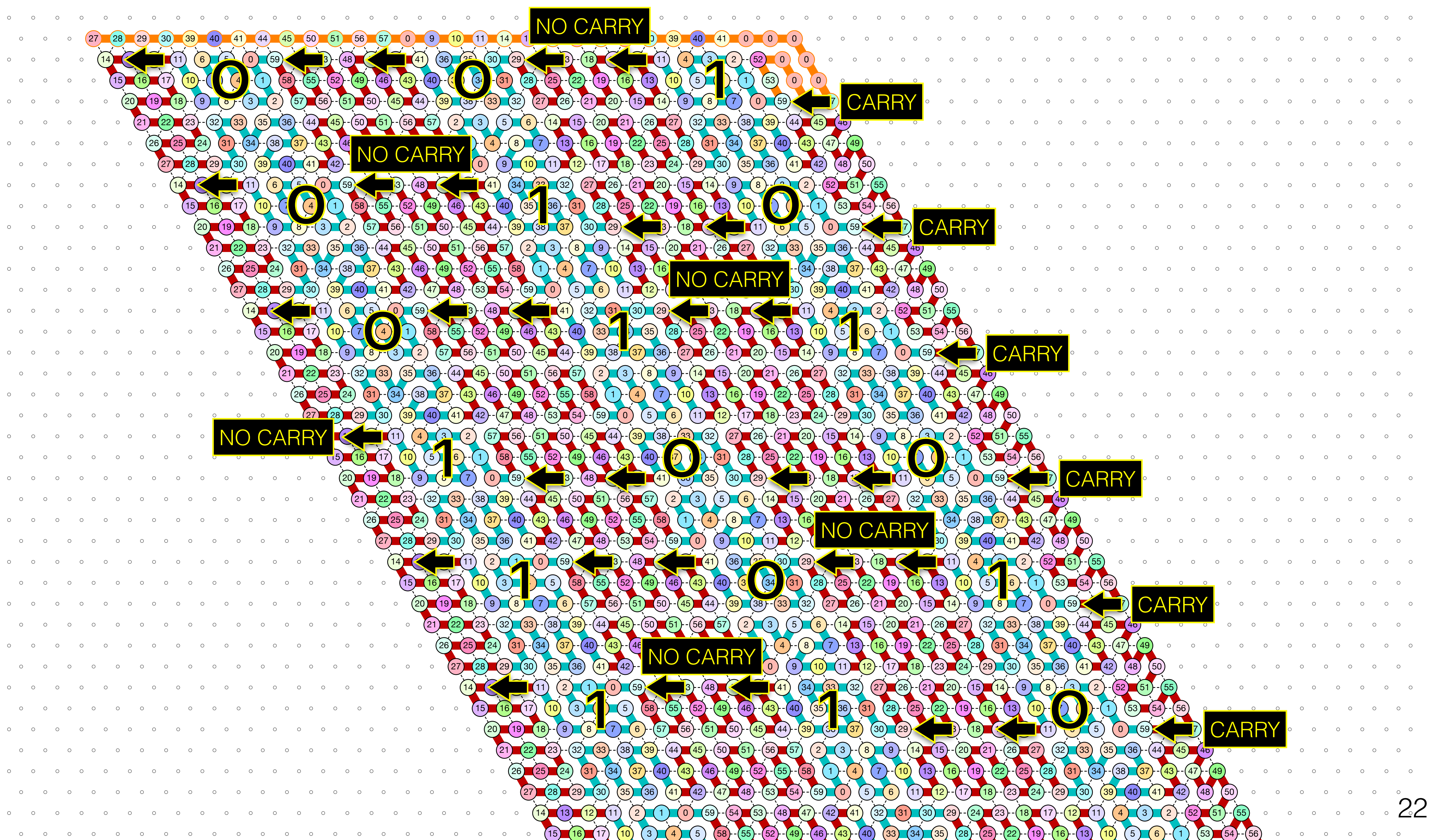


Oritatami

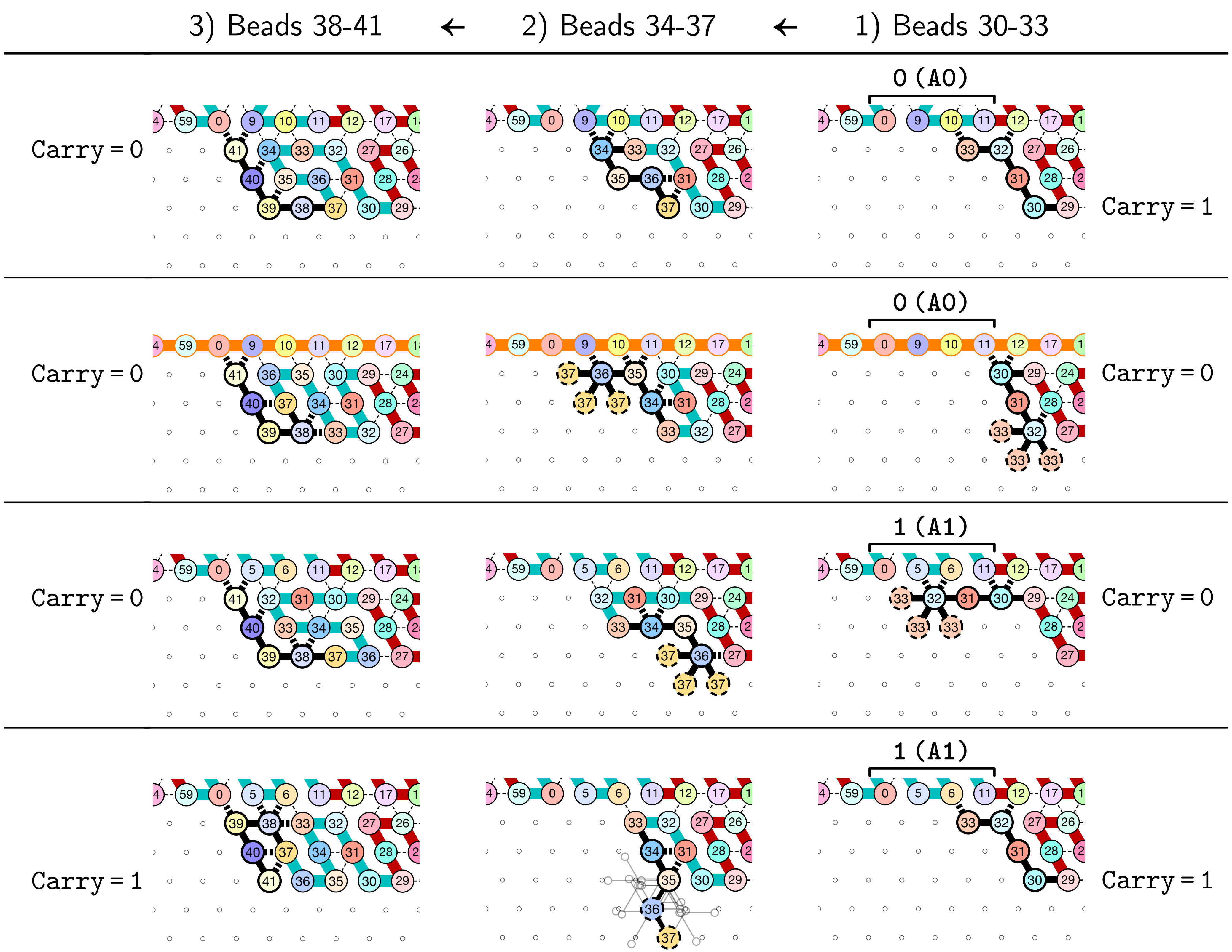
- A **binary counter** [Geary, Meunier, S., Seki, 2016]
- Heighdragon **fractal** [Masuda, Seki, Ubukata, 2018]
- Folding **arbitrary shapes** [Demaine et al, 2018]
- **NP-hardness** for oritatami design [Geary et al, 2016; Ota, Seki, 2017; Han, Kim, 2017] and for non-determinisitic oritatami equivalence [Han et al, 2016]
- **Efficient Turing Machine simulation** through tag-systems [Geary et al, 2018]
- Intrinsic simulation of **1D cellular automata** [Pchelina et al, 2020]
- Intrinsic simulation of **Turedo**: **uncomputable** and **arbitrary dense** limit configurations, building **arbitrary object** from asymptotically minimal seed [Pchelina et al, 2022]

Oritatami. A binary counter

Information is encoded in the geometry



How does computation work?



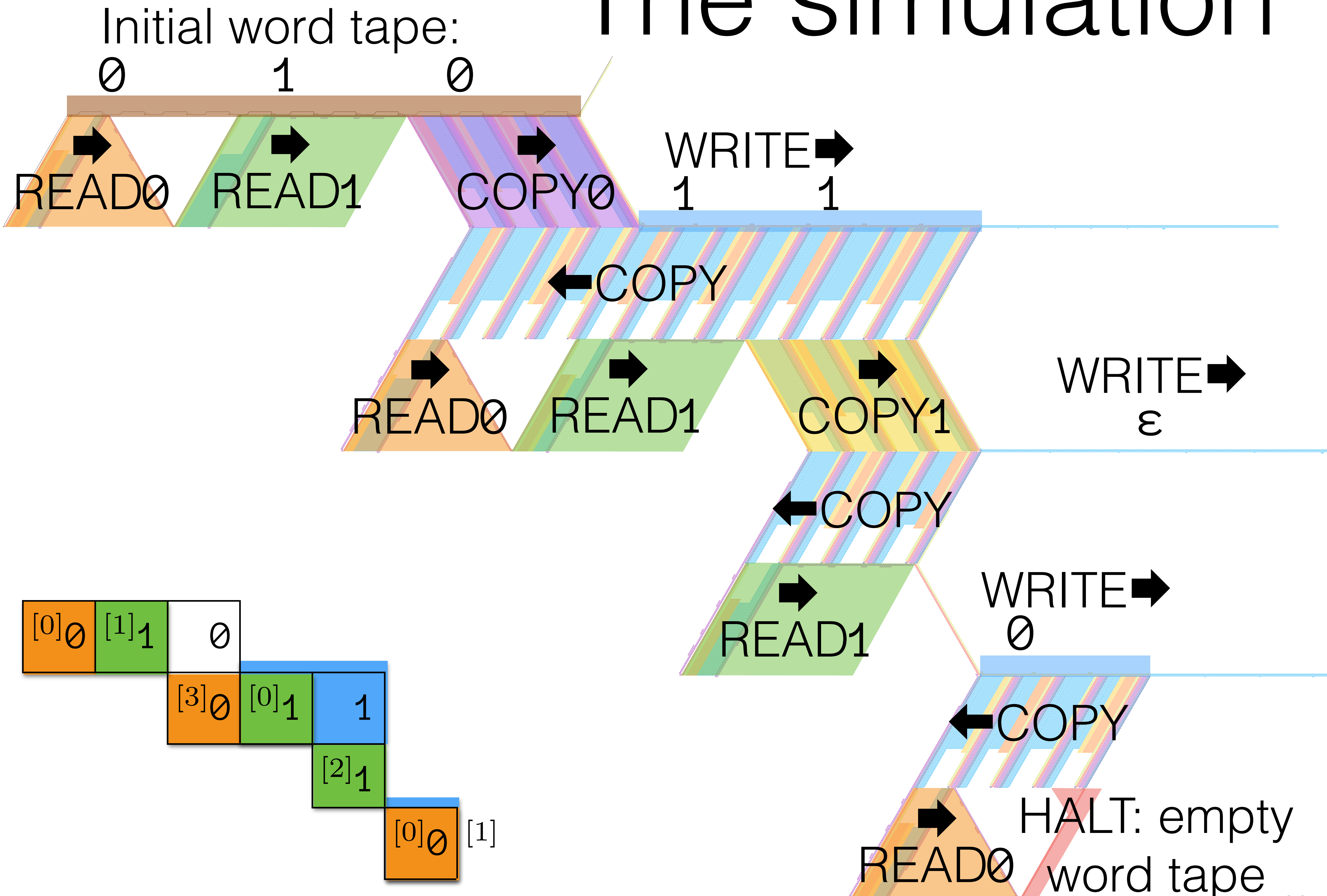
Oritatami is Turing complete

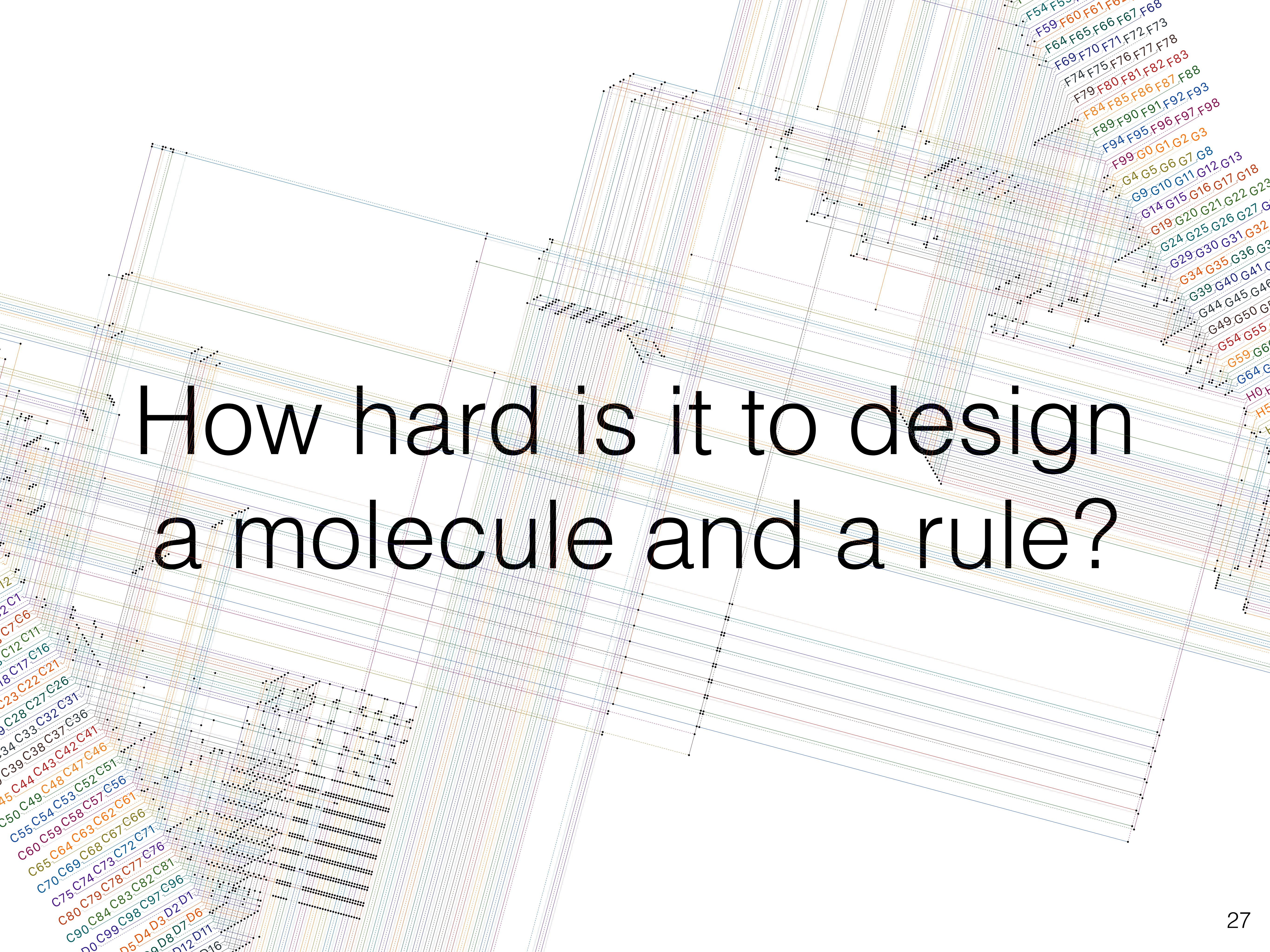
Trimmed space-time diagram

Consider the following productions: $p = \langle \overset{[0]}{\texttt{110}}, \overset{[1]}{\epsilon}, \overset{[2]}{\texttt{11}}, \overset{[3]}{\texttt{0}} \rangle$

$$^{[0]}\texttt{010} \rightarrow {}^{[1]}\texttt{10} \xrightarrow[\text{[2]:11}]{\text{Append}} {}^{[3]}\texttt{011} \rightarrow {}^{[0]}\texttt{11} \xrightarrow[{[1]:\epsilon}]{\text{Append}} {}^{[2]}\texttt{1} \xrightarrow[\text{[3]:0}]{\text{Append}} {}^{[0]}\texttt{0} \rightarrow {}^{[1]}\ \texttt{Halt}$$

The simulation

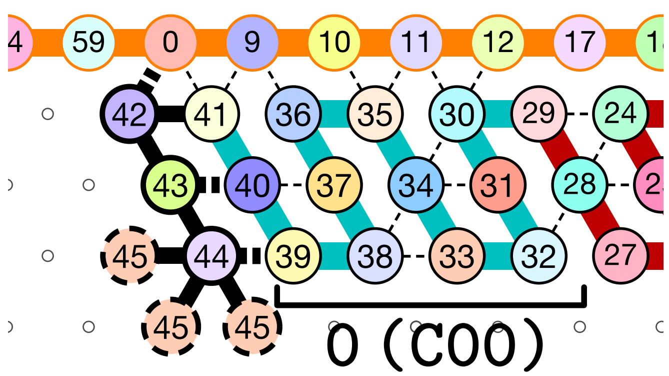




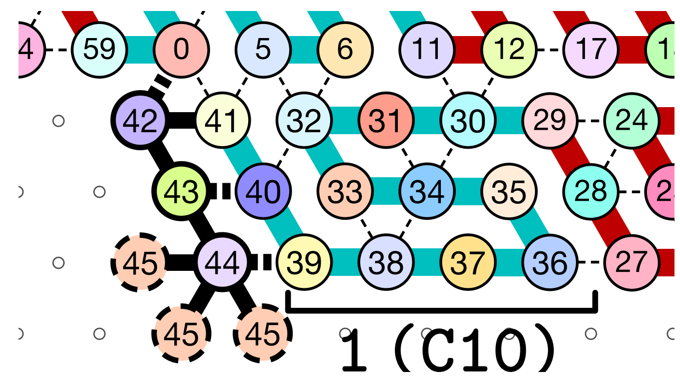
How hard is it to design a molecule and a rule?

The first challenge: Designing the desired shapes

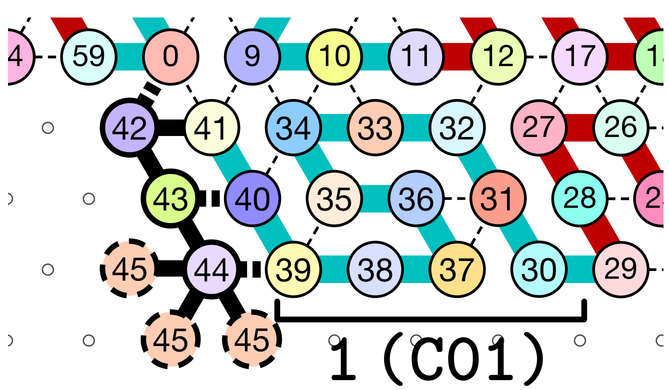
- Design shapes for which a **common** rule ❤️ exists



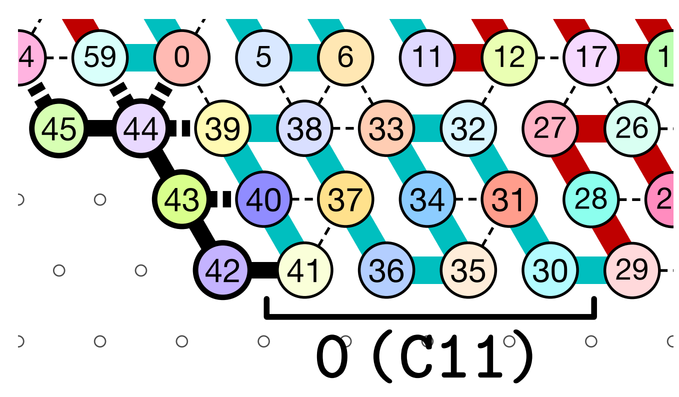
0+0 = 0 + no C



1+0 = 1 + no C



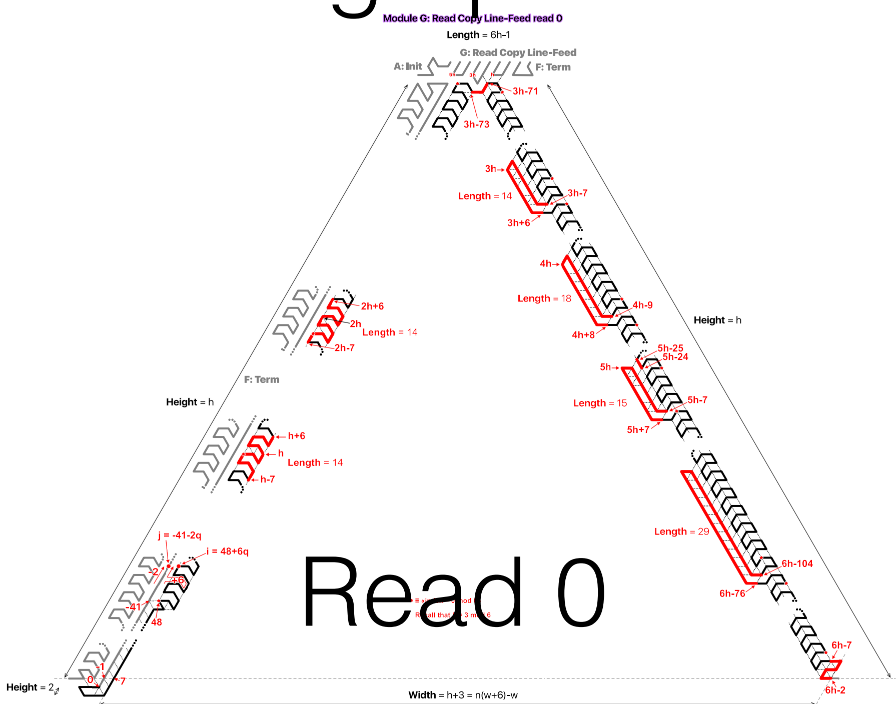
0+1 = 1 + no C



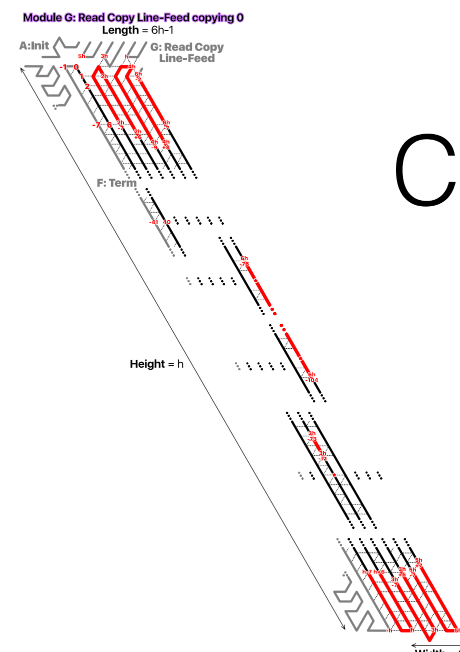
1+1 = 0 + C

The first challenge: Designing the desired paths

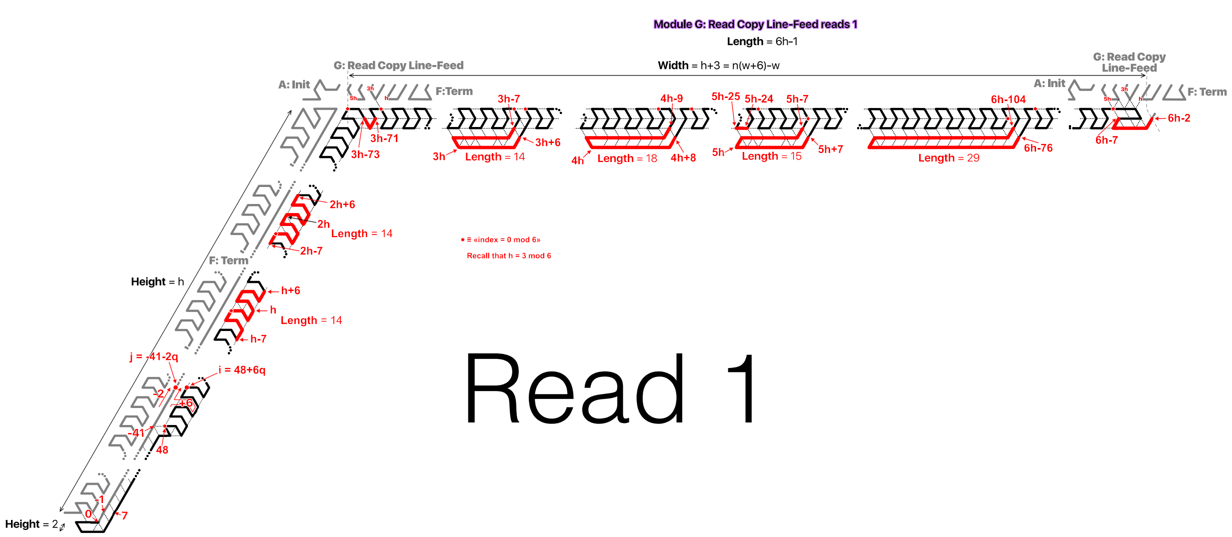
- Design paths for which a **common** rule ❤️ exists



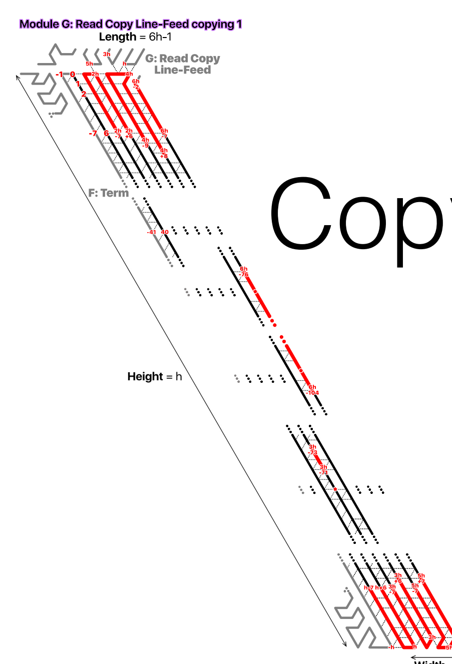
Read 0



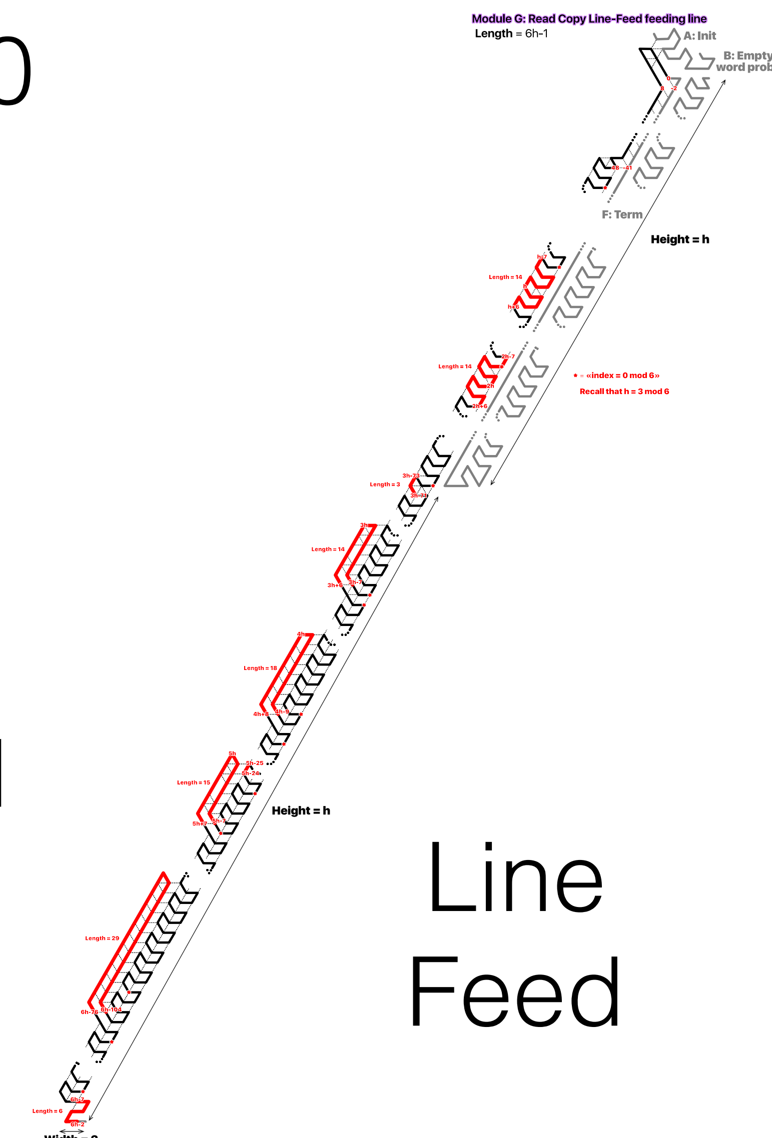
Copy 0



Read 1



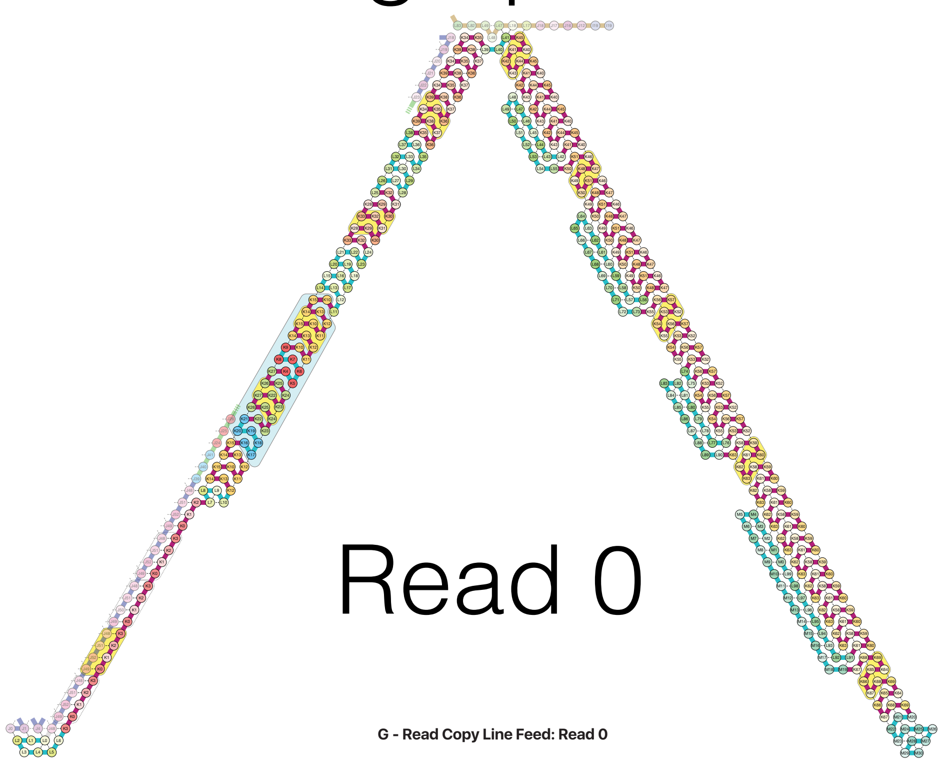
Copy 1



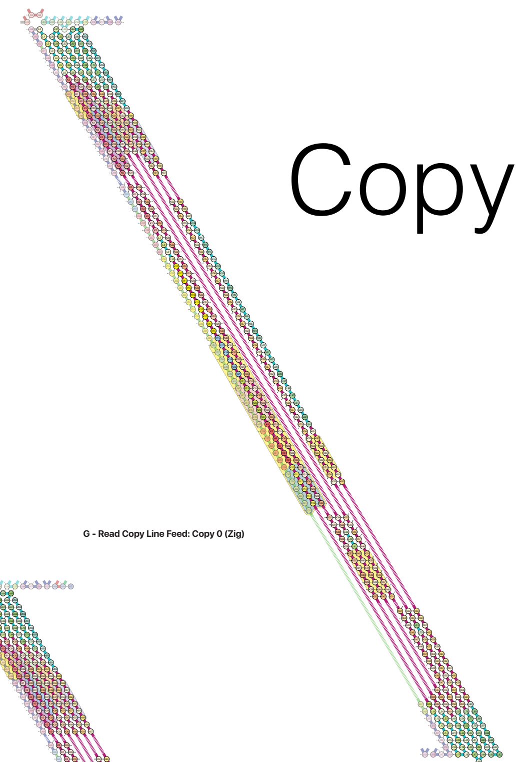
Line Feed

The first challenge: Designing the paths

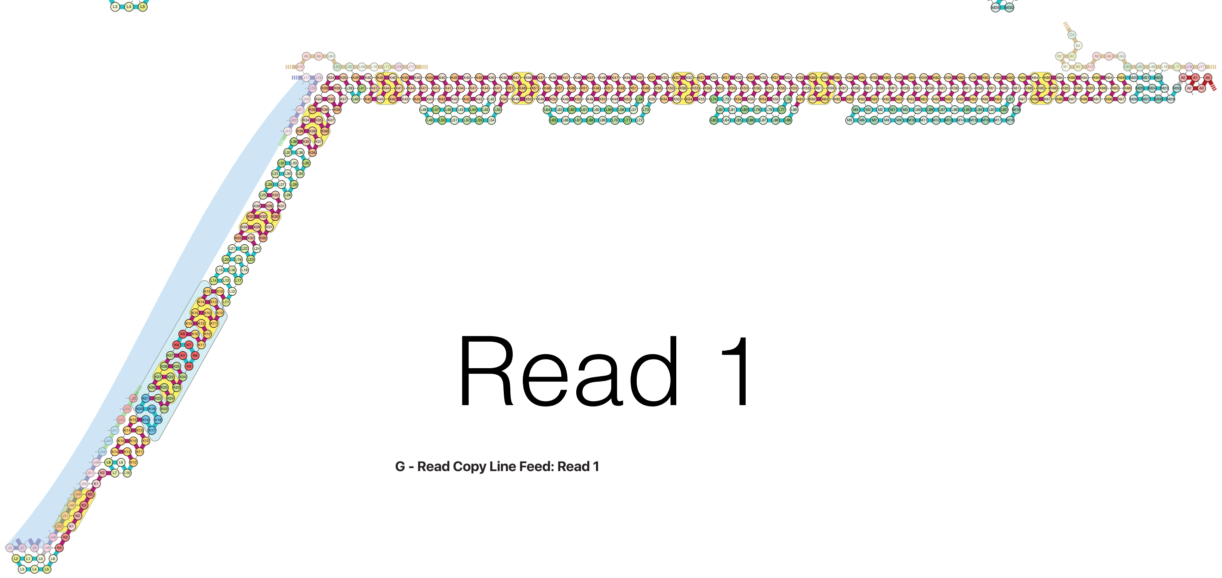
- Design paths for which a common rule ❤️ exists



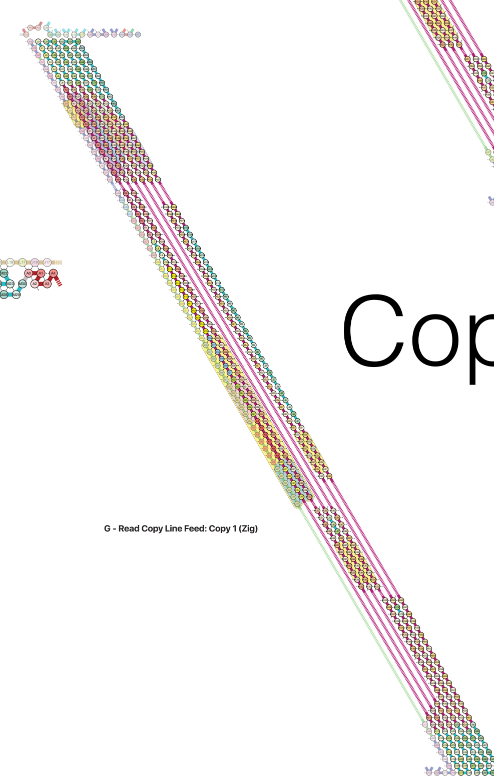
Read 0



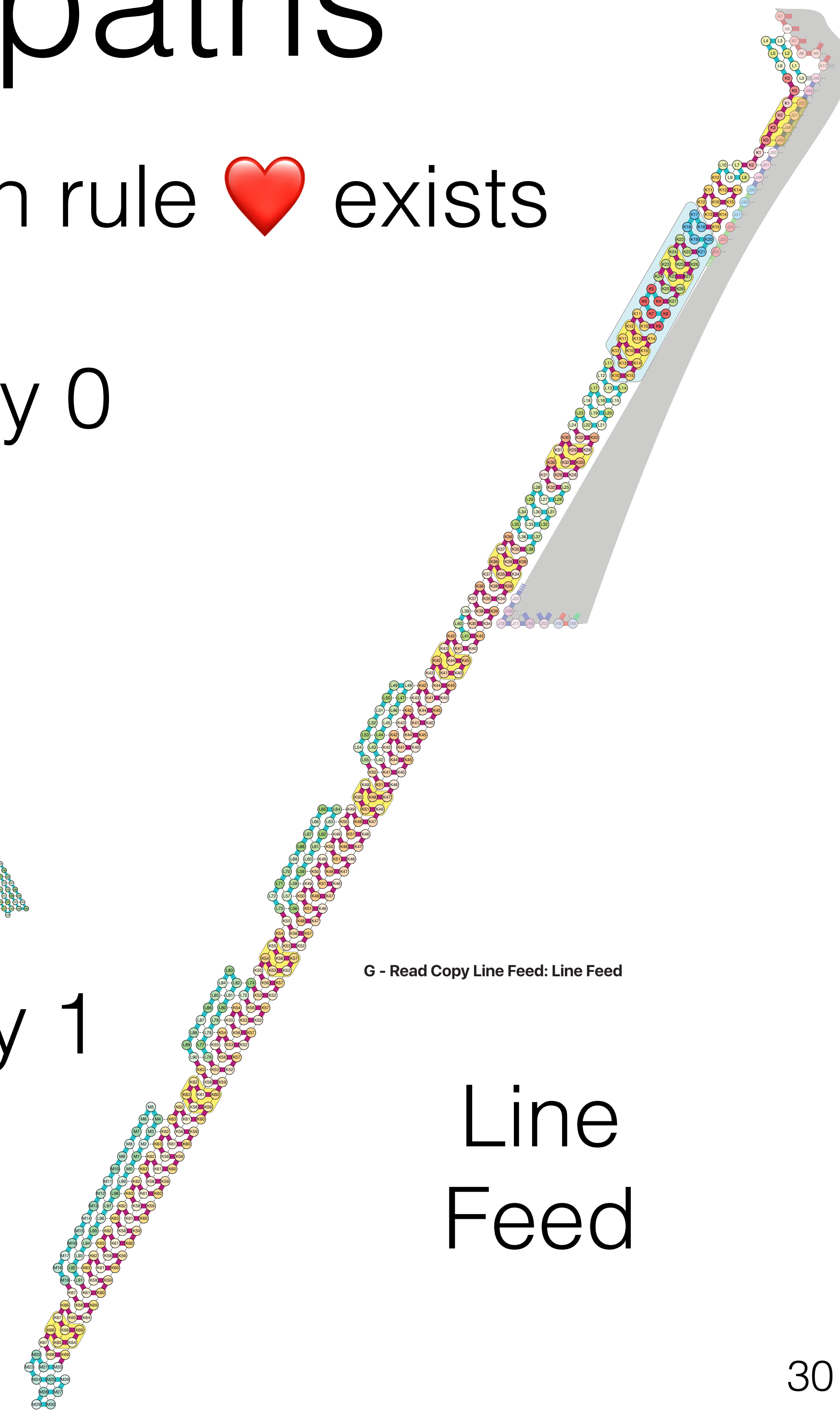
Copy 0



Read 1



Copy 1



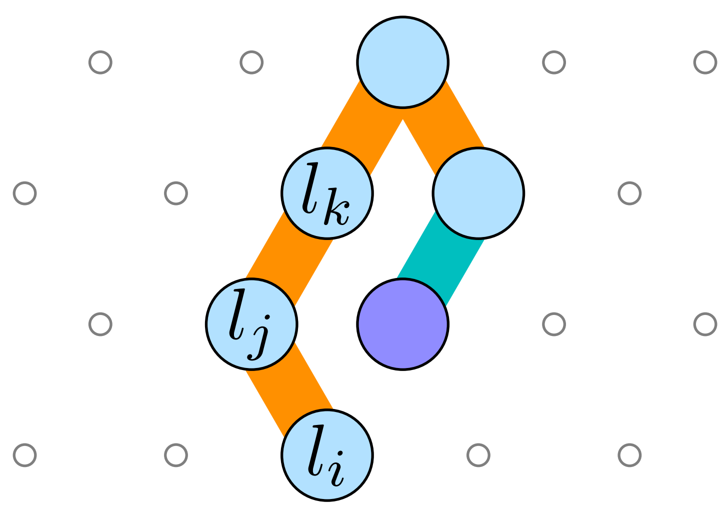
Line Feed

Oritatami design is NP-hard

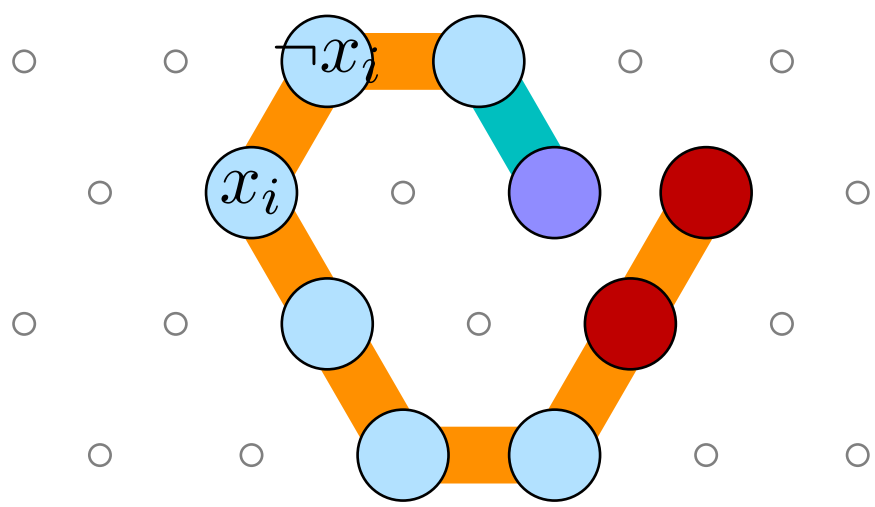
INPUT:	a delay time δ, a list of $n > 0$ seeds $\sigma_1, \sigma_2, \ldots, \sigma_n$, and a list of n conformations $c_1, c_2, \ldots, c_n$ of the same length l
OUTPUT:	an attraction rule $\heartsuit$ such that for all $i \in \{1, 2, \ldots, n\}$, Oritatami system $\mathcal{O}_i = (s, \sigma_i, \heartsuit, \delta)$ deterministically folds into conformation c_i, where s is the sequence of length l such that for all $i \in \{1, 2, \ldots, l\}$, $s_i = i$.

The reduction *(length=1, δ arbitrary)*

Ensures it binds to at least one litteral in l_i v l_j v l_k



Ensures it binds to at most one of x_i and $\neg x_i$



The second challenge: Designing the rule ❤️

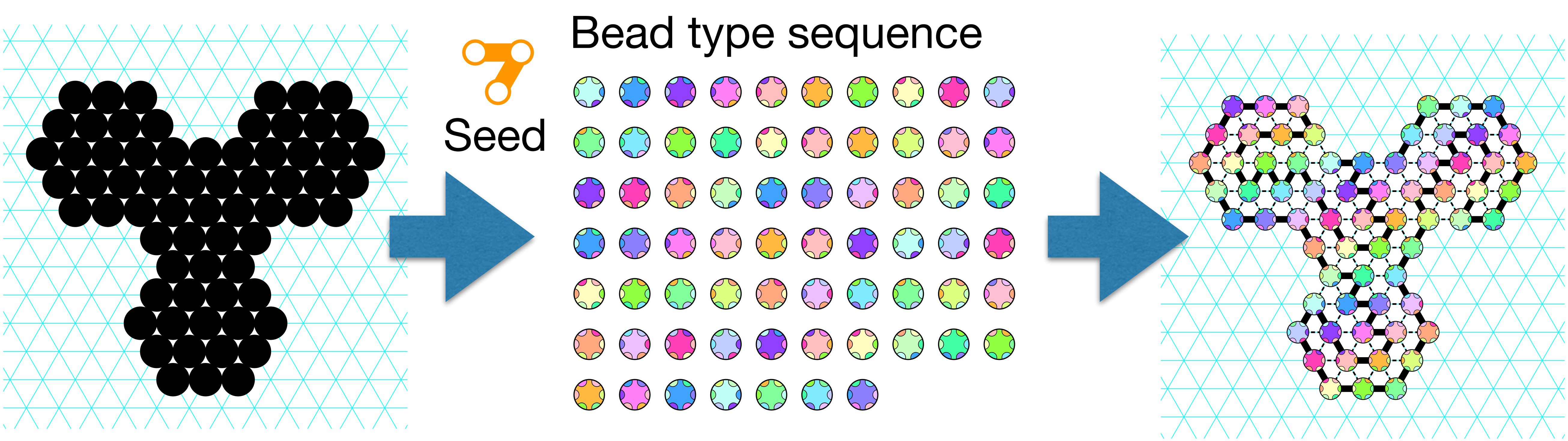
Theorem. There is a **FPT algorithm** with respect to ***L*** that designs **in linear time in *L*** (but exponential in ***k*** and ***δ***) a **rule** ❤️ that folds the **sequence *1,...,L* of length *L*** into ***k*** **prescribed conformations** when folded in ***k*** **prescribed environments.**

Proof. • **Locality:** each bead only sees a bounded number (exponential in δ) of other beads when folded.

- Then, compute all valid local rules for each of these neighborhoods
- And use dynamic programming to decide whether there is a global rule compatible with at least one of the local rule for each environment.

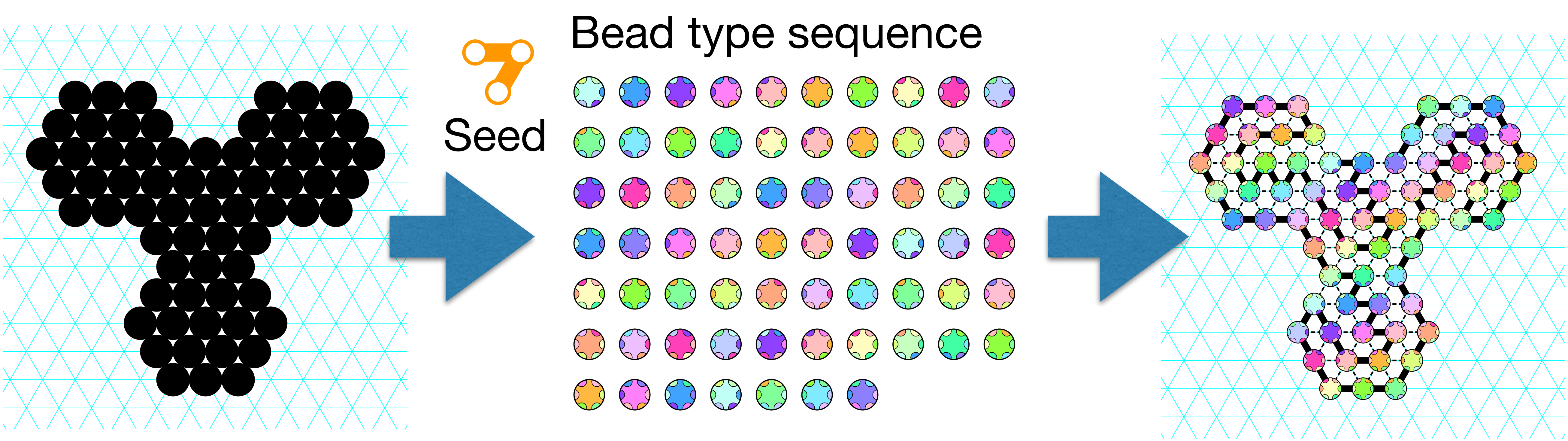
Building shapes

Goal: Given a shape S, Find an oritatami system, i.e. a *sequence of bead types*, that folds into S



An Oritatami system folds a shape if:

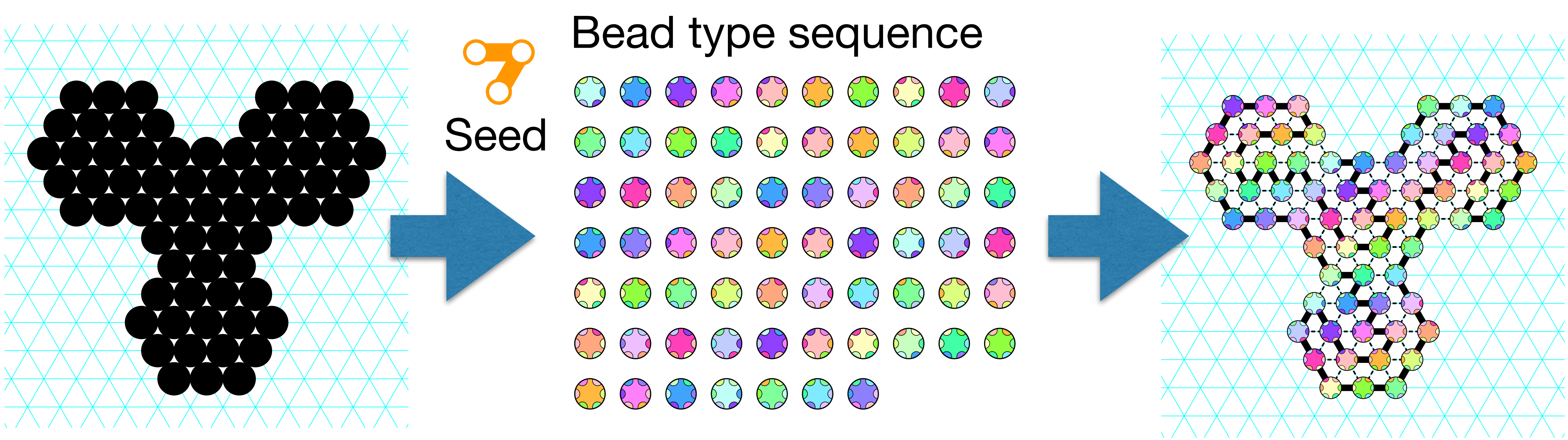
Starting from the seed configuration,
it folds deterministically to occupy all the positions of the shape and only them



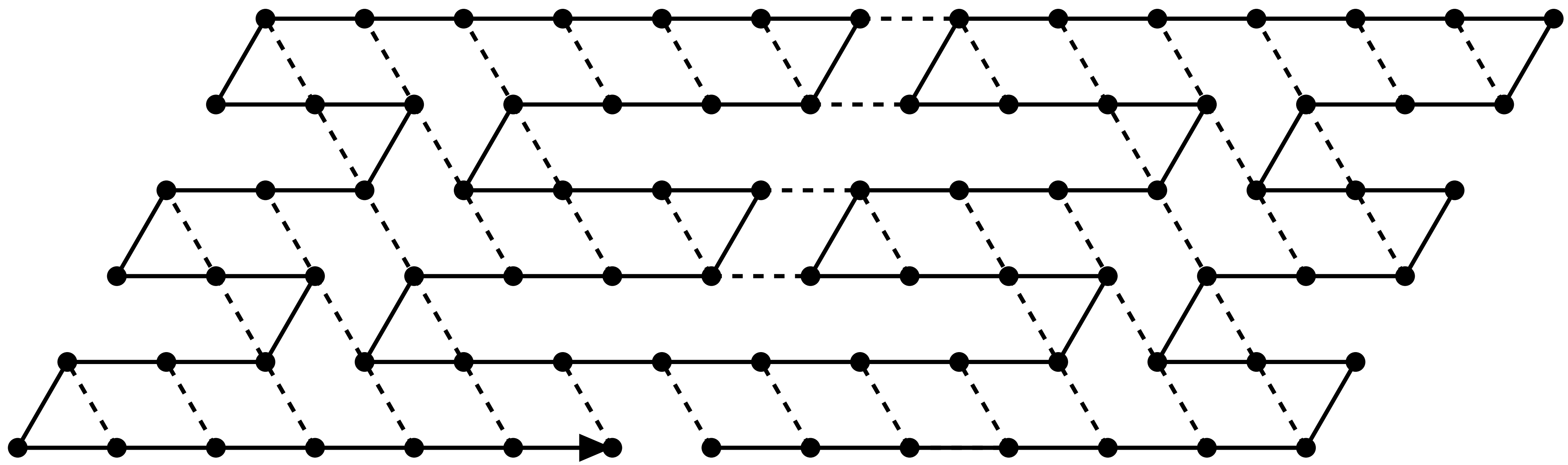
Seek an Universal construction

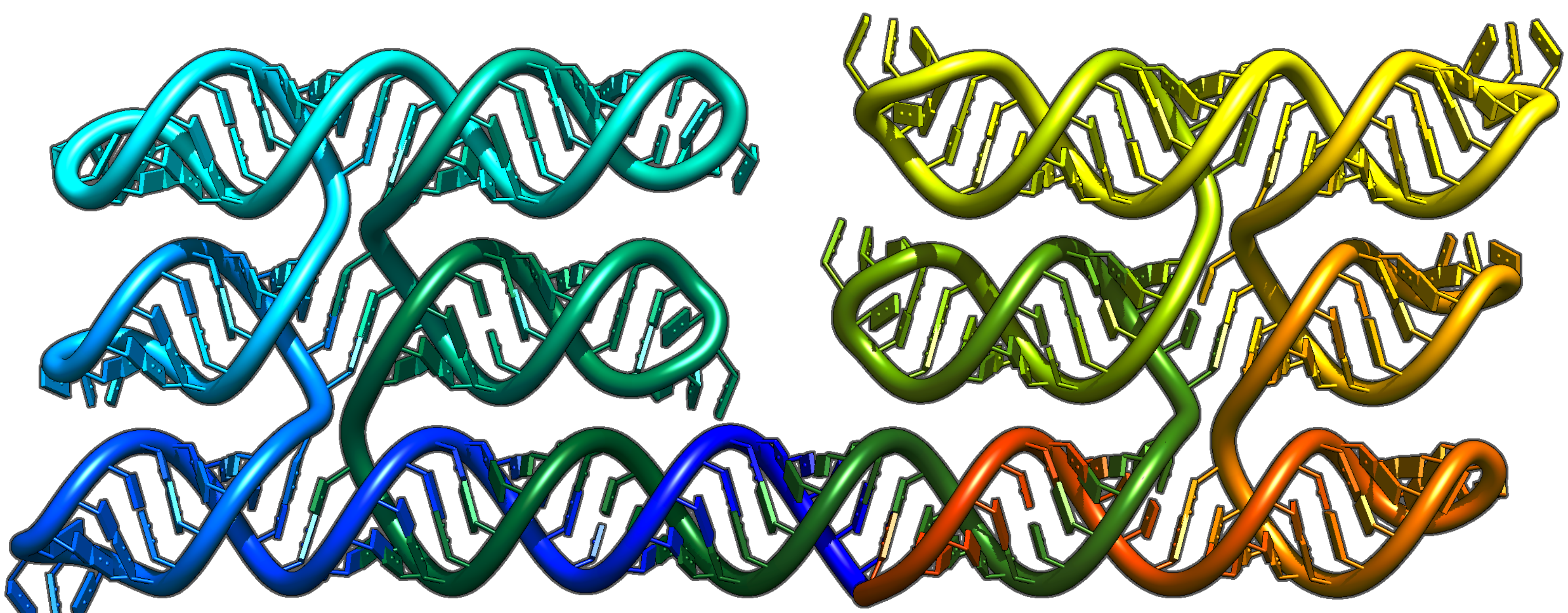
- **Fixed** finite size seed
- **Fixed** finite set of bead types

  (independent of the shape)

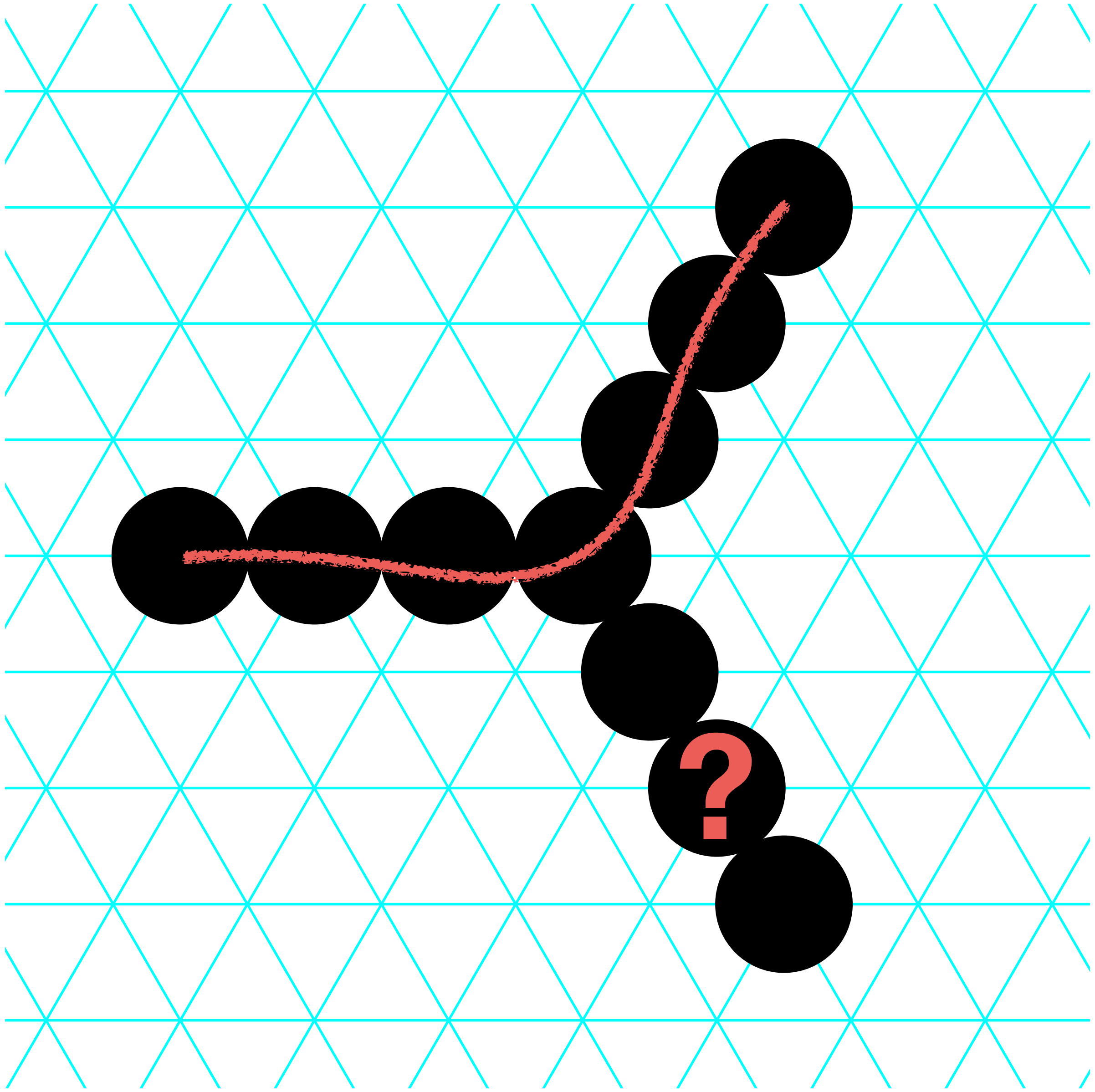


Example of such molecule

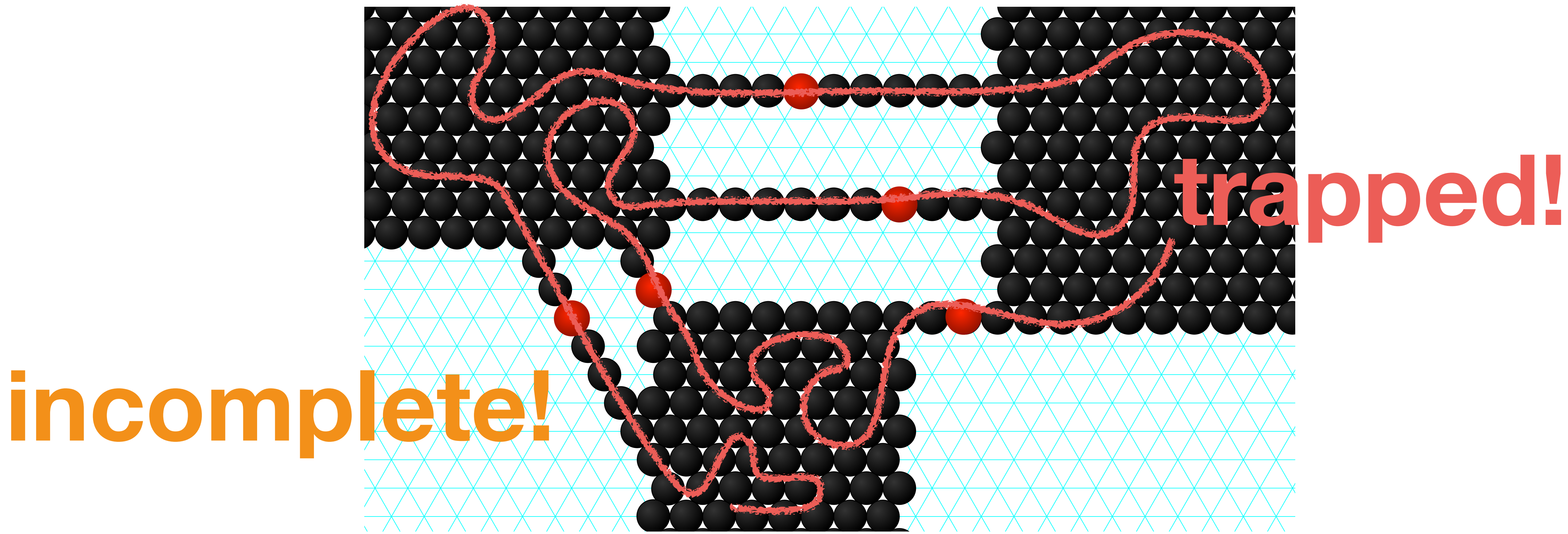




Trivial fact:
Foldable shapes are Hamiltonian



Fact: Finitely cutable *infinite* shapes cannot be folded

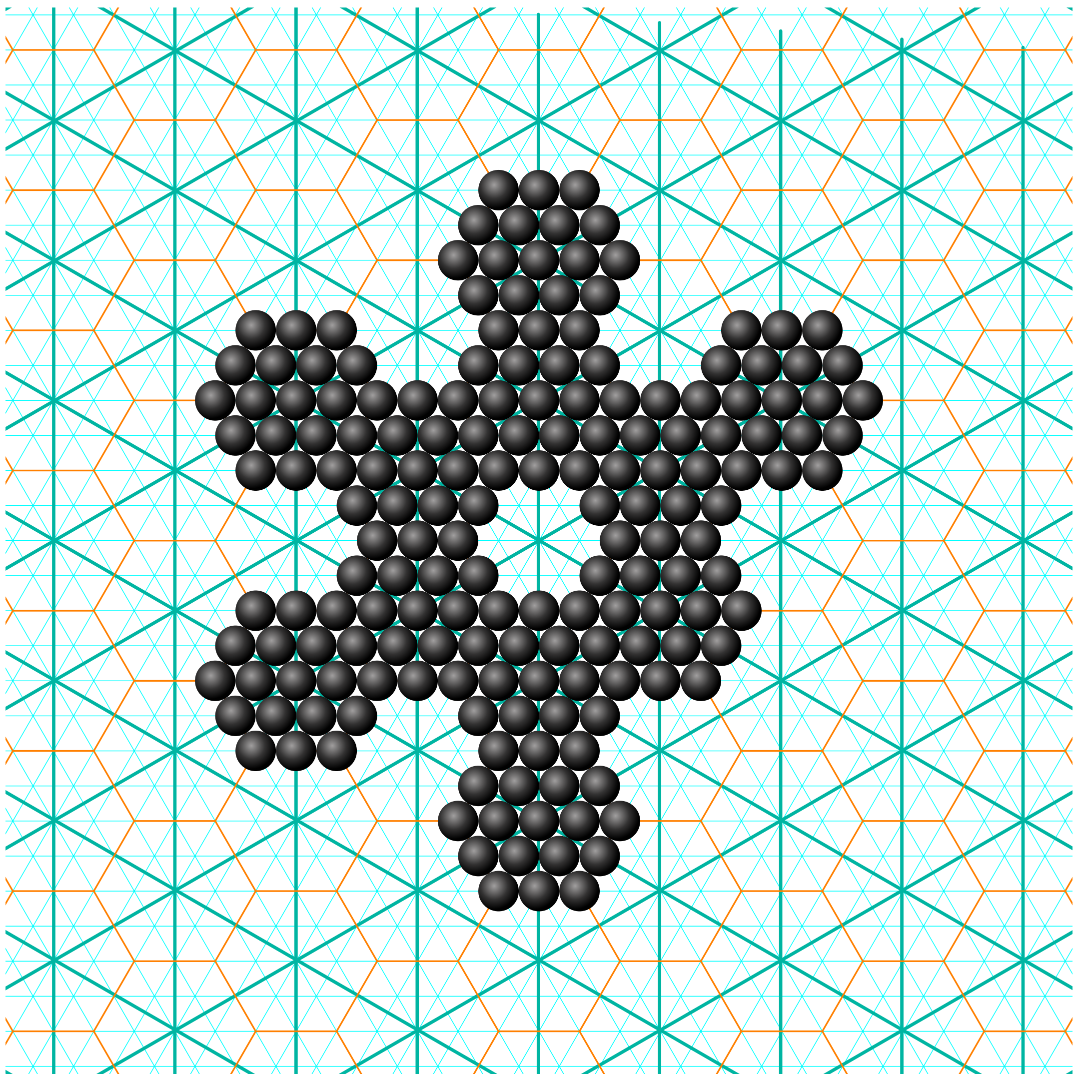


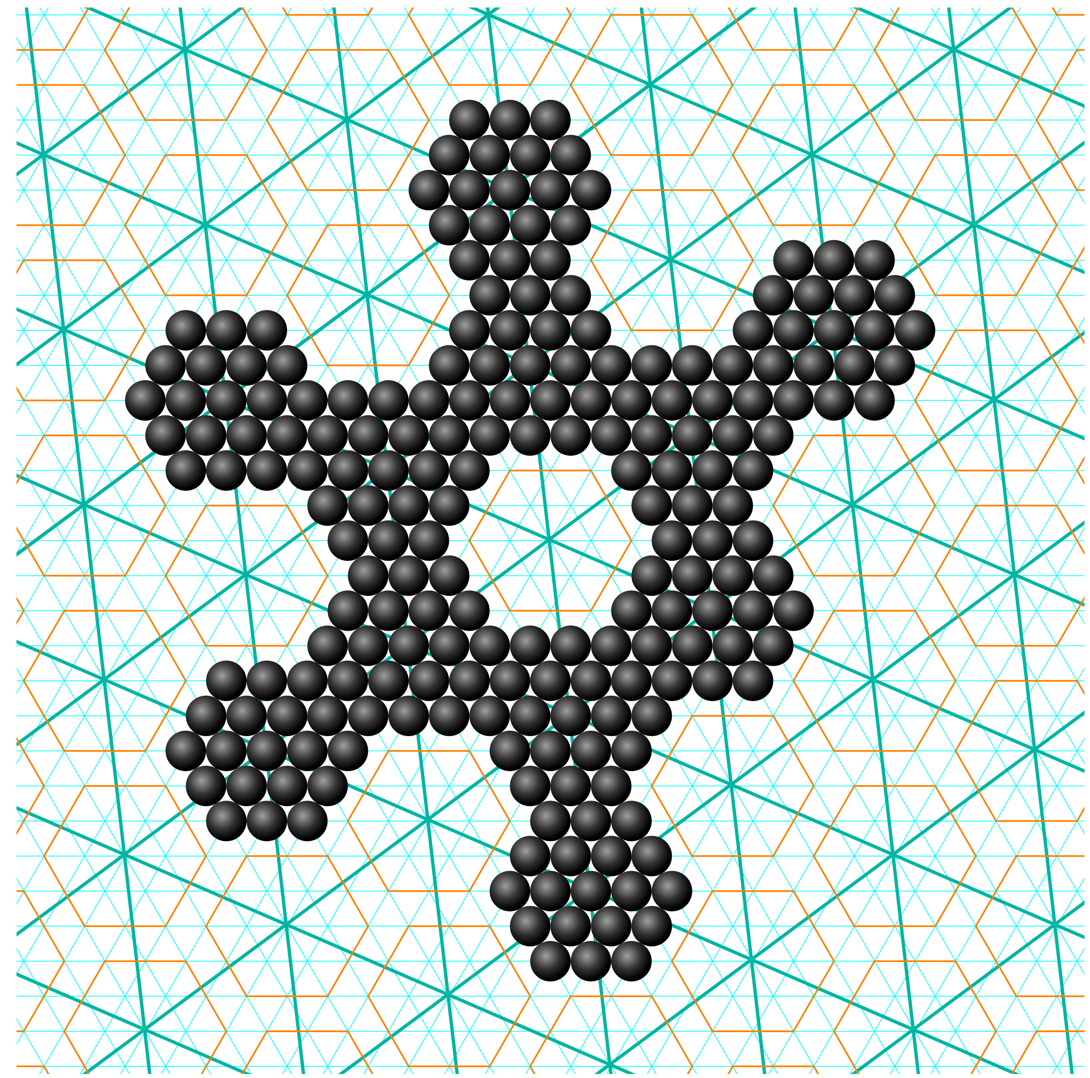
A **finite set of points** cutting the shape into several infinite pieces

Oritatami systems are thus essentially different from tile assembly systems (aTam)

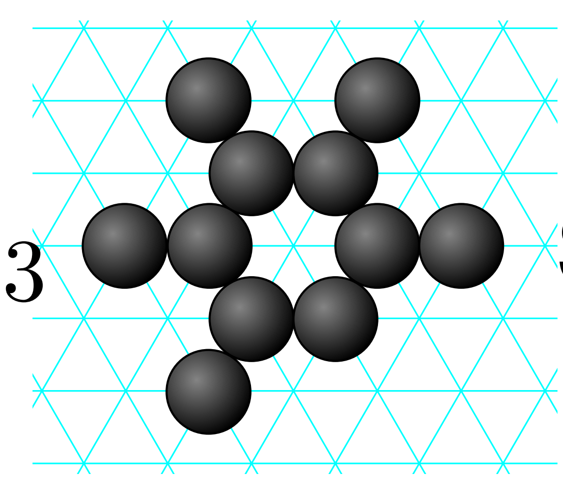
Consider upscaling schemes

Upscaling schemes



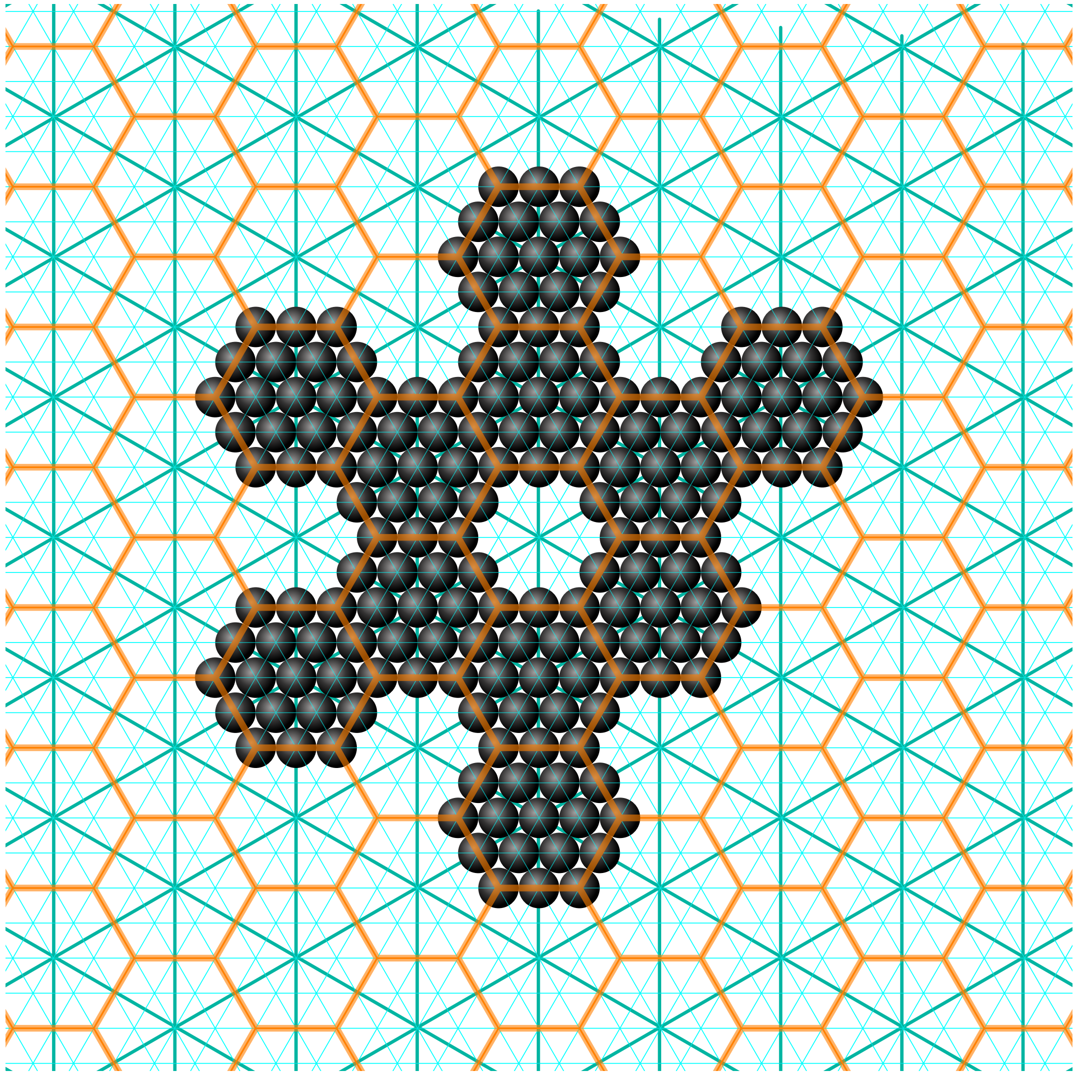


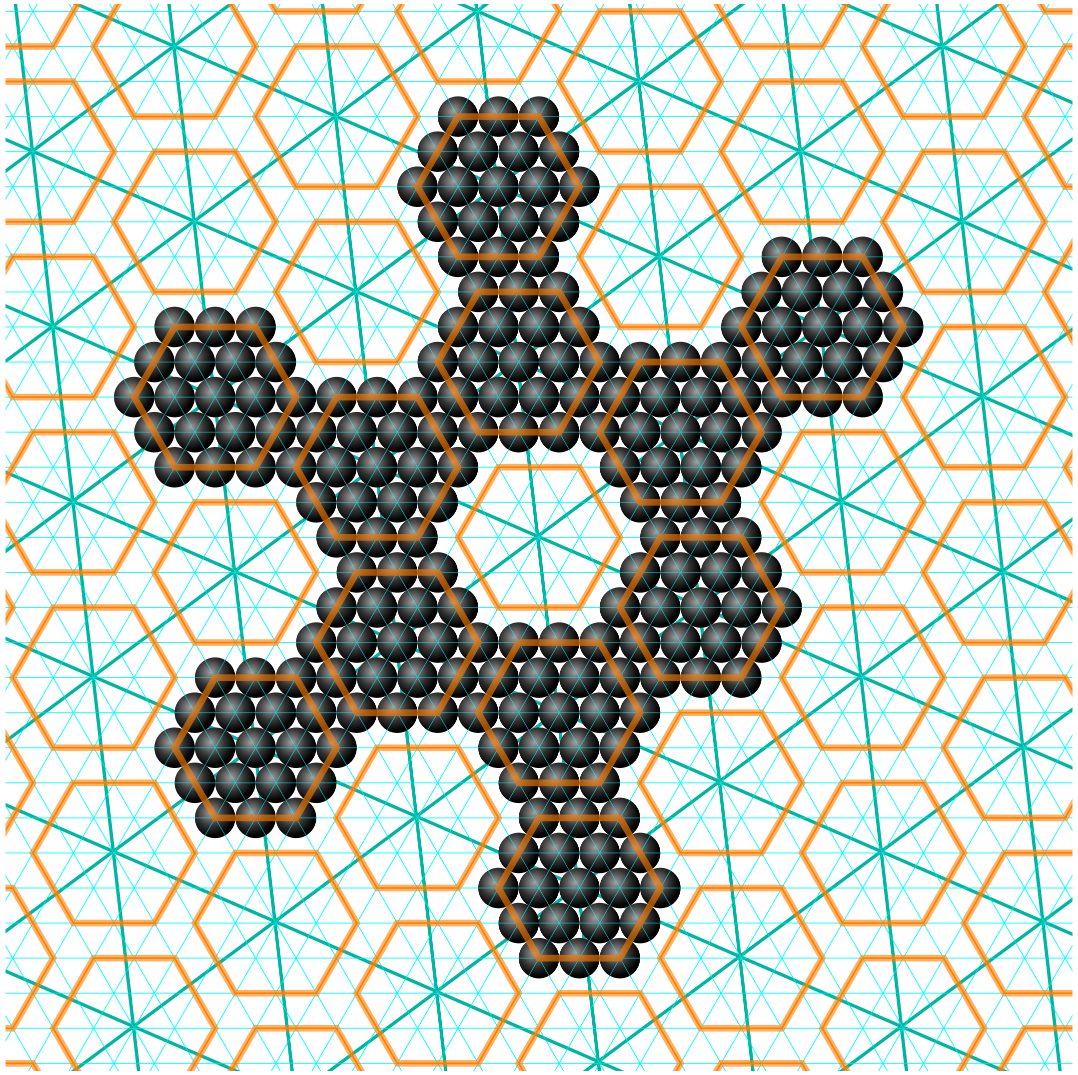
scale $\mathscr{A}_{n},\ n = 3$



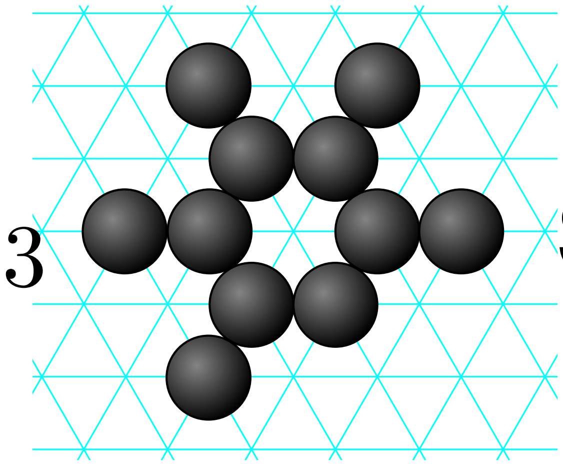
scale $\mathscr{B}_{n},\ n = 3$

Upscaling schemes





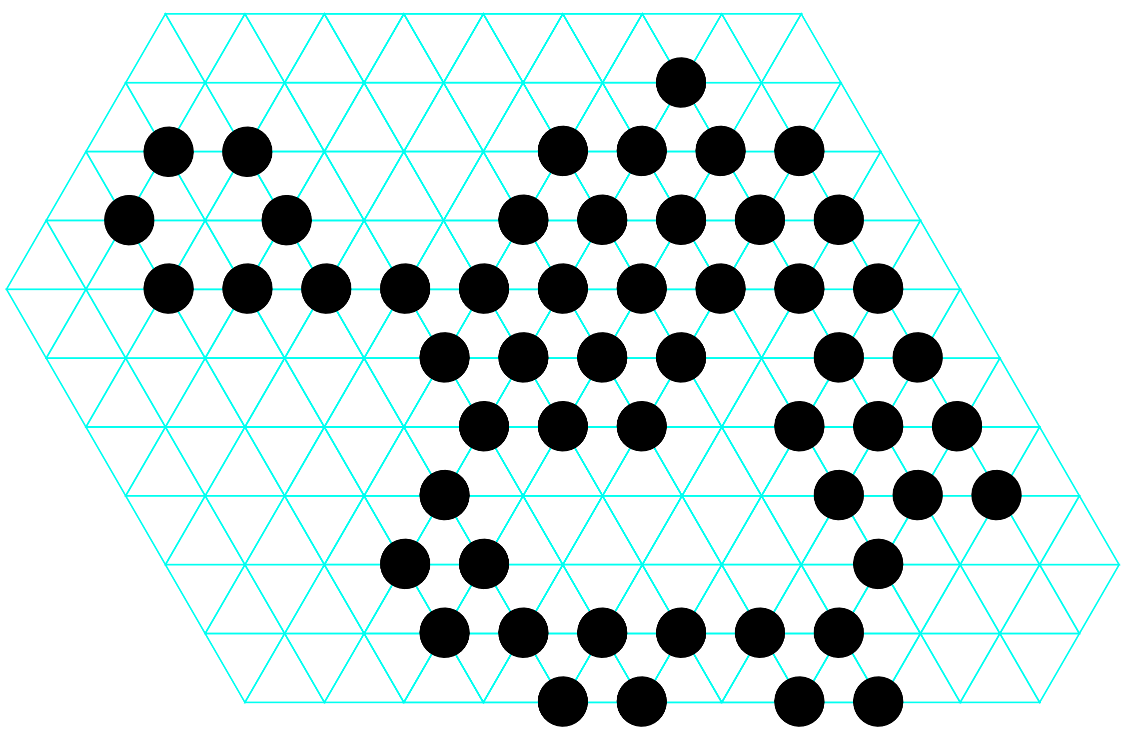
scale $\mathscr{A}_{n},\ n = 3$

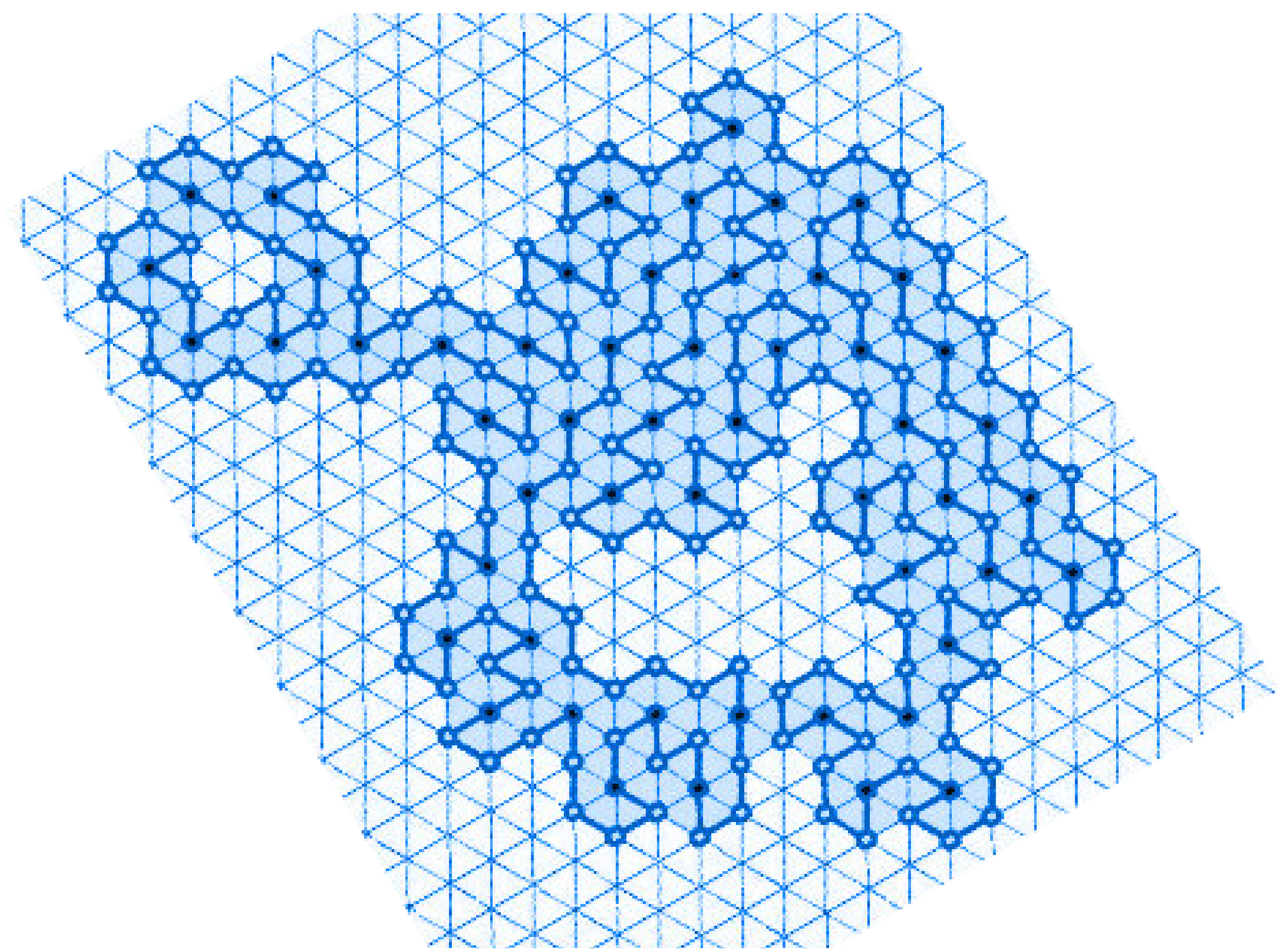


scale $\mathscr{B}_{n},\ n = 3$

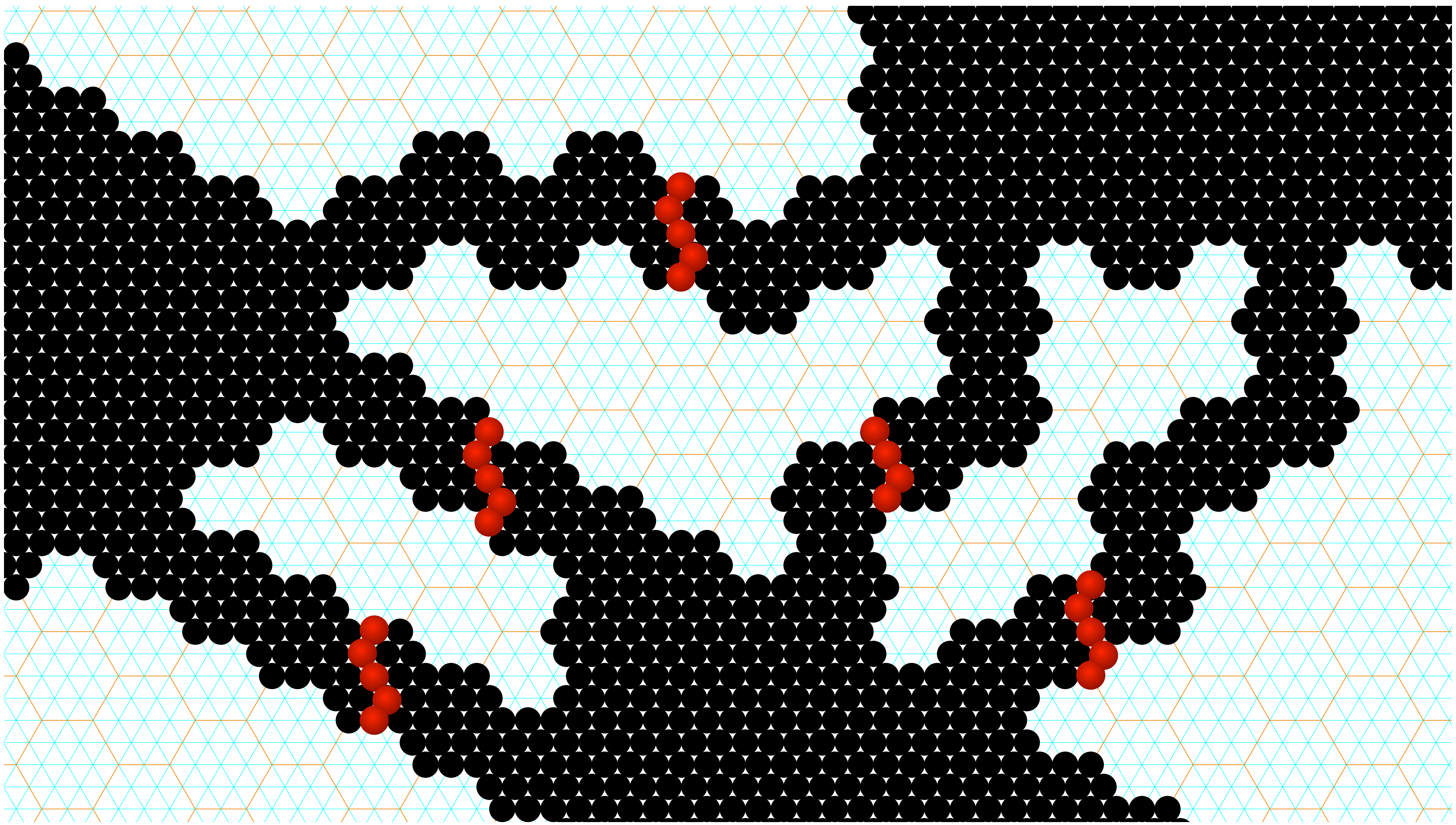
Finite shapes are Hamiltonian at scale $\mathcal{A}_2$

Theorem. There is a quadratic algorithm that computes an Hamiltonian path for any finite shape at scale $\mathcal{A}_2$





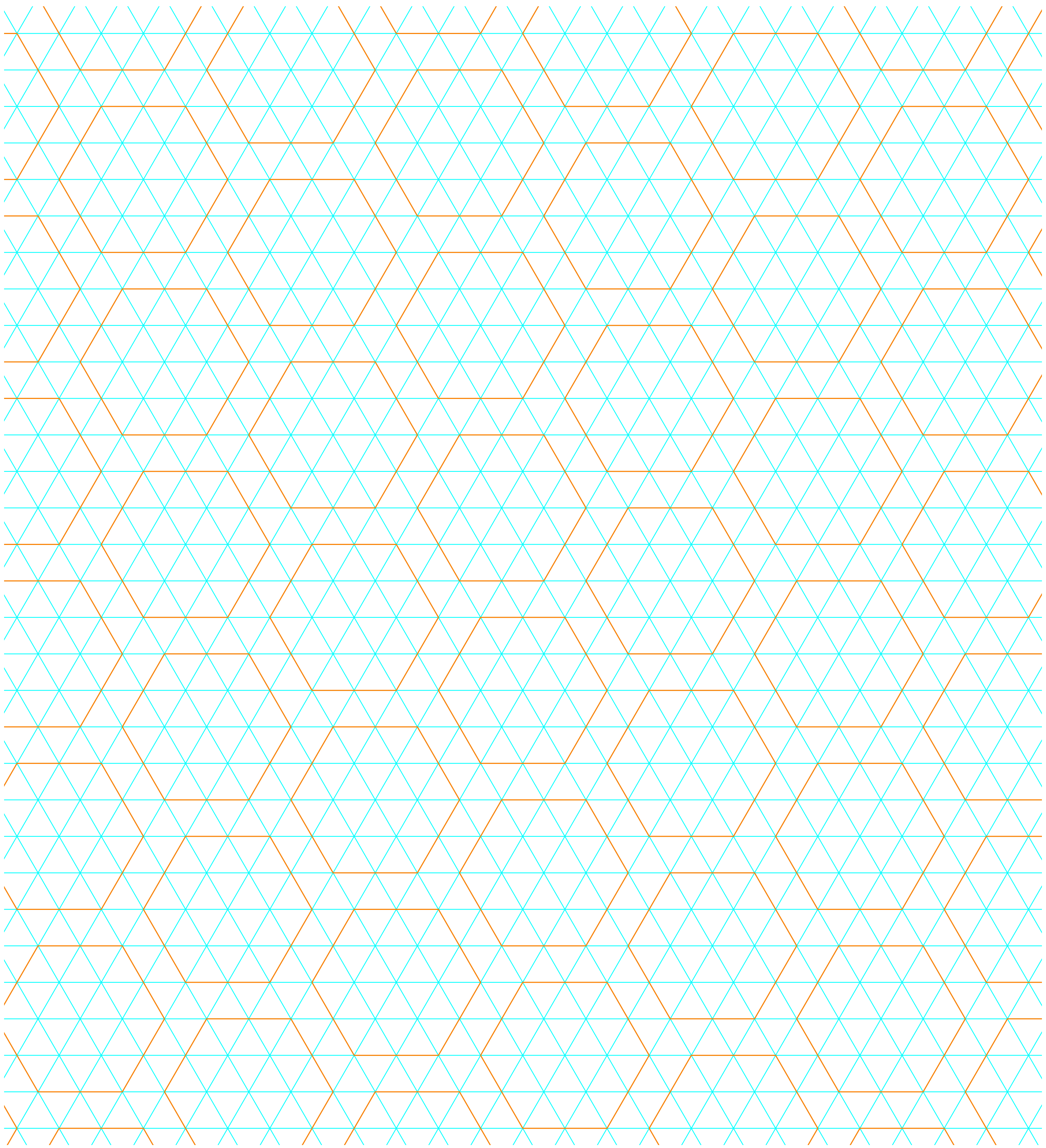
Upscaling does not help with finitely cutable infinite shapes



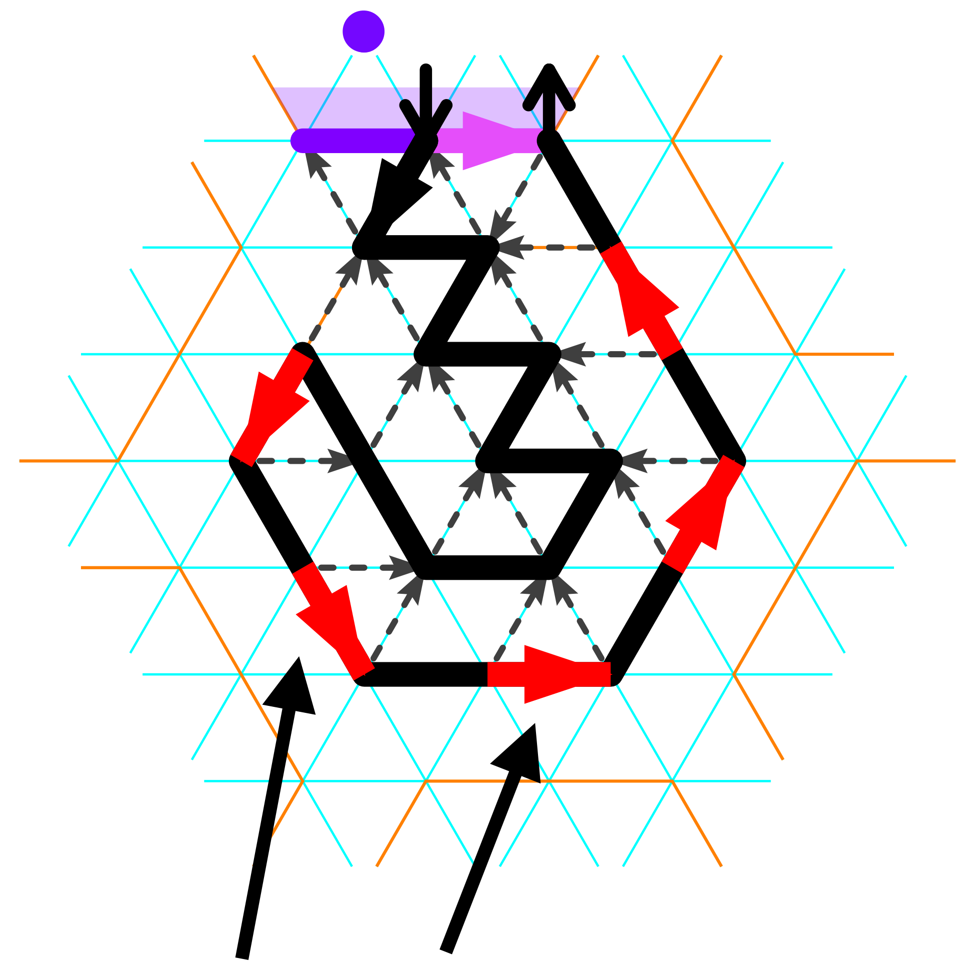
Thus, we focus on **finite shapes**

Scale $\mathscr{B}_n$

scale $\mathscr{B}_n$

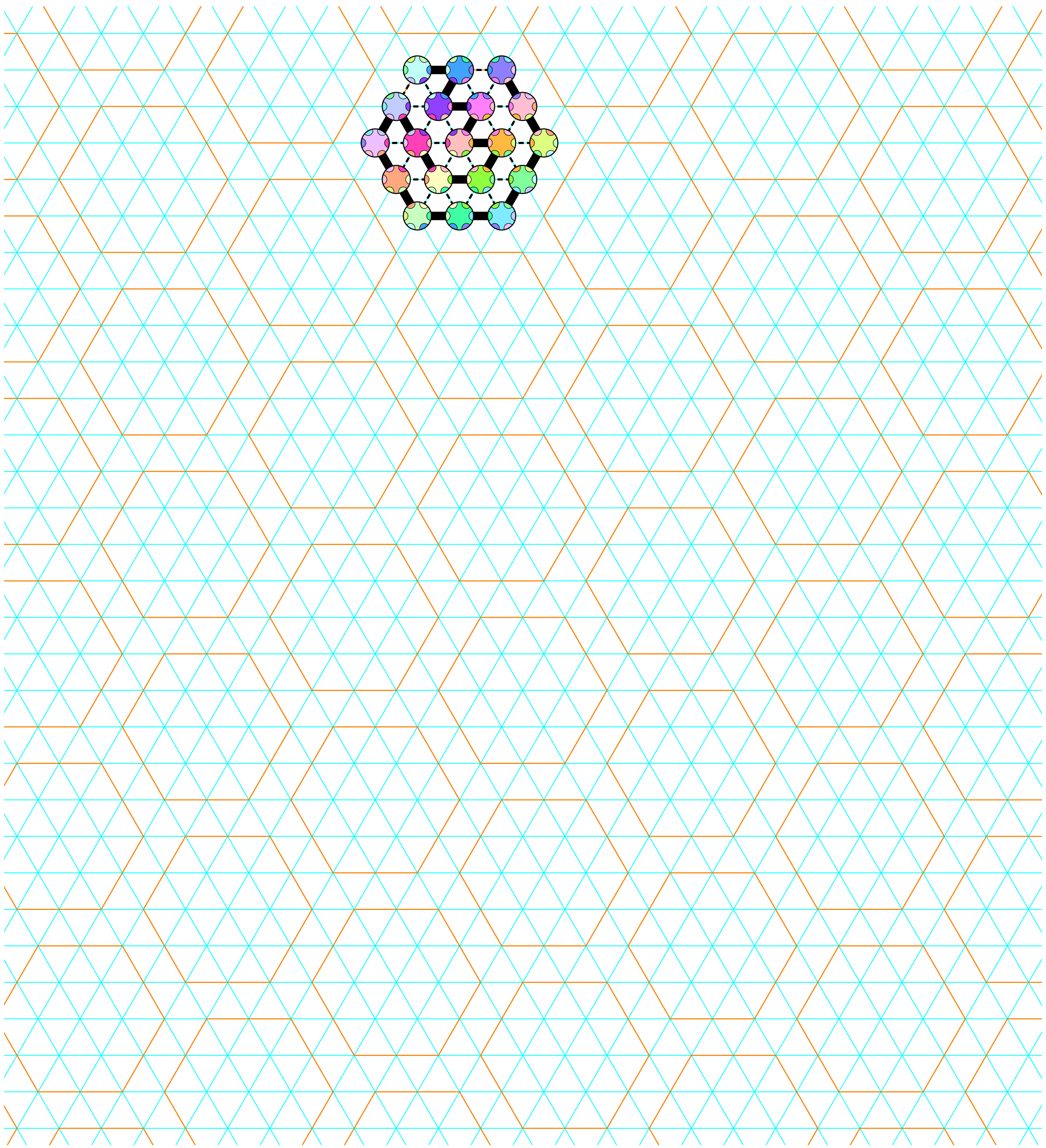


Use a unique pattern

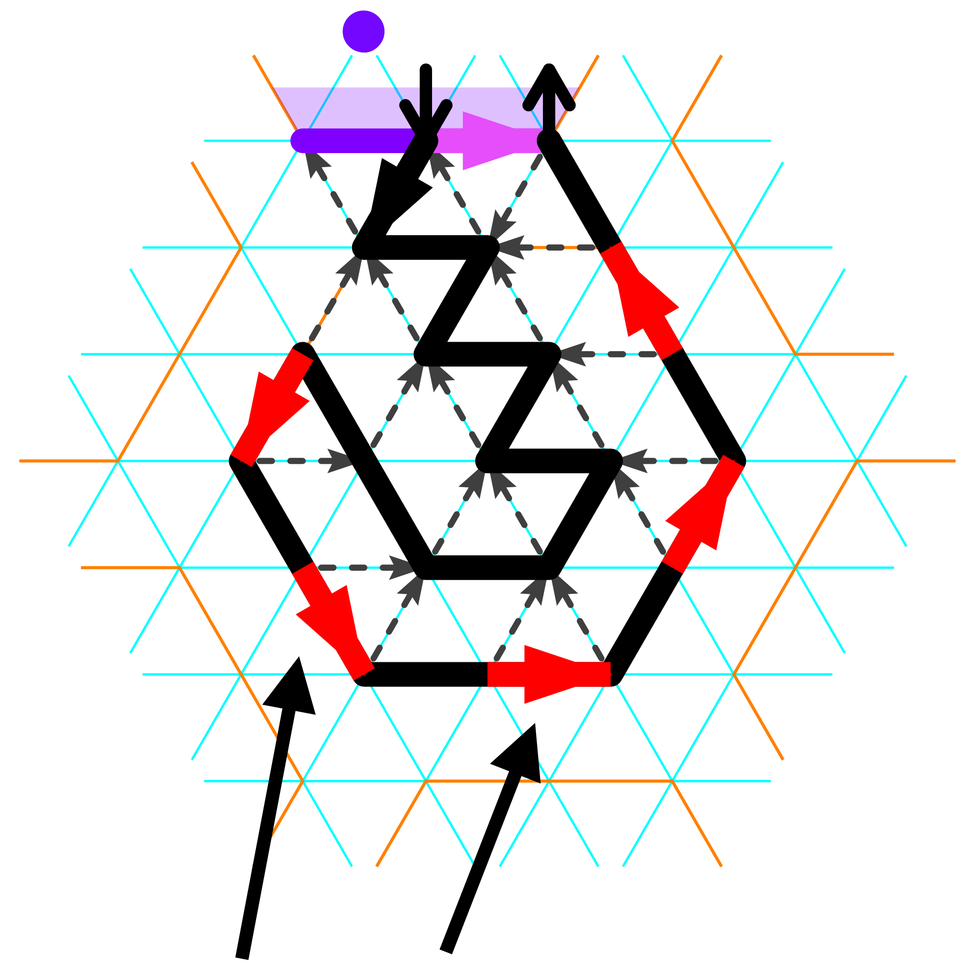


docking edges

scale $\mathscr{B}_n$

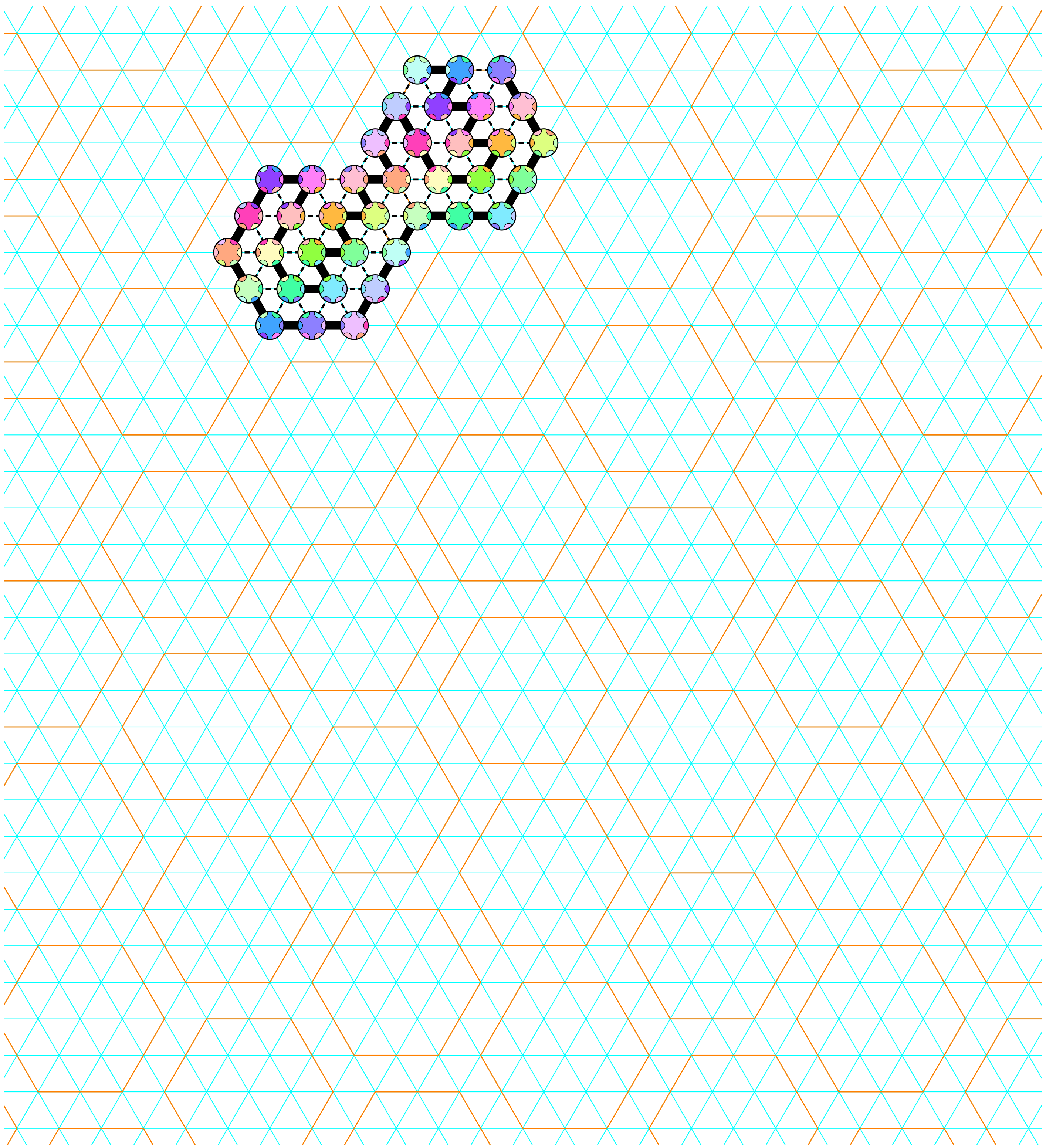


Use a unique pattern

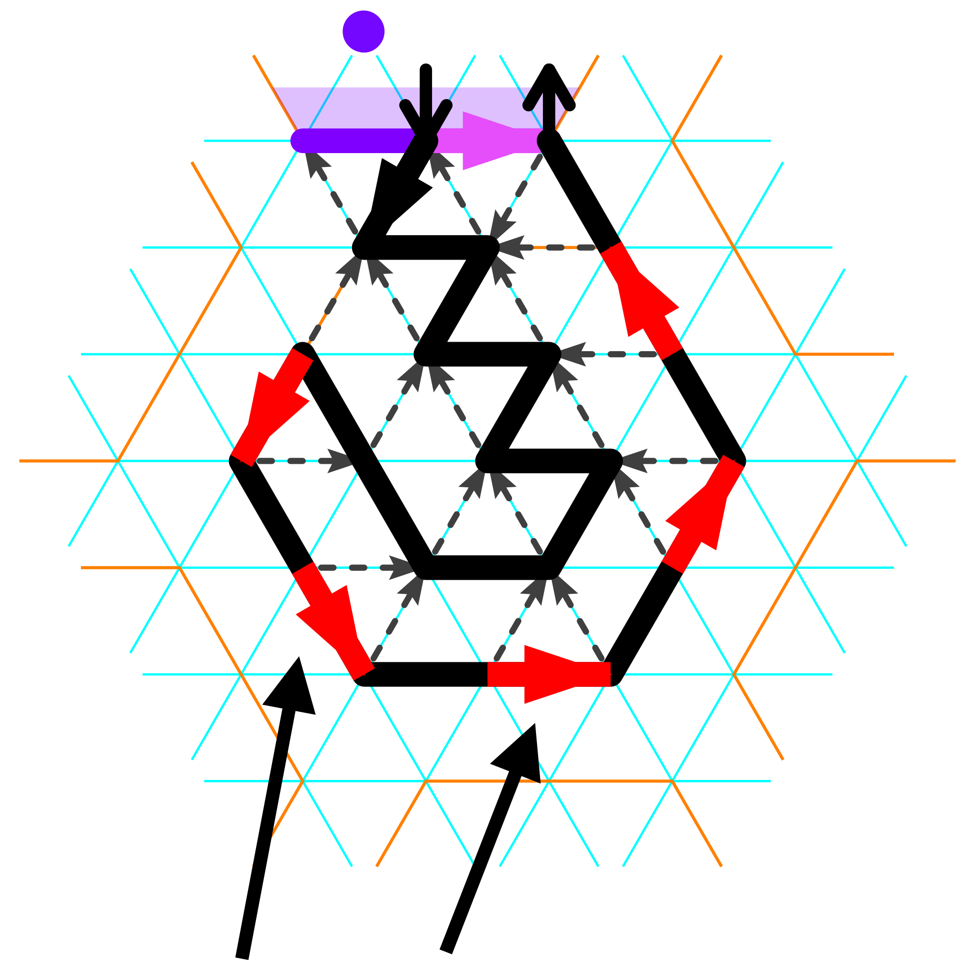


docking edges

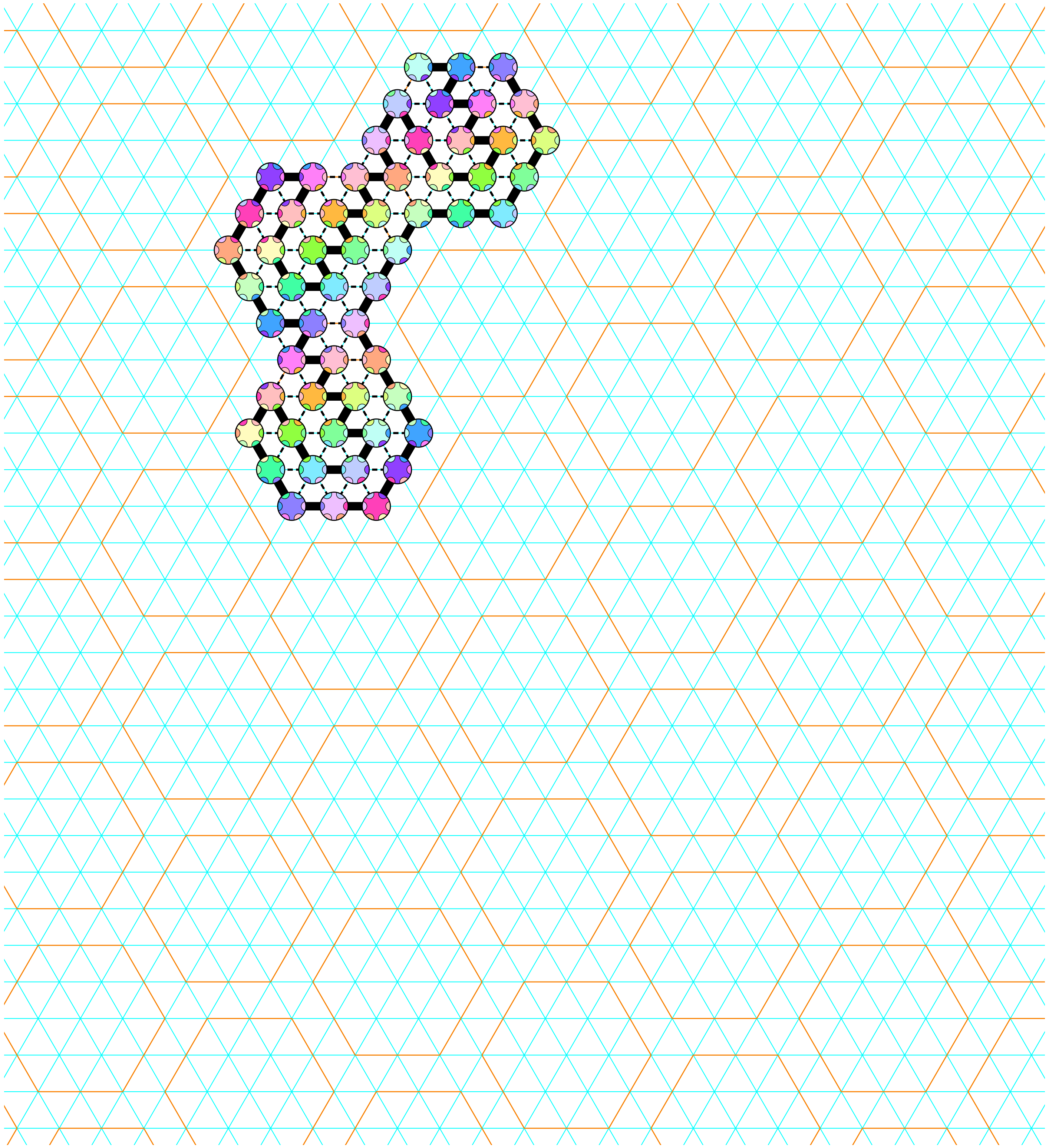
scale $\mathscr{B}_n$



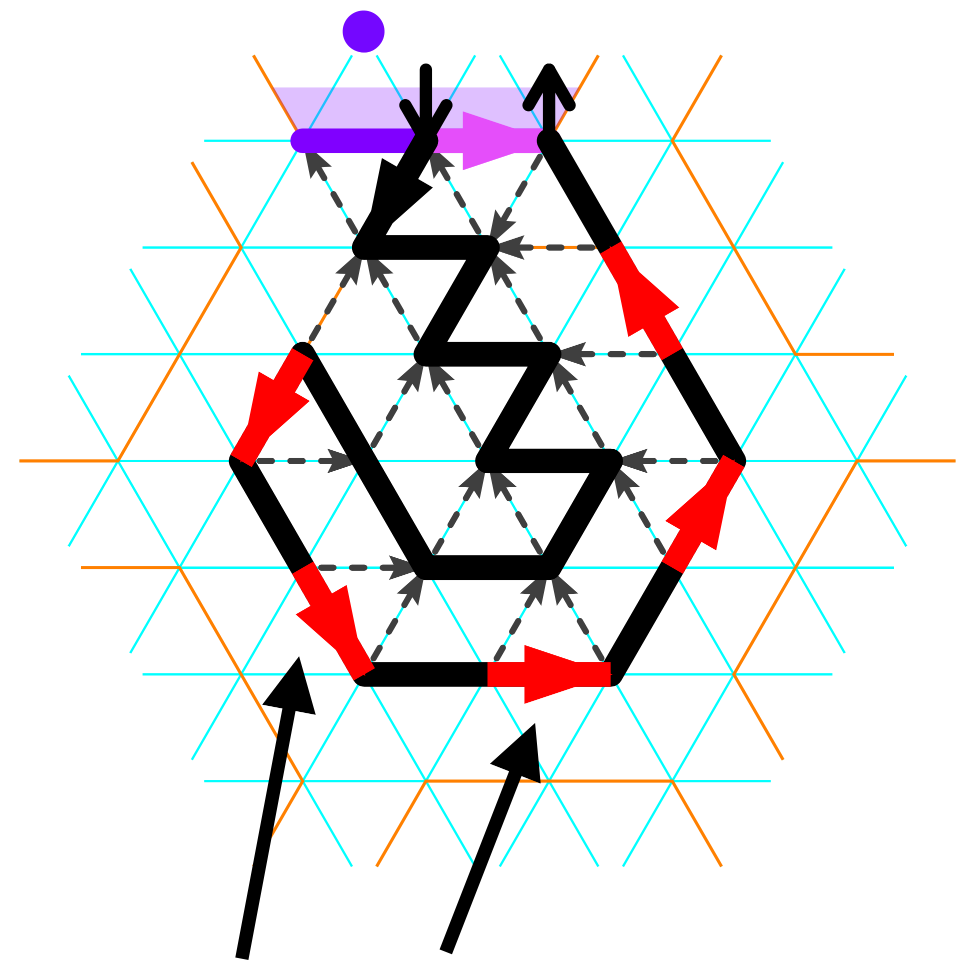
Use a unique pattern



scale $\mathscr{B}_n$

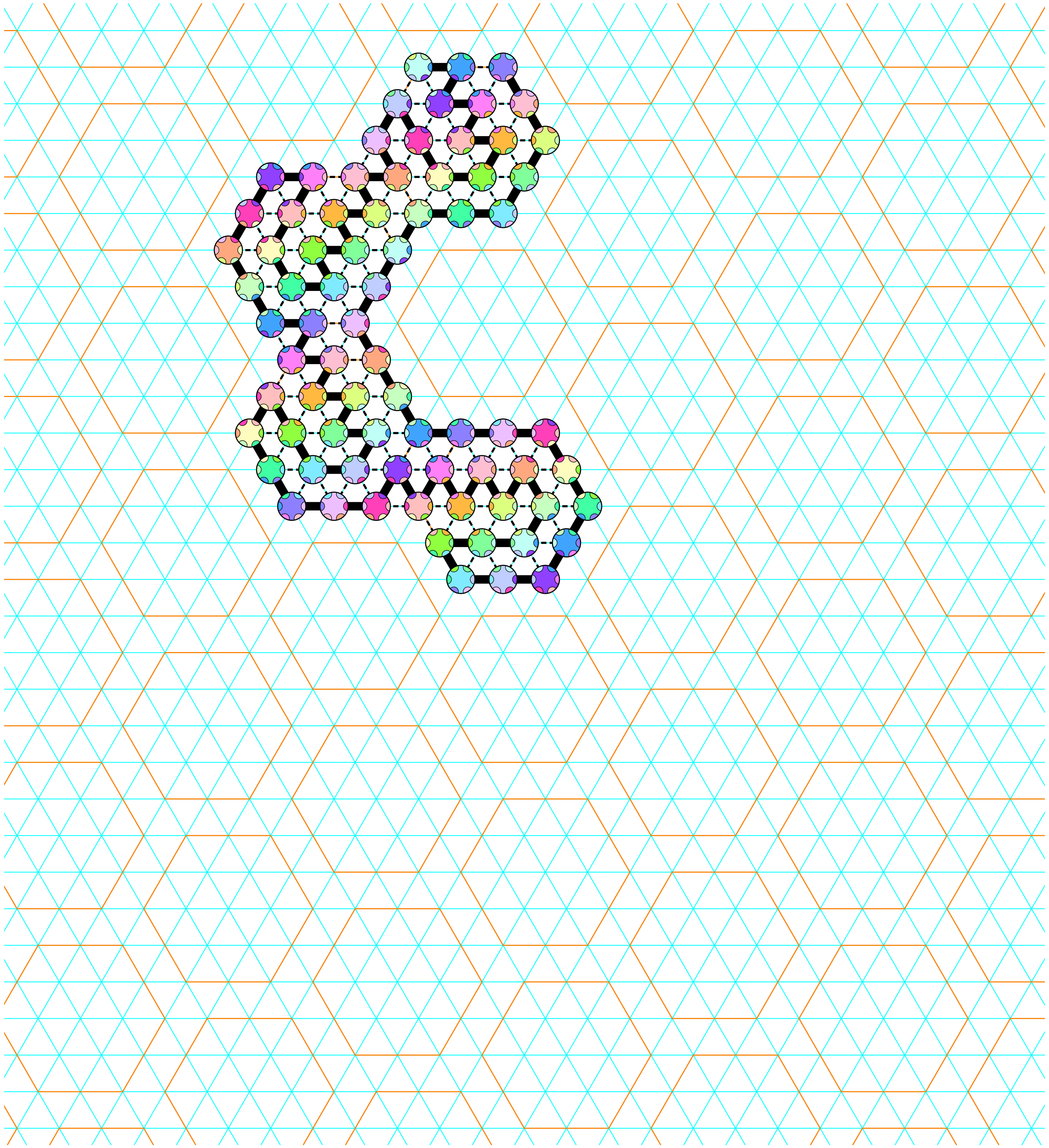


Use a unique pattern

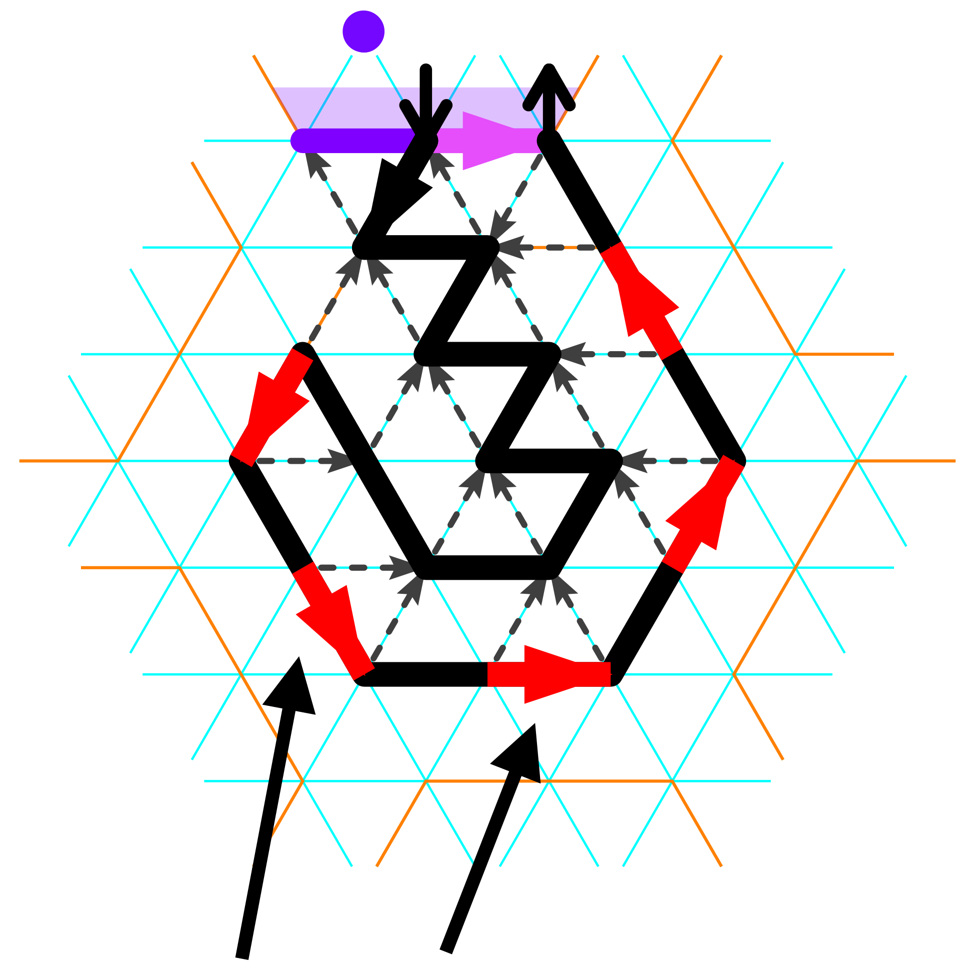


docking edges

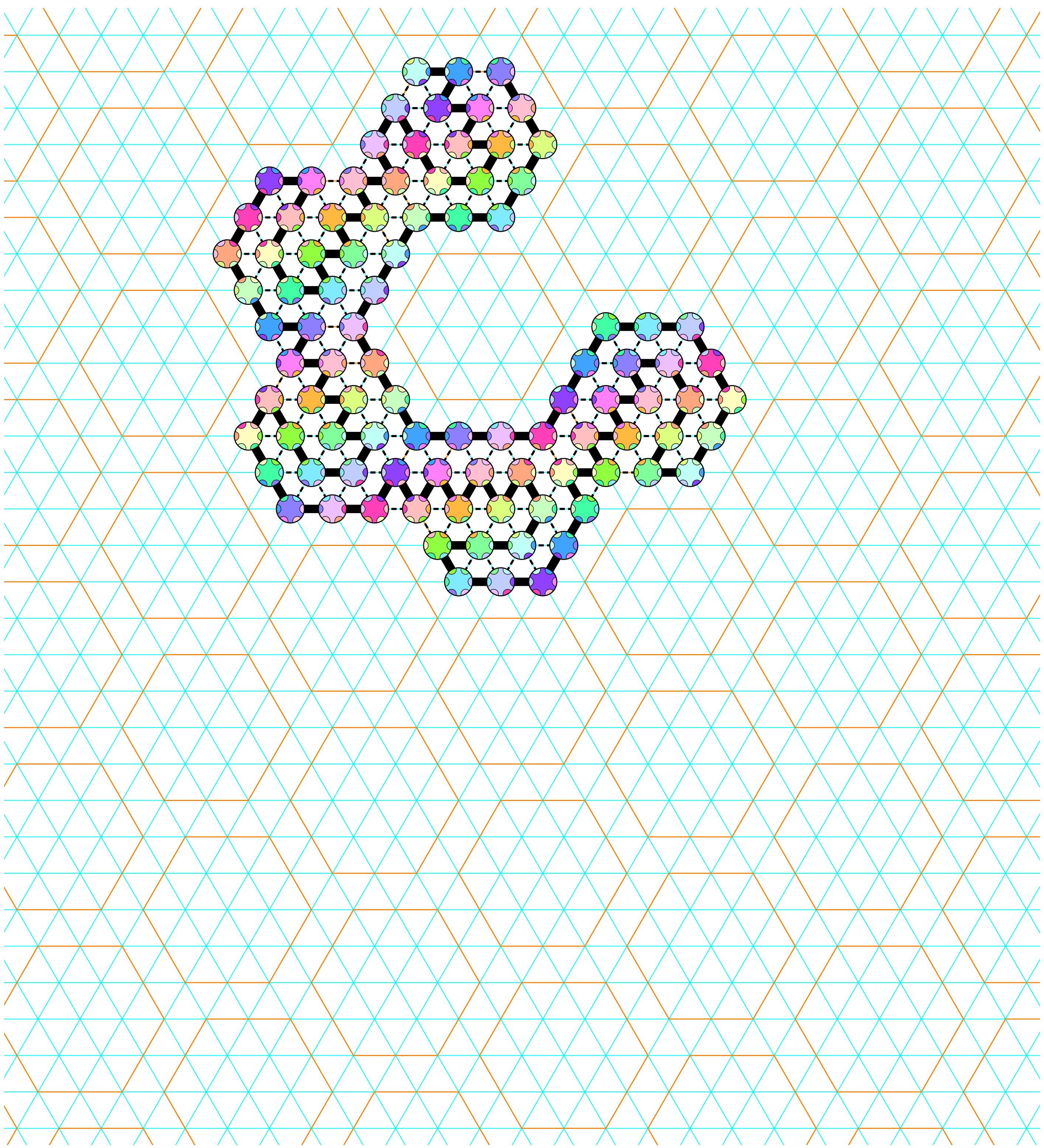
scale $\mathscr{B}_n$



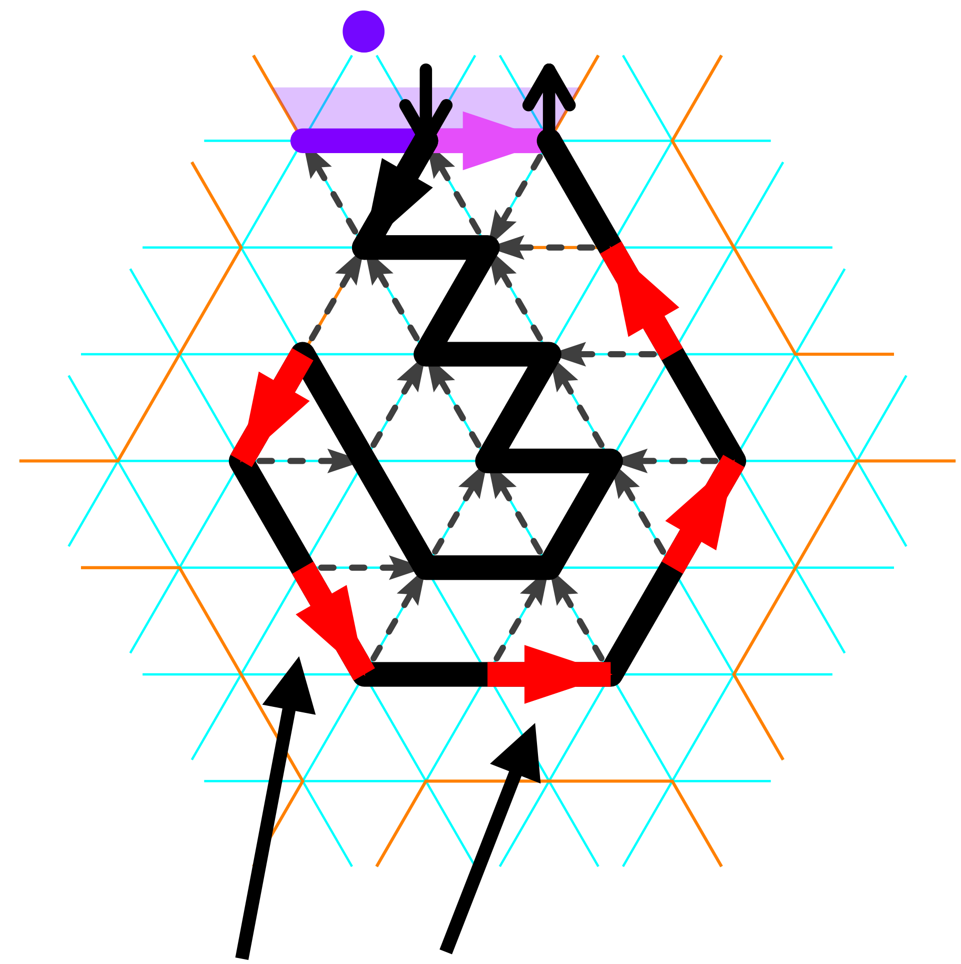
Use a unique pattern



scale $\mathscr{B}_n$

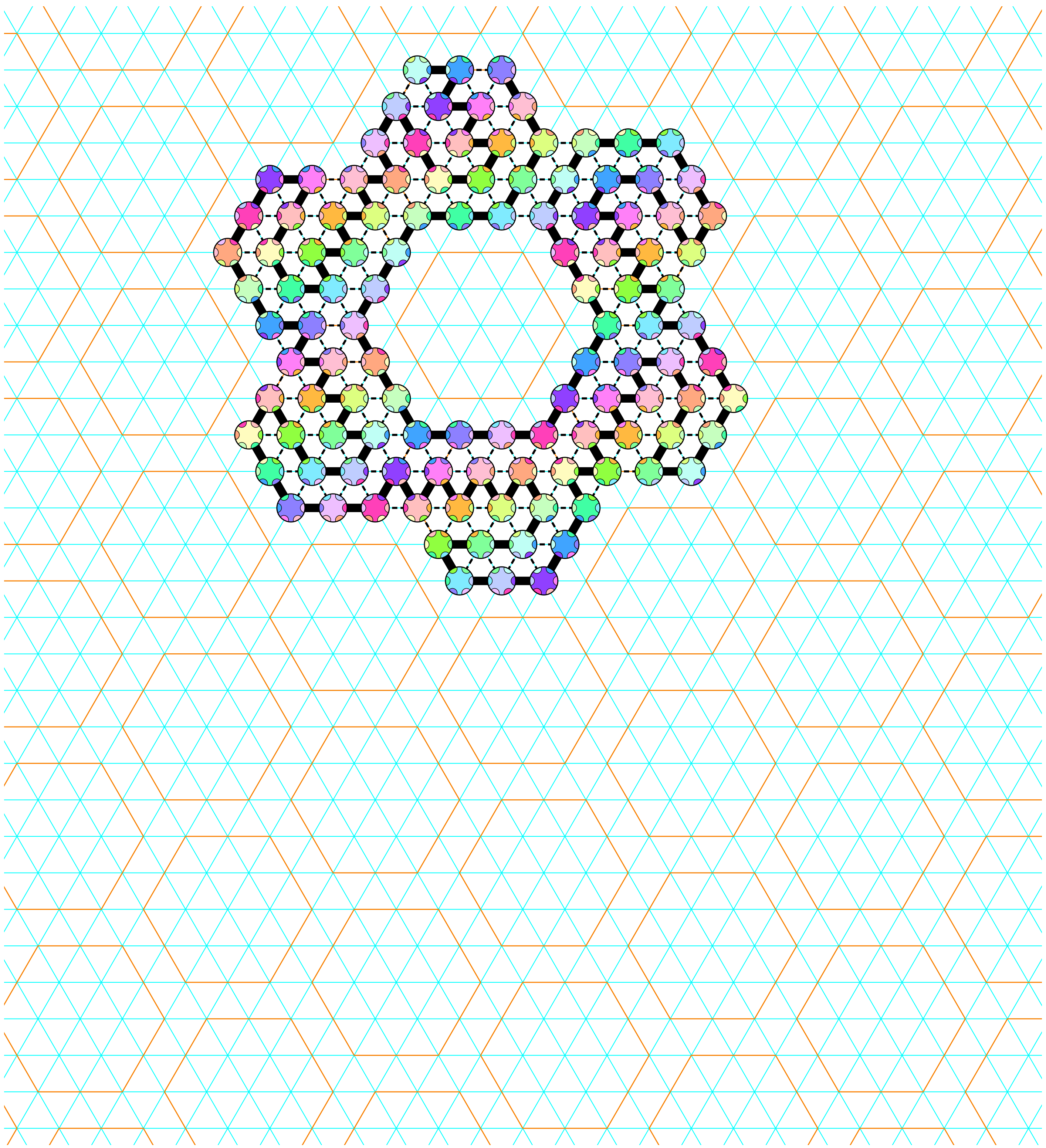


Use a unique pattern

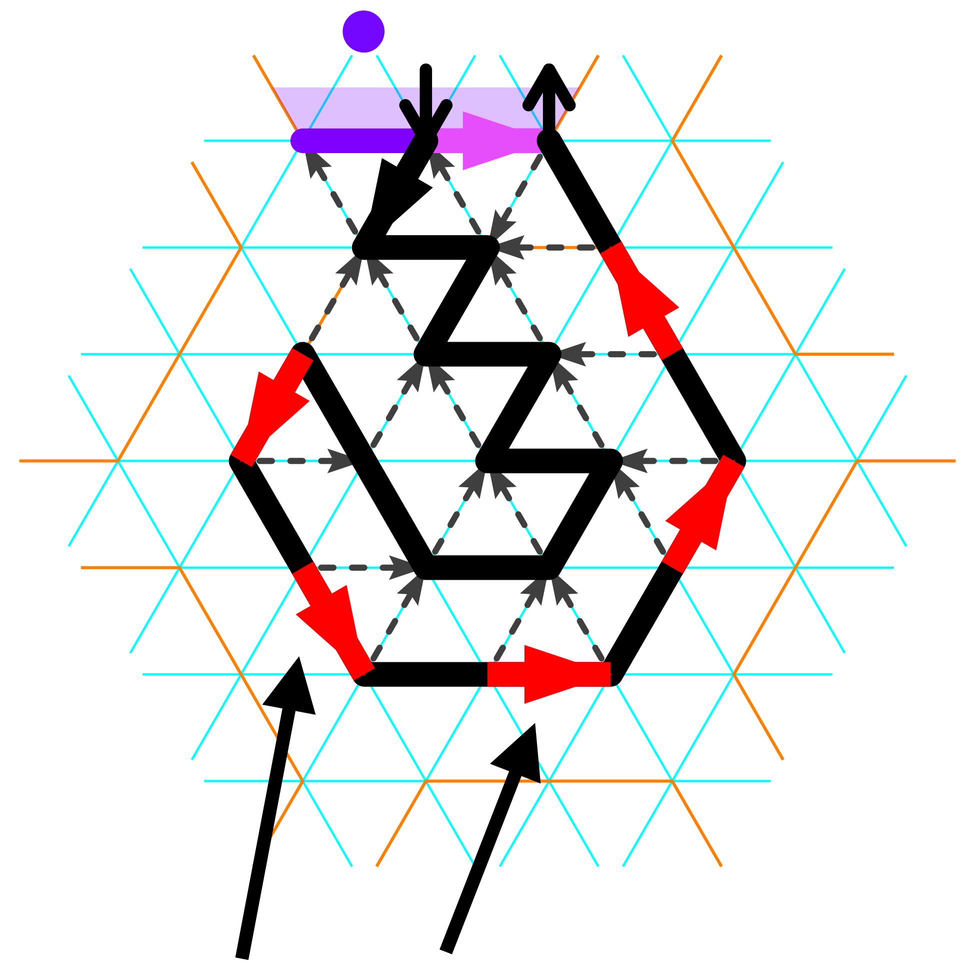


docking edges

scale $\mathscr{B}_n$

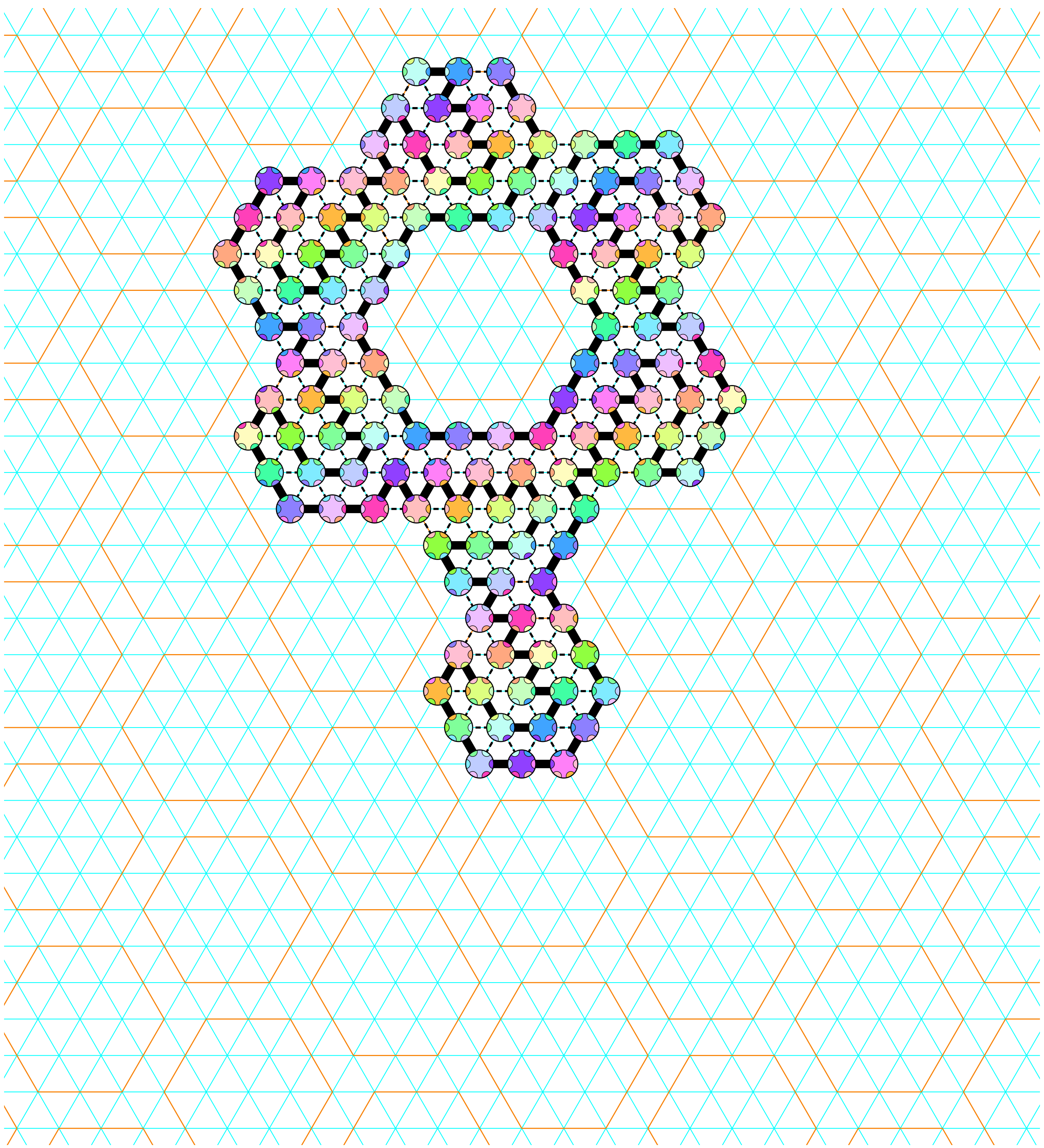


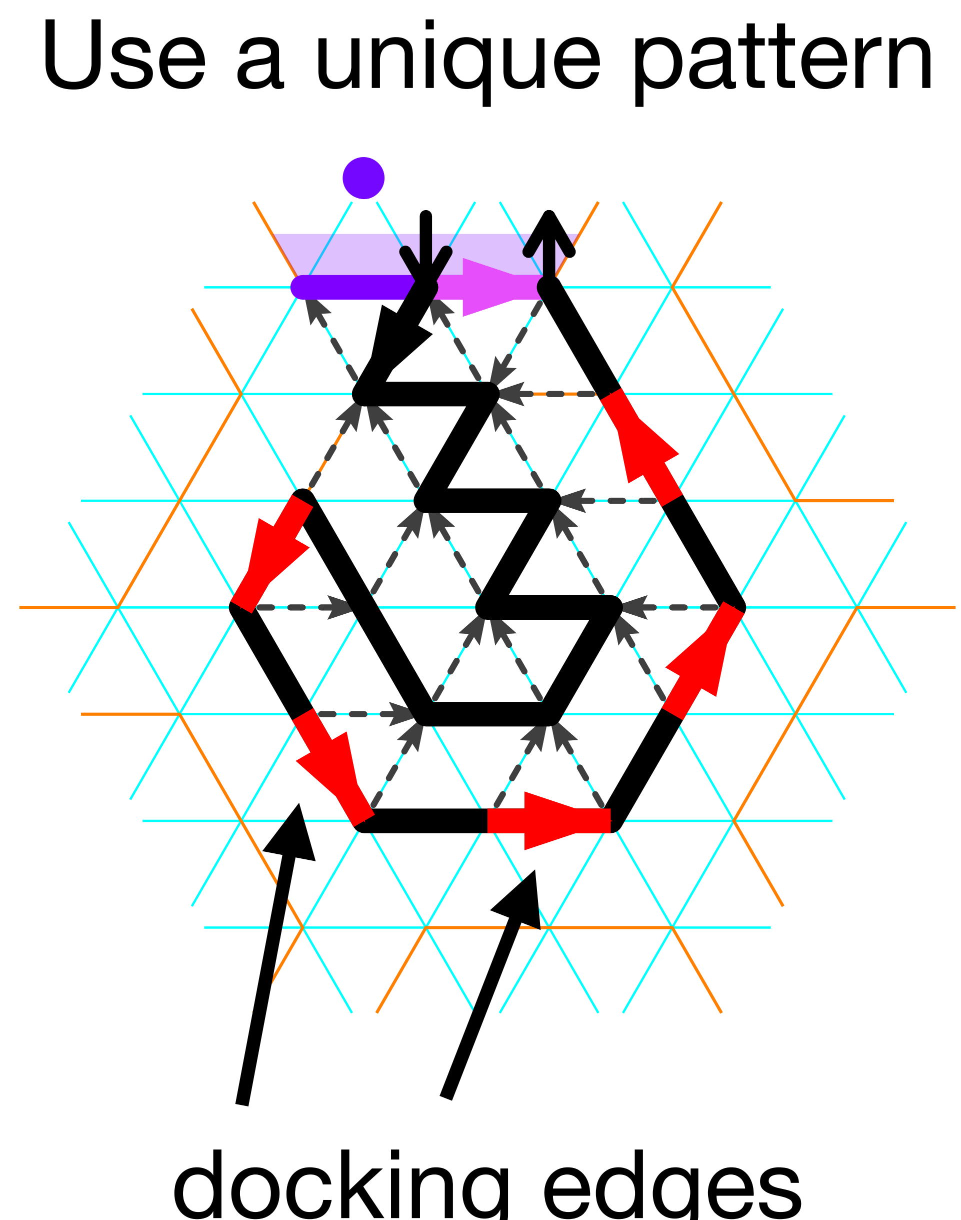
Use a unique pattern



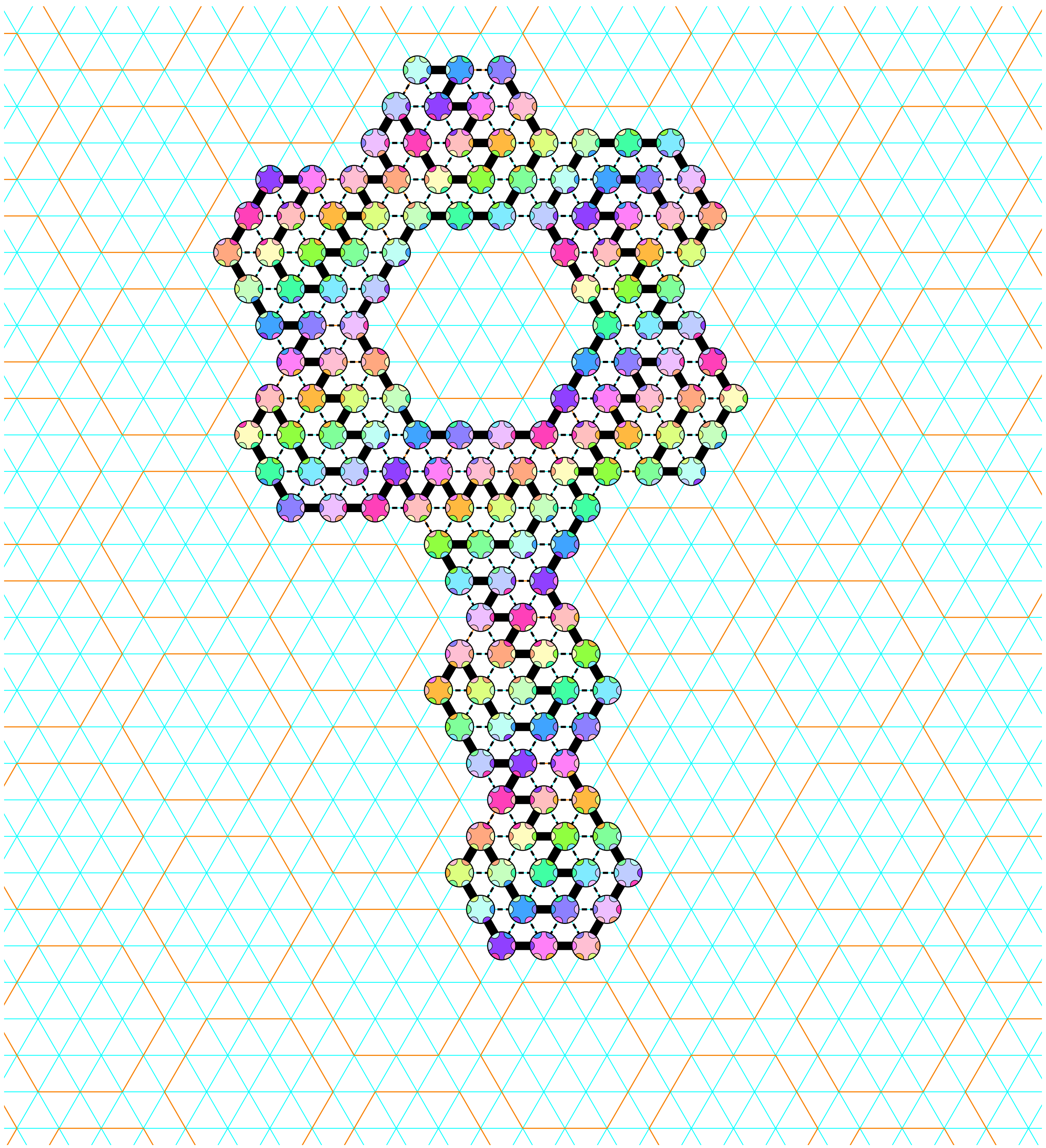
docking edges

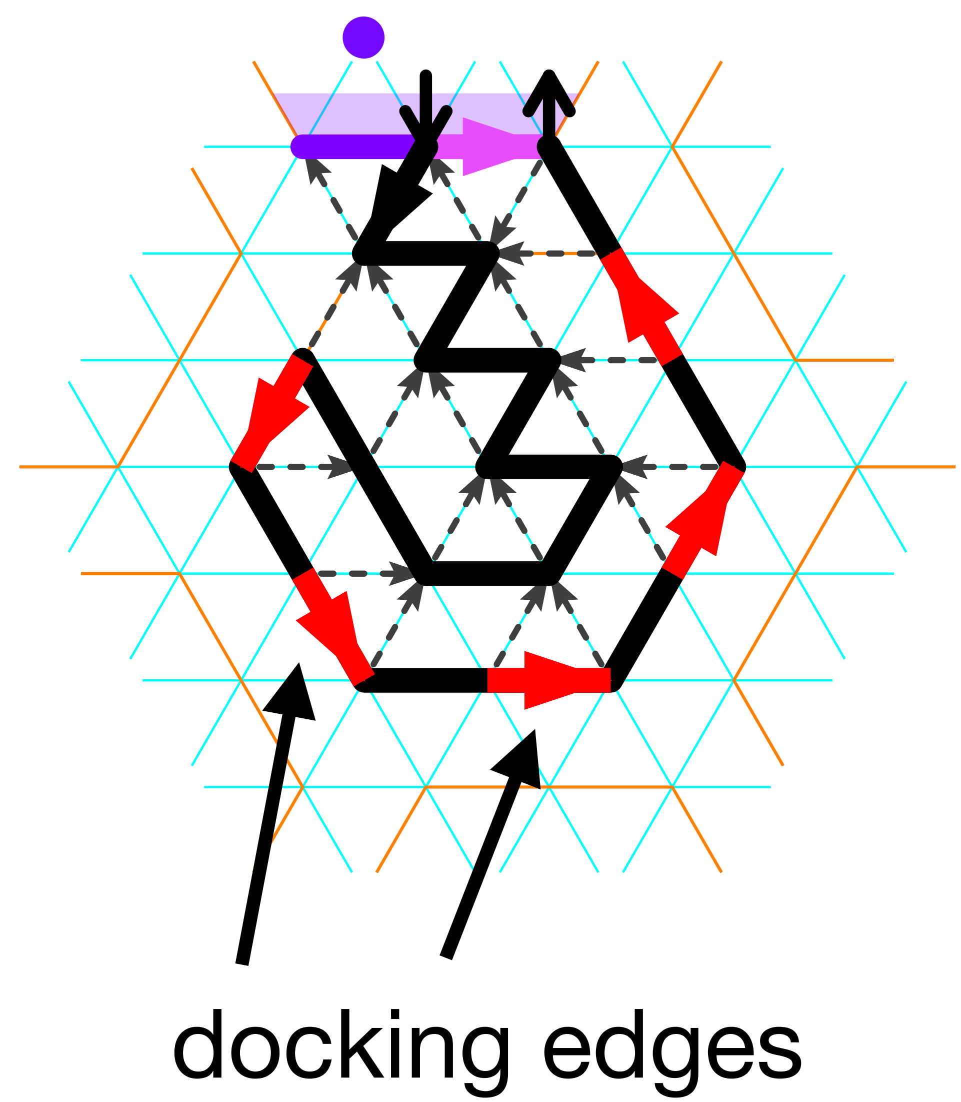
scale $\mathscr{B}_n$



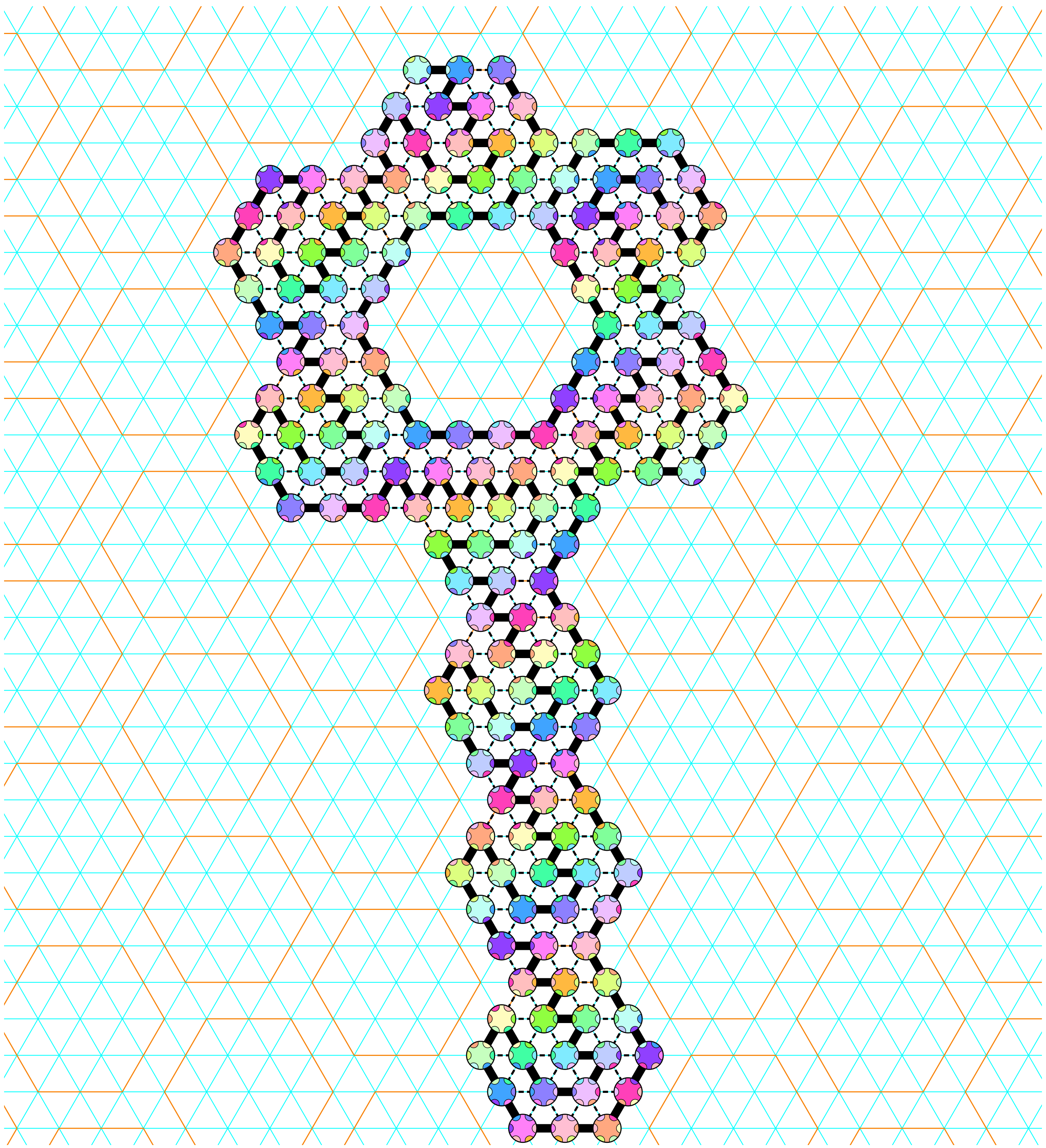


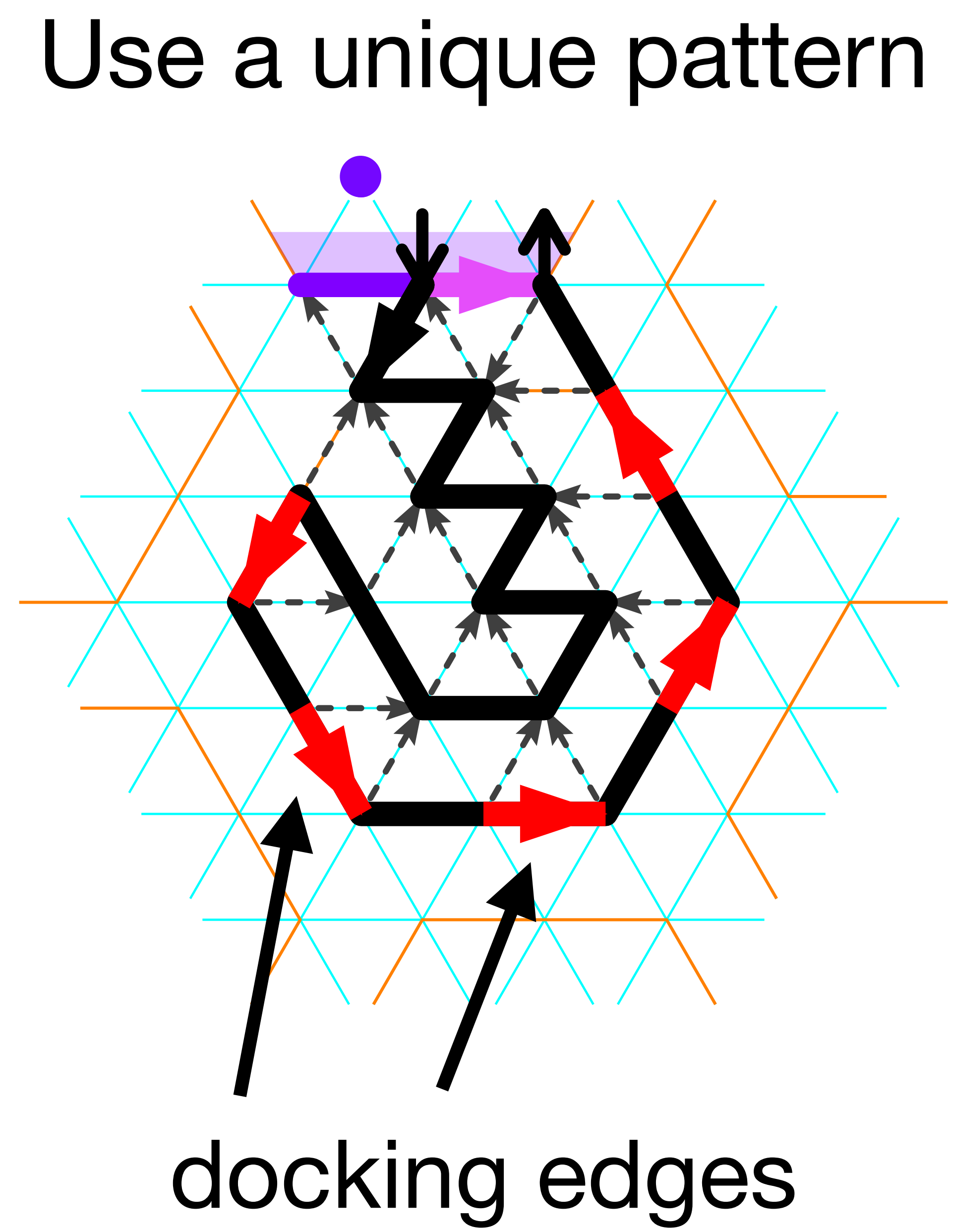
scale $\mathscr{B}_n$



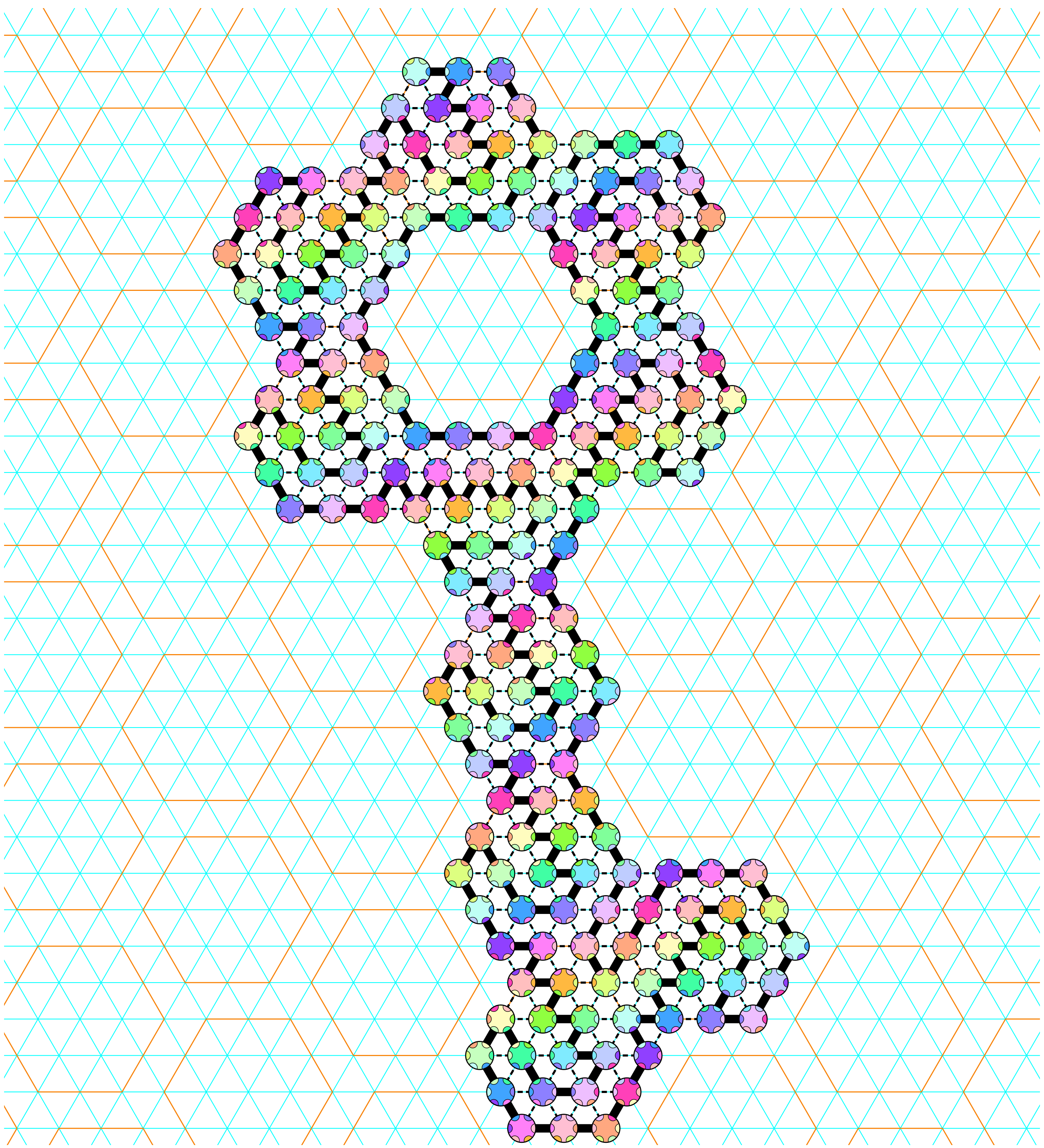


scale $\mathscr{B}_n$

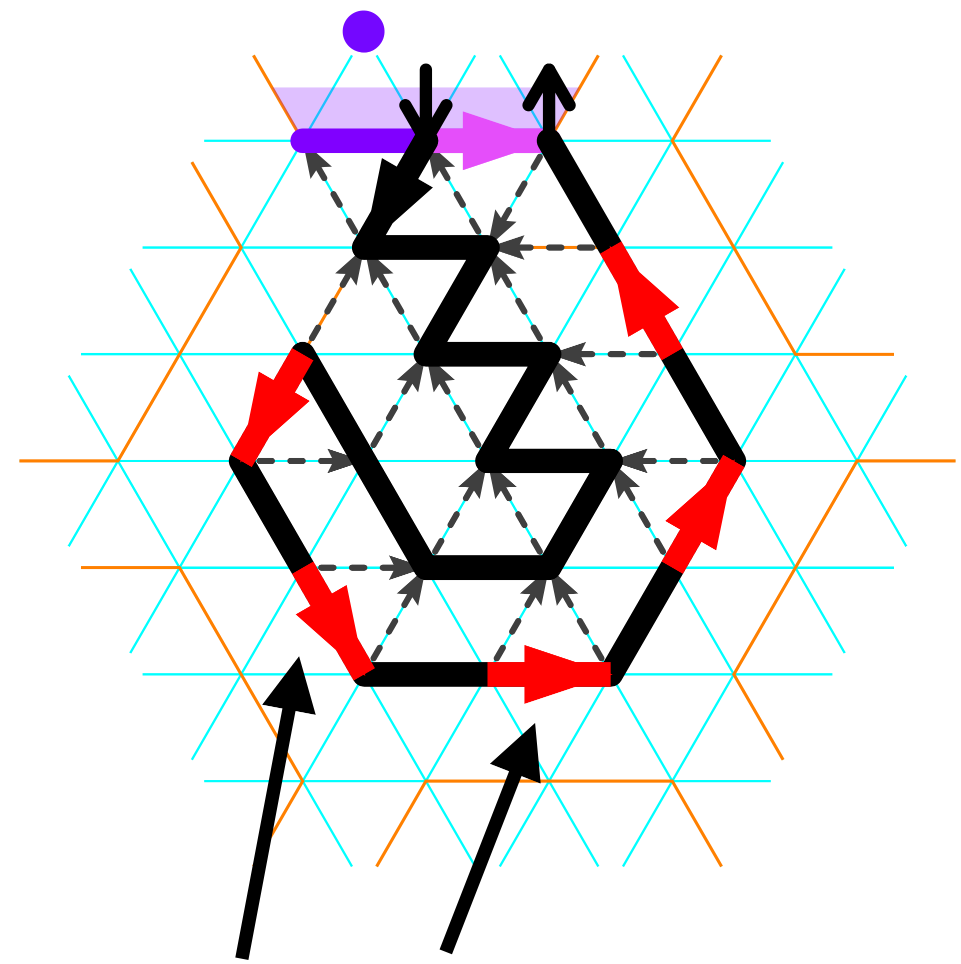




scale $\mathscr{B}_n$

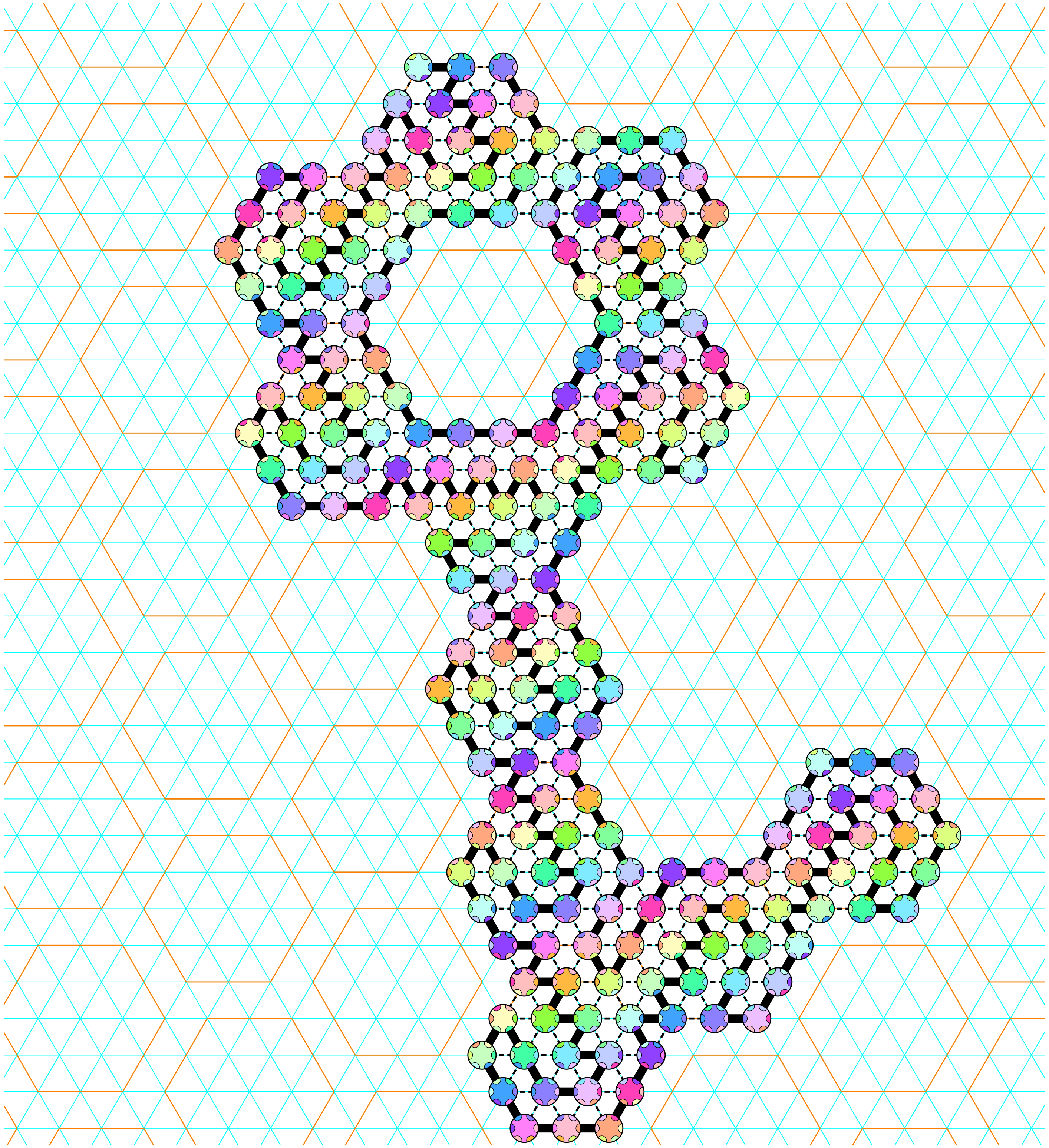


Use a unique pattern

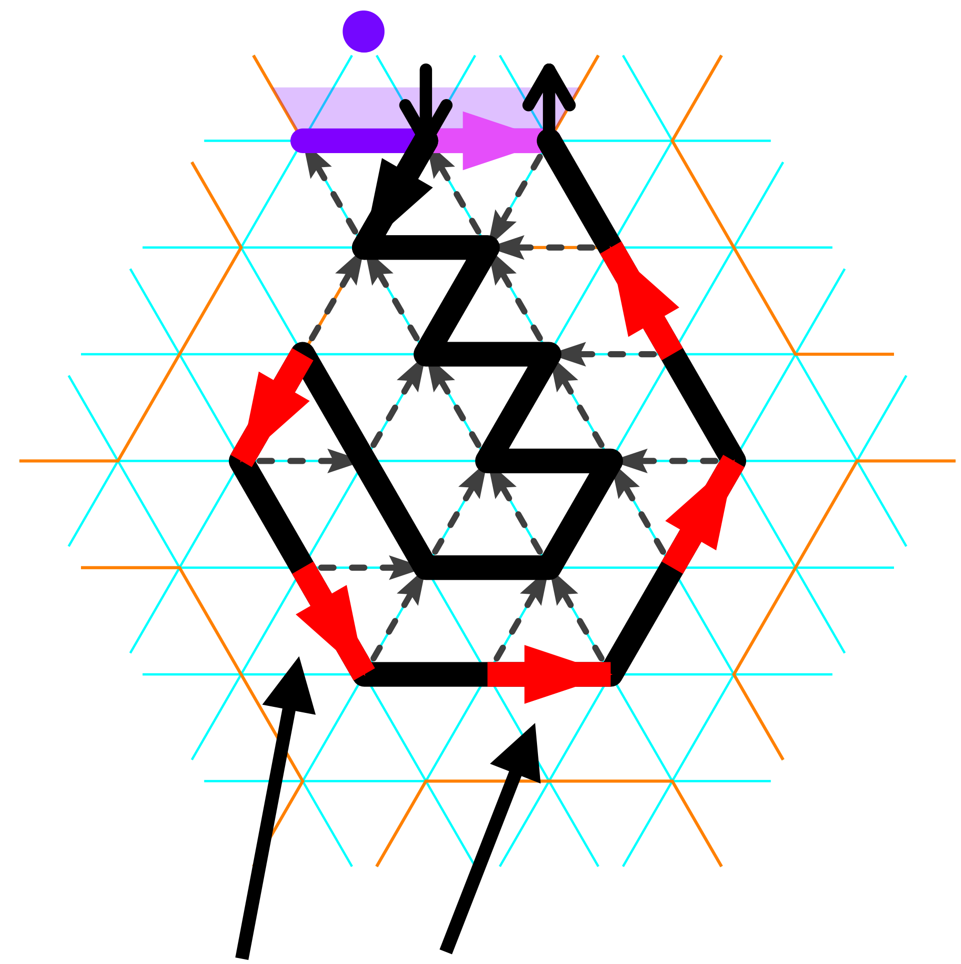


docking edges

scale $\mathscr{B}_n$

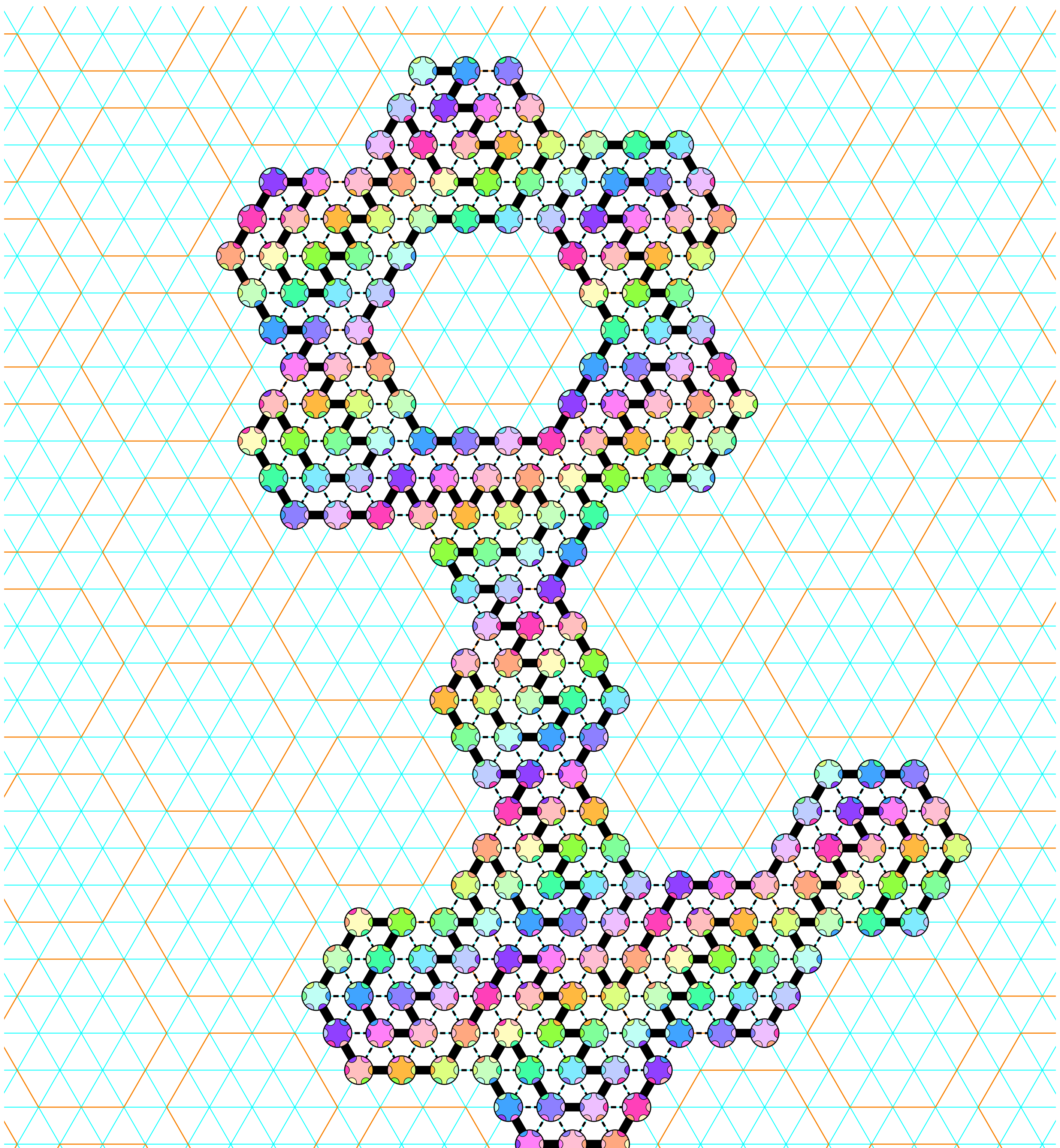


Use a unique pattern

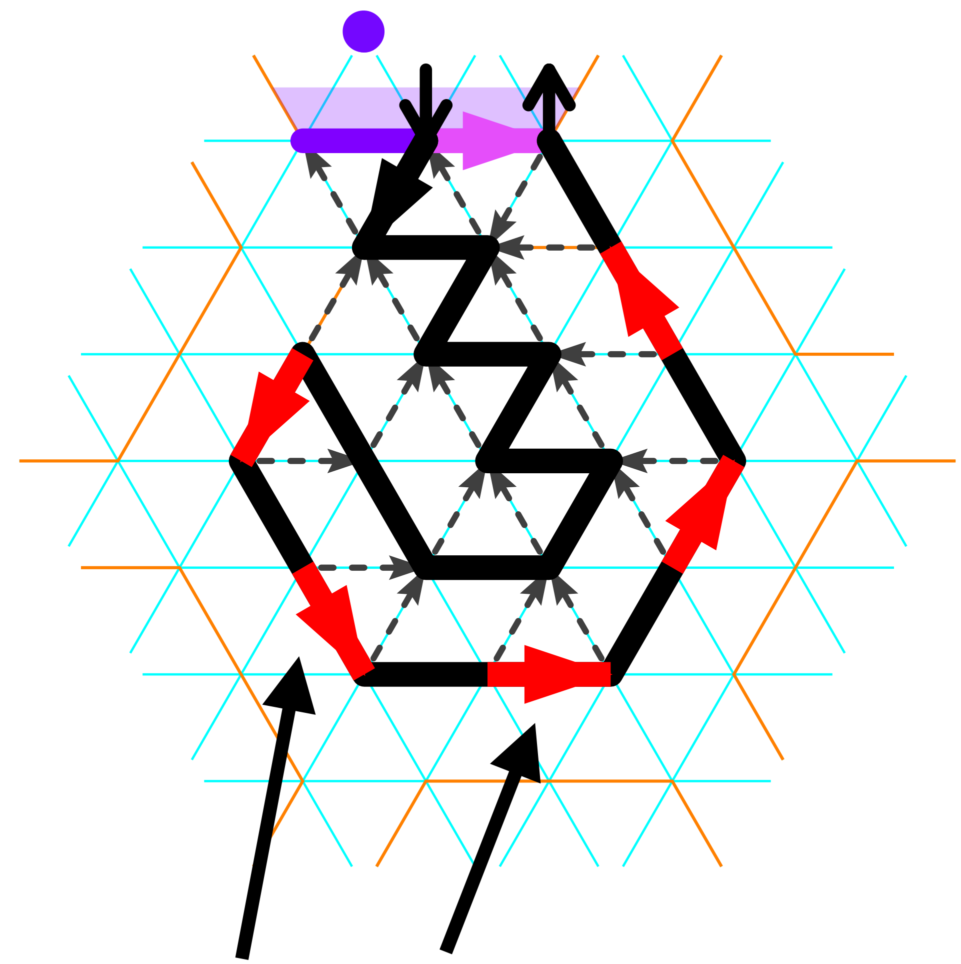


docking edges

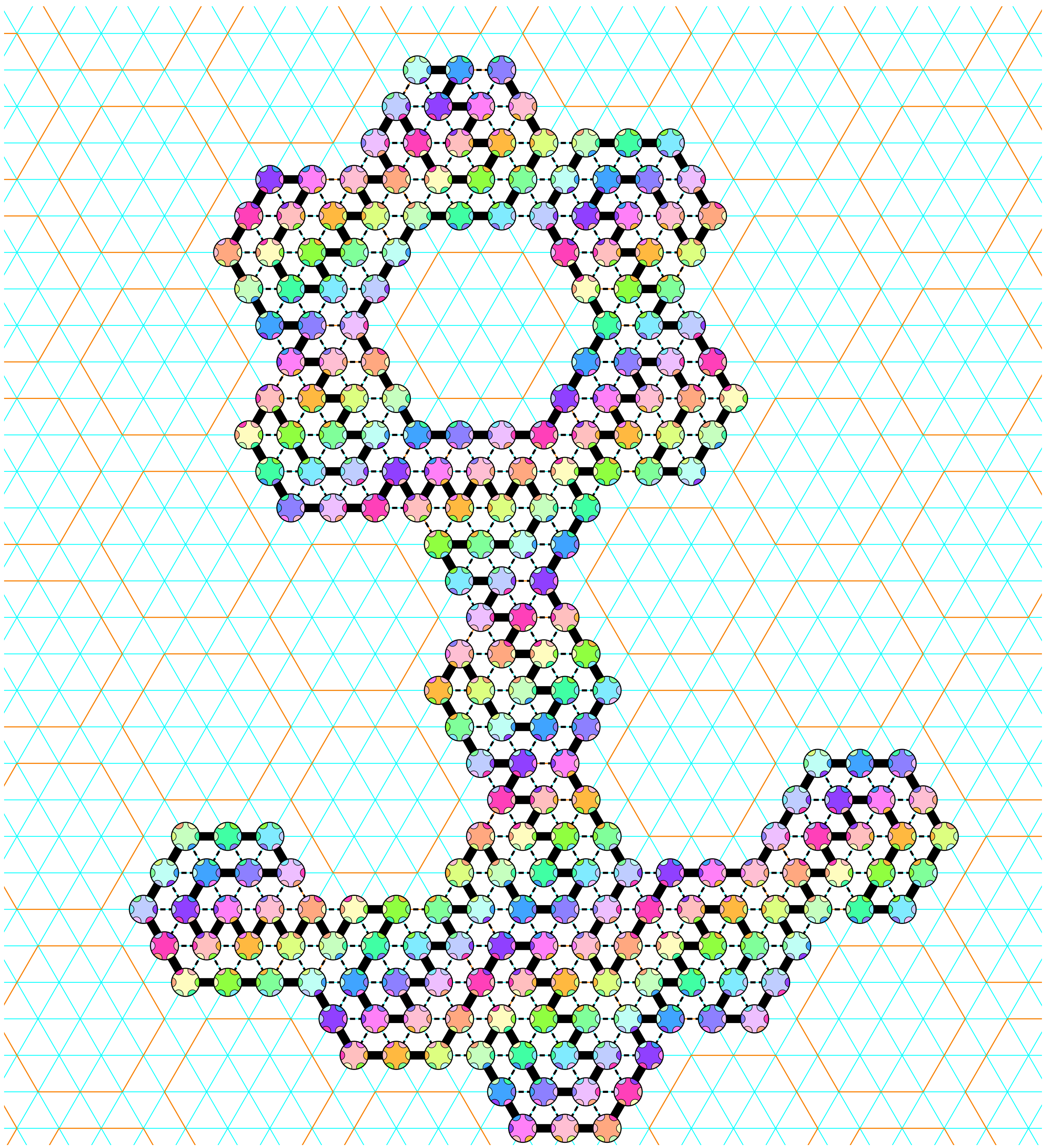
scale $\mathscr{B}_n$



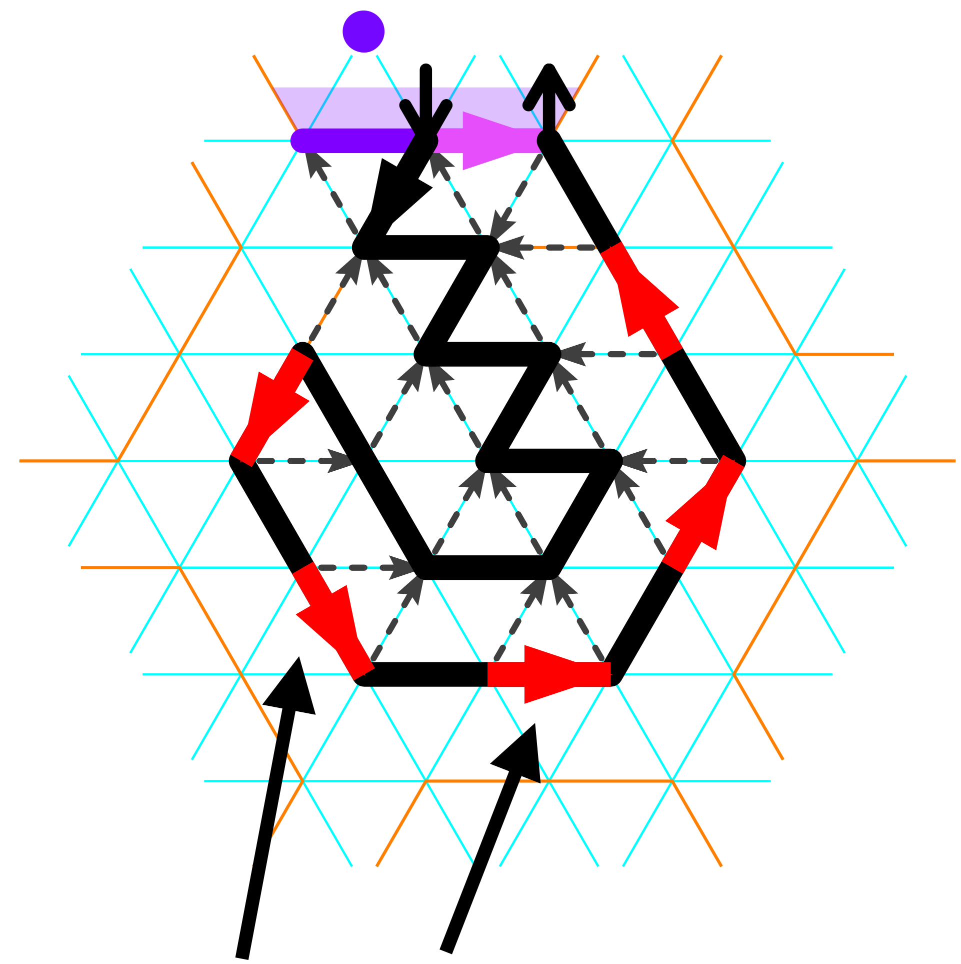
Use a unique pattern



scale $\mathscr{B}_n$

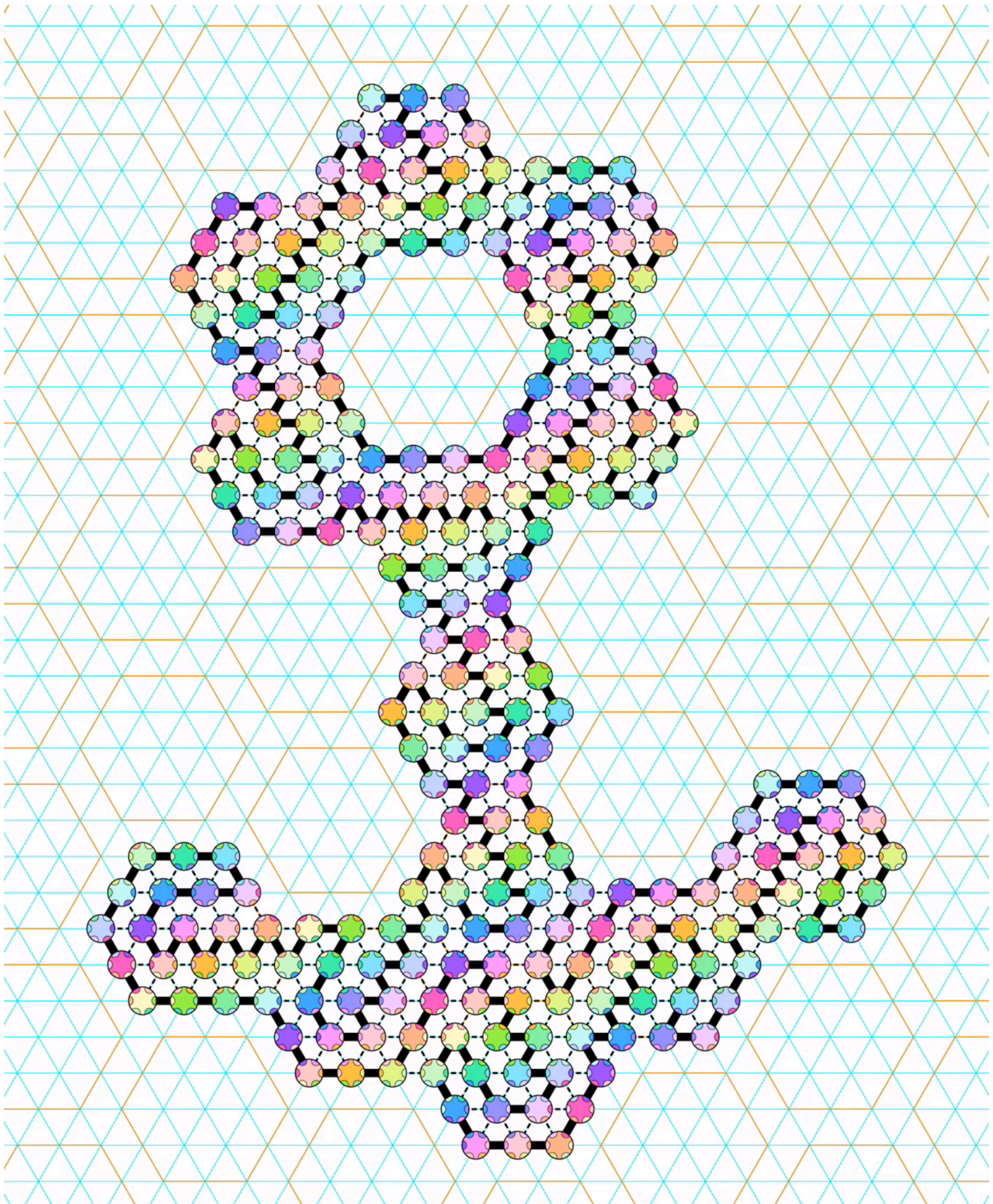


Use a unique pattern

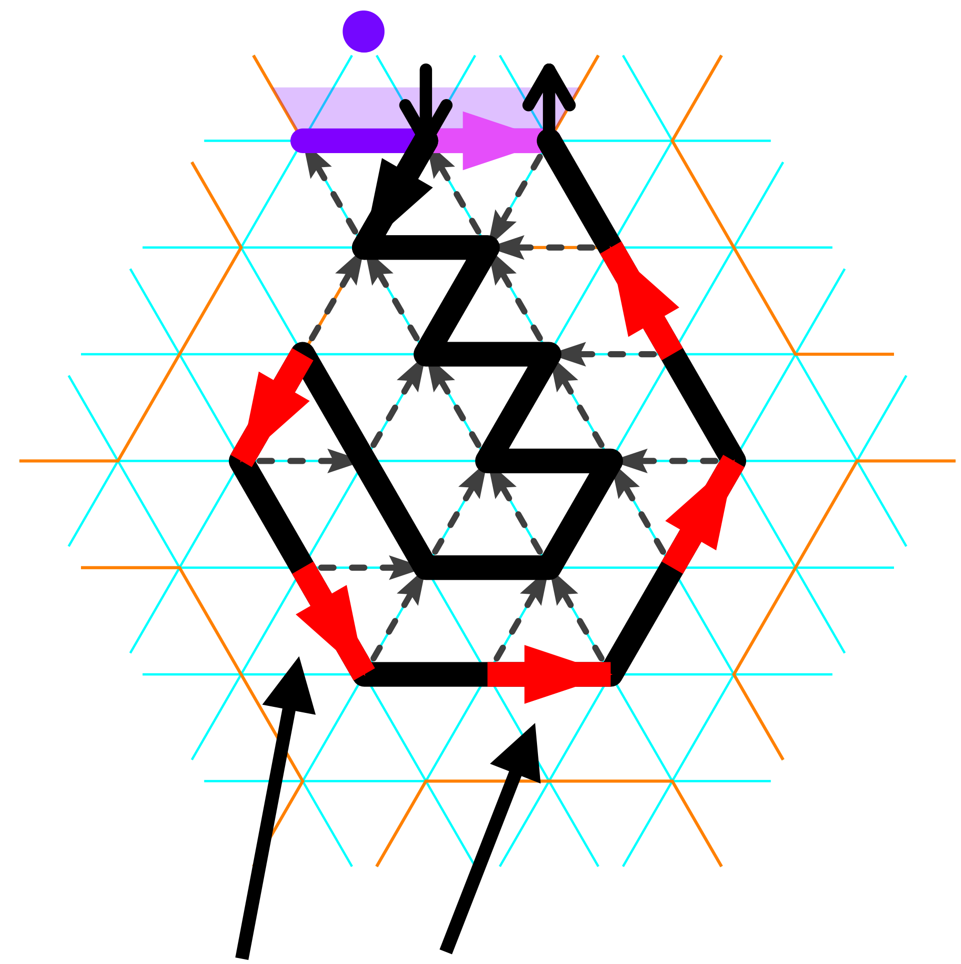


docking edges

scale $\mathcal{B}_n$



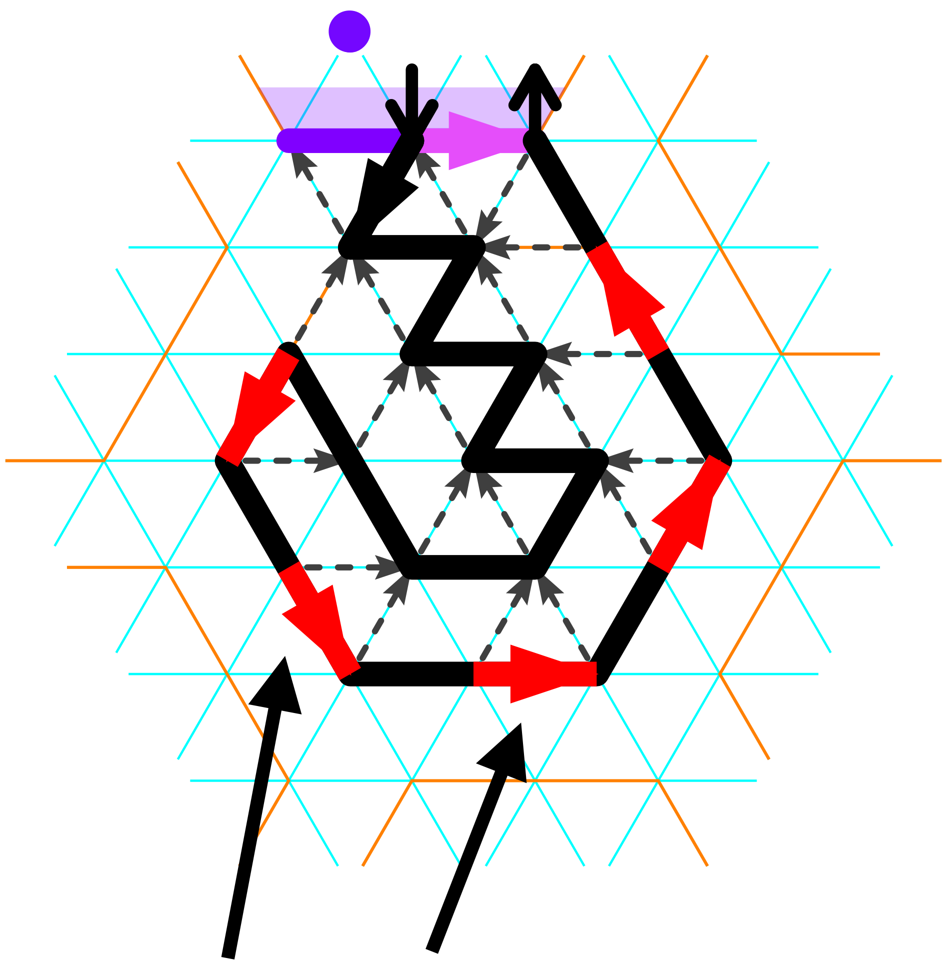
Use a unique pattern



docking edges

scale $\mathscr{B}_n$

Use a unique pattern



docking edges

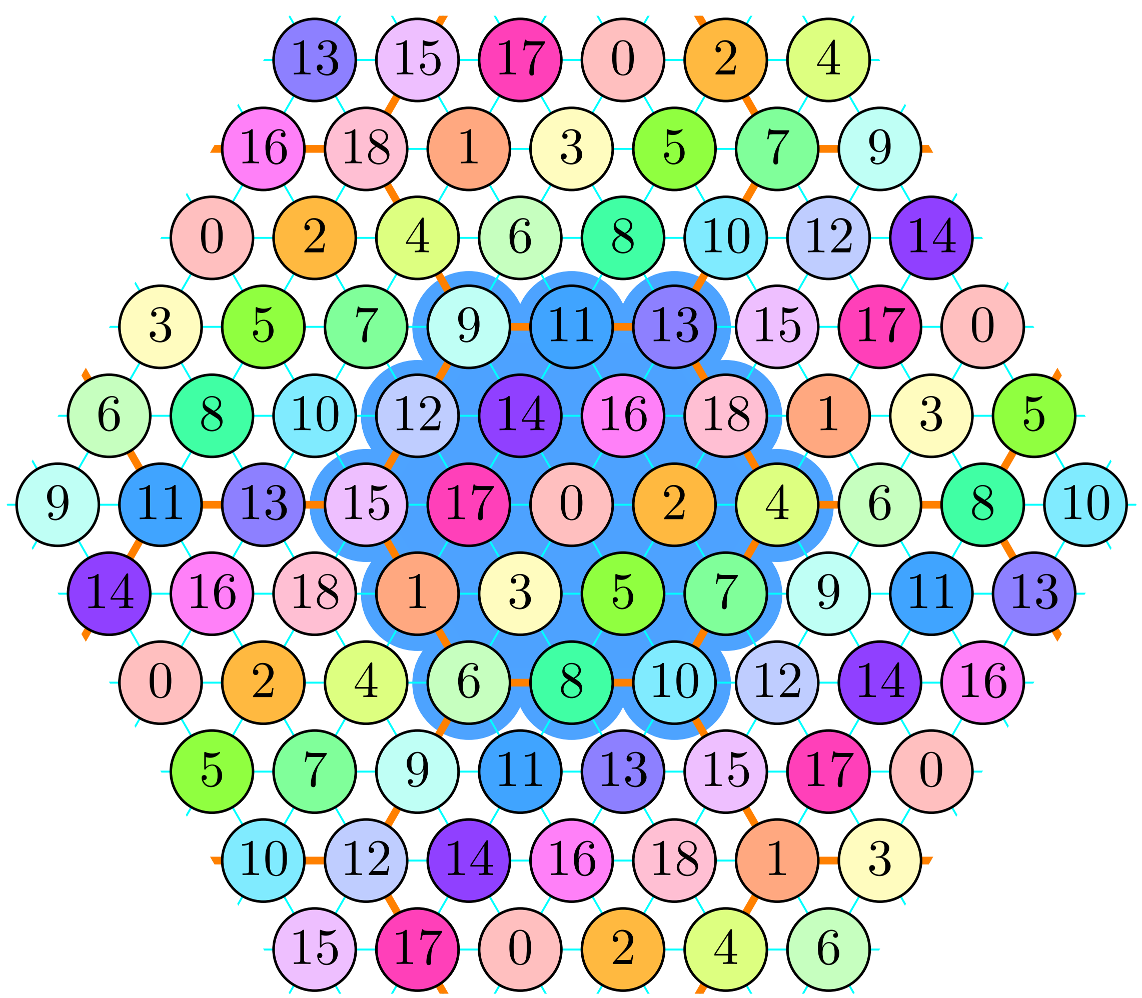
Theorem. All finite shapes can be folded at scale $\mathscr{B}_n$ *for* $n \geq 3$

Proof. By induction:

For **all red edges**, the corresponding **three purple positions** are filled before.

How many bead types are needed?

Affine coloring of hexagons



Theorem. Let H_n be the hexagon of radius n,

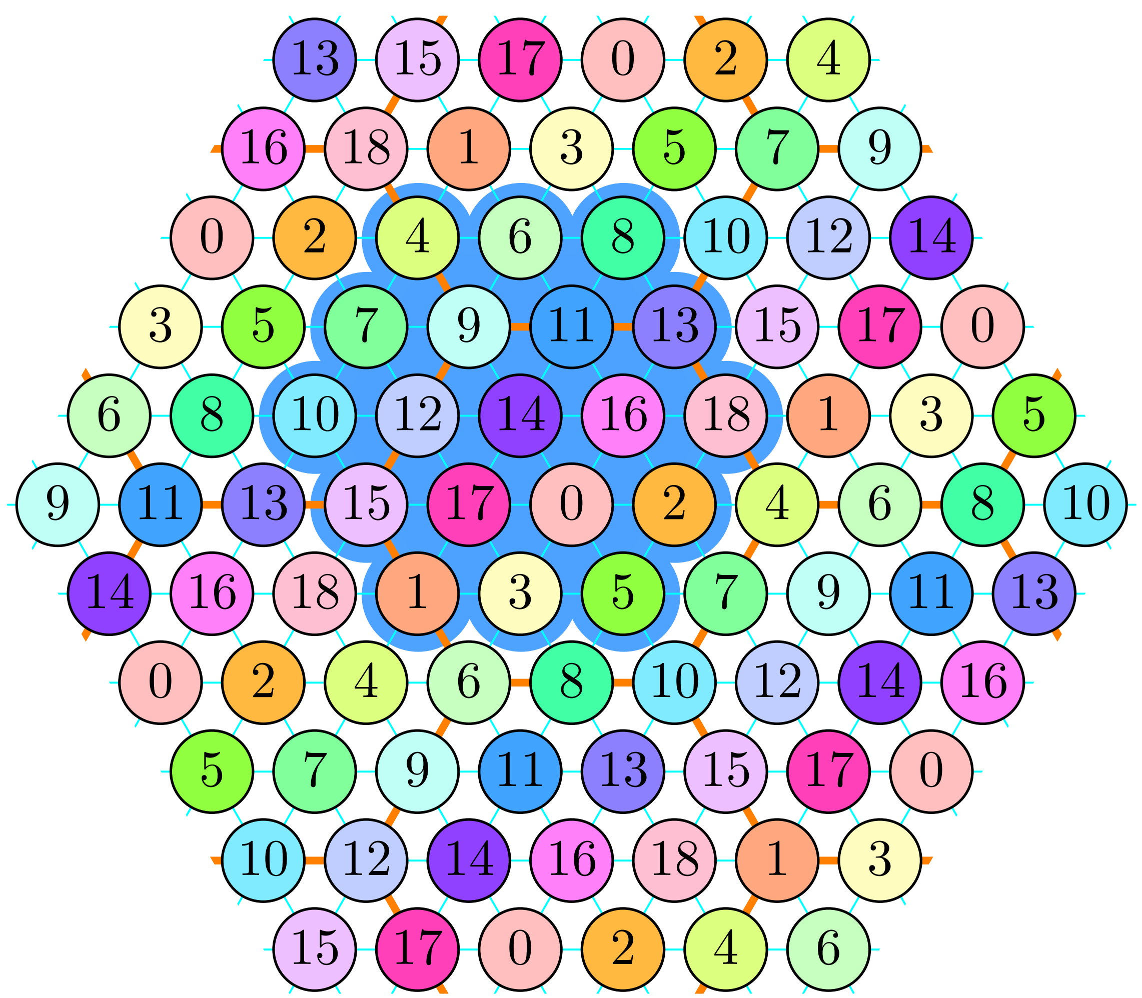
$$c(i,j) = ni + (n+1)j \bmod |H_n|$$

is a proper coloring of H_n

Corolary 1. As it is affine, it is a proper coloring of *any* translation of H_n

Corolary 2. Furthermore, the colors of the neighbors of a given node are fixed translations modulo $|H_n|$ of its own color

Affine coloring of hexagons



Theorem. Let H_n be the hexagon of radius n,

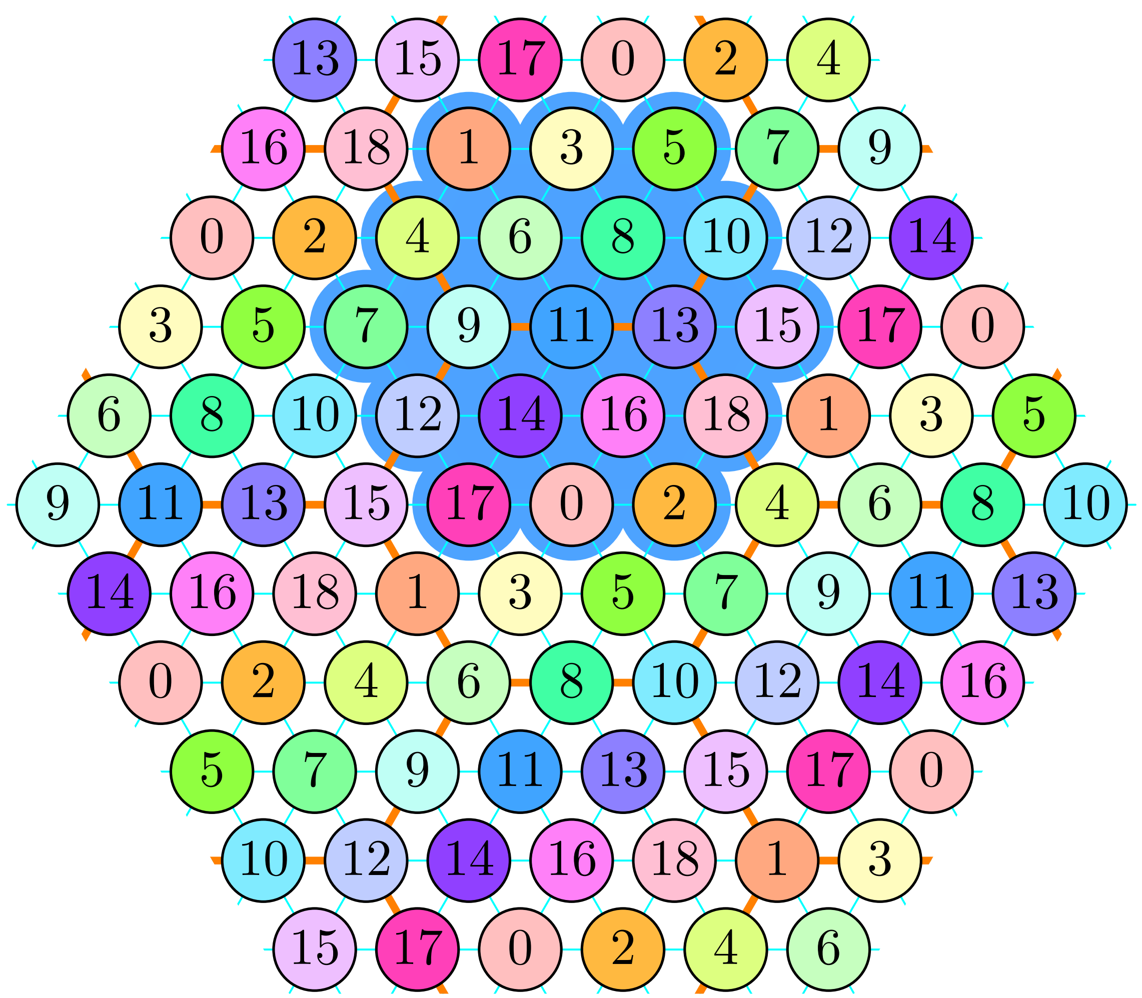
$$c(i,j) = ni + (n+1)j \bmod |H_n|$$

is a proper coloring of H_n

Corolary 1. As it is affine, it is a proper coloring of *any* translation of H_n

Corolary 2. Furthermore, the colors of the neighbors of a given node are fixed translations modulo $|H_n|$ of its own color

Affine coloring of hexagons



Theorem. Let H_n be the hexagon of radius n,

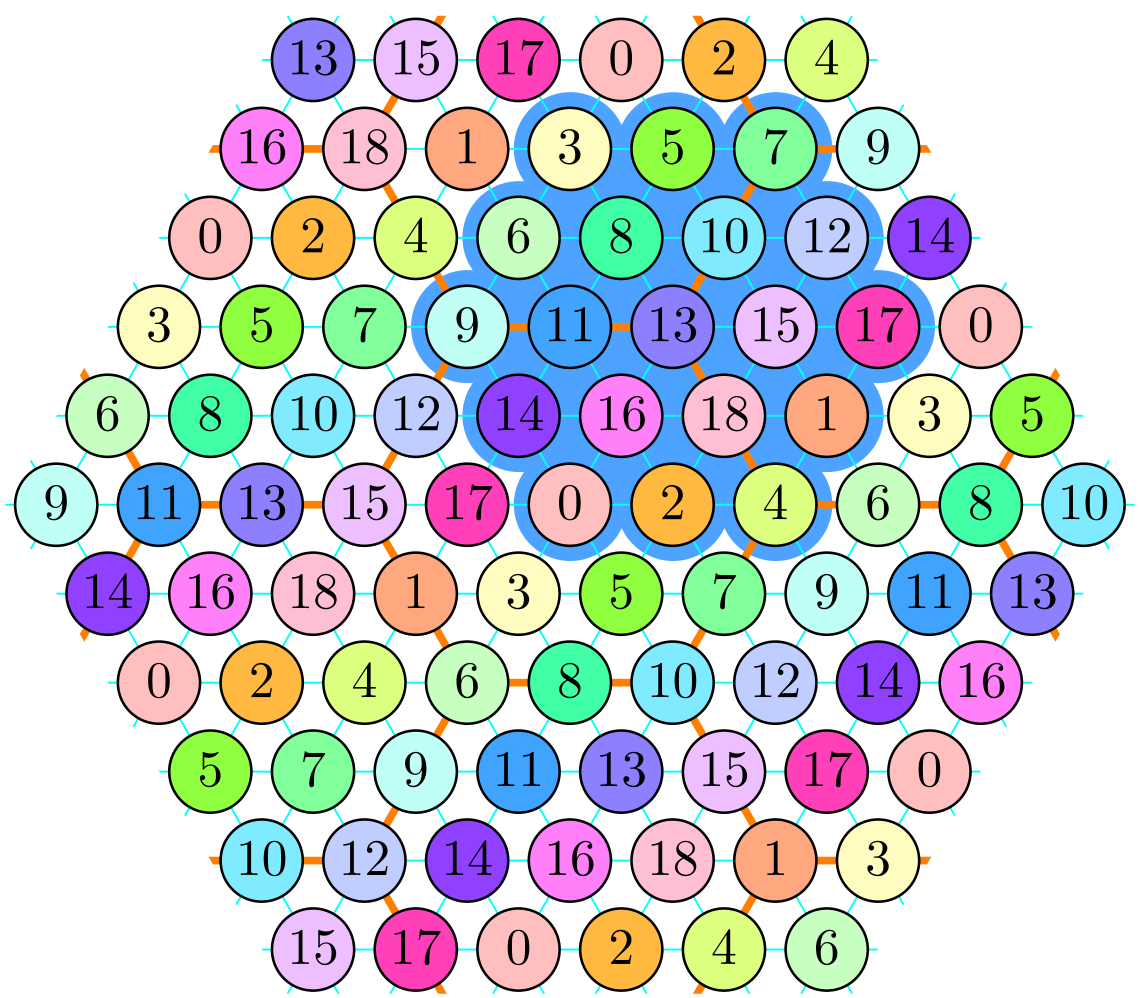
$$c(i,j) = ni + (n+1)j \bmod |H_n|$$

is a proper coloring of H_n

Corolary 1. As it is affine, it is a proper coloring of *any* translation of H_n

Corolary 2. Furthermore, the colors of the neighbors of a given node are fixed translations modulo $|H_n|$ of its own color

Affine coloring of hexagons



Theorem. Let H_n be the hexagon of radius n,

$$c(i,j) = ni + (n+1)j \bmod |H_n|$$

is a proper coloring of H_n

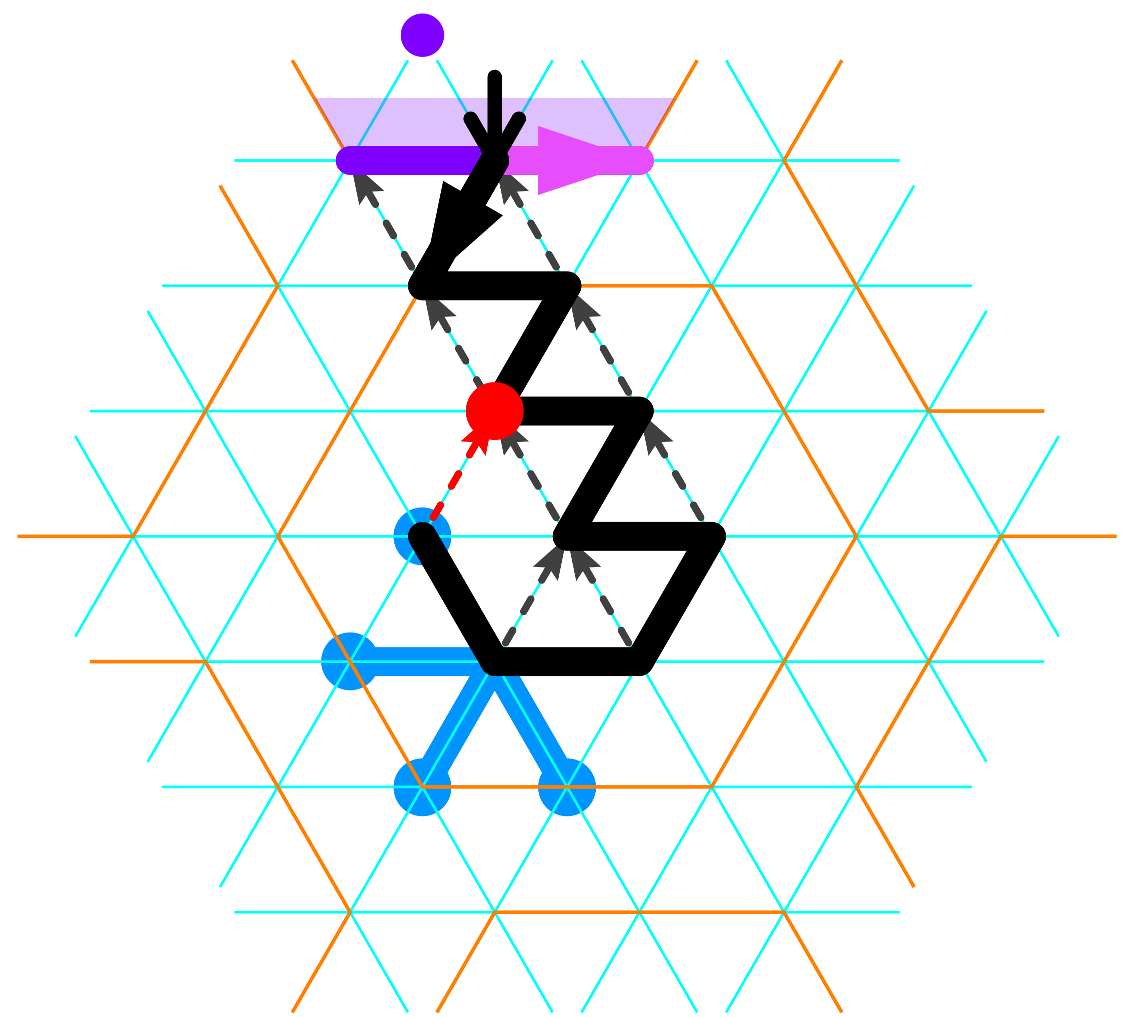
Corolary 1. As it is affine, it is a proper coloring of *any* translation of H_n

Corolary 2. Furthermore, the colors of the neighbors of a given node are fixed translations modulo $|H_n|$ of its own color

Tight oritatami Systems

An oritatami system is *tight* if:

- delay **δ = 1**
- **every bead destination** has a **tight neighbor,** i.e. such that there is only **one available position** next to it

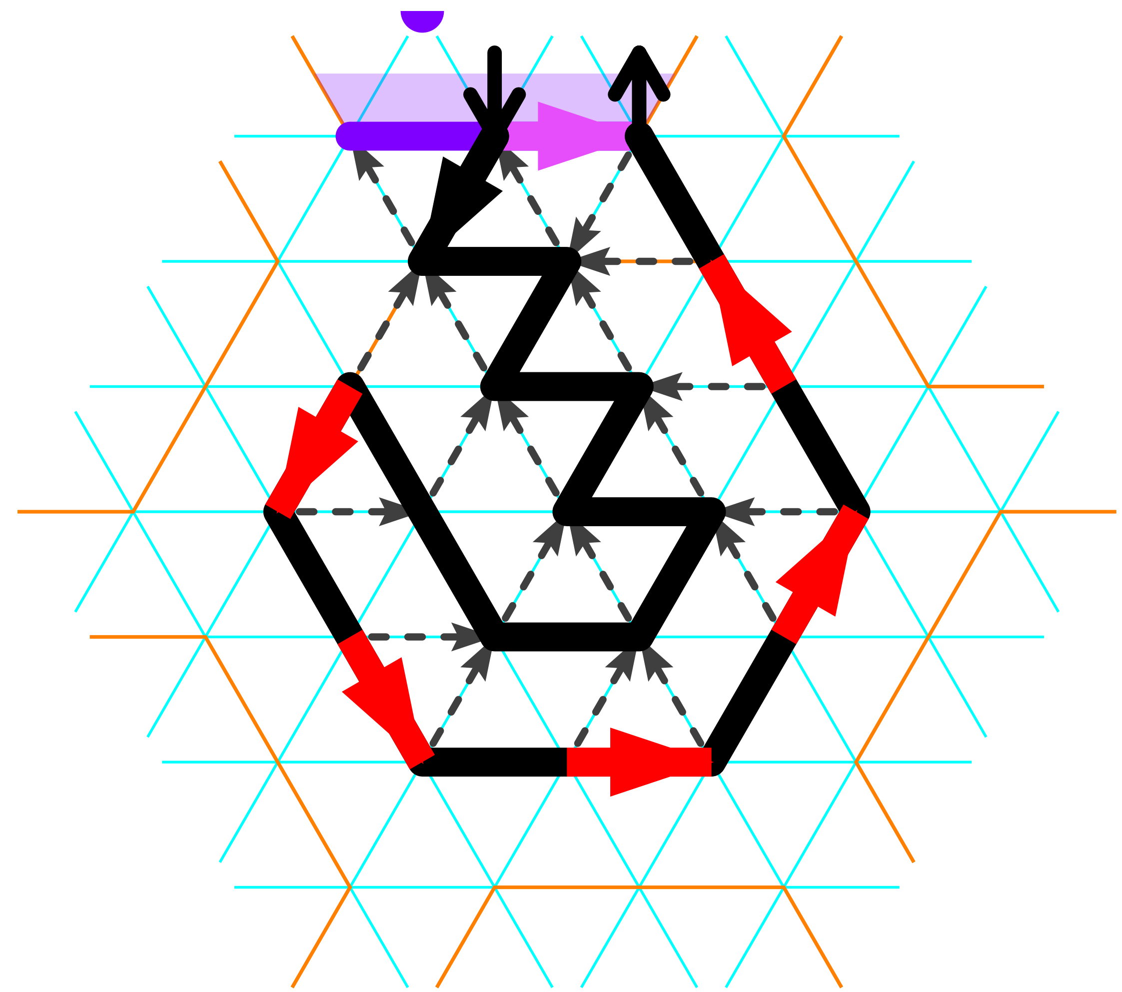


For tight oritatami system, each bead's position is uniquely determined by whom it is attracted to

Tight oritatami Systems

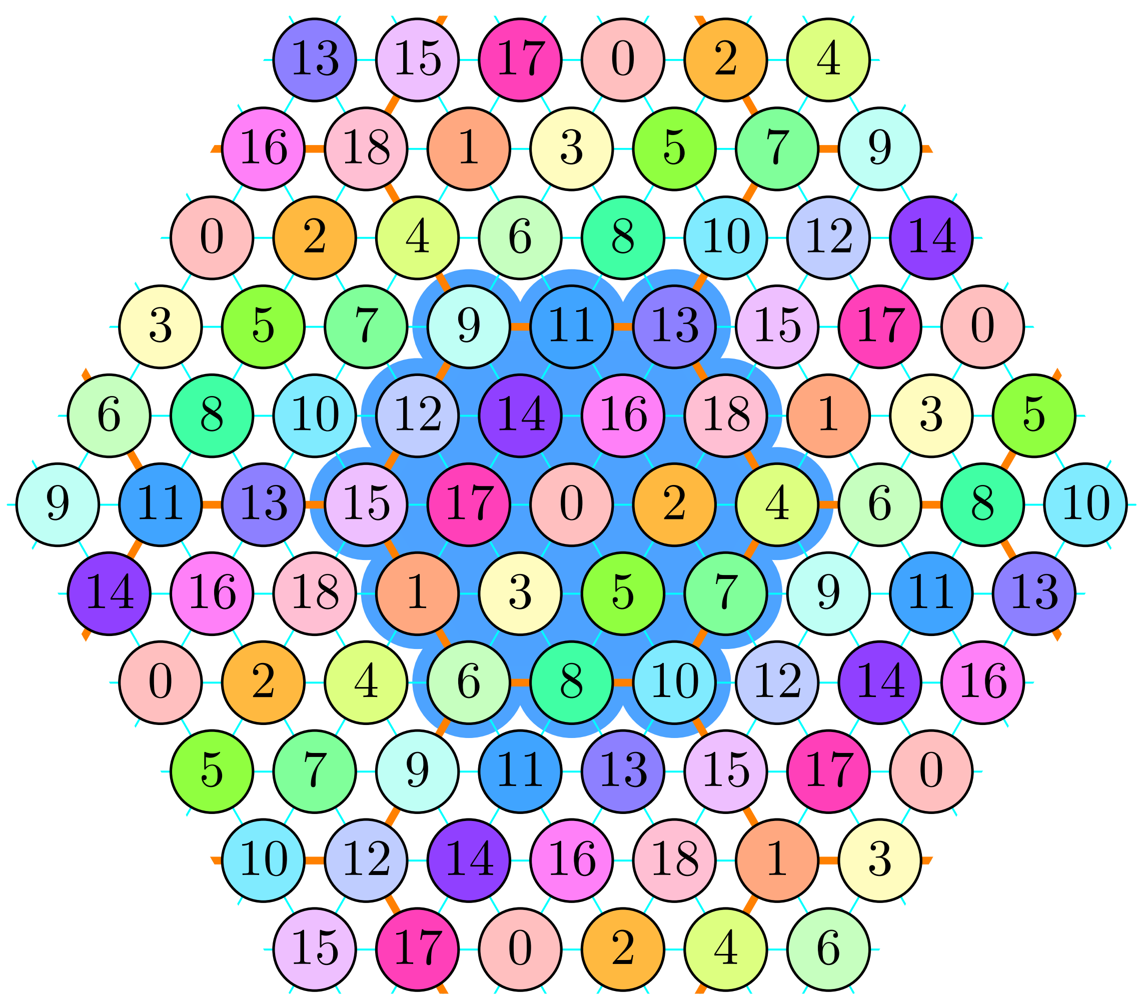
An oritatami system is *tight* if:

- delay **δ = 1**
- **every bead destination** has a **tight neighbor,** i.e. such that there is only **one available position** next to it



For tight oritatami system, each bead's position is uniquely determined by whom it is attracted to

19 bead types are enough for tight oritatami systems



Each bead located at (i,j) receives bead type:

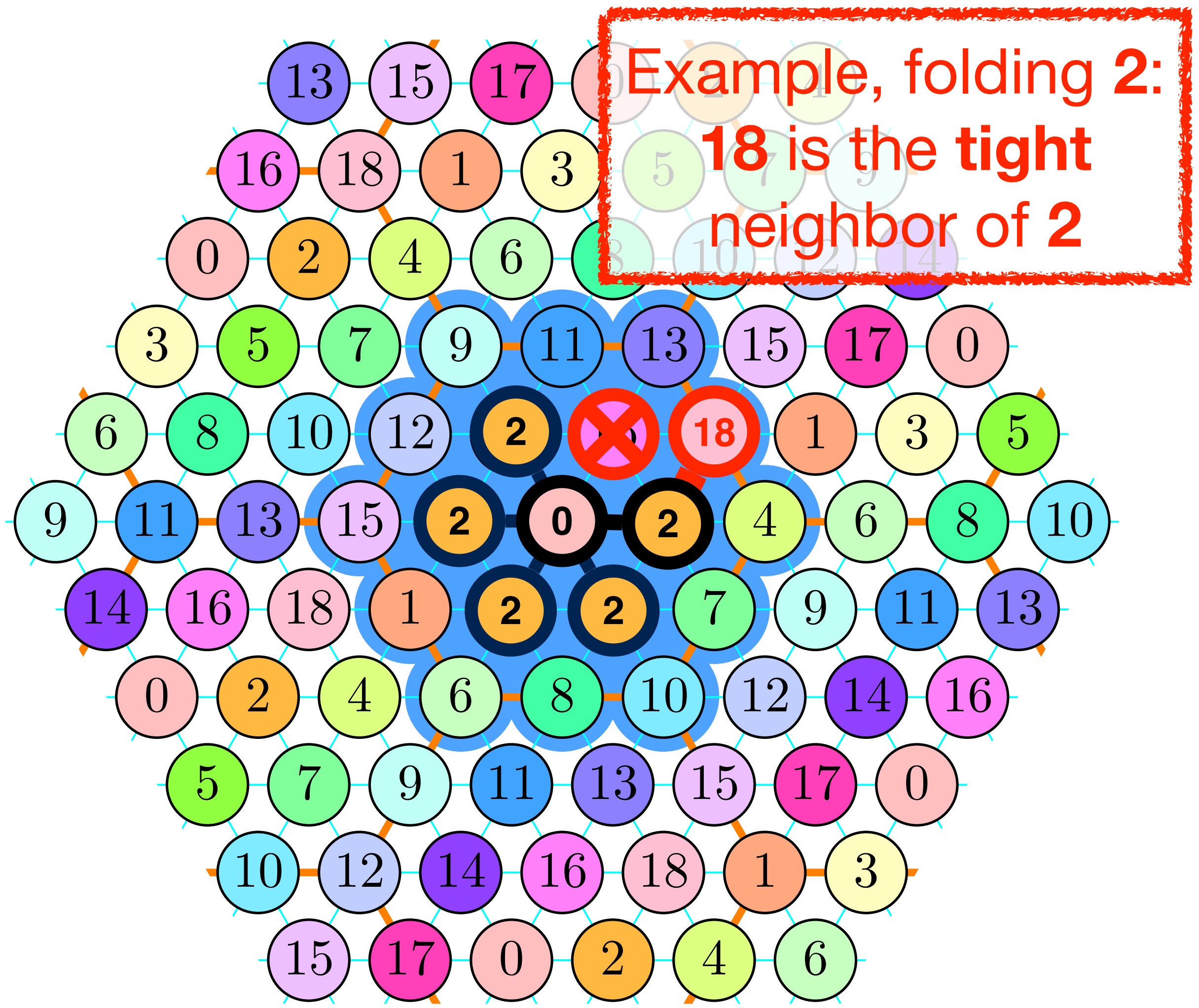
$$\boldsymbol{c}(\boldsymbol{i},\boldsymbol{j})$$

and $\boldsymbol{c}$❤️$\boldsymbol{c}$' iff

$$\boldsymbol{c}' = \boldsymbol{c} + \boldsymbol{\Delta c}(\boldsymbol{d}) \ \mathbf{mod}\ \mathbf{19}$$

For tight oritatami system, each bead's position is fully determined by whom it is attracted to

19 bead types are enough for tight oritatami systems



Each bead located at (i,j) receives bead type:

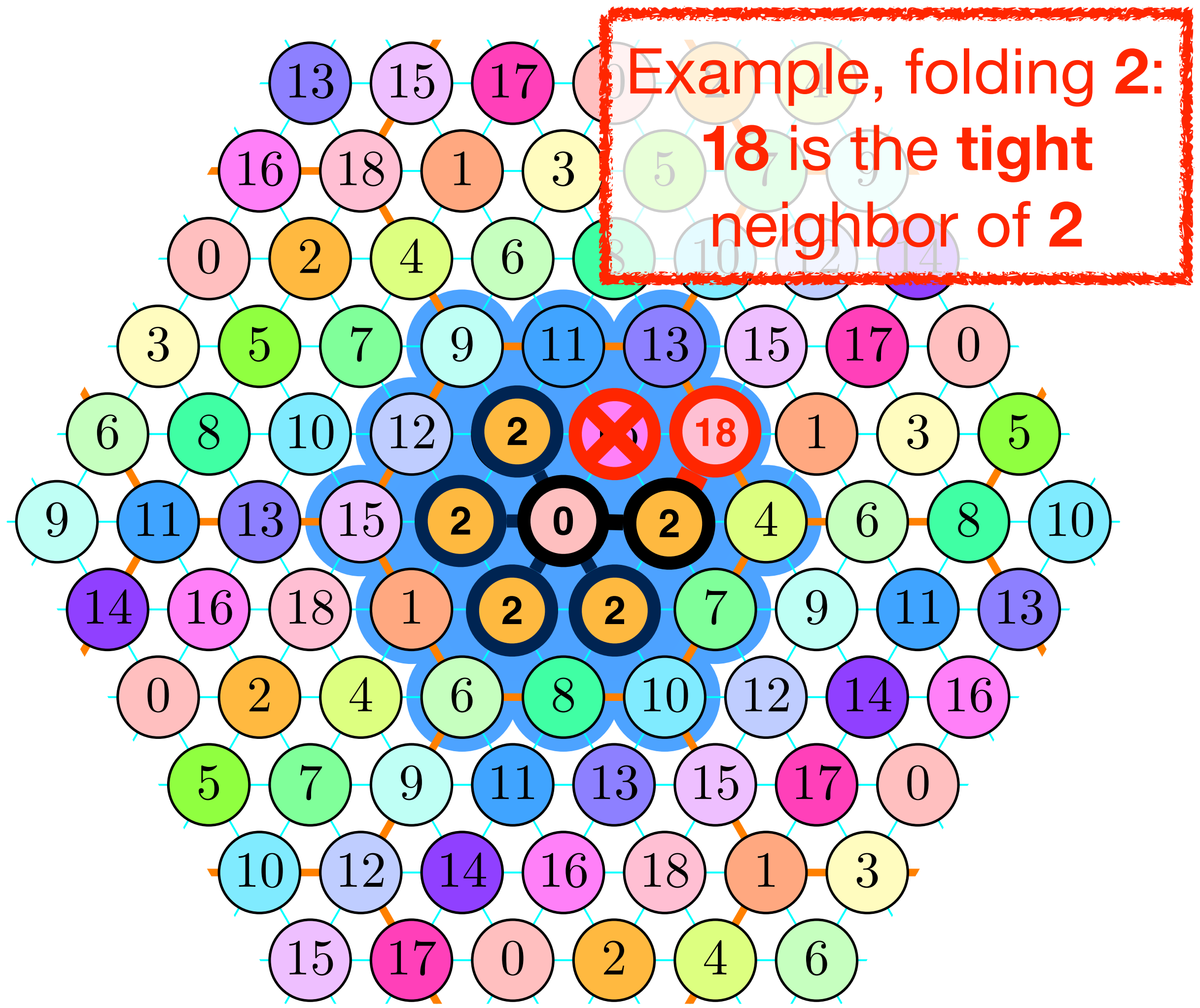
$$c(i,j)$$

and c❤️c' iff

$$c' = c + \Delta c(d) \bmod 19$$

For tight oritatami system, each bead's position is fully determined by whom it is attracted to

19 bead types are enough for tight oritatami systems



Each bead located at (i,j) receives bead type:

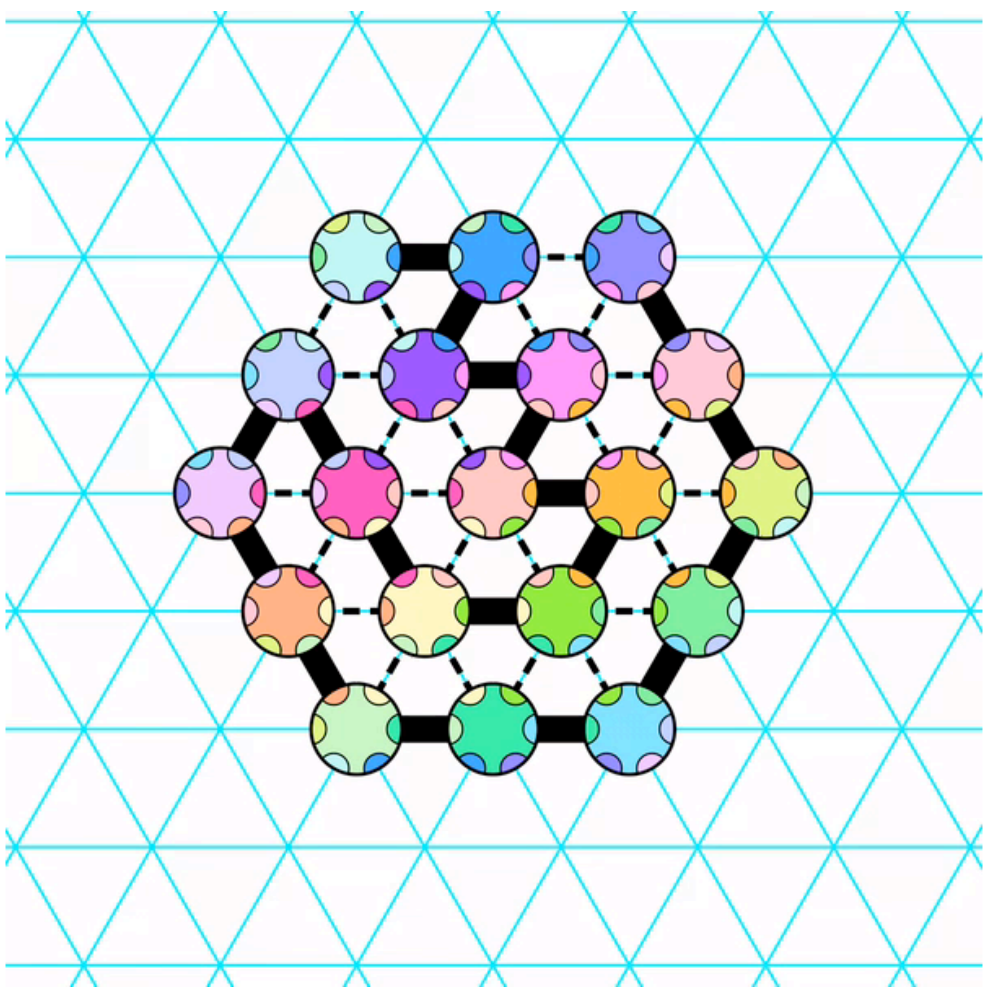
$$c(i,j)$$

and c ❤️ c' iff

$$c' = c + \Delta c(d) \mod 19$$

Theorem. 19 beads types are enough

19 bead types are enough for tight oritatami systems

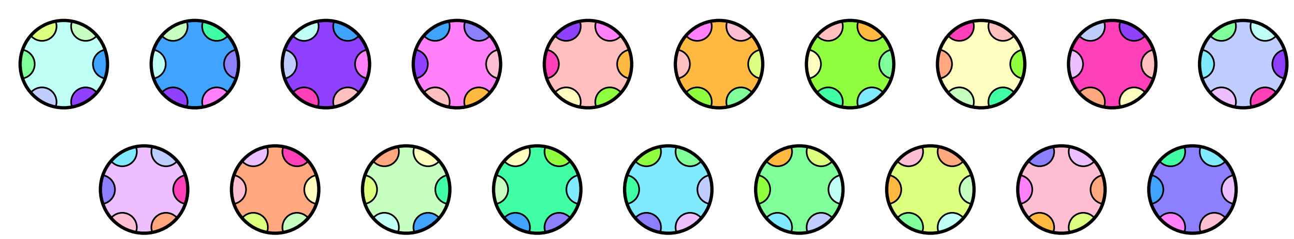


Each bead located at (i,j) receives bead type:

$$c(i,j)$$

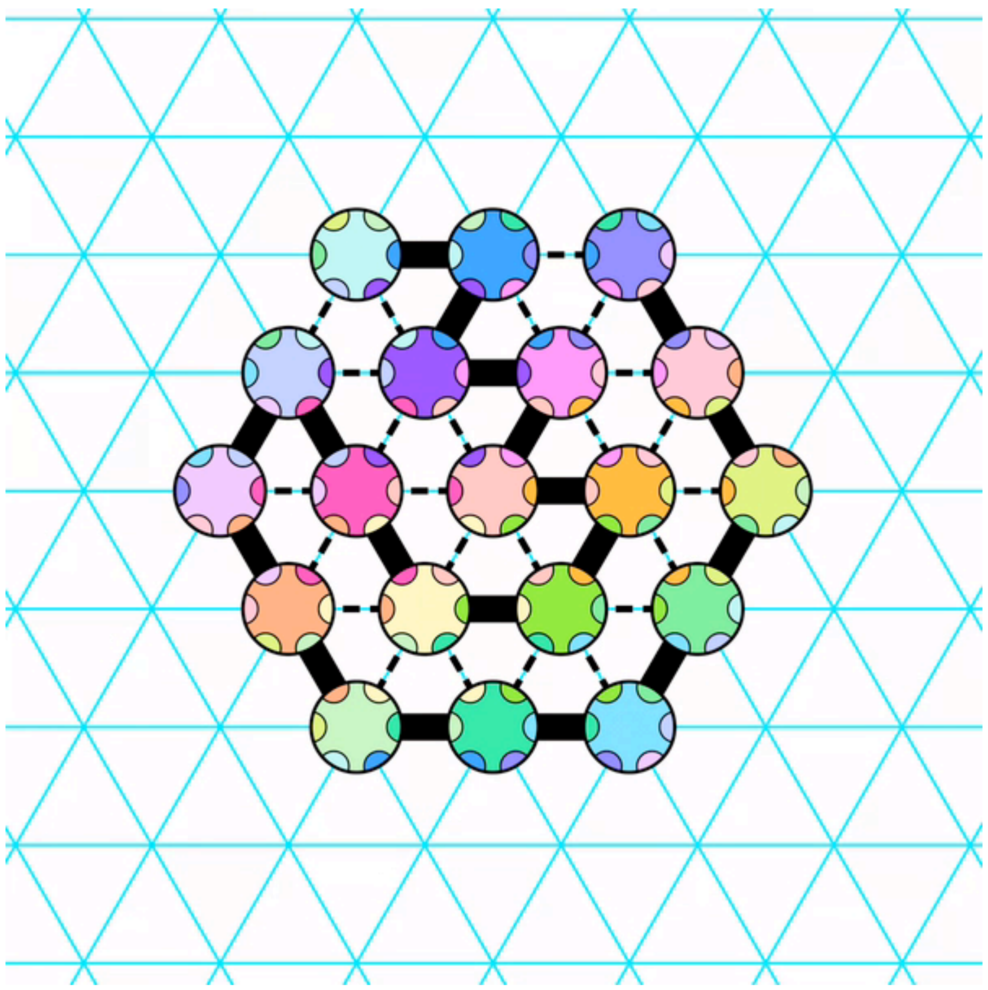
and c❤c' iff

$$c' = c + \Delta c(d) \bmod 19$$



Theorem. 19 beads types are enough

19 bead types are enough for tight oritatami systems

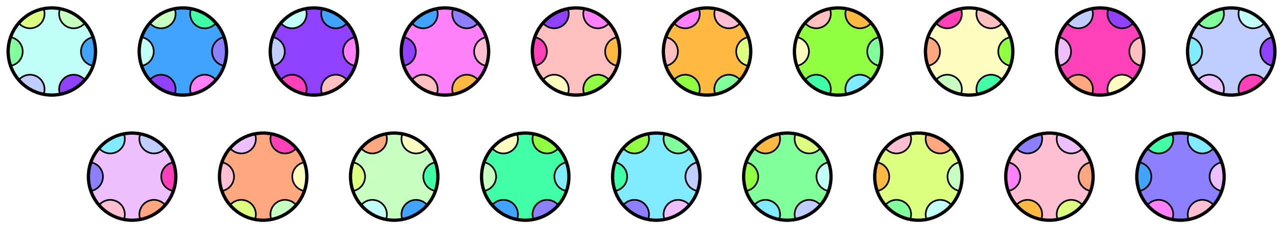


Each bead located at (i,j) receives bead type:

$$\boldsymbol{c(i,j)}$$

and $\boldsymbol{c}$❤️$\boldsymbol{c'}$ iff

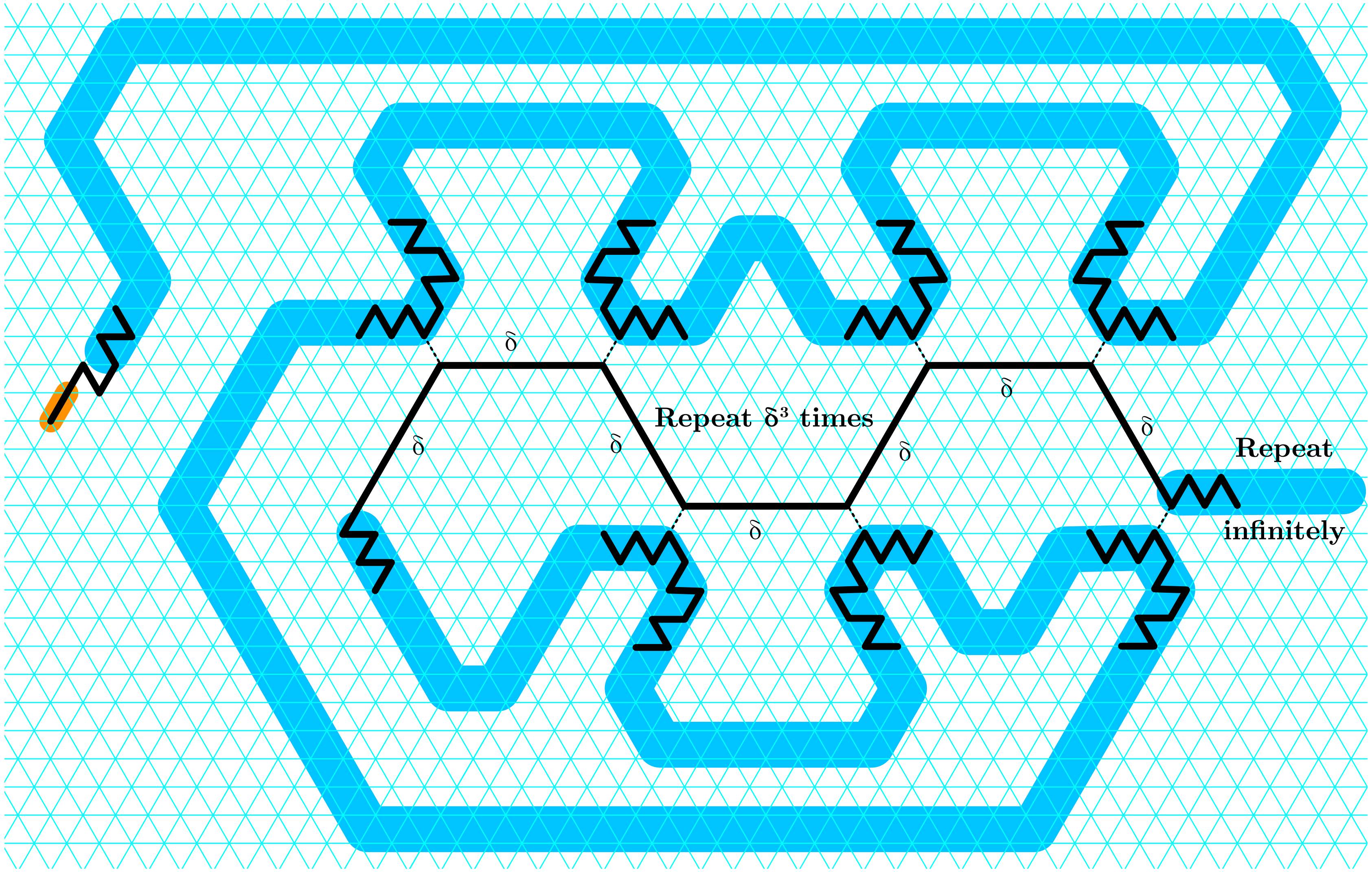
$$\boldsymbol{c' = c + \Delta c(d) \bmod 19}$$



Theorem. There is a **constant-time incremental** algorithm that outputs a tight oritatami system using **19 bead types** that folds any finite shape at scale $\mathscr{B}_{n\,\geq\,3}$ from a **seed of size 3**

Would increasing the delay instead of upscaling help?

Theorem. For any delay δ, there is an infinite shape that cannot be folded by no oritatami system with delay δ



Turedo: building nanobots with oritatami

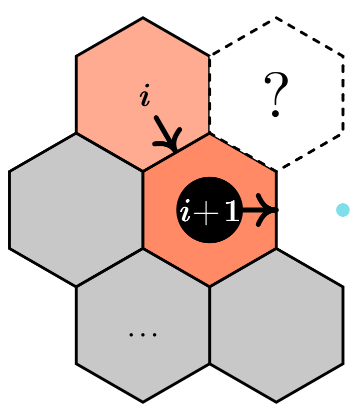
[Pchelina, S., Seki and Theyssier, *STACS* 2022]

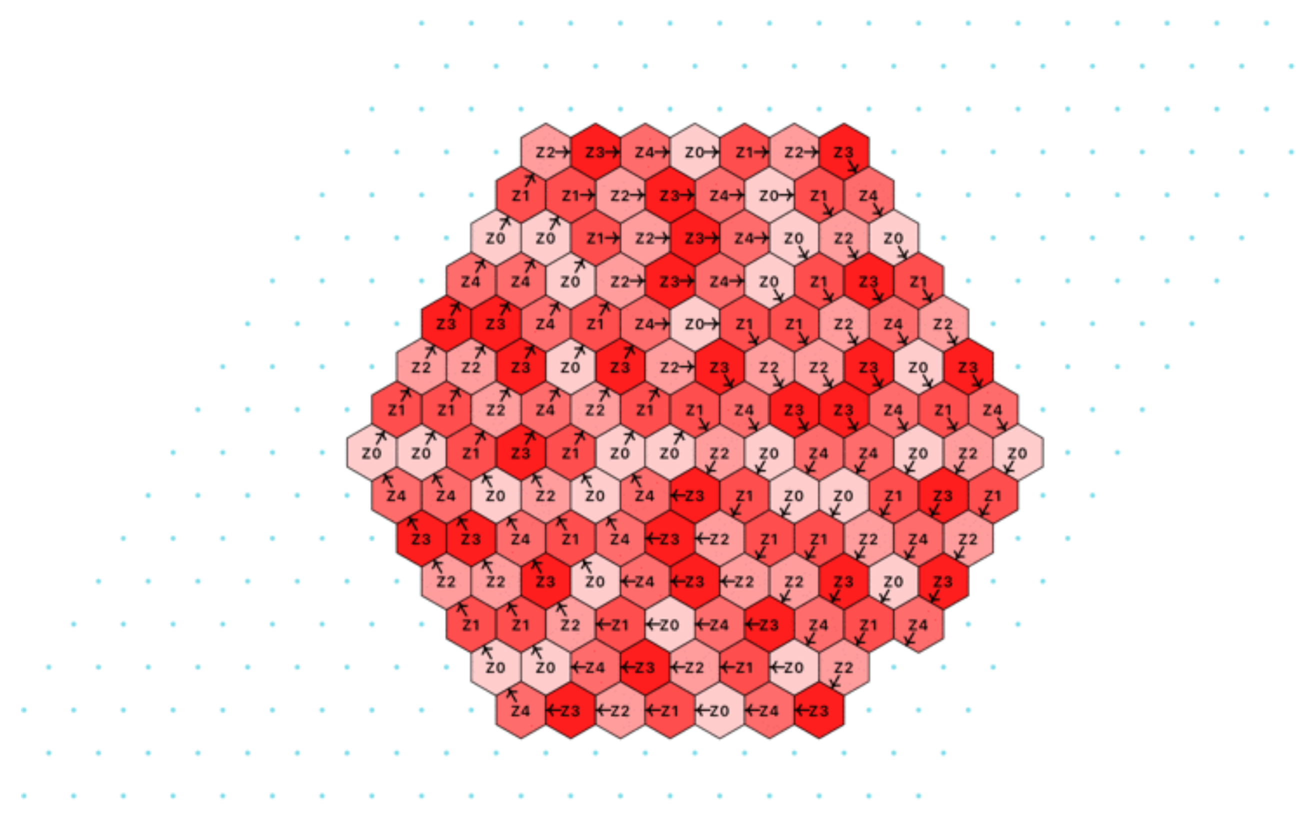
Turedos

A **finite automata** follows a **self-avoiding path, moving** and **writing** a state according to a **uniform local rule**

A clockwise walker

The rule:



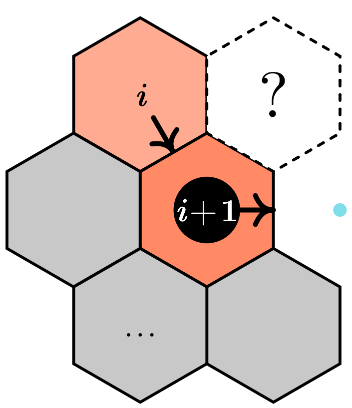


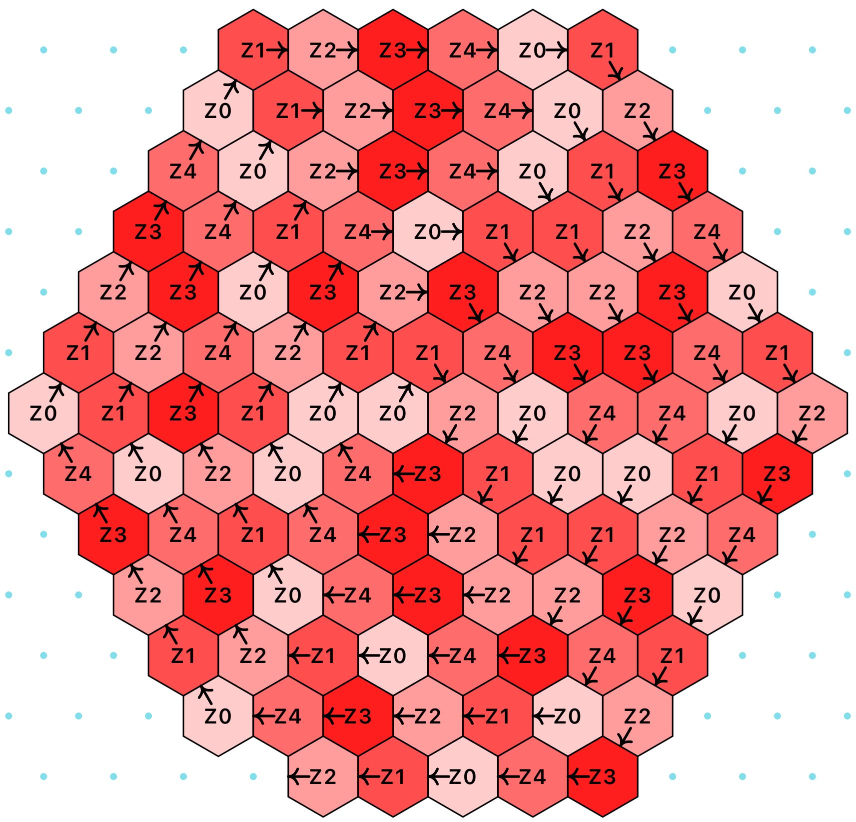
Turedo

A **finite automata** follows a **self-avoiding path, moving** and **writing** a state according to a **uniform local rule**

A clockwise walker

The rule:

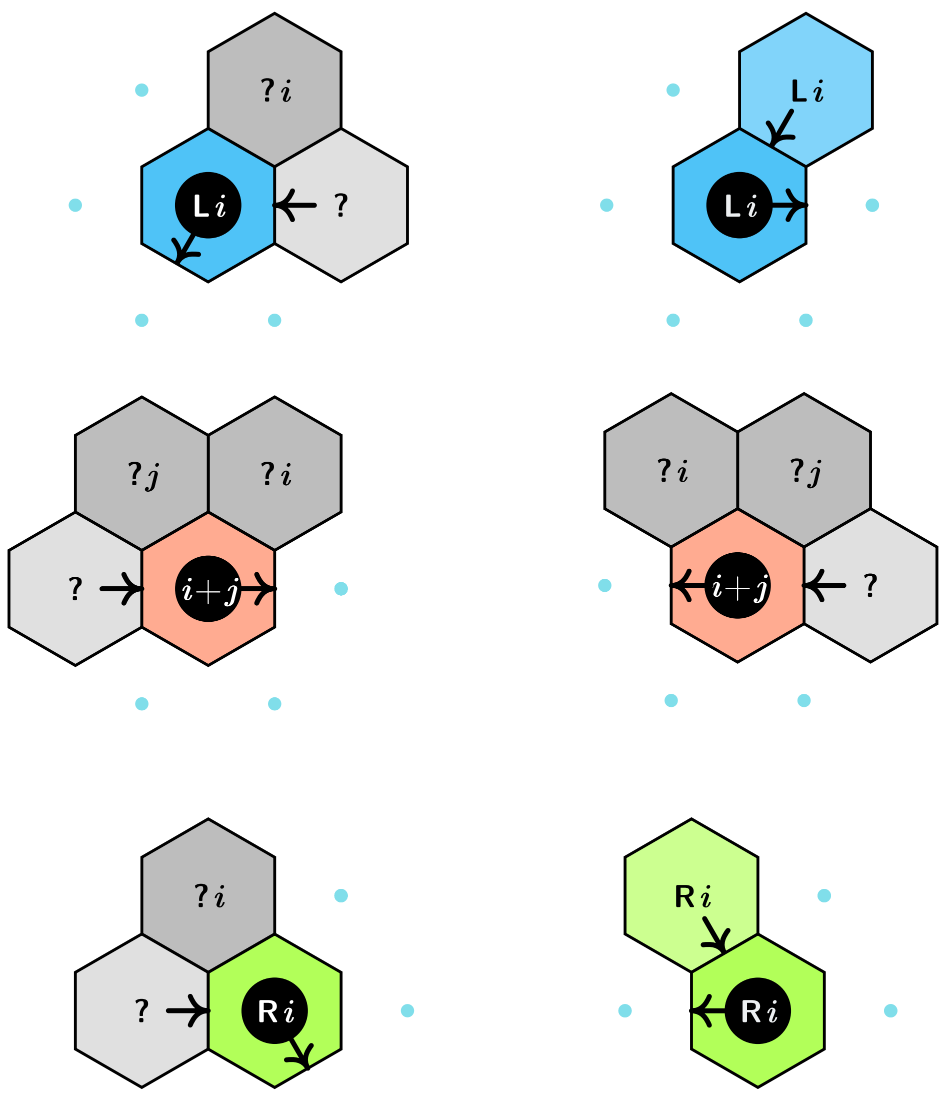


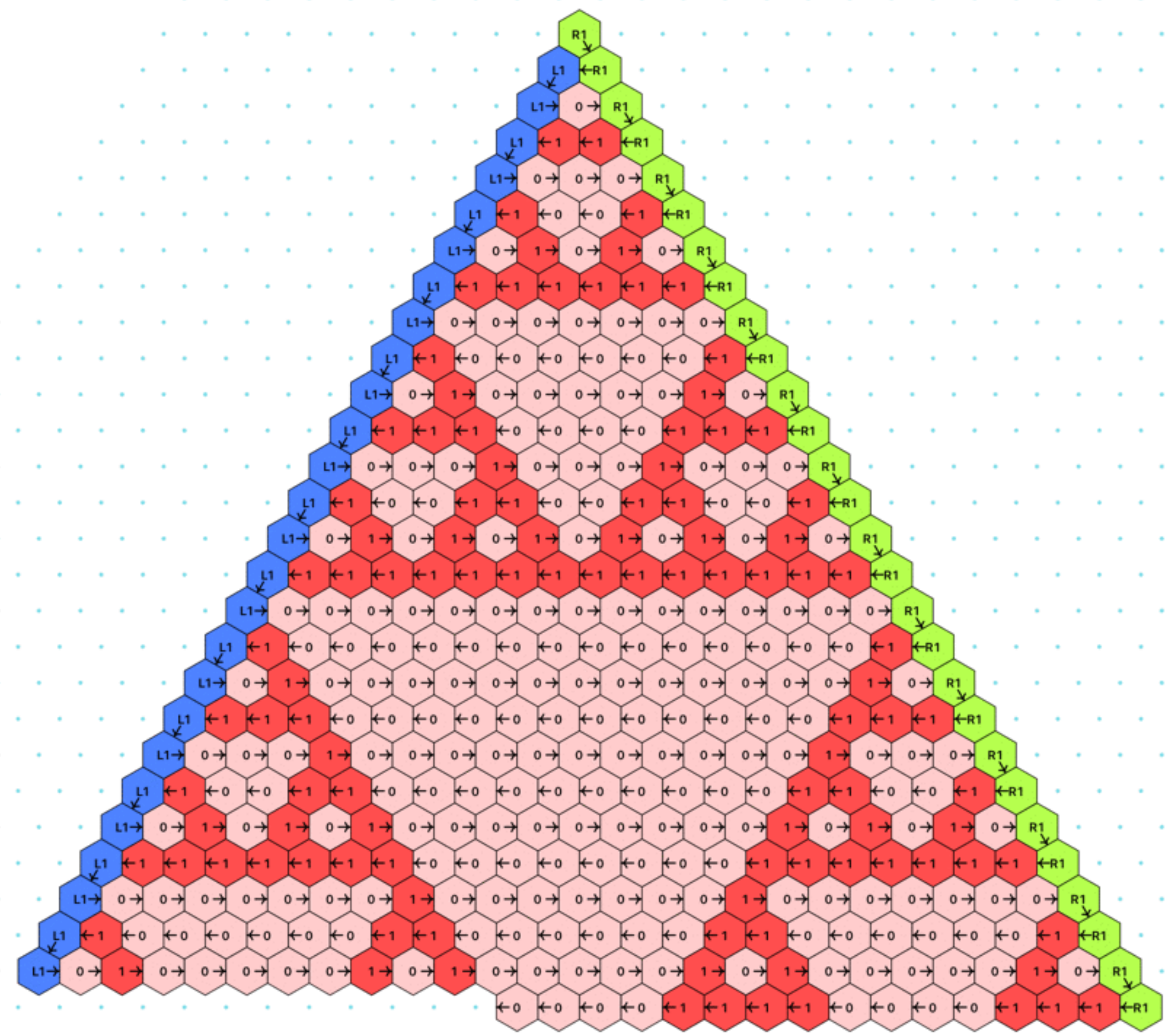


Radius-1 Turedos implement cellular automata

Left/Right Swiping

The rule:

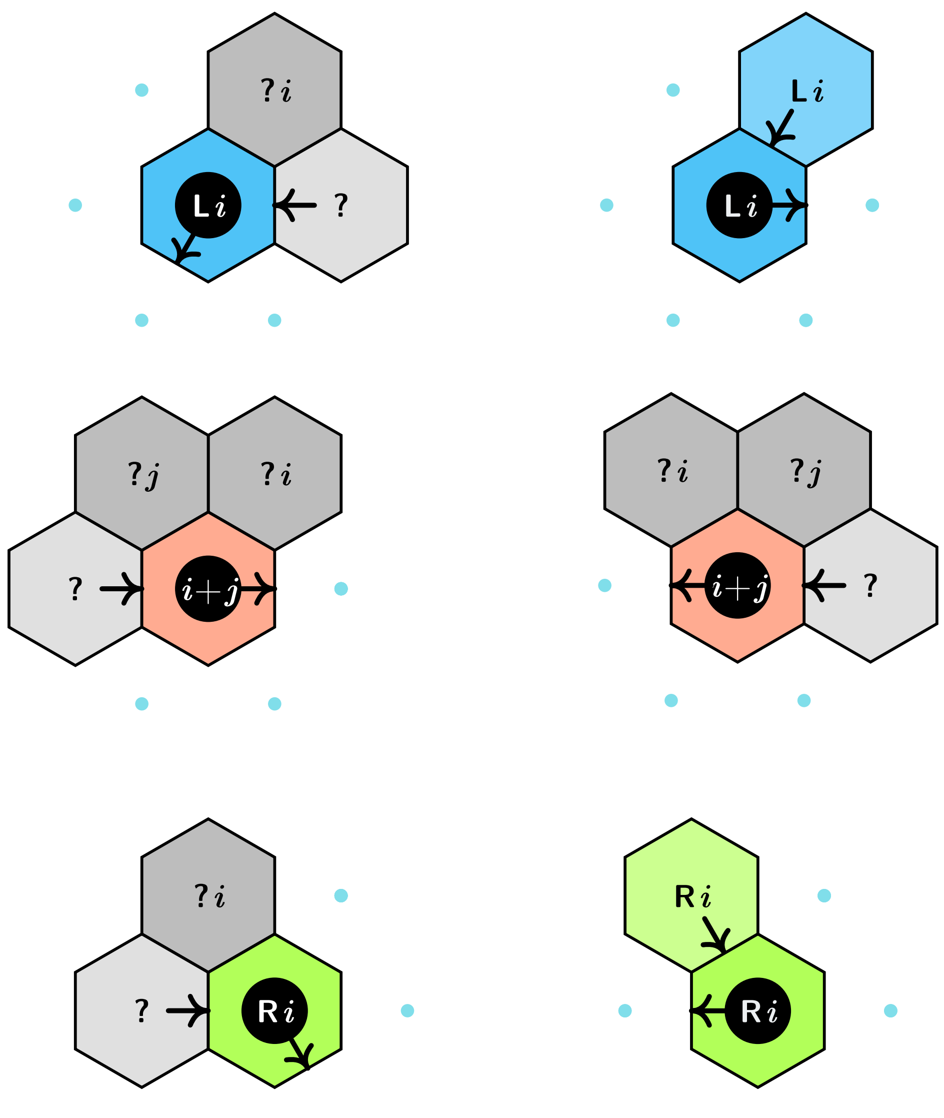


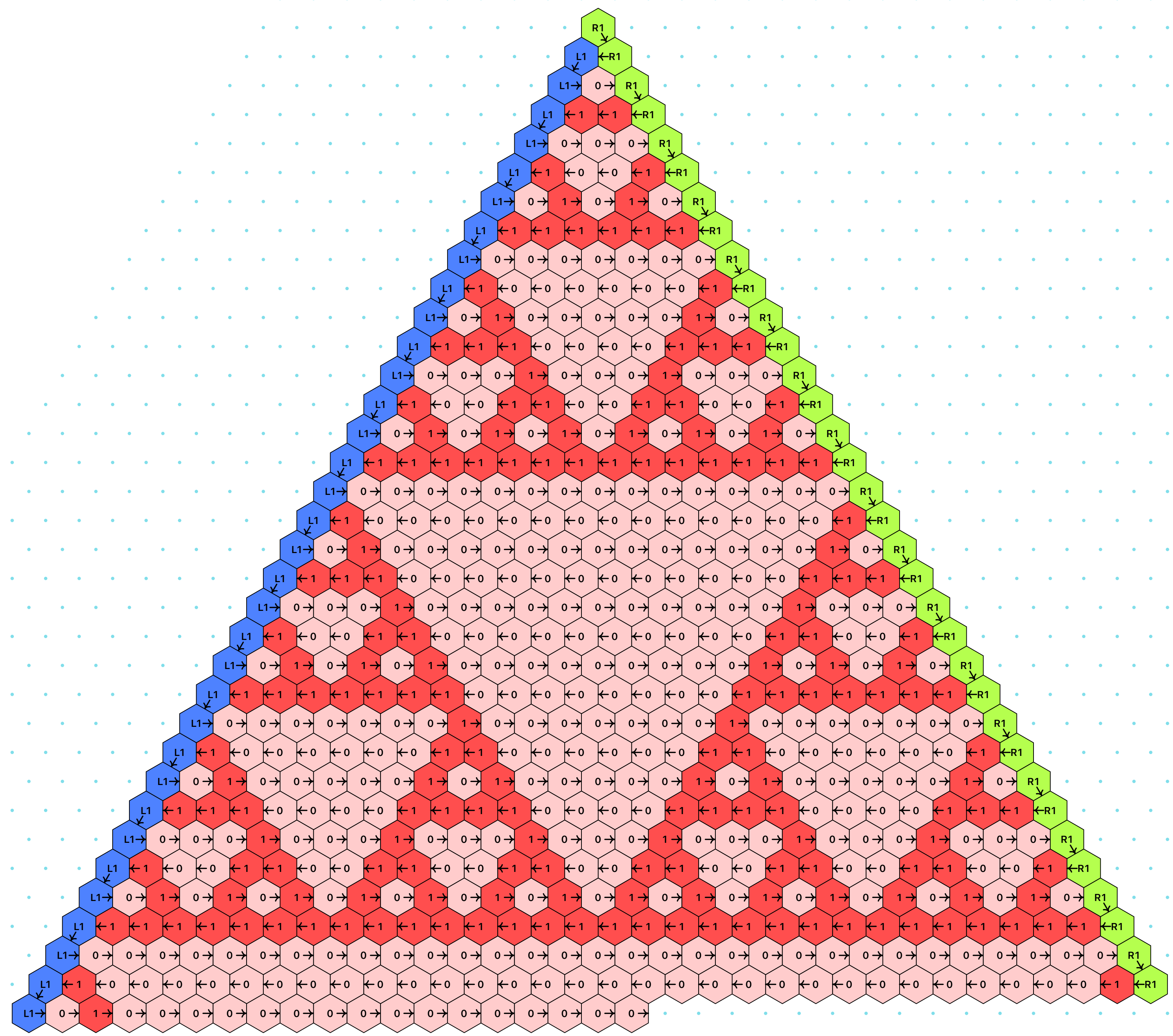


Radius-1 Turedos implement cellular automata

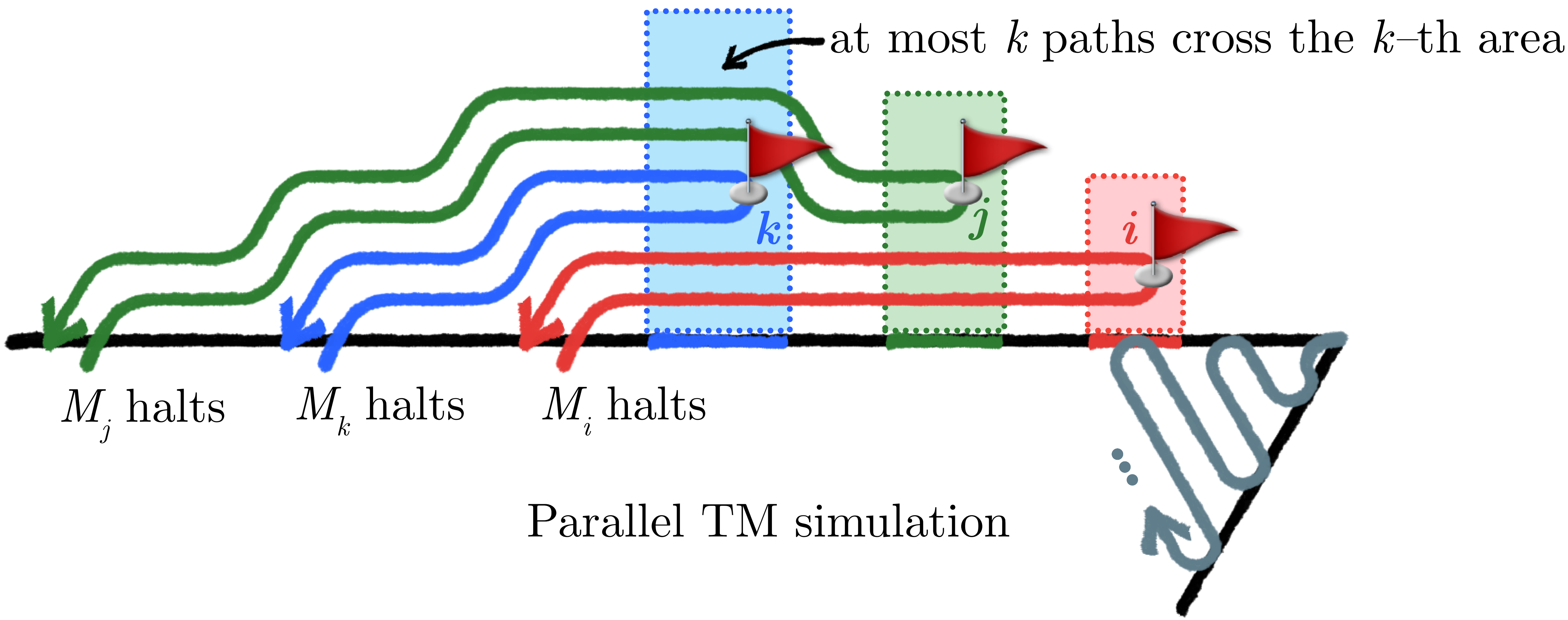
Left/Right Swiping

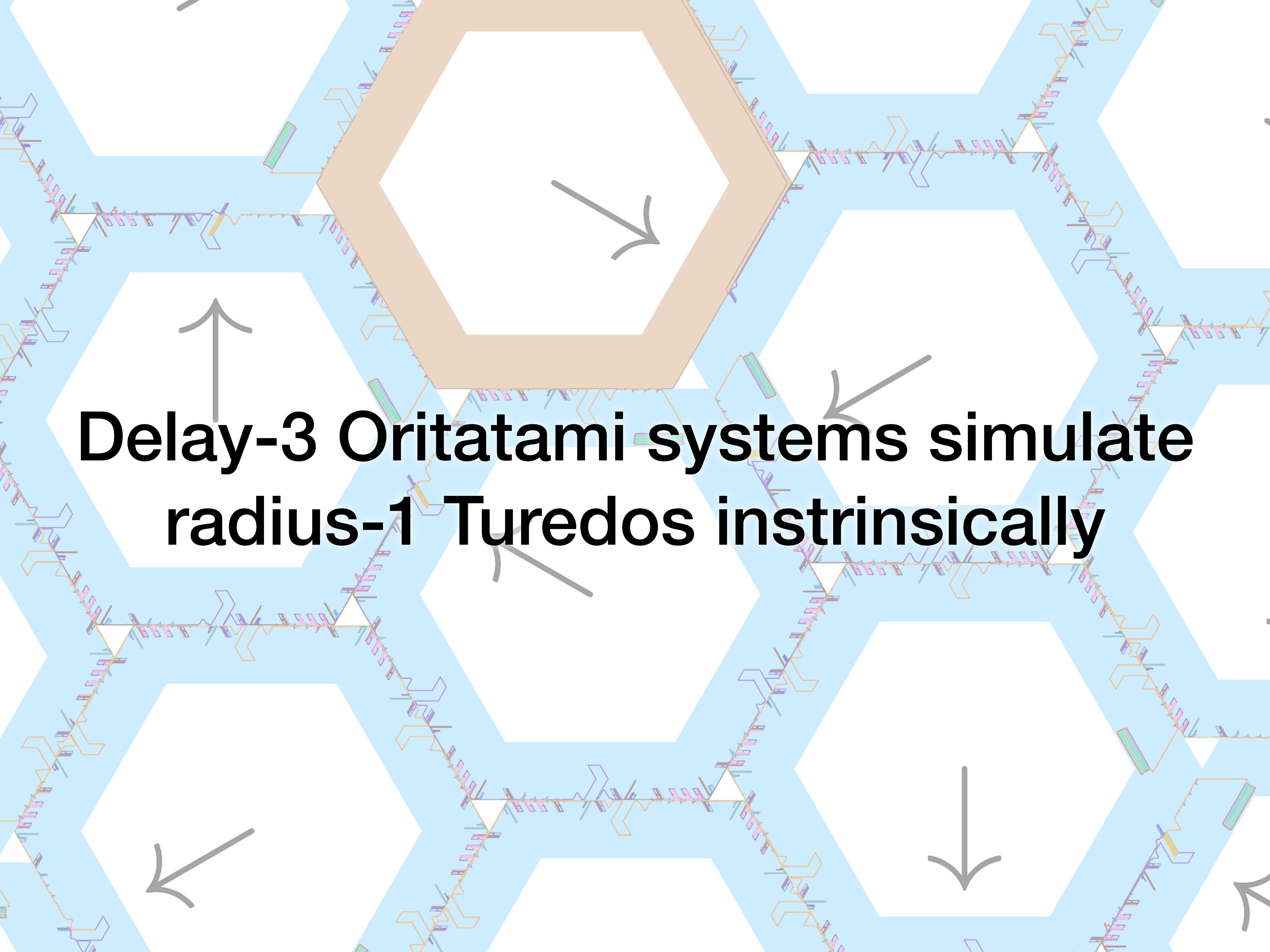
The rule:





Theorem 1.
Radius-1 turedos doodle uncomputably

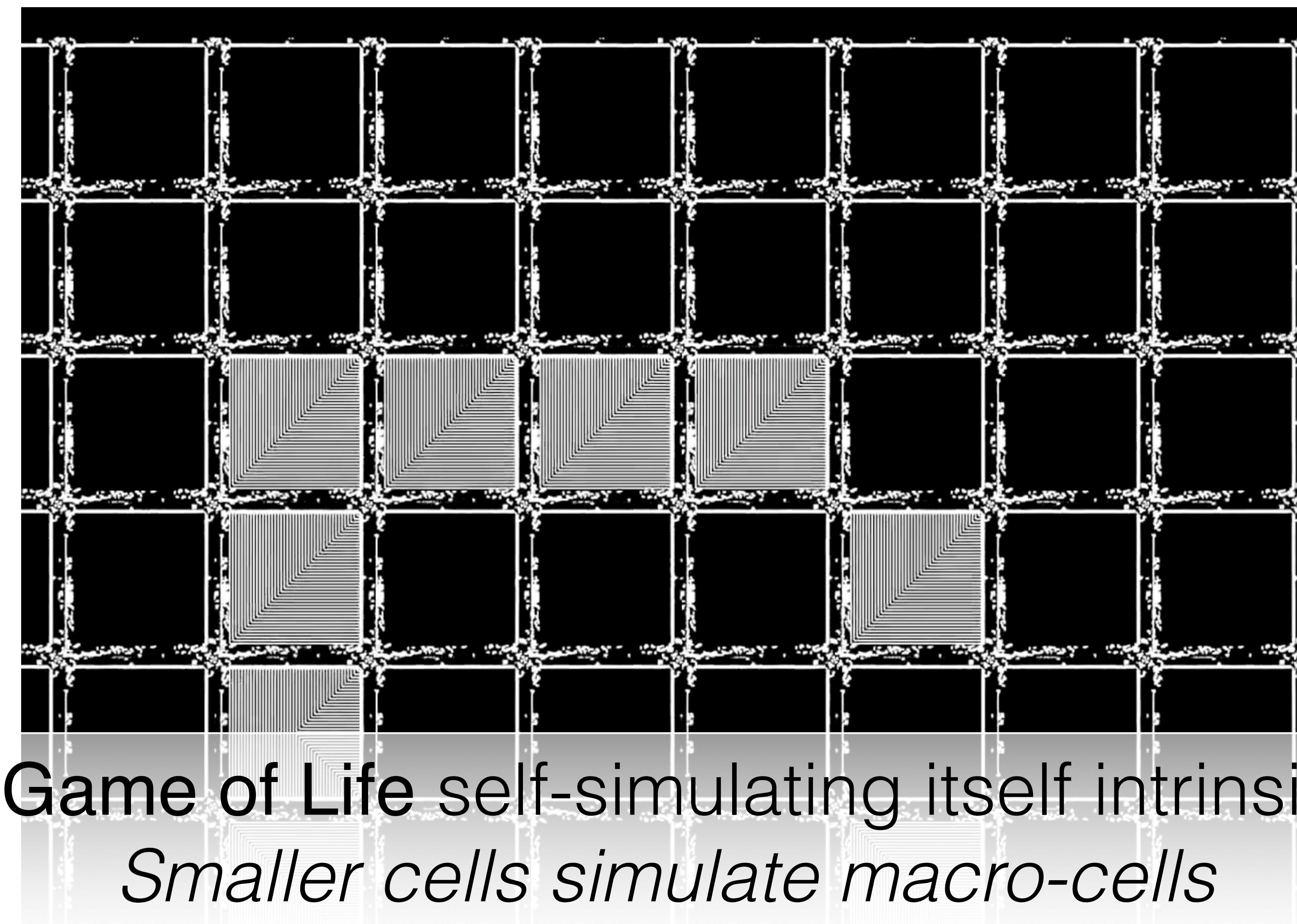




Delay-3 Oritatami systems simulate radius-1 Turedos instrinsically

Intrinsic simulation

- Linear time & scape rescaling



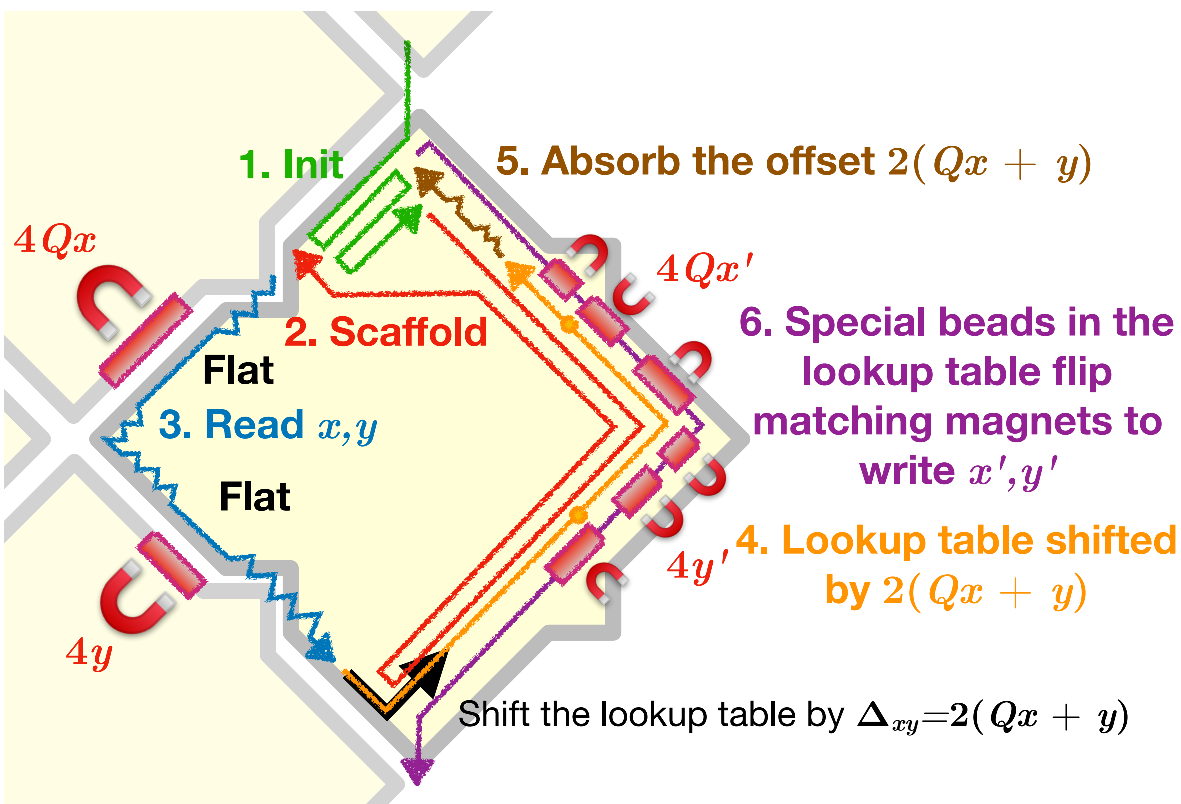
Brice Due 2006

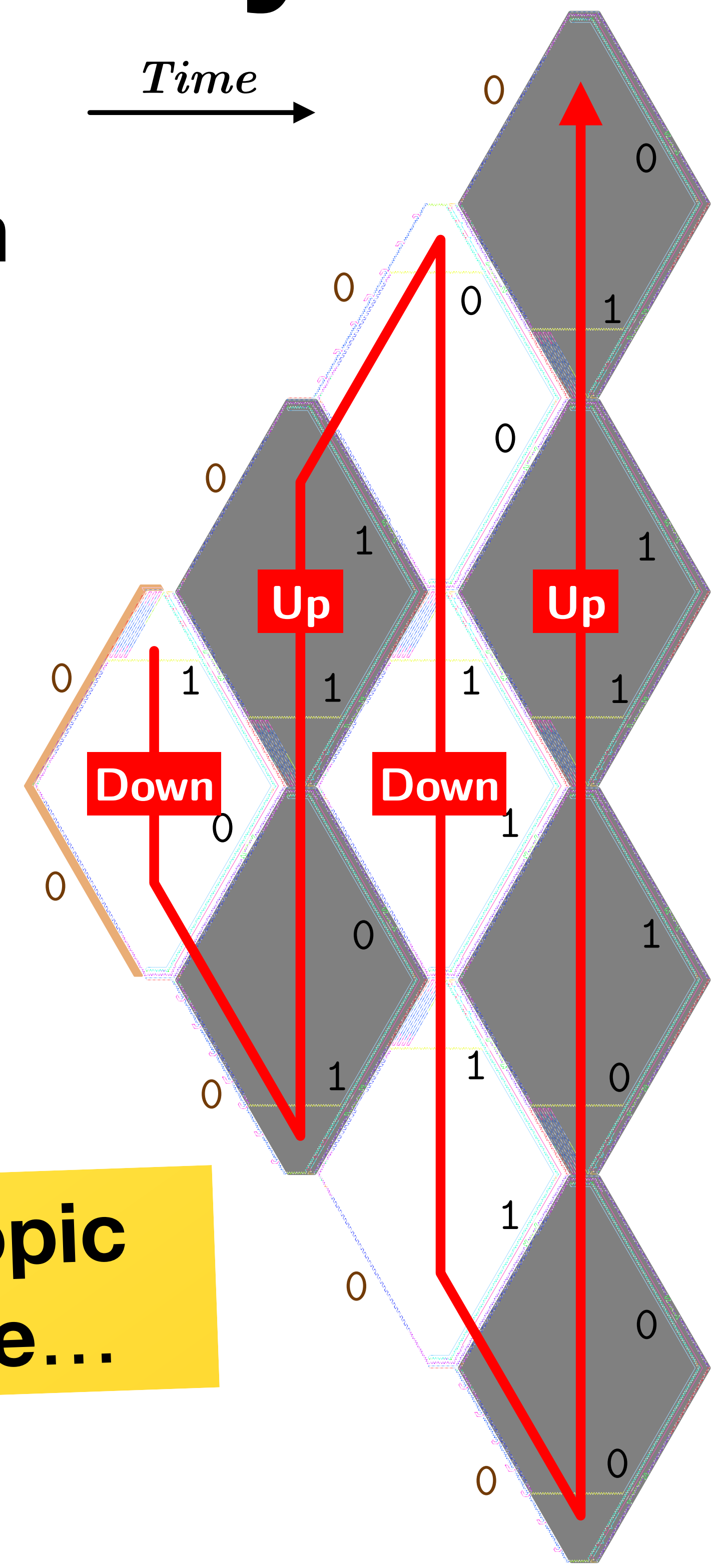
The Game of Life self-simulating itself intrinsically:

Smaller cells simulate macro-cells

Oritatami systems simulate 1D CA instrinsically

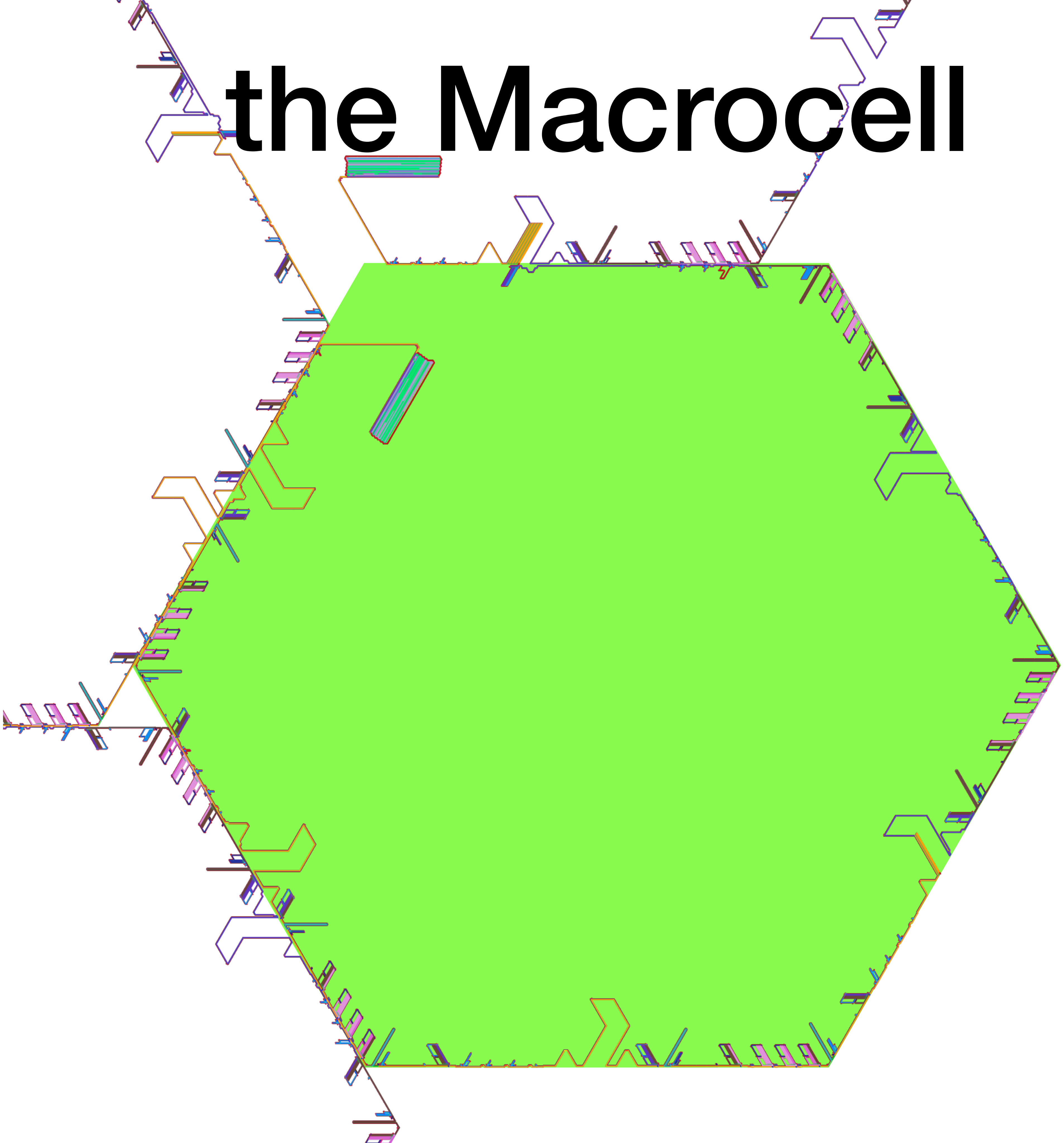
- **Previous work.** [PSSU, 2020]
  1D Cellular automata intrinsic simulation



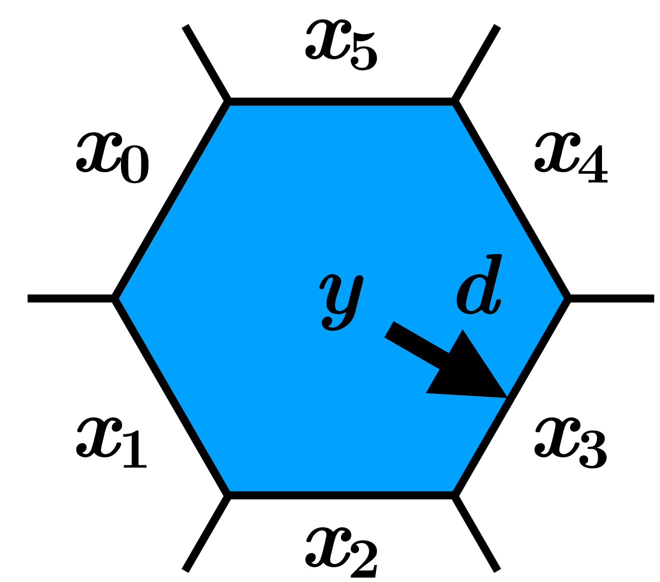


2 Problems. Macrocells must be **isotropic**
We need to **exit from an arbitrary side…**

the Macrocell



Bit-weighted encoding for Turedos

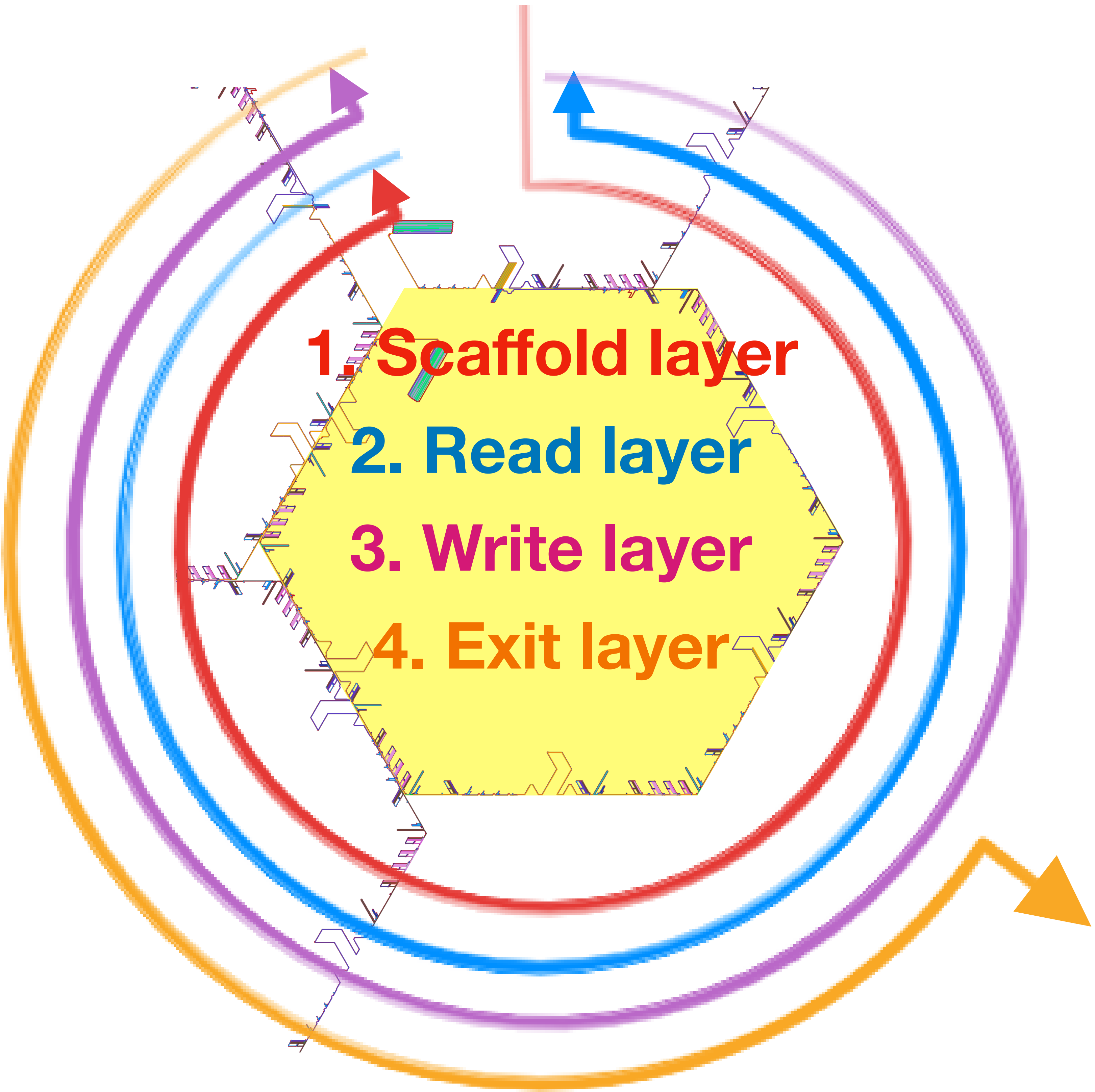


- **Turedo**
  $Q = 2^q - 1$ states and the empty state $\perp$
  Transition function:
  $$F : (Q \cup \{\perp\})^6 \to Q \times \{\leftarrow, \nwarrow, \nearrow, \rightarrow, \searrow, \swarrow\}$$
  $$(x_0, \ldots, x_5) \longmapsto (y, d)$$

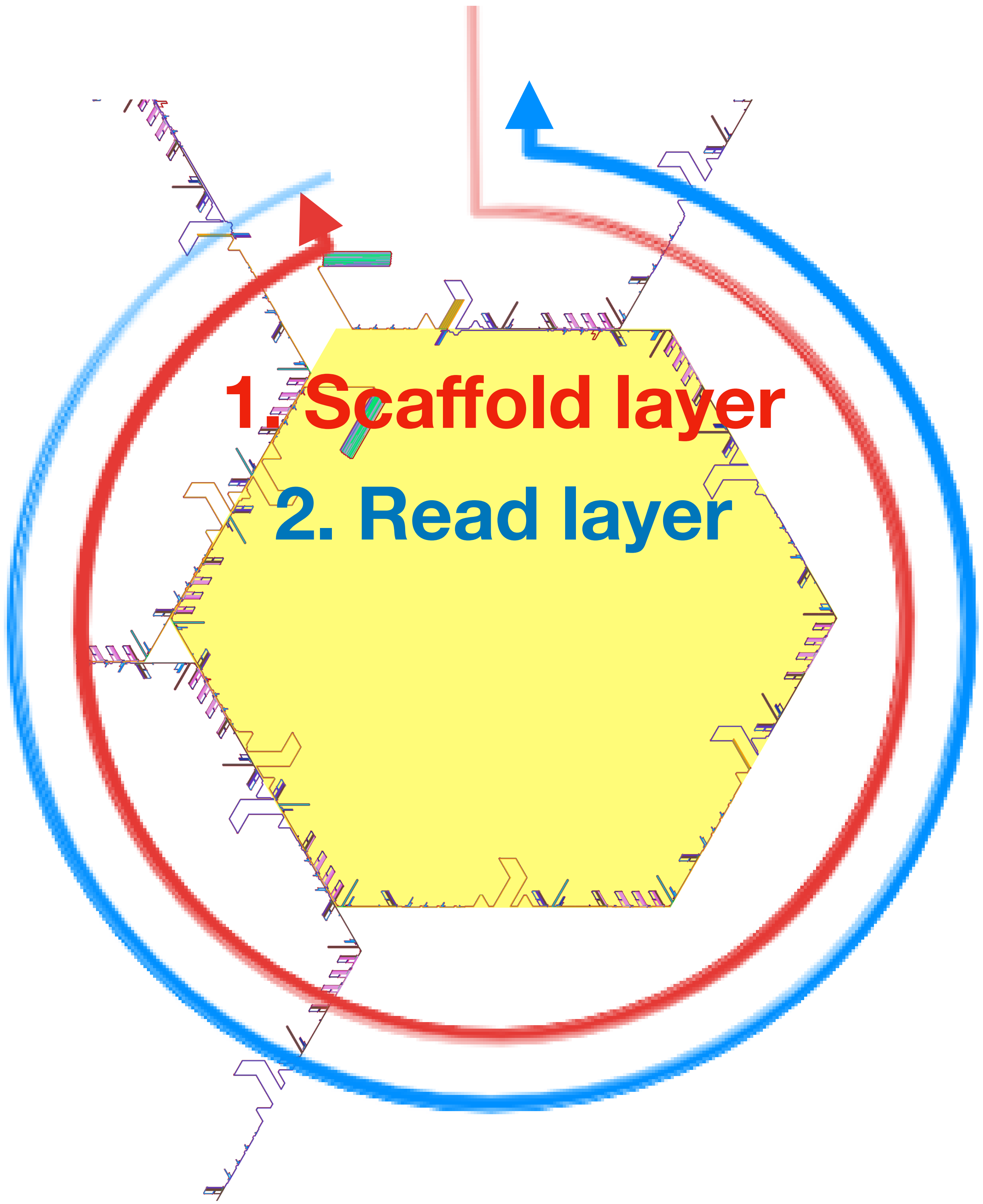
- **Bit-weight encoding**
  w_{ij} **= weight of the** j**-th bit** x_{ij} **of** x_i
  $$F(x) = \Phi(w(x)) \text{ where } w(x) = \sum_{ij} w_{ij}\, x_{ij}$$

- $w_{ij} = 2^{qi+j}$ works for all F

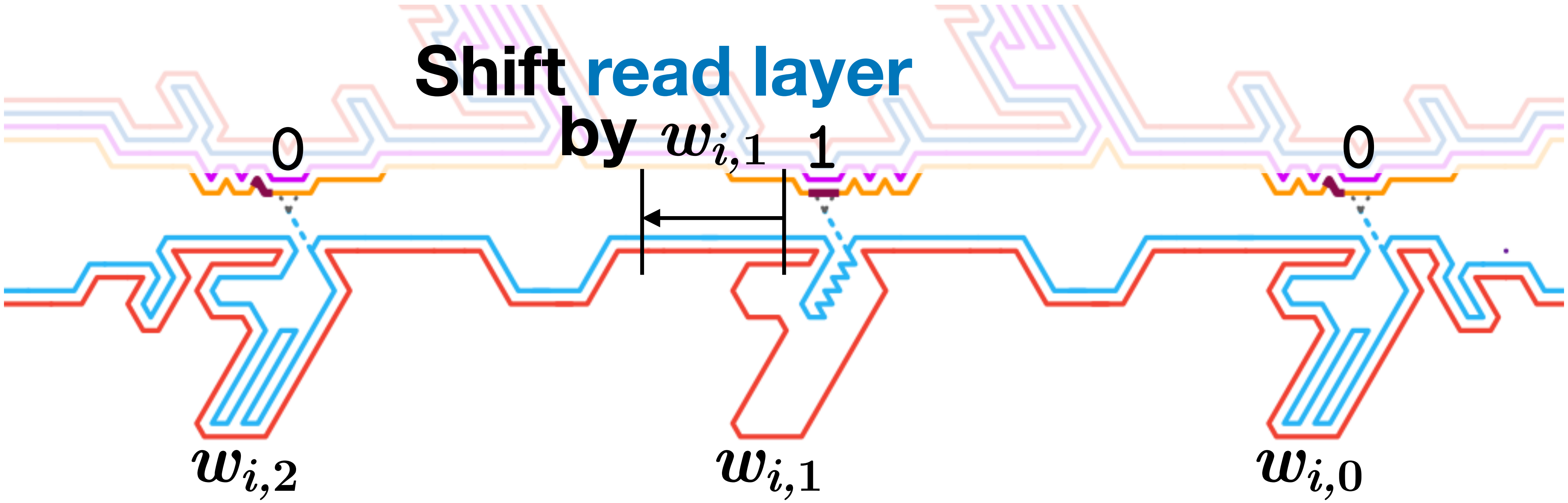
the Macrocell



the Read layer



Reading. Reading pockets

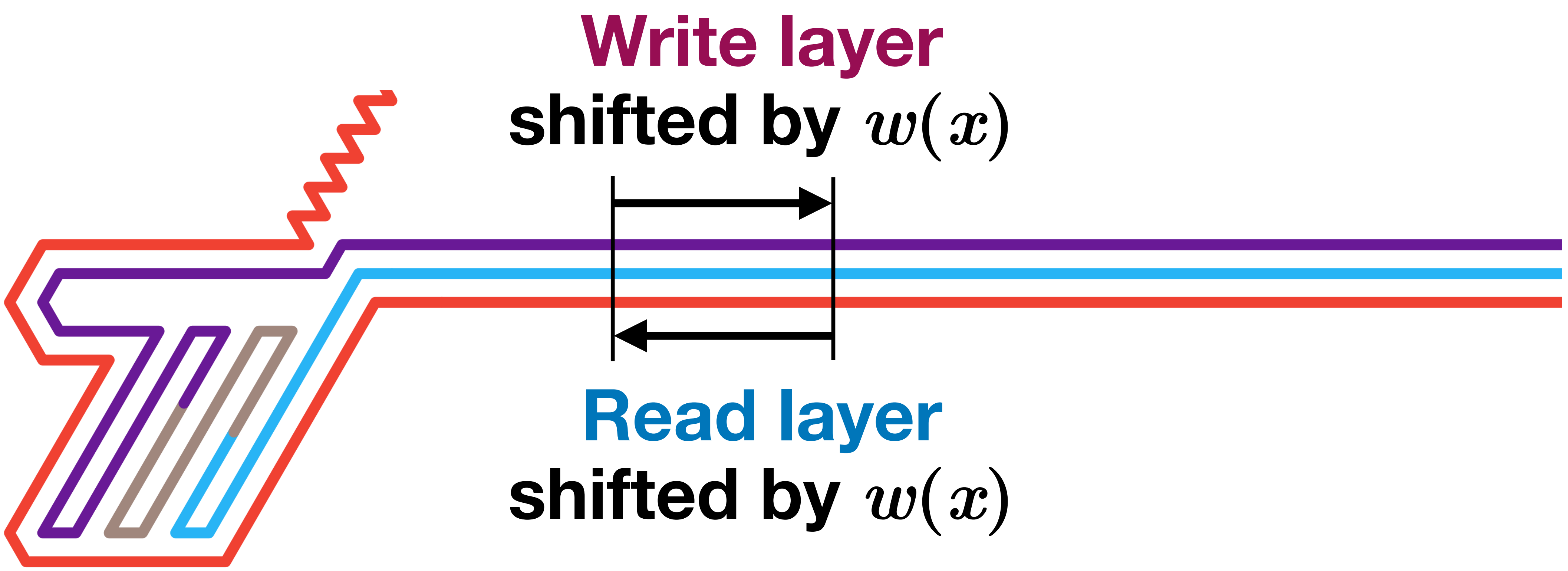


If j–th bit = 1, then Shift += w_{ij}

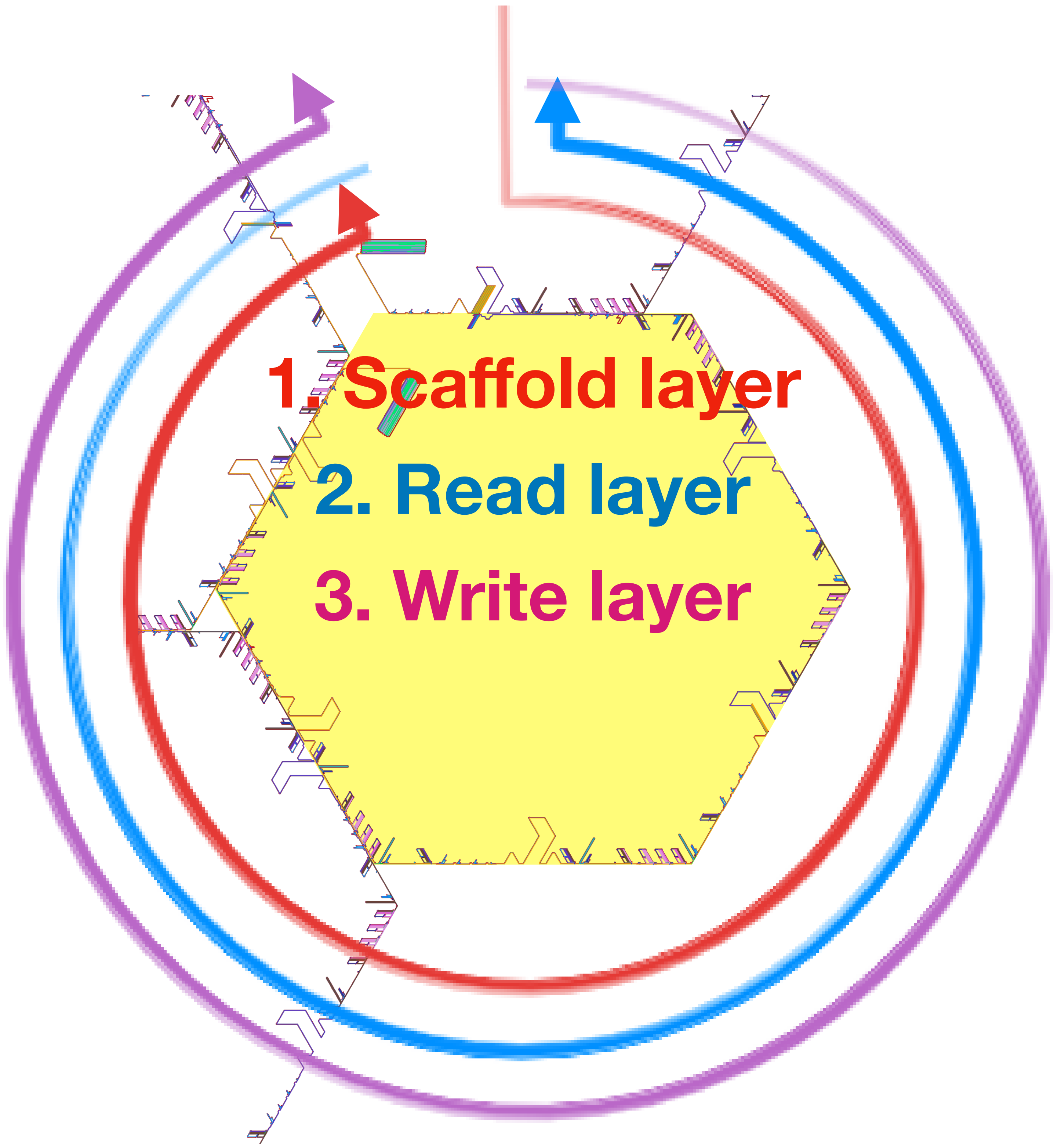
⇒ Shift on i–th side = $\sum_j w_{ij}\ x_{ij} = w(x_i)$

⇒ Total Shift on all side = $w(x)$

Uturn pocket. Read to write

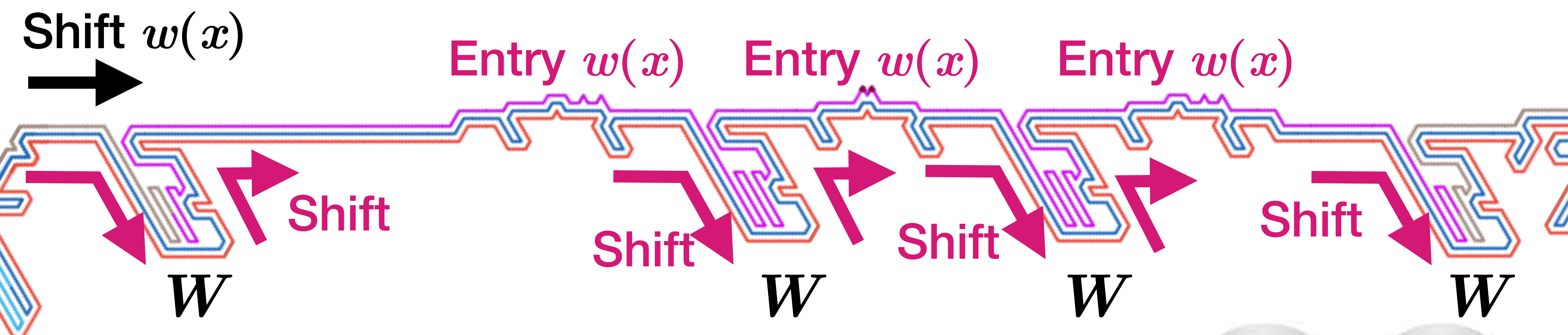


the Macrocell



Writing. Shift pushes the transition table to the right

bits 0 and 1 fold differently

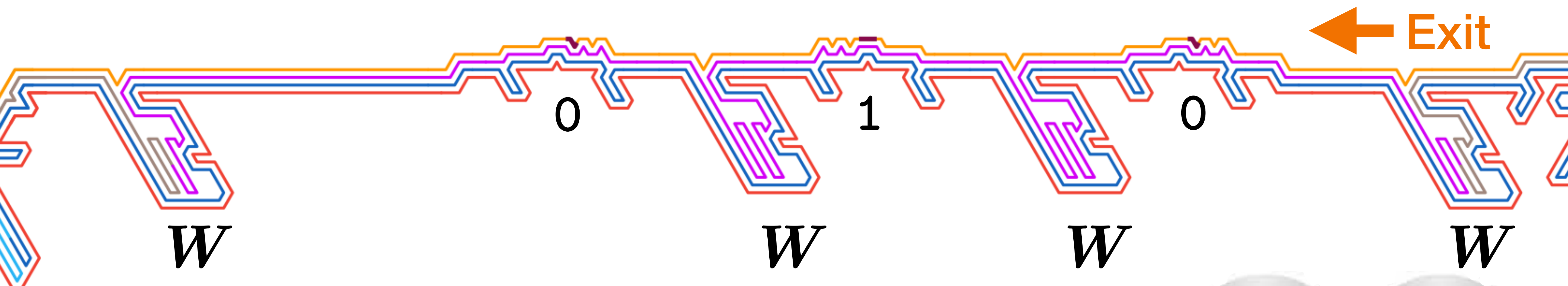


The pockets hide the $W = \sum_{ij} w_{ij}$ unused entries in the transition table

Writing. Shift pushes the transition table to the right

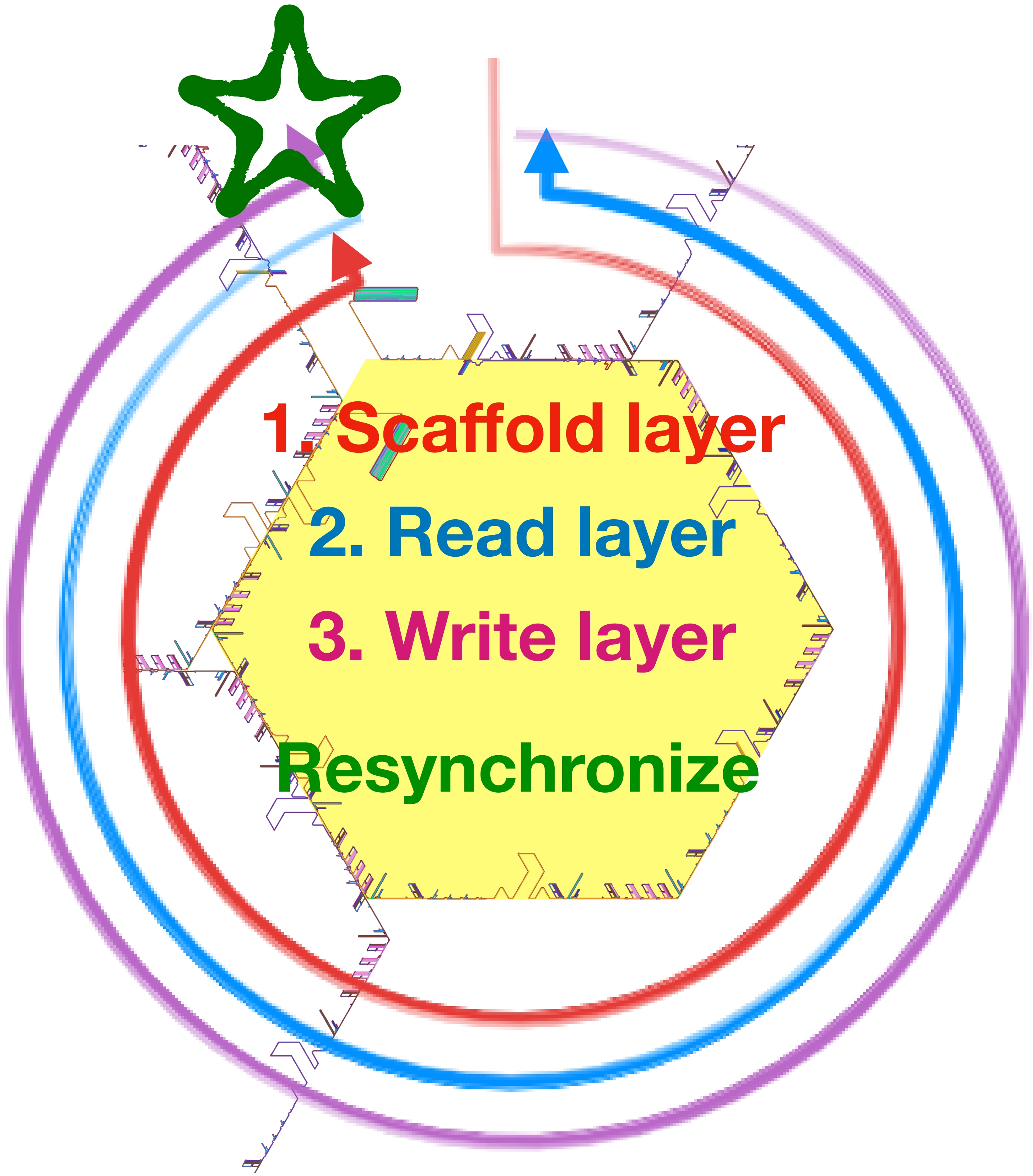
bits 0 and 1 fold differently

⇒ the exit layer shows or hide the special beads



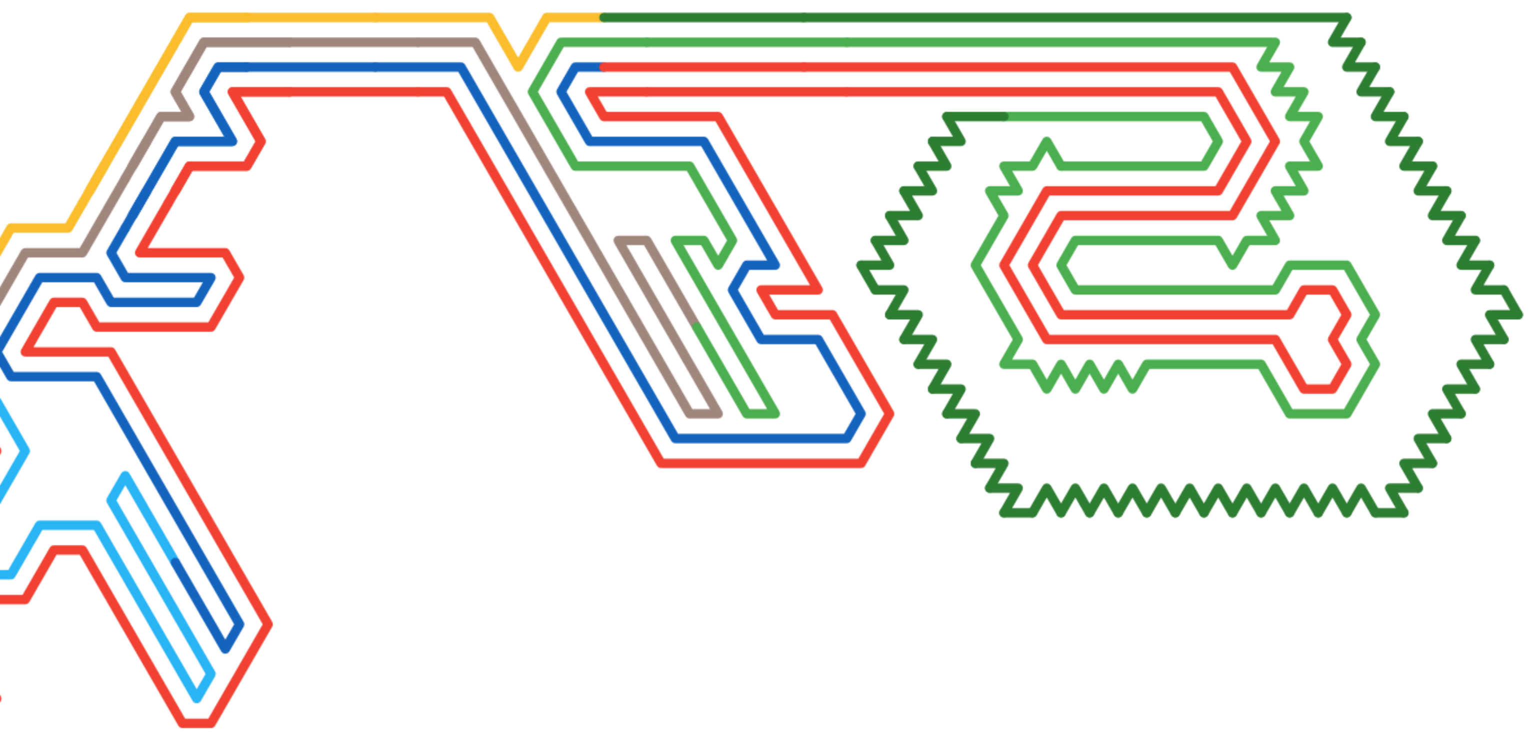
The pockets hide the $W = \sum_{ij} w_{ij}$ unused entries in the transition table

the Macrocell



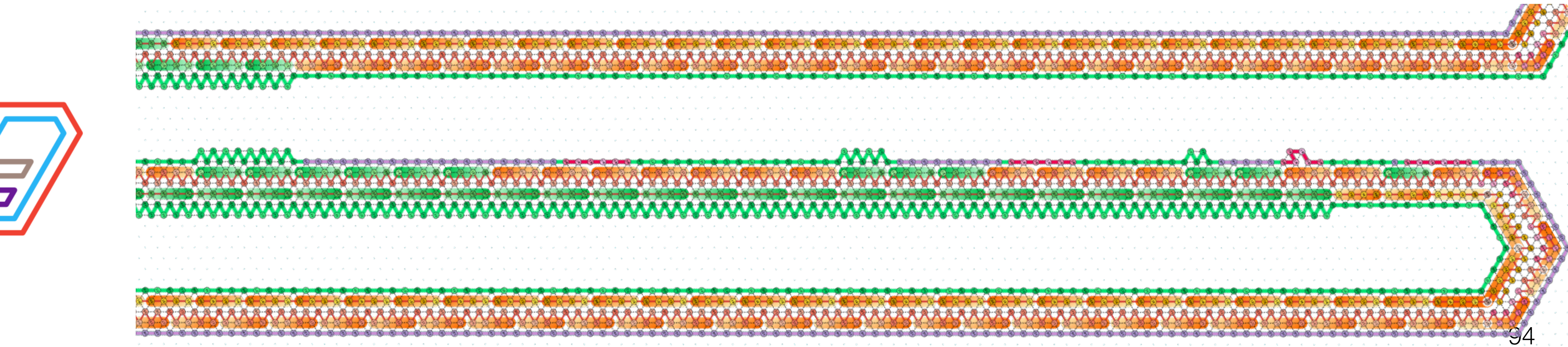
Resynchronizing.

Speedbumps [PSSU2020]

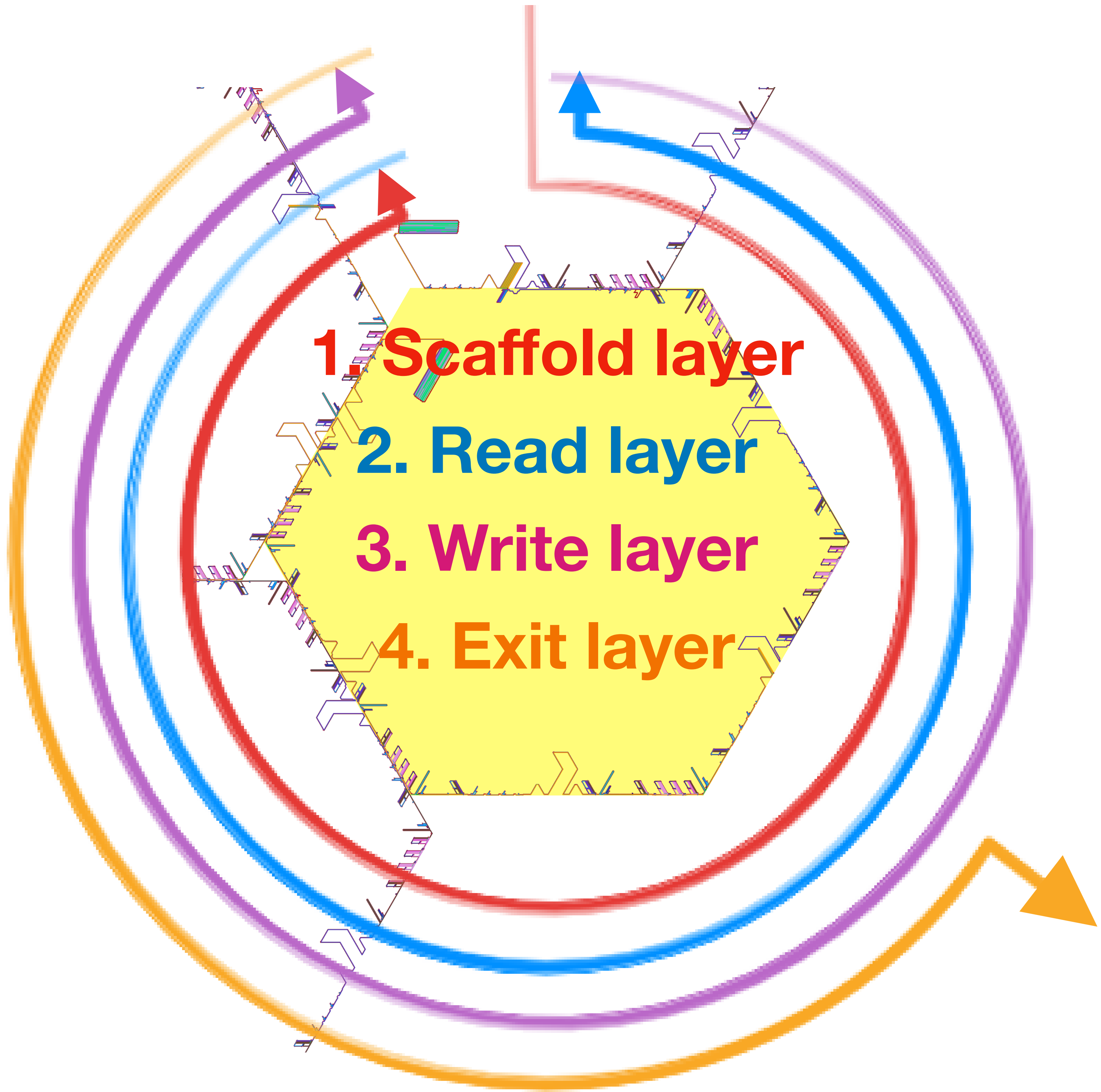


Can absorb any offset ≤ *W*

(in Zig-Zags! 🙃)



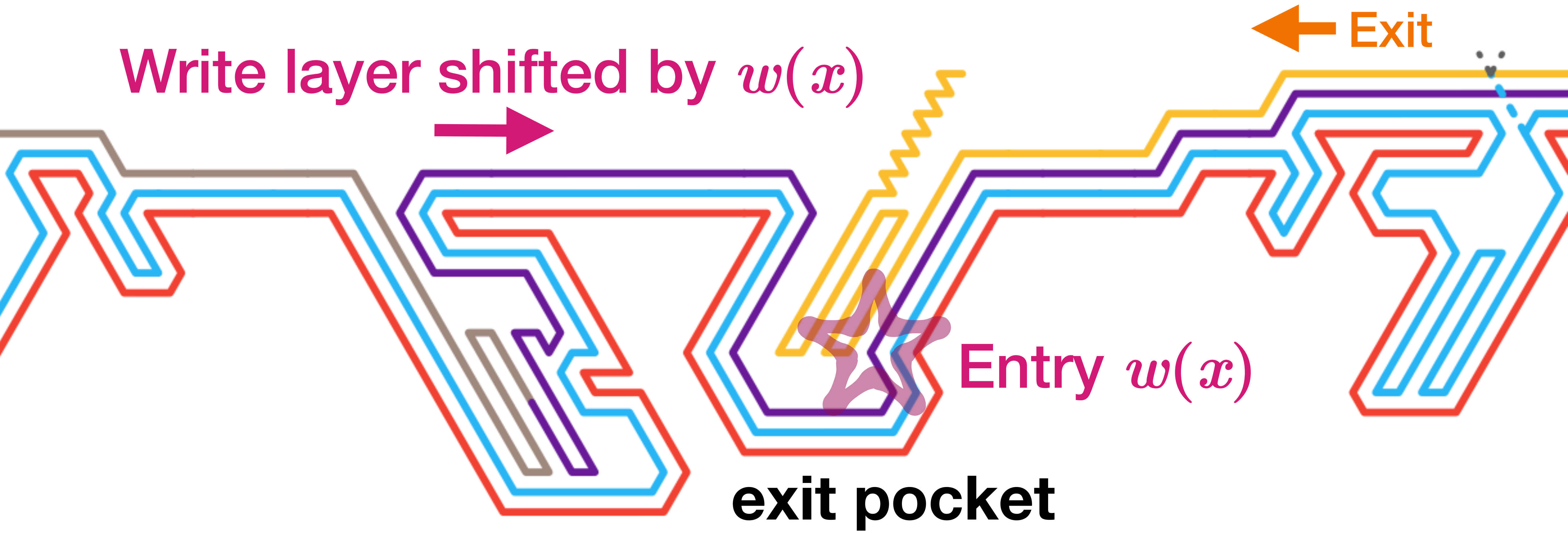
the Macrocell



Exiting... or not

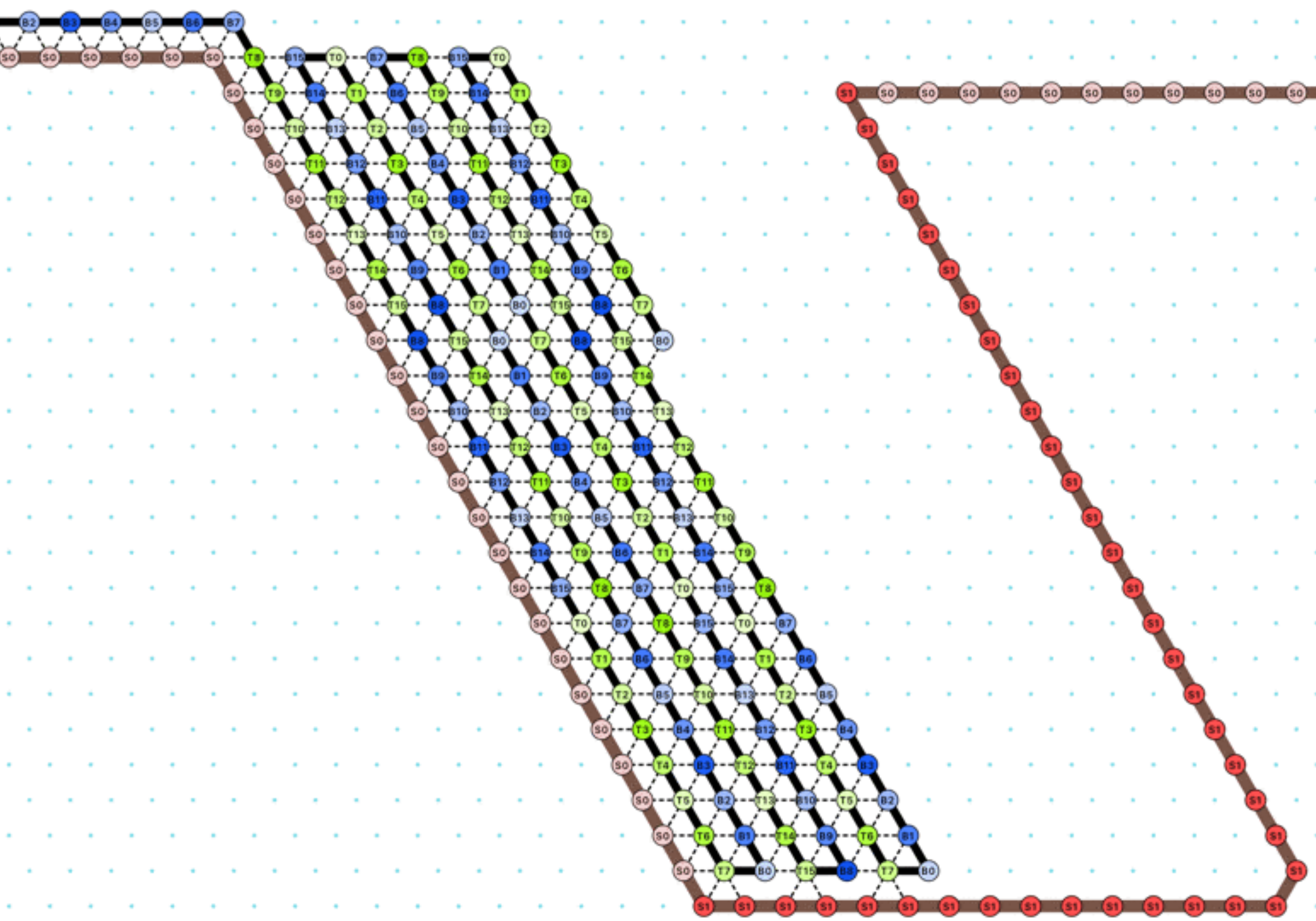
By default, exit layer follows the border of the exit box

Exiting... or not



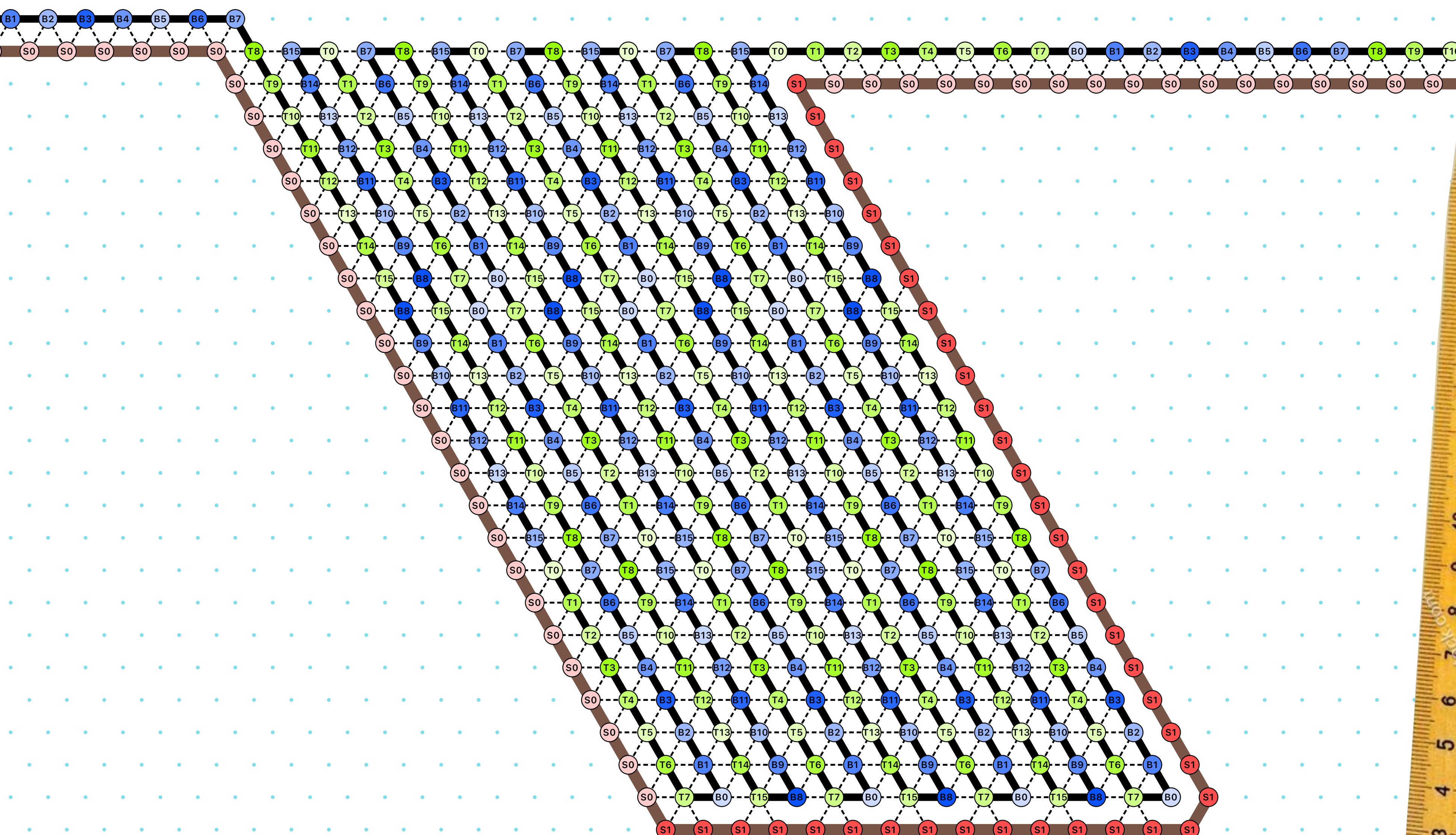
With the proper signal (offset!), exit layer folds upon itself and... exit!

Key new tool: Folding meter



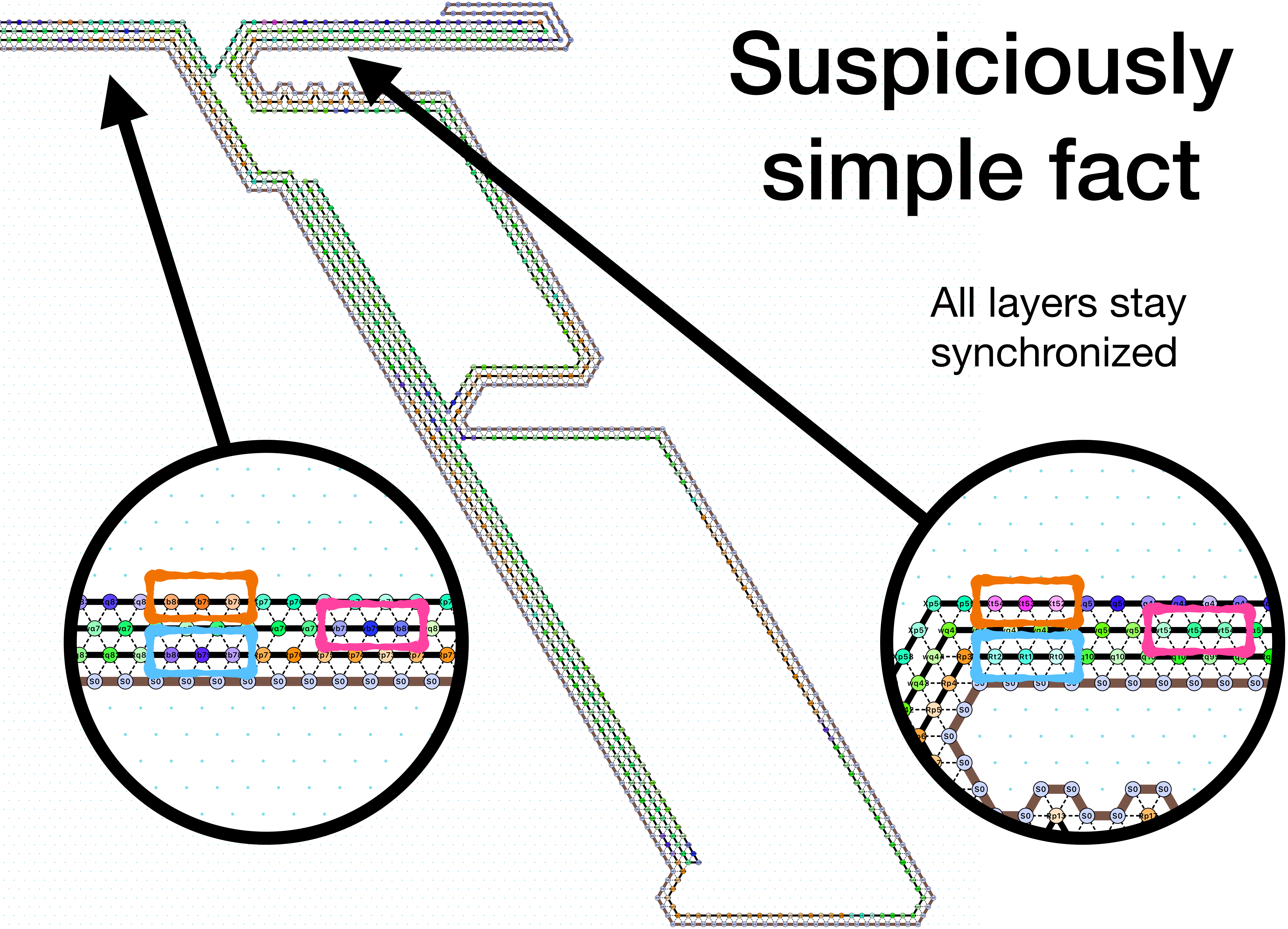
it folds upon itself into pockets

Key new tool: Folding meter



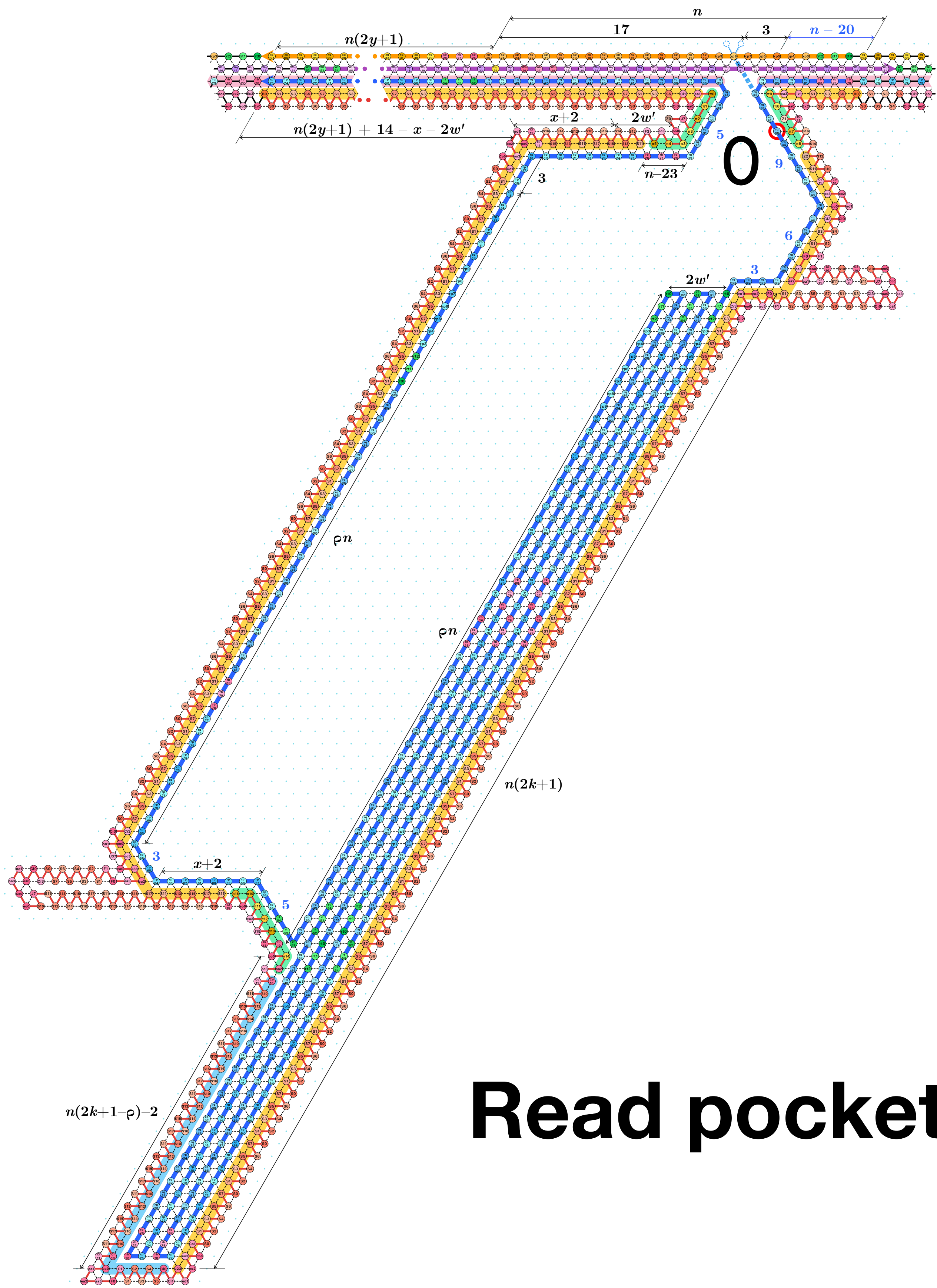
it folds upon itself into pockets

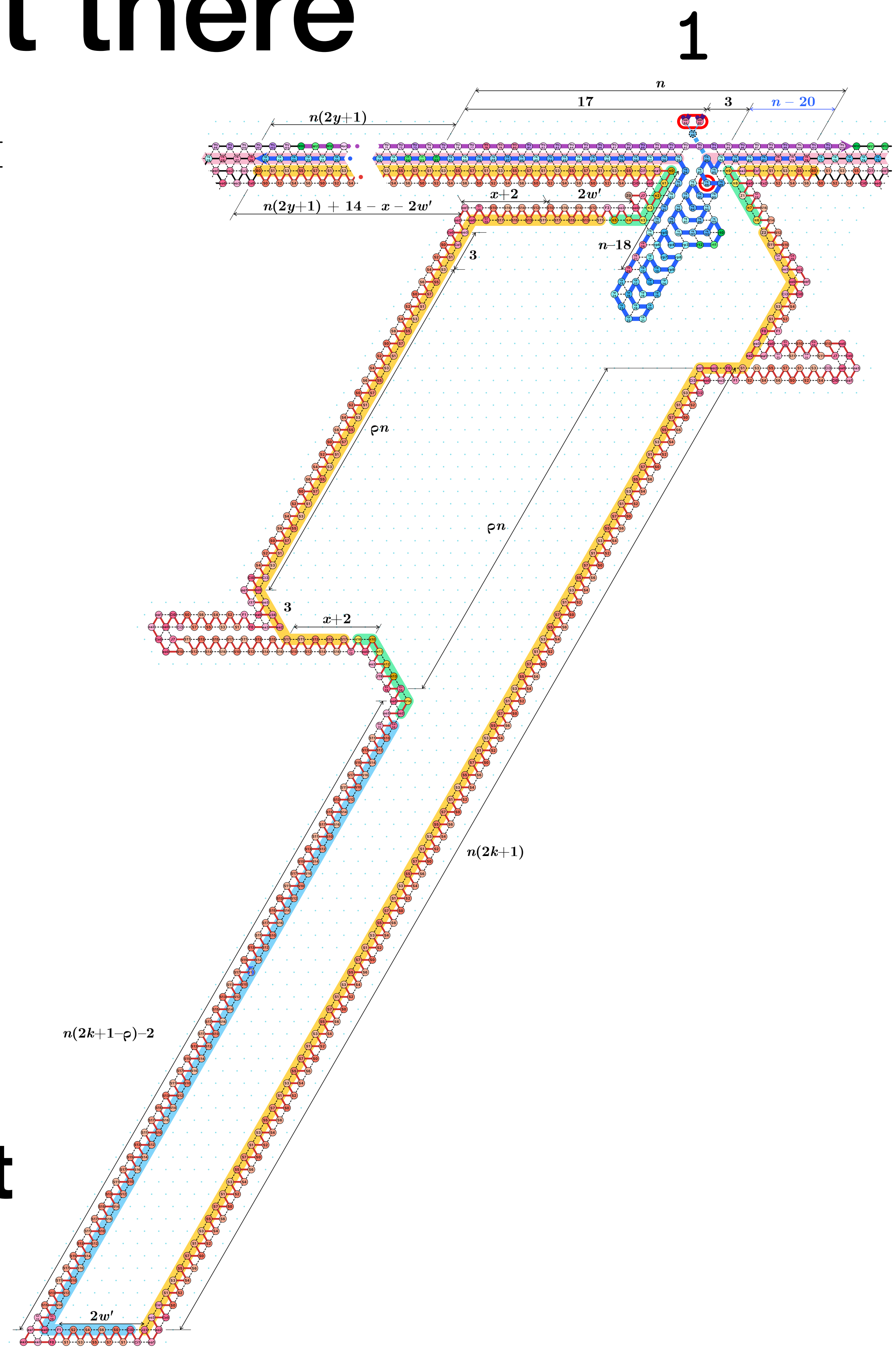
Suspiciously simple fact



All layers stay synchronized

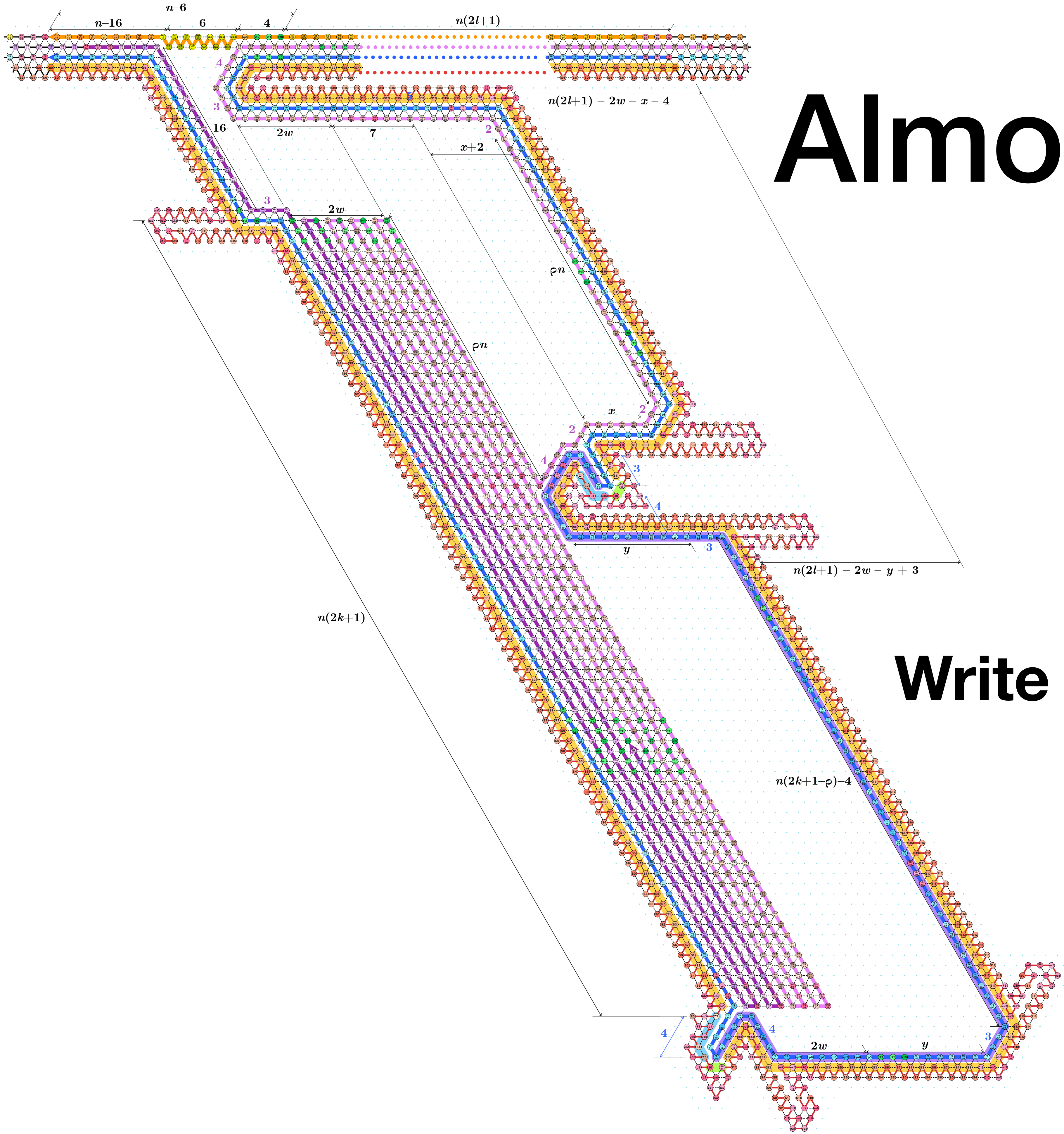
Almost there





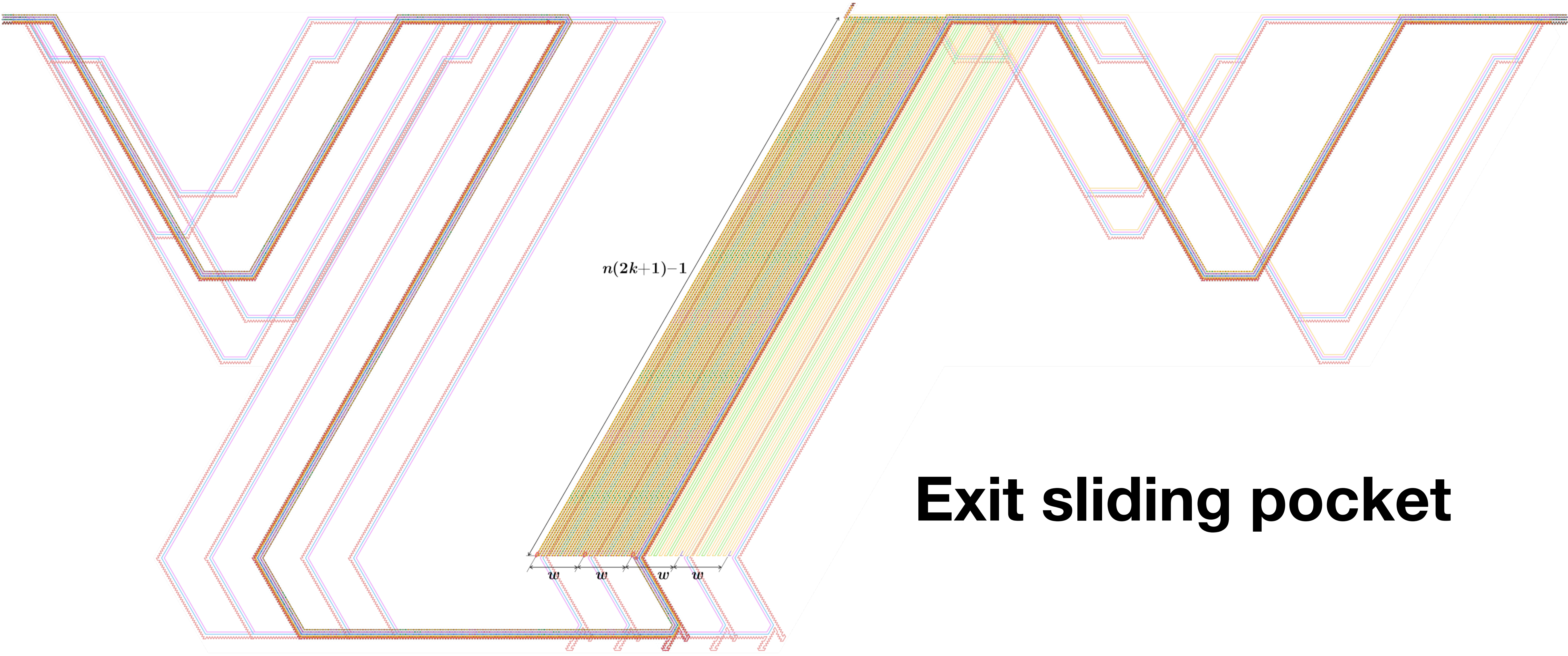
Read pocket

Almost there



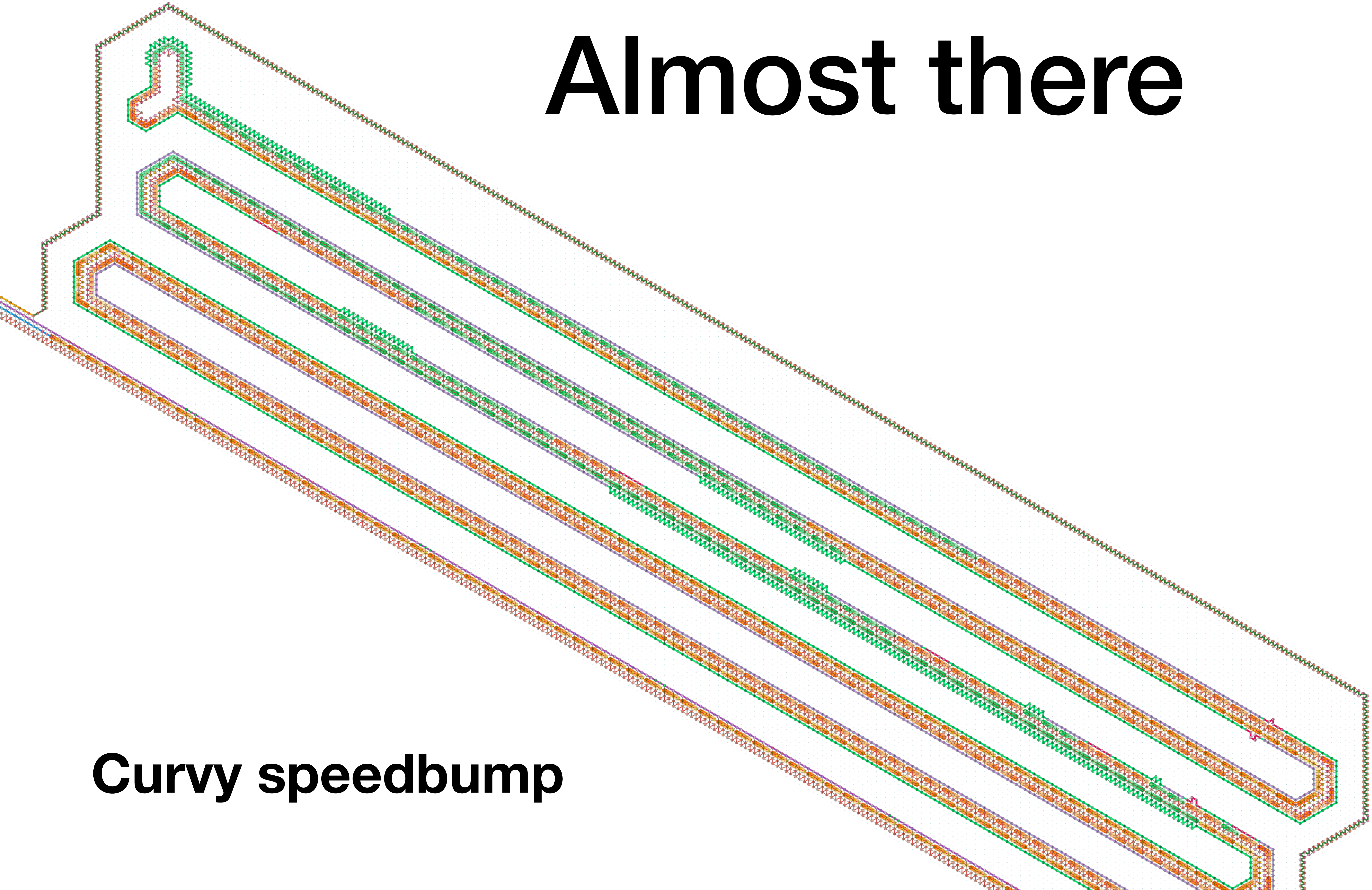
Write pocket

Almost there

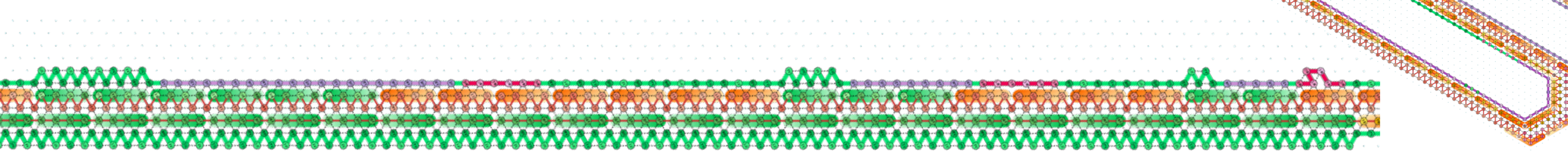


Exit sliding pocket

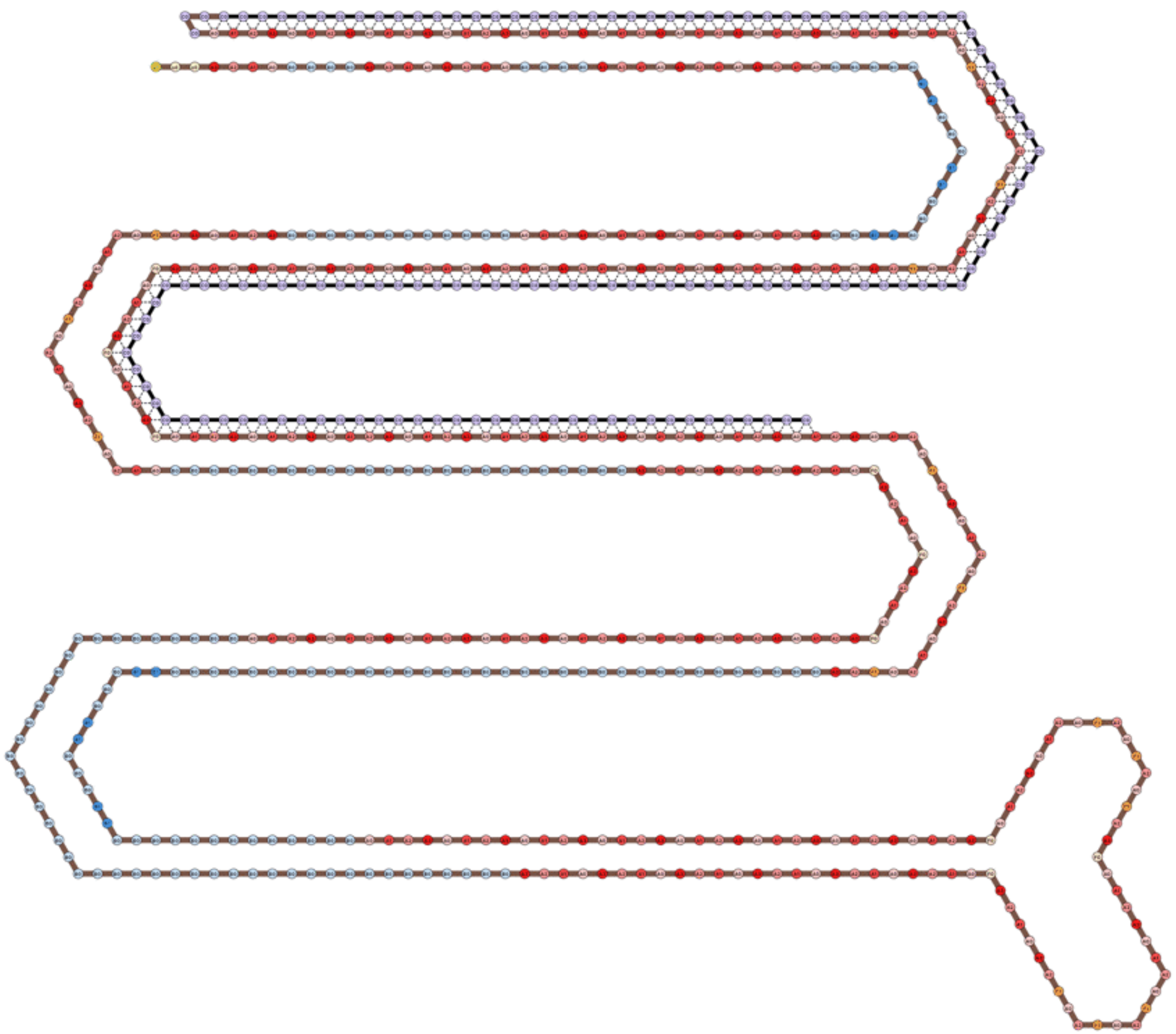
Almost there

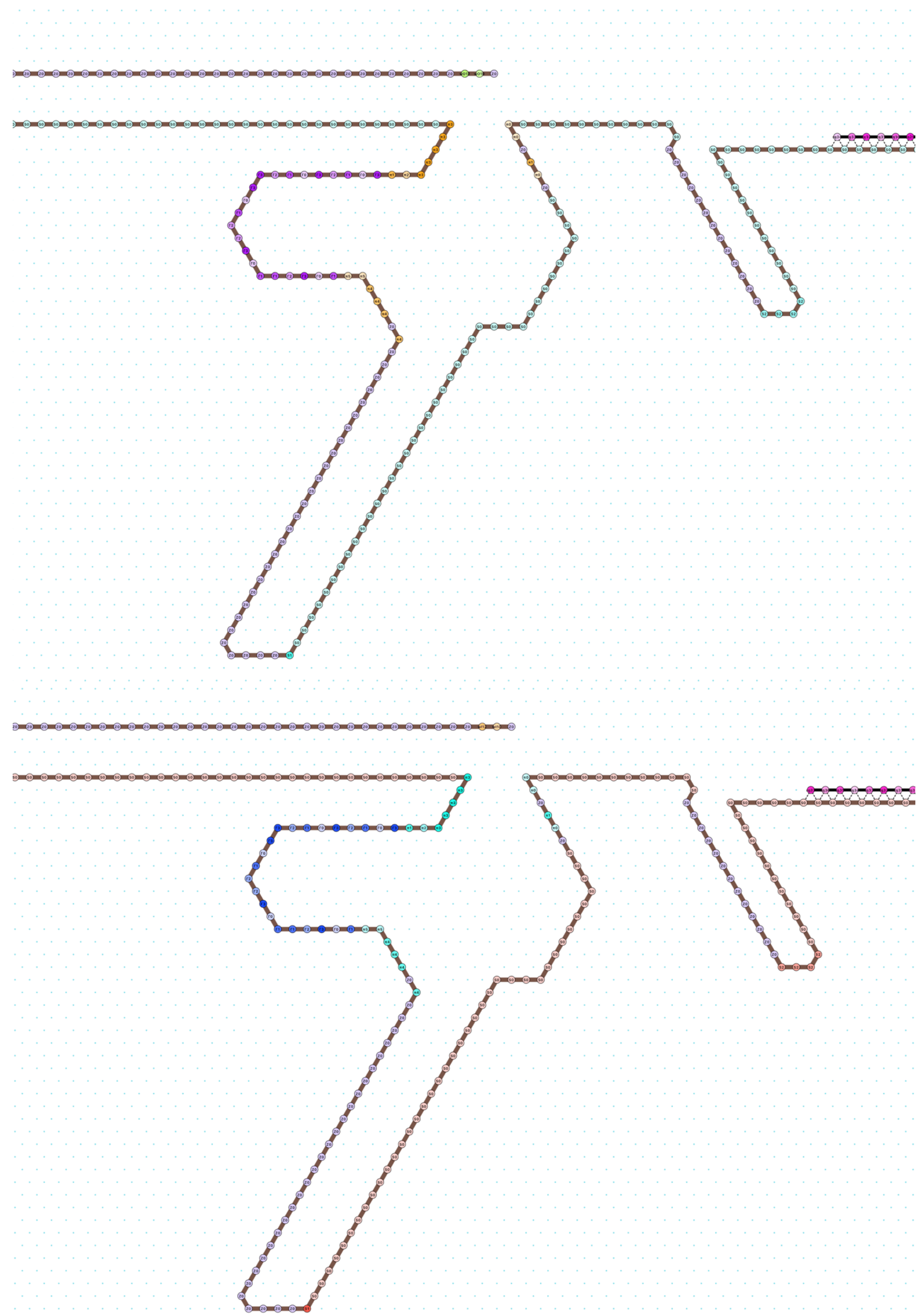


Curvy speedbump

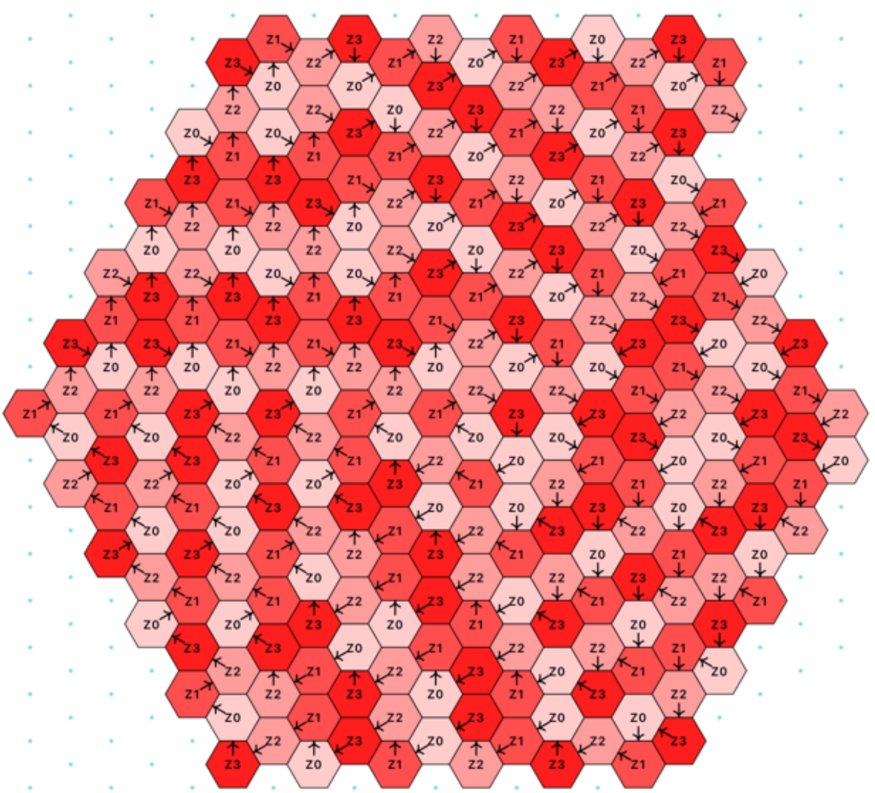


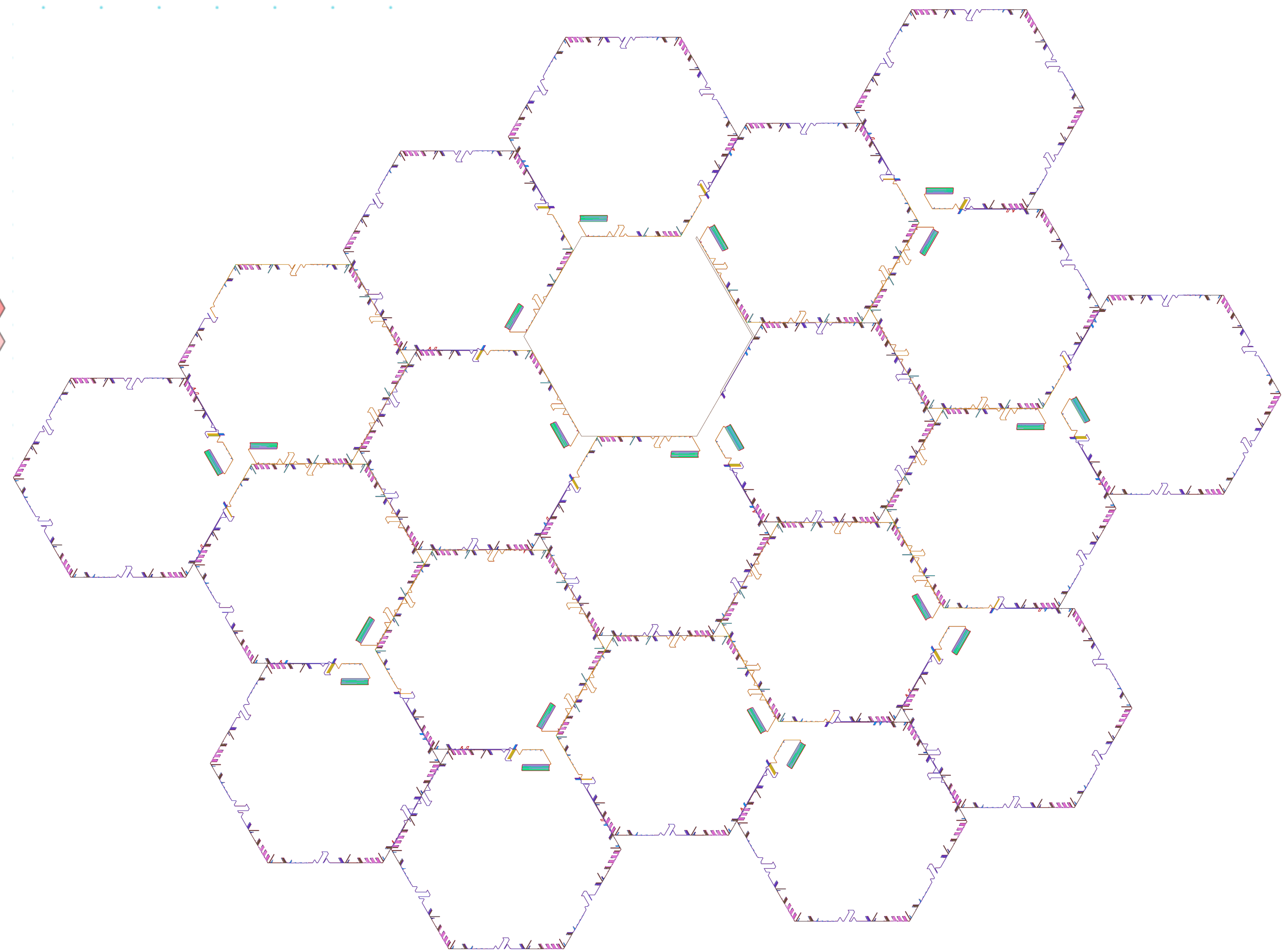
Almost there





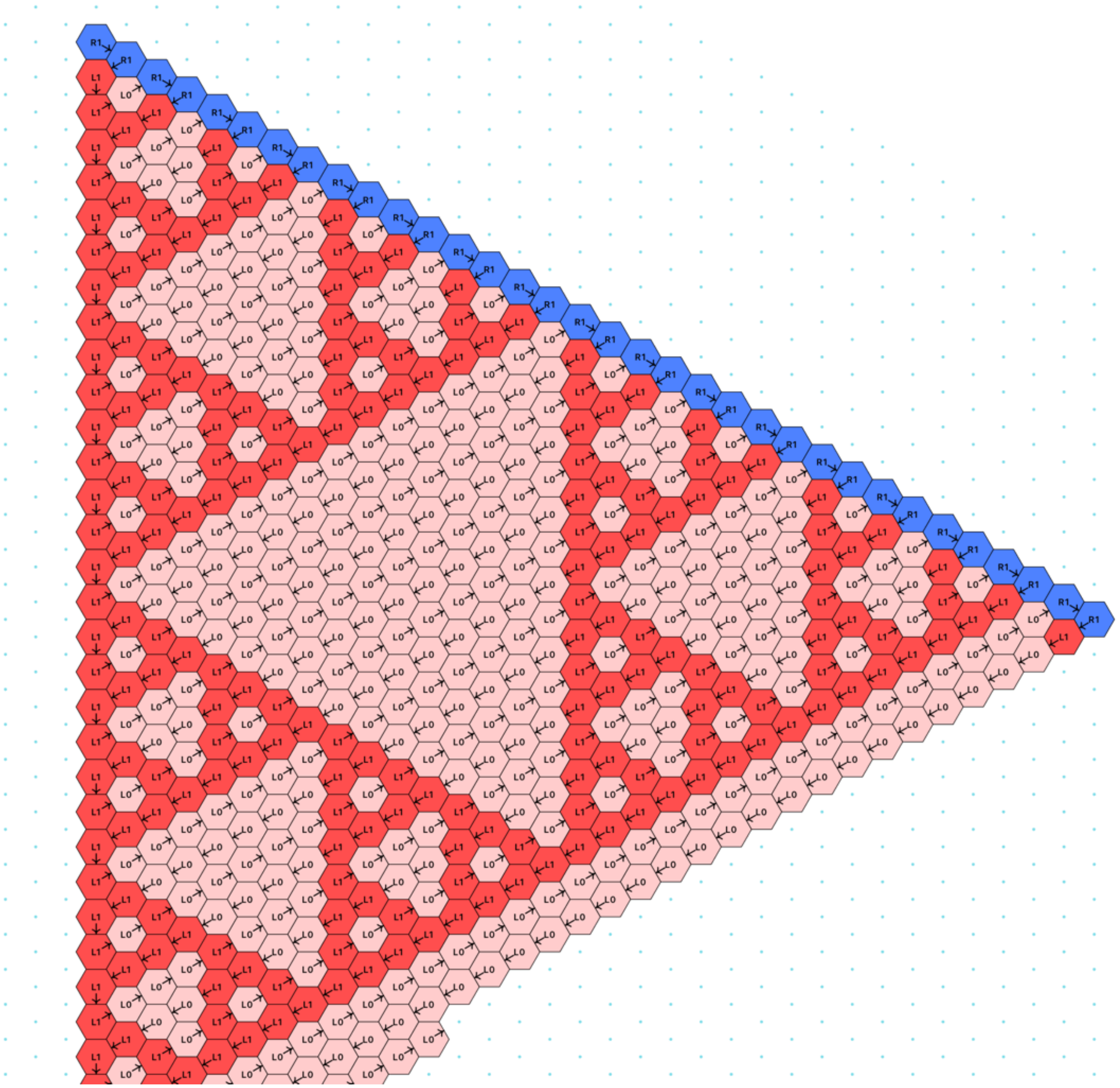
Theorem 2. Delay-3 oritatami simulates intrinsically radius-1 turedos

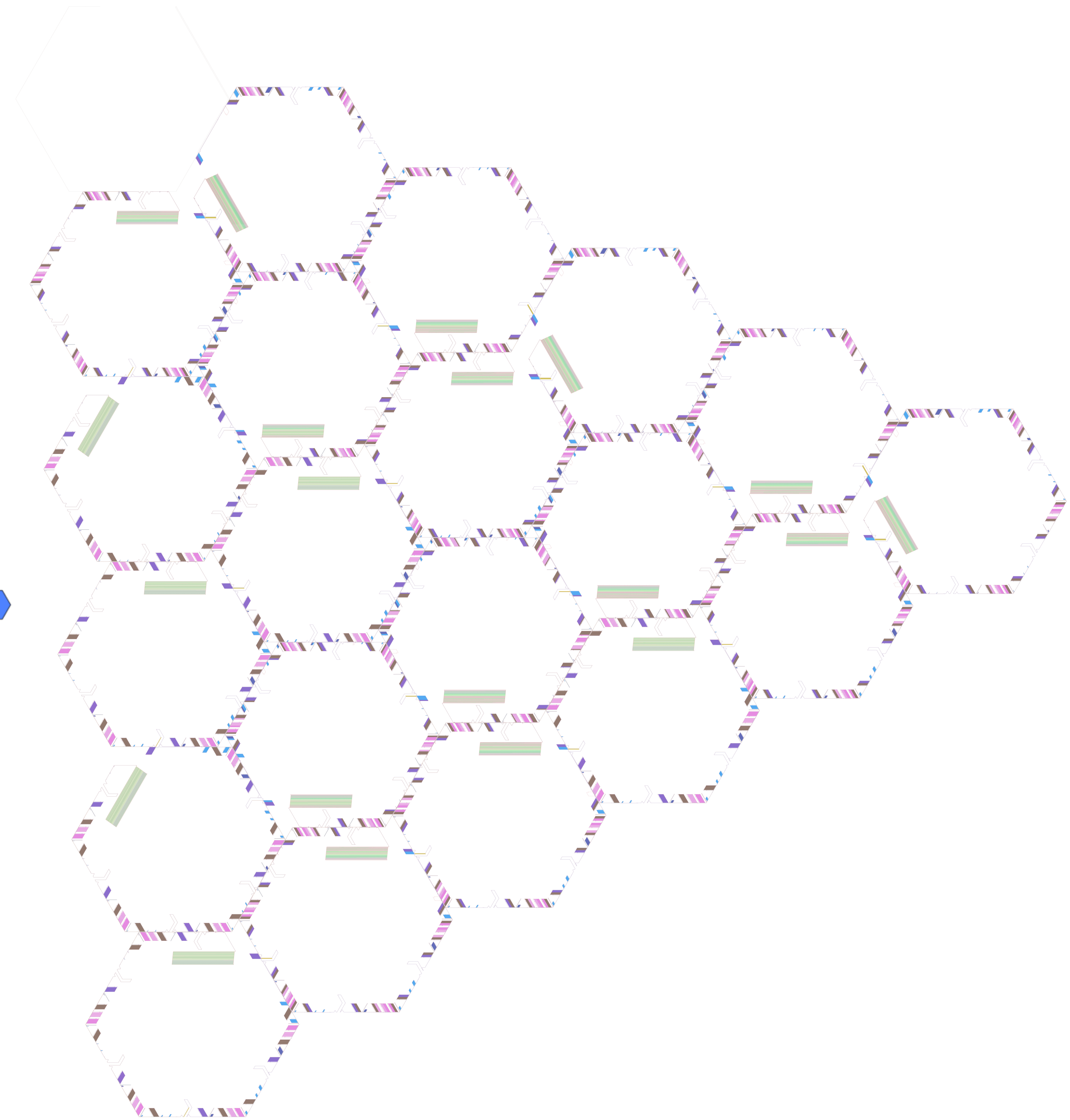




Example: A bouncy right hand turedo

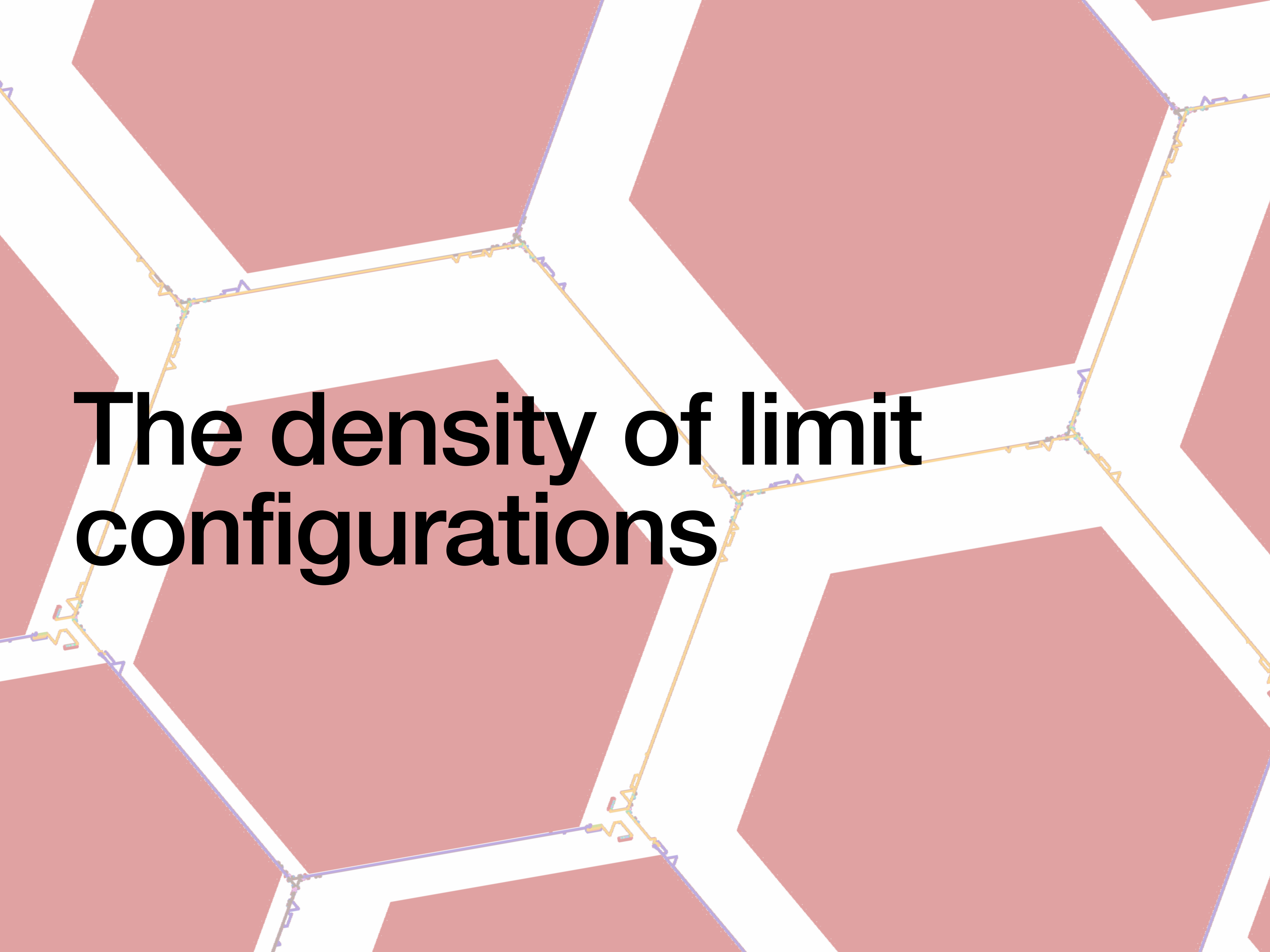
Some examples





A Sierpinsky turedo

The density of limit configurations

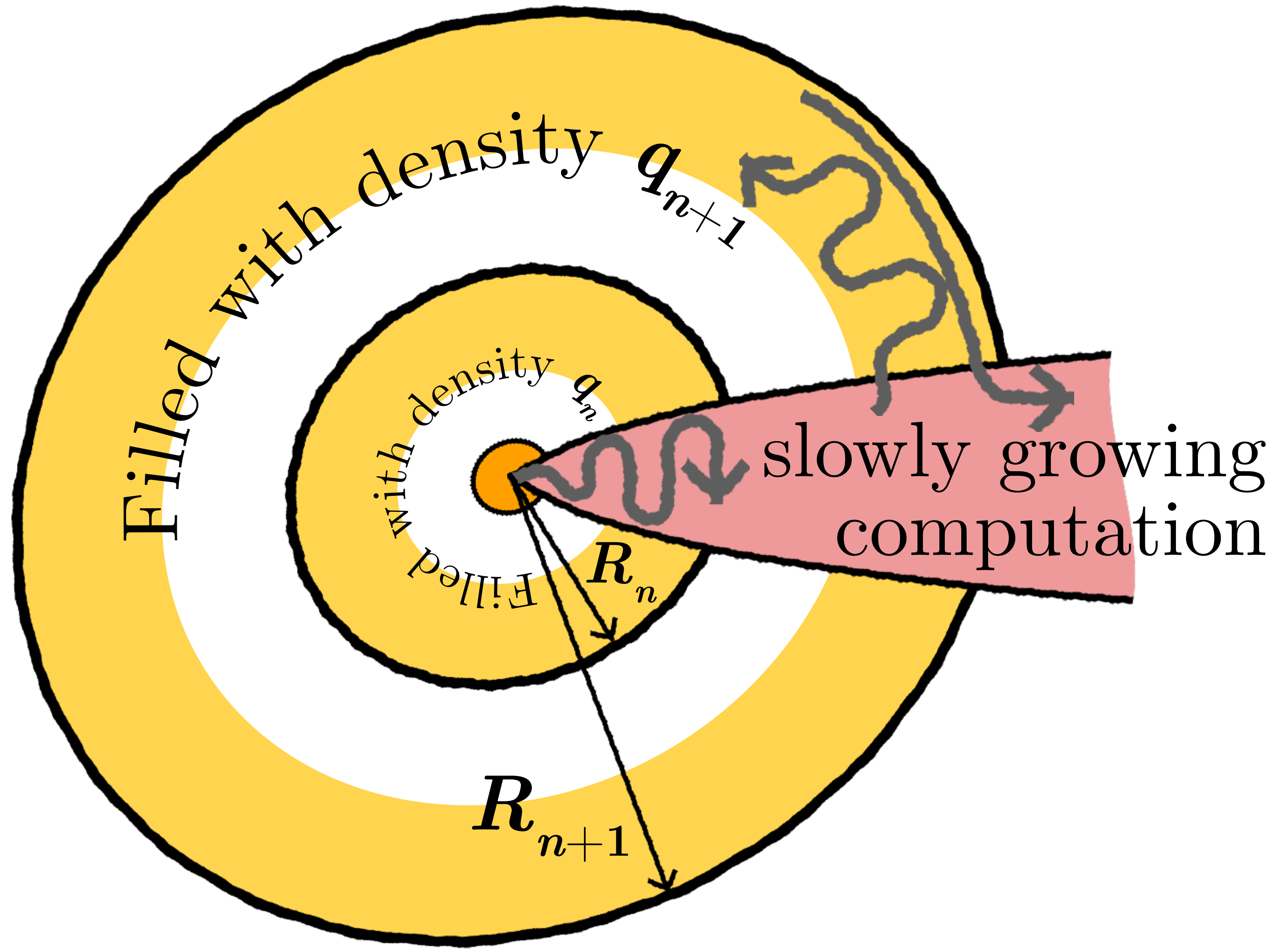


Theorem 3. The densities of radius-1 Turedos limit configuration are all Π_2

limsup$_n$ $q_n = q \in \Pi_2$

Next annulus erases previous one by computing R_n such that:

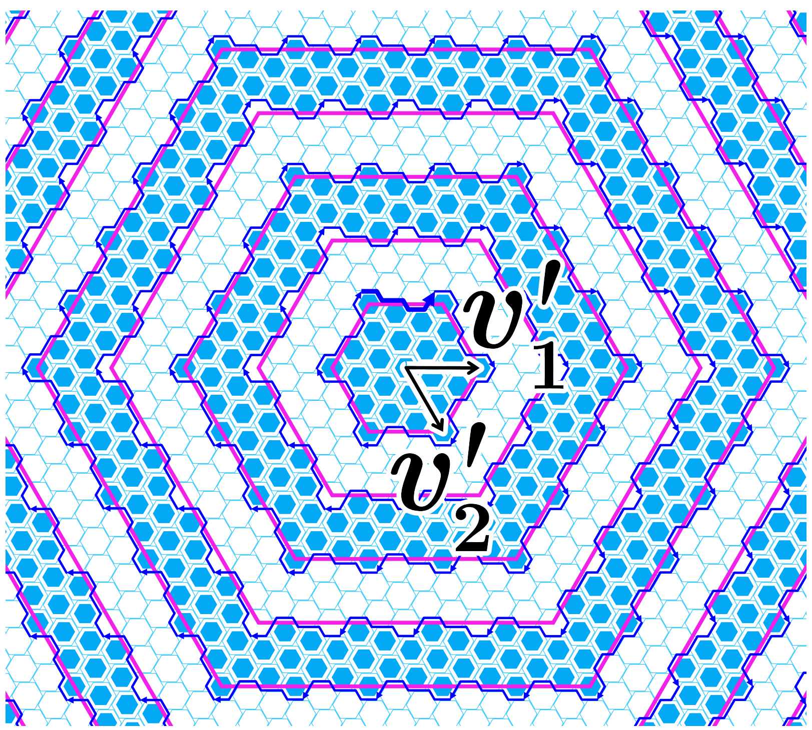
$(R_{n+1})^2 q_{n+1} \gg (R_n)^2 q_n$



Fact. The densities of limit configurations for directed aTAM and freezing cellular automata belong to Π_2 also.

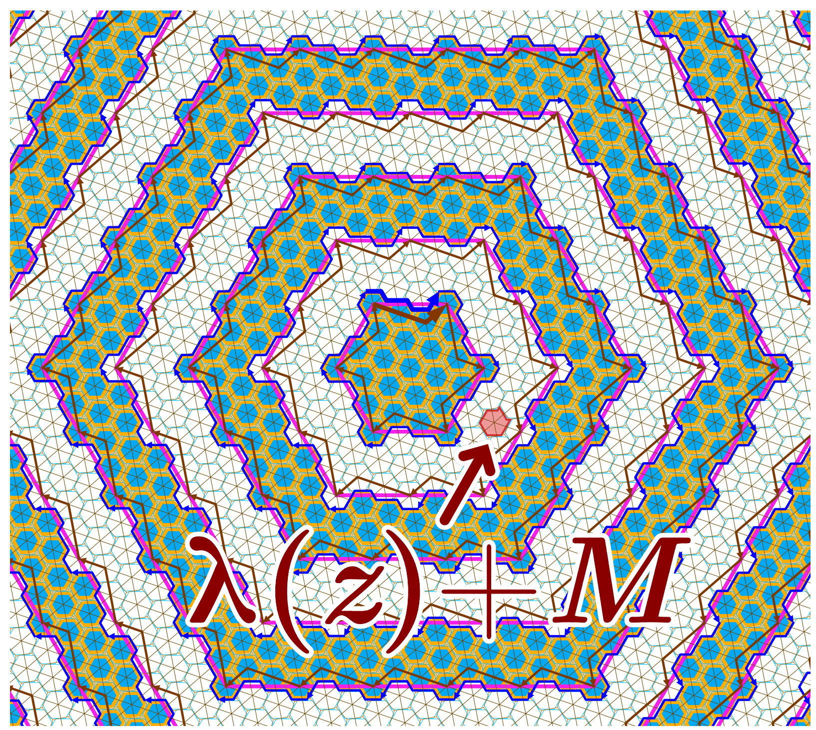
Corollary. The densities of oritatami limit configuration are all Π_2

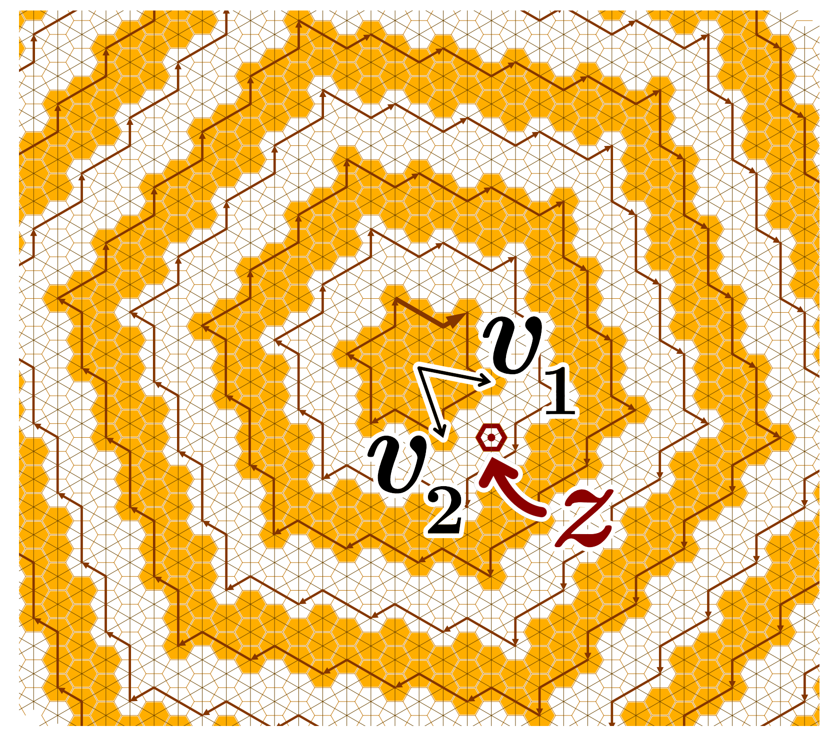
Must beware of the **rotation induced** by the oritatami simulation



Oritatami world

The target balls



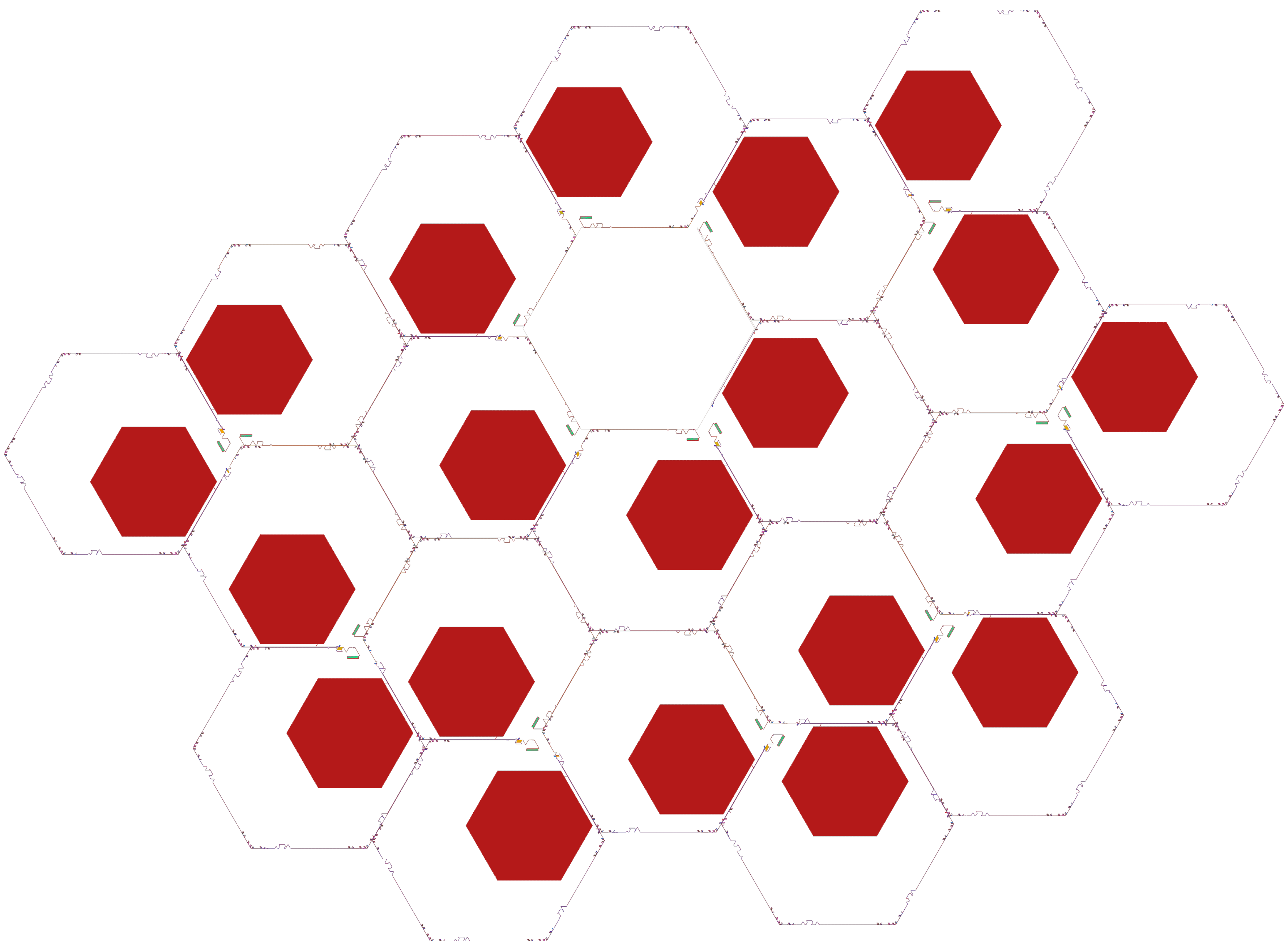


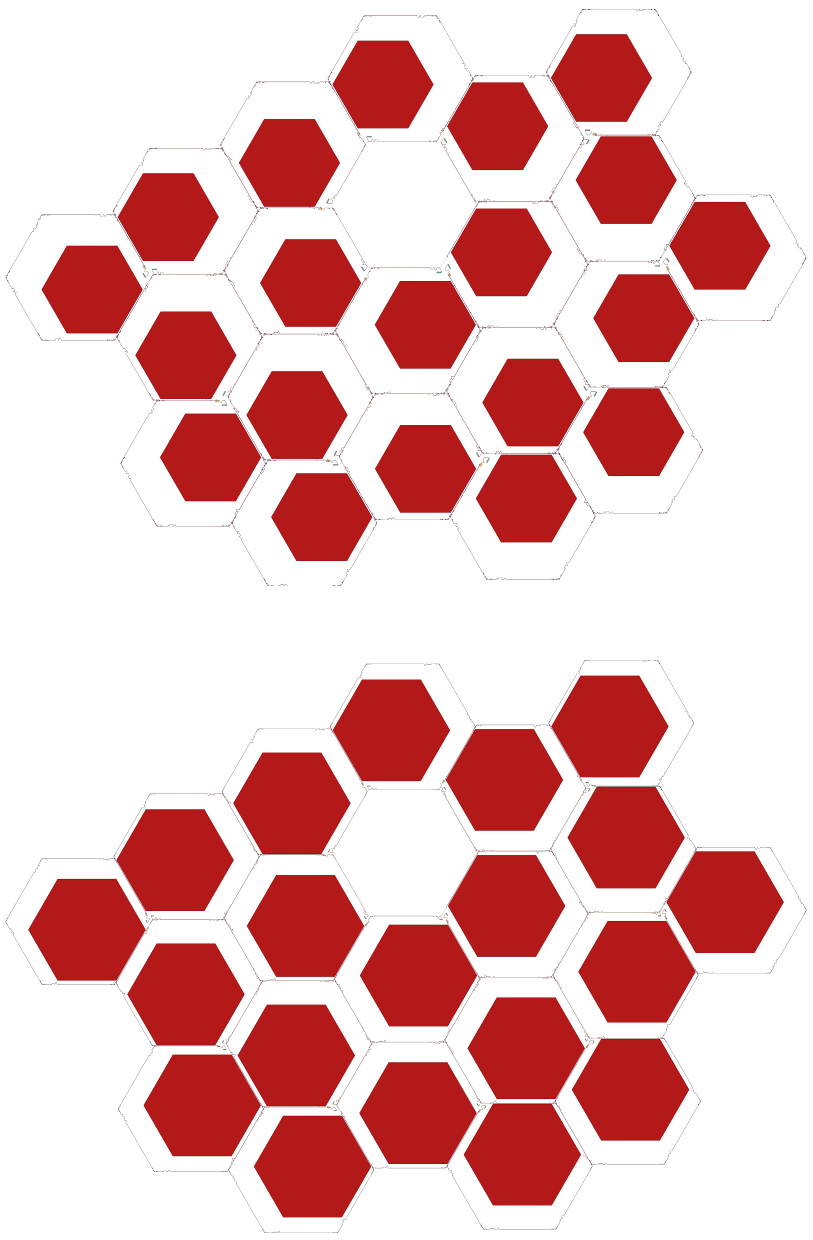
Turedo world

The simulated balls

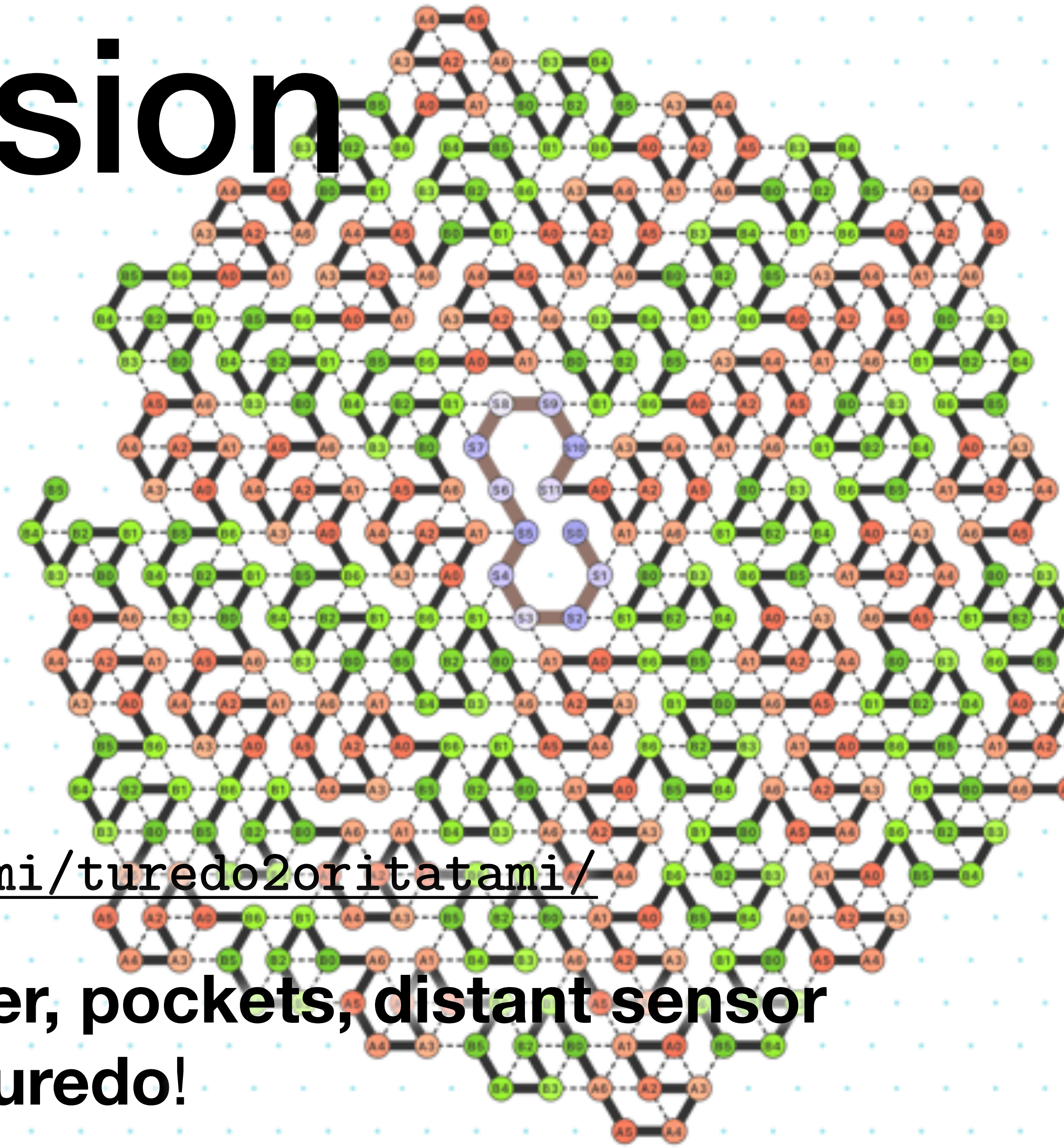
Corollary. The densities of oritatami limit configuration are all Π_2

Fill a hardcoded hexagon in the center of an extended macrocell





Conclusion



- **No need for parallelism**
- **No need for 3D**
- Lines just don't cross!
- We have a running implementation:
  https://hub.darcs.net/turedo2oritatami/turedo2oritatami/
- New tools for Oritatami: **Folding meter, pockets, distant sensor & crazy-curvy speedbump, and… Turedo**!

- Some interesting turedos implemented in oritatami and… RNA ?
  e.g.: **simple plane filling oritatami**
- **What about turedos ?** → S. Nalin & G. Theyssier *(coming soon)*