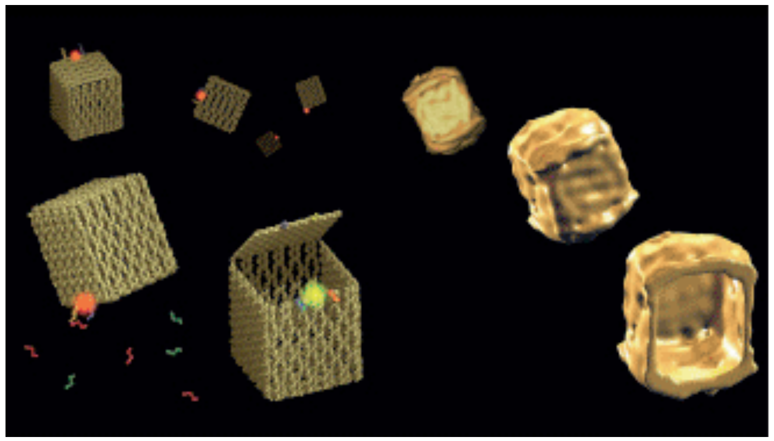
# Oritatami: A computational model for cotranscriptional folding

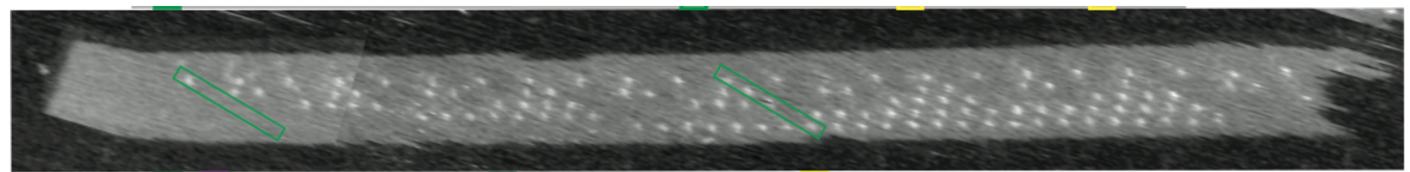
Nicolas **Schabanel**
CNRS - LIP, ENS Lyon & IXXI - France

# Context: Biomolecular Computing & Engineering

~100 nm

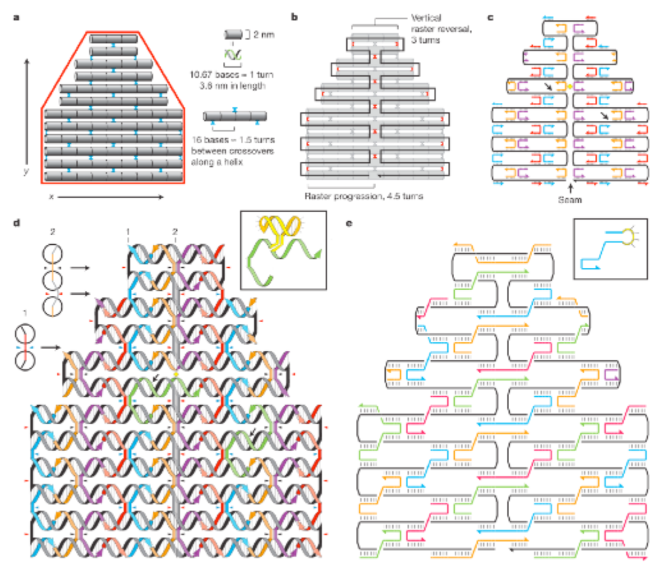


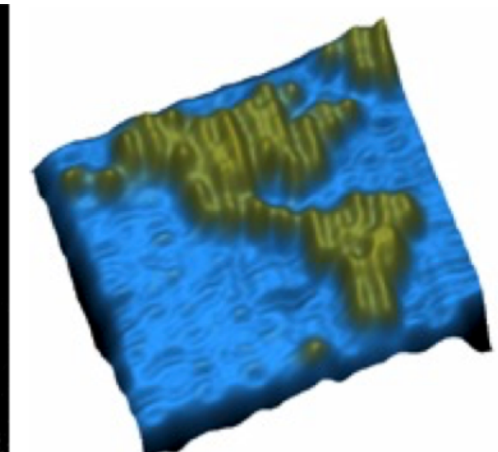
*Andersen et al, 2009*



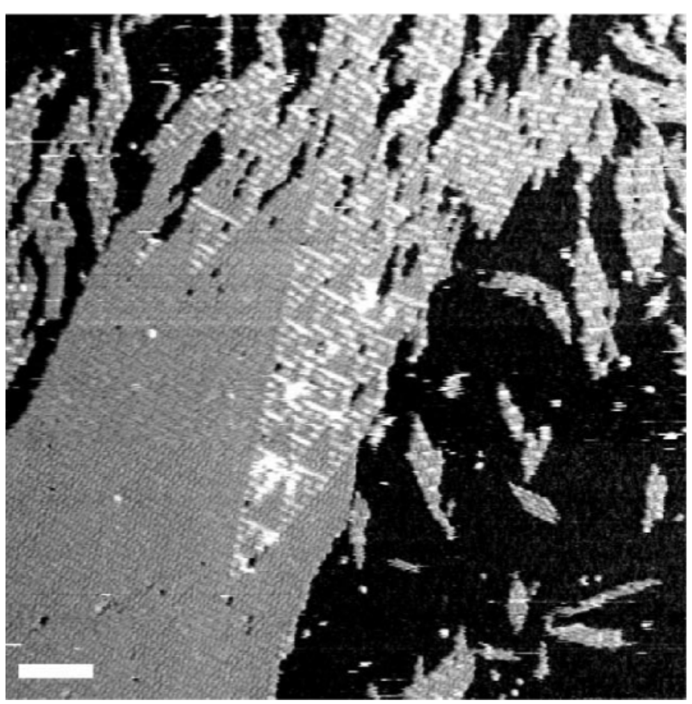
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

**Constantine Evans, PhD Thesis, Caltech 2014**

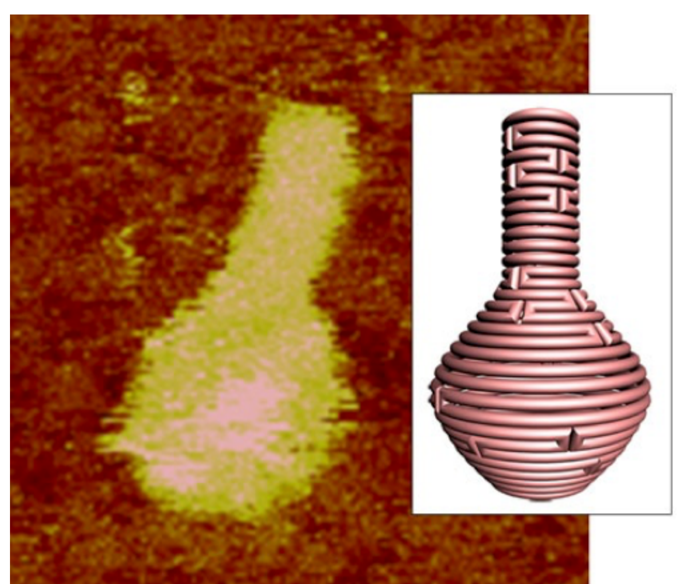




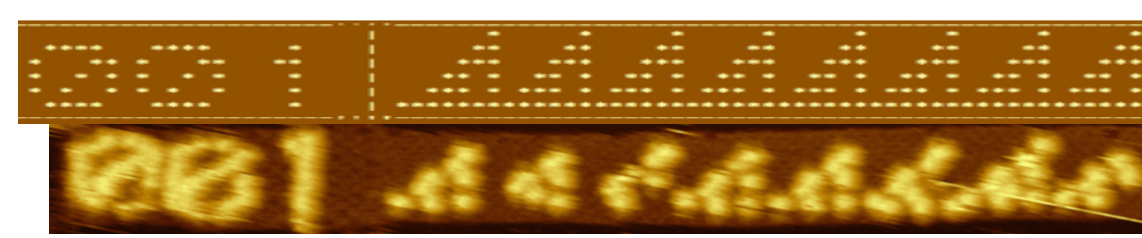
**Rothemund, Nature 2006**



*Fujibayashi et al, 2007*



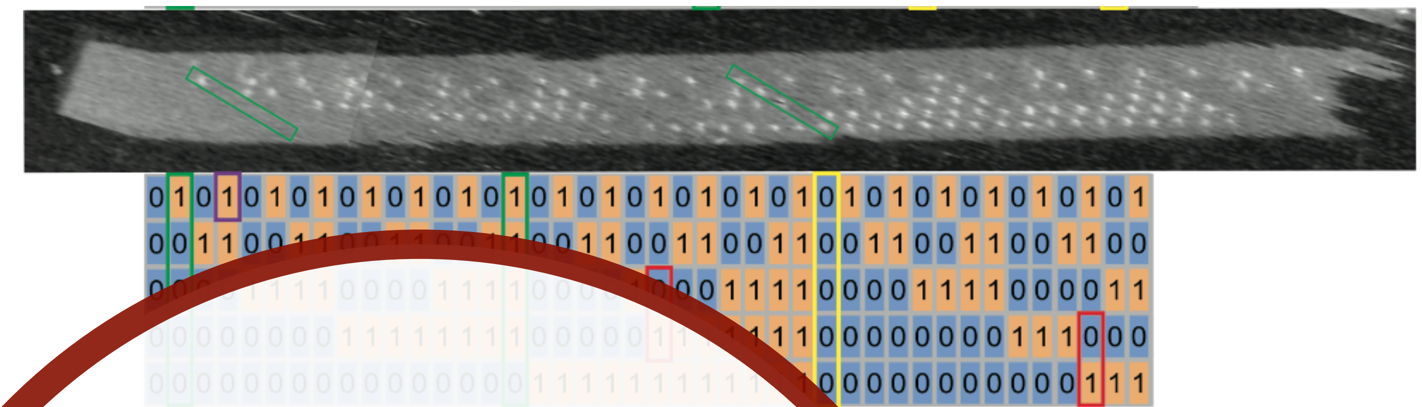
**Han et al, Science 2011**



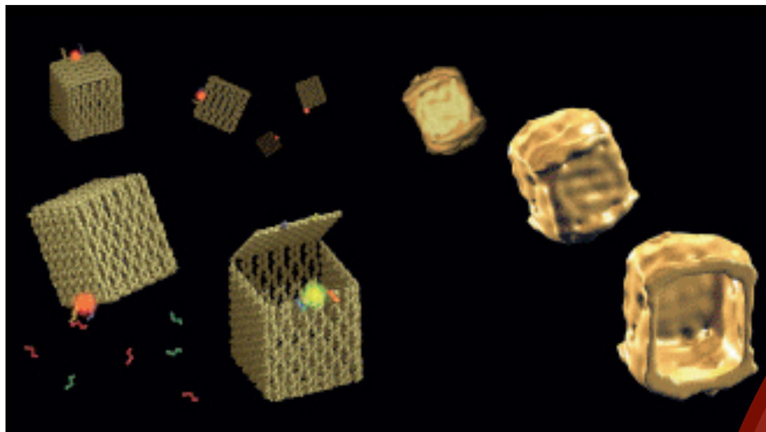
*Rule 110 on input 001 - Woods et al, Nature 2019*

# Context: Biomolecular Computing & Engineering

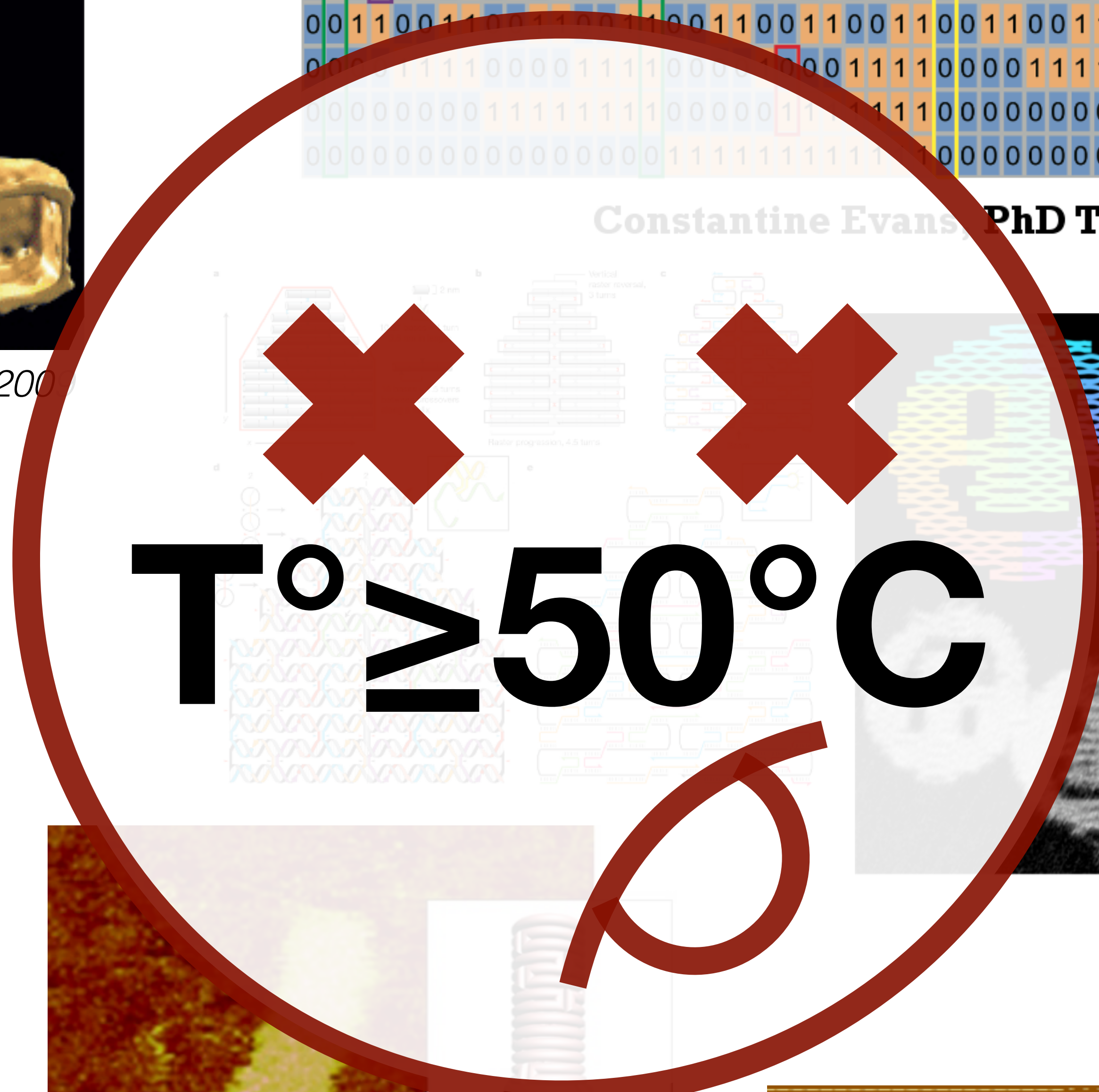
~100 nm

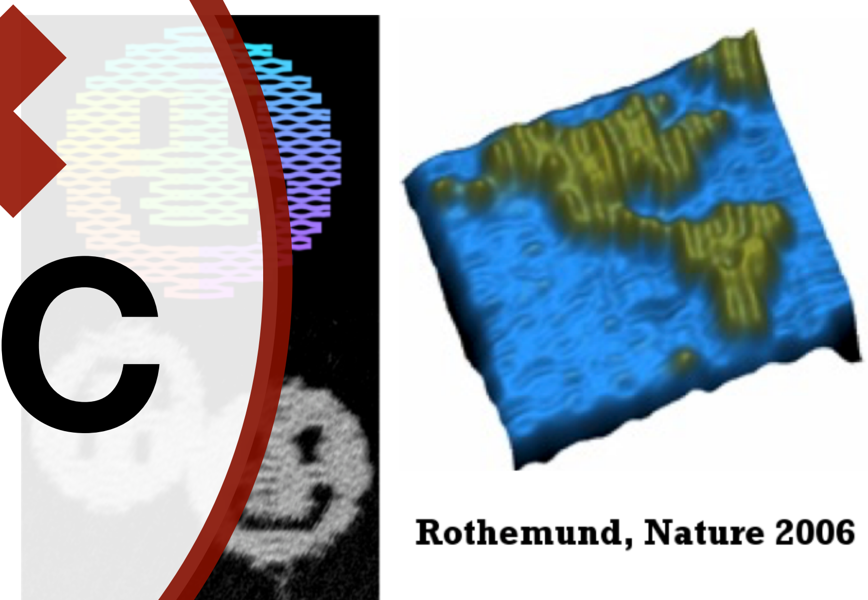


Constantine Evans, PhD Thesis, Caltech 2014

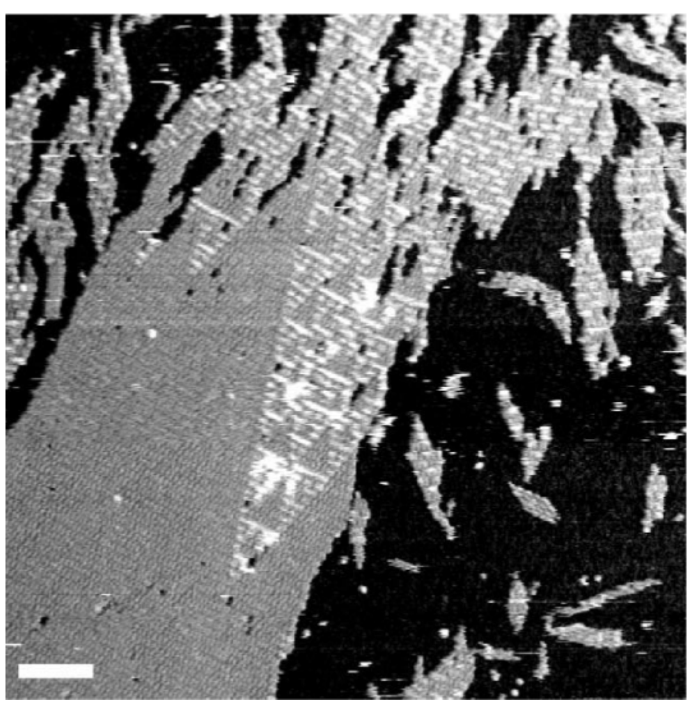


*Andersen et al, 200*

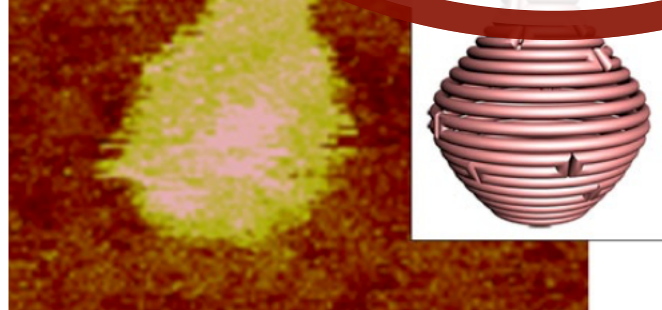


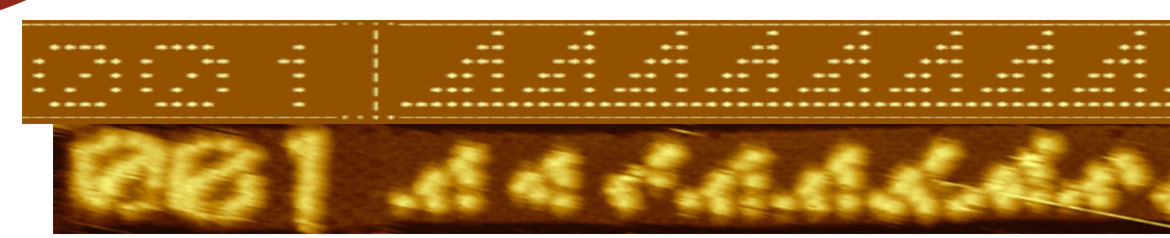
Rothemund, Nature 2006



*Fujibayashi et al, 2007*

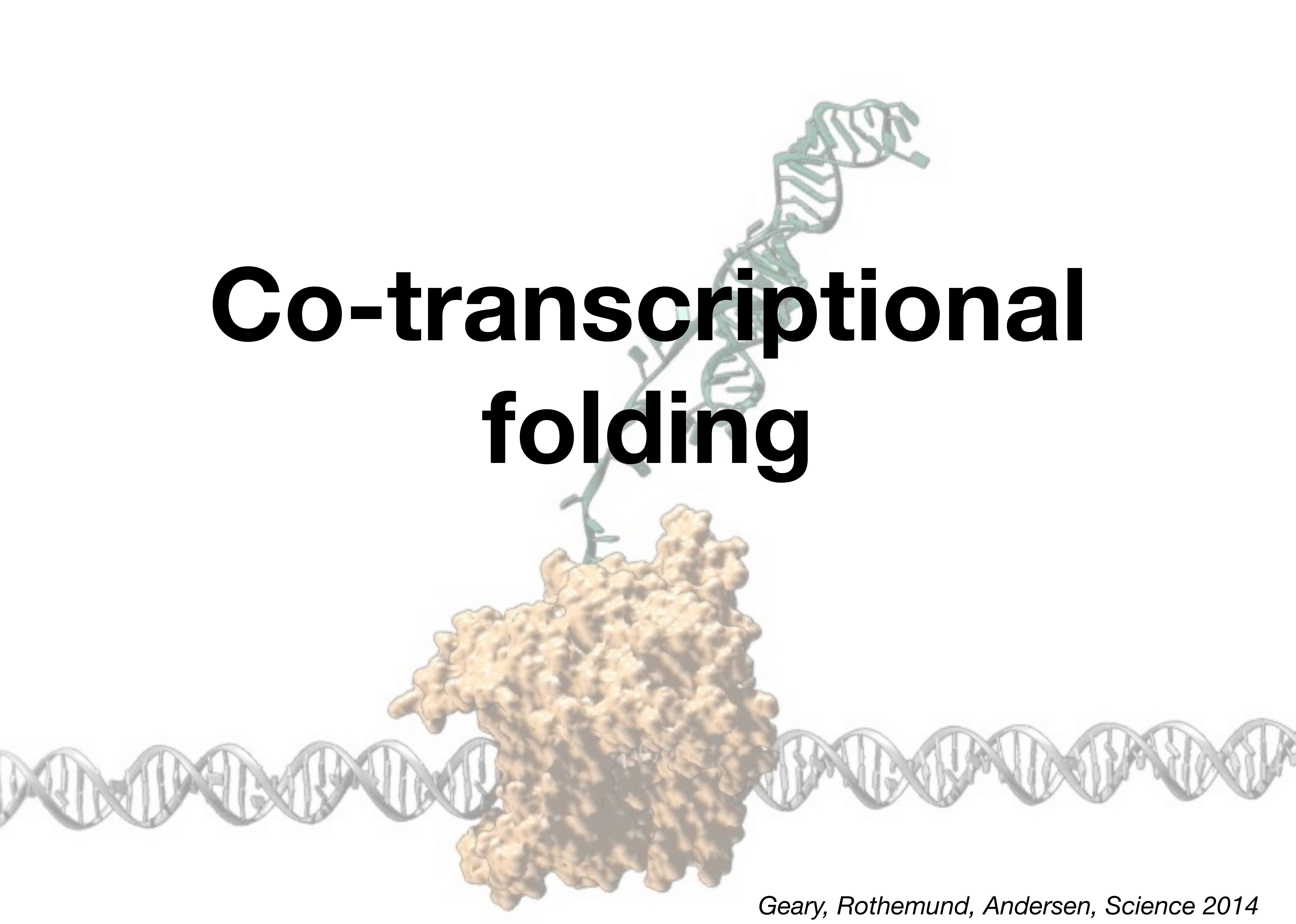


Han et al, Science 2011



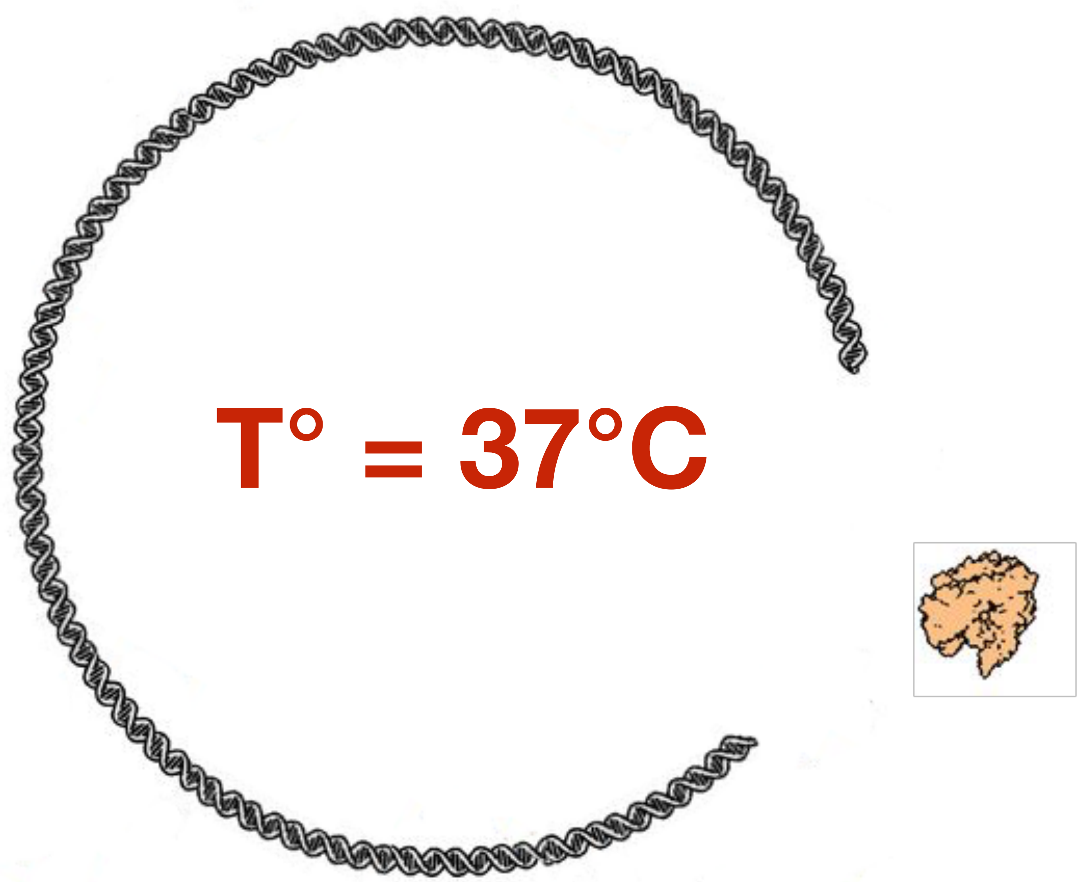
*Rule 110 on input 001 - Woods et al, Nature 2019*

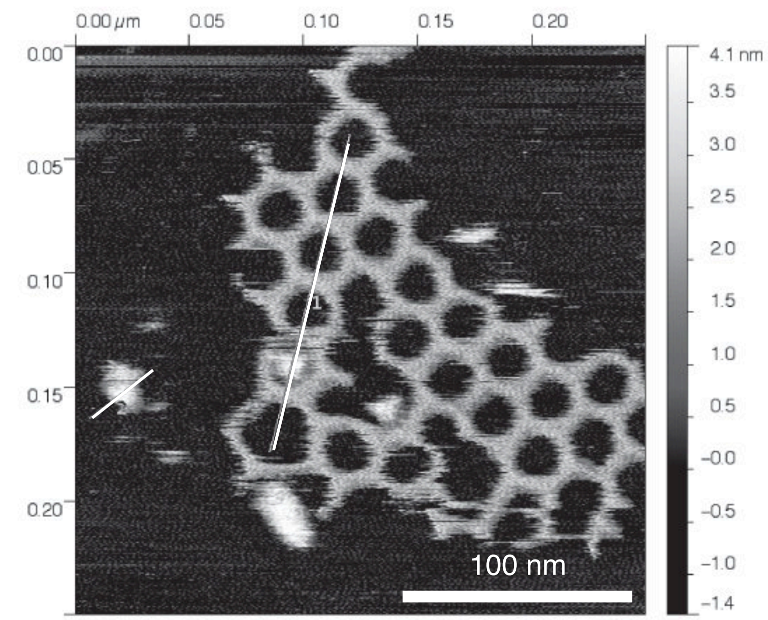
# Co-transcriptional folding

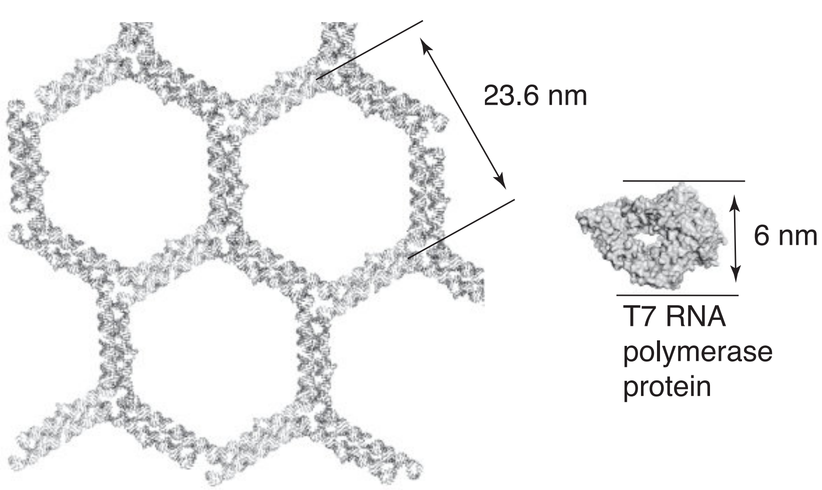


*Geary, Rothemund, Andersen, Science 2014*

# RNA
# co-transcriptional folding

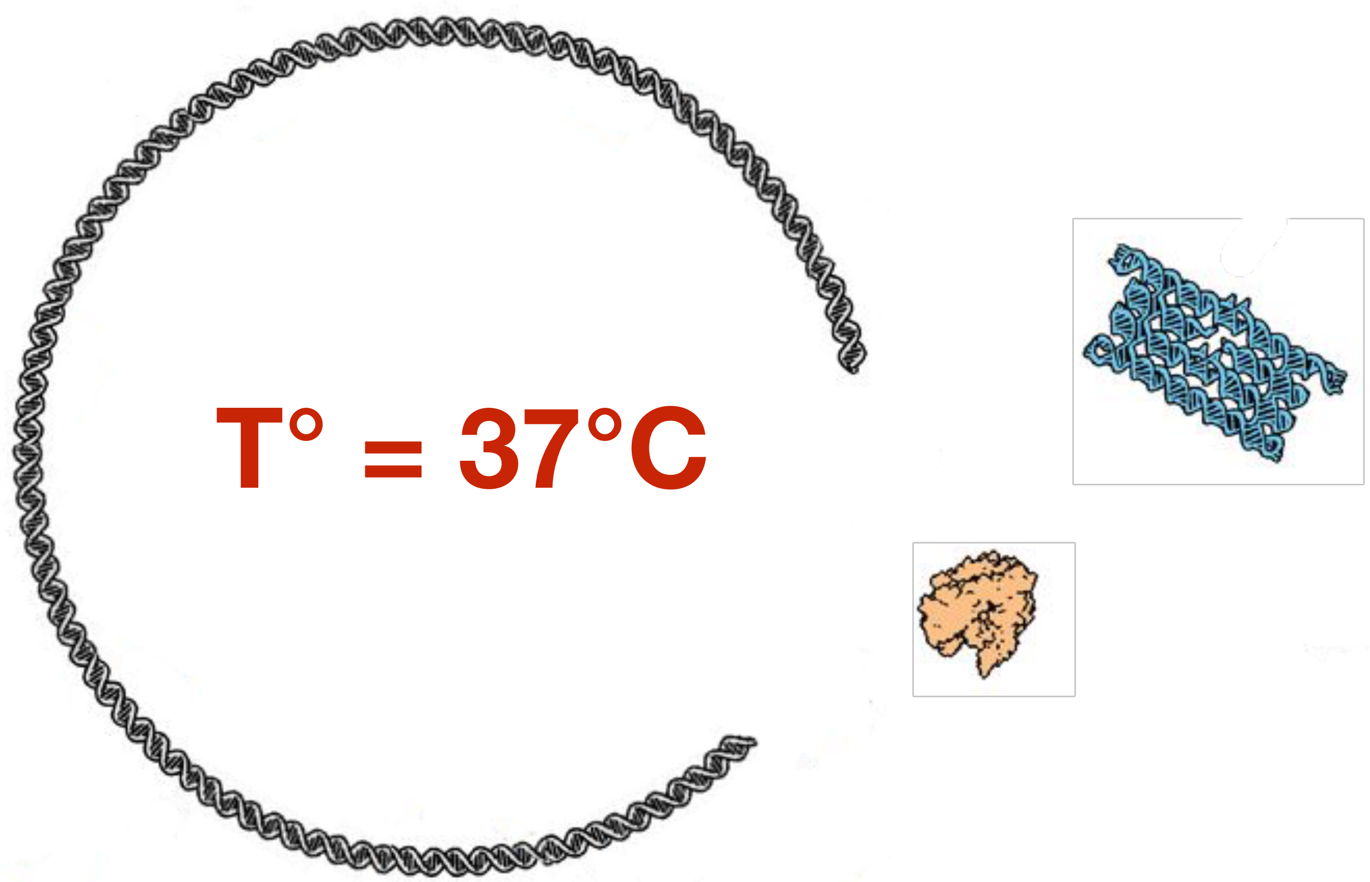


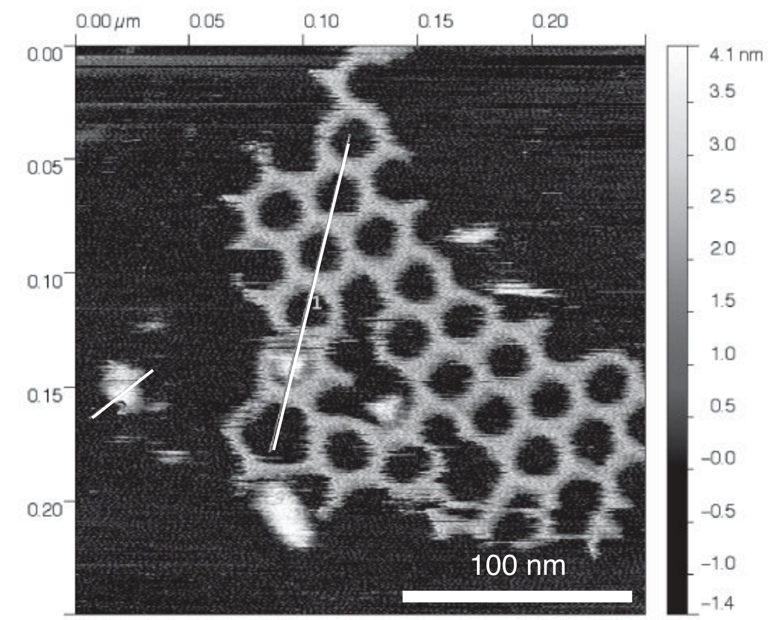



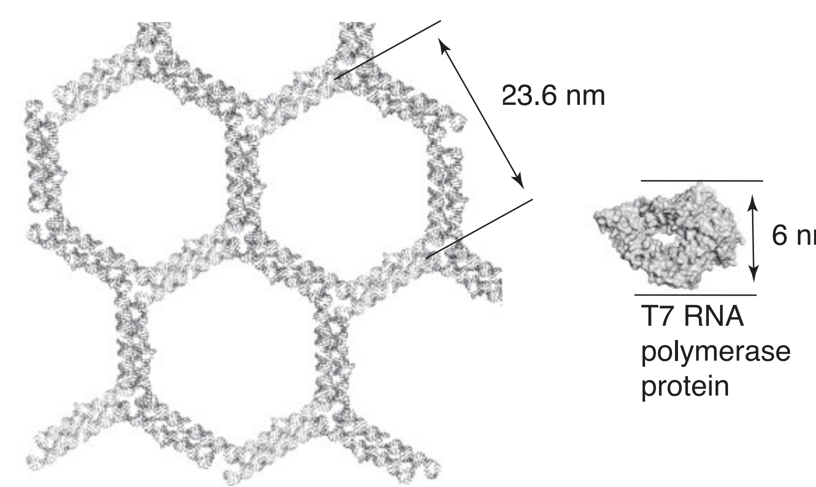




*Geary, Rothemund, Andersen, Science 2014*

# RNA co-transcriptional folding

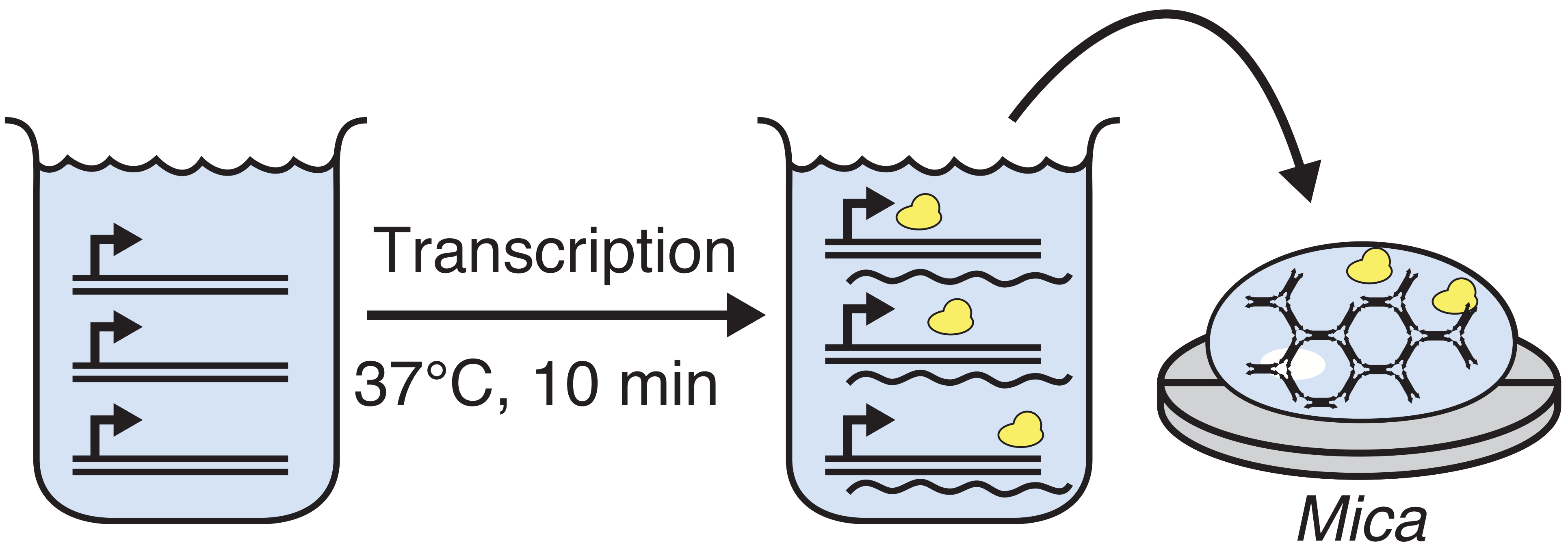


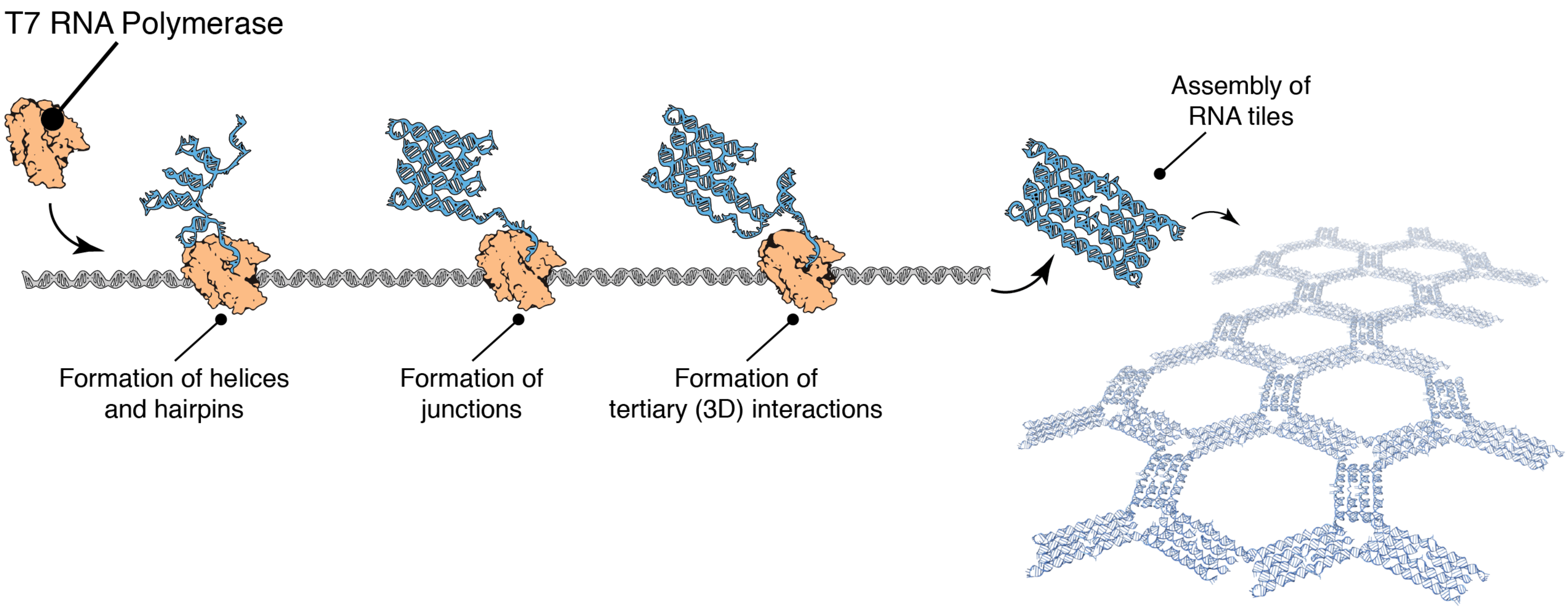




*Geary, Rothemund, Andersen, Science 2014*

# Protocol



# RNA Origami in Real Time



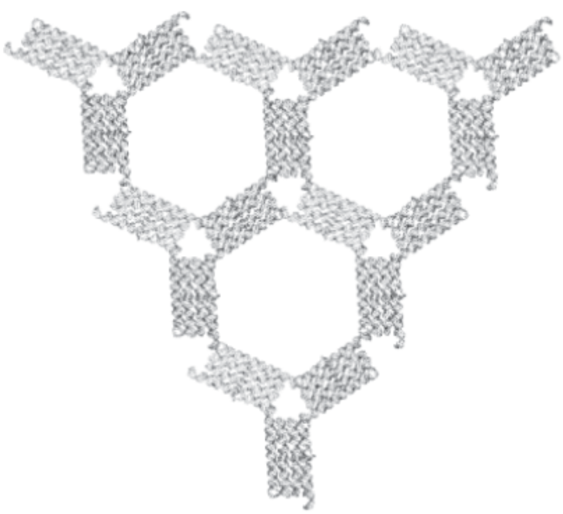

T7 RNA polymerase produces RNA directionally from 5’ to 3’, **at a rate much slower than the RNA folds up (few microseconds).**

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

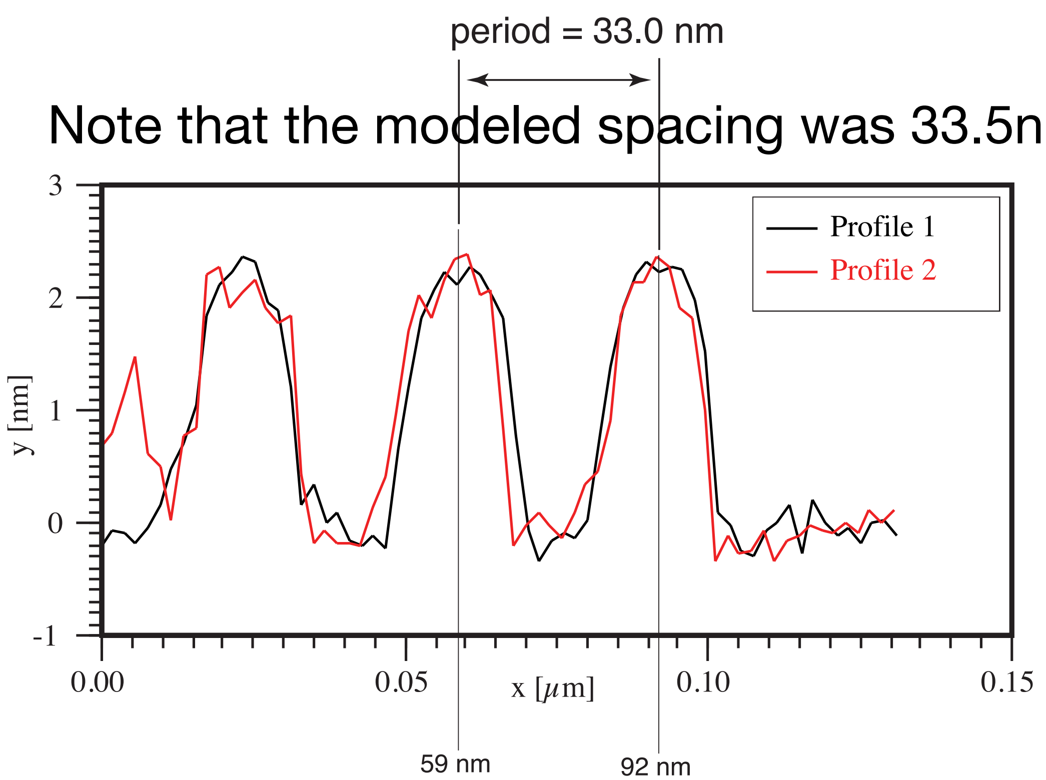
*Slide by Cody Geary*

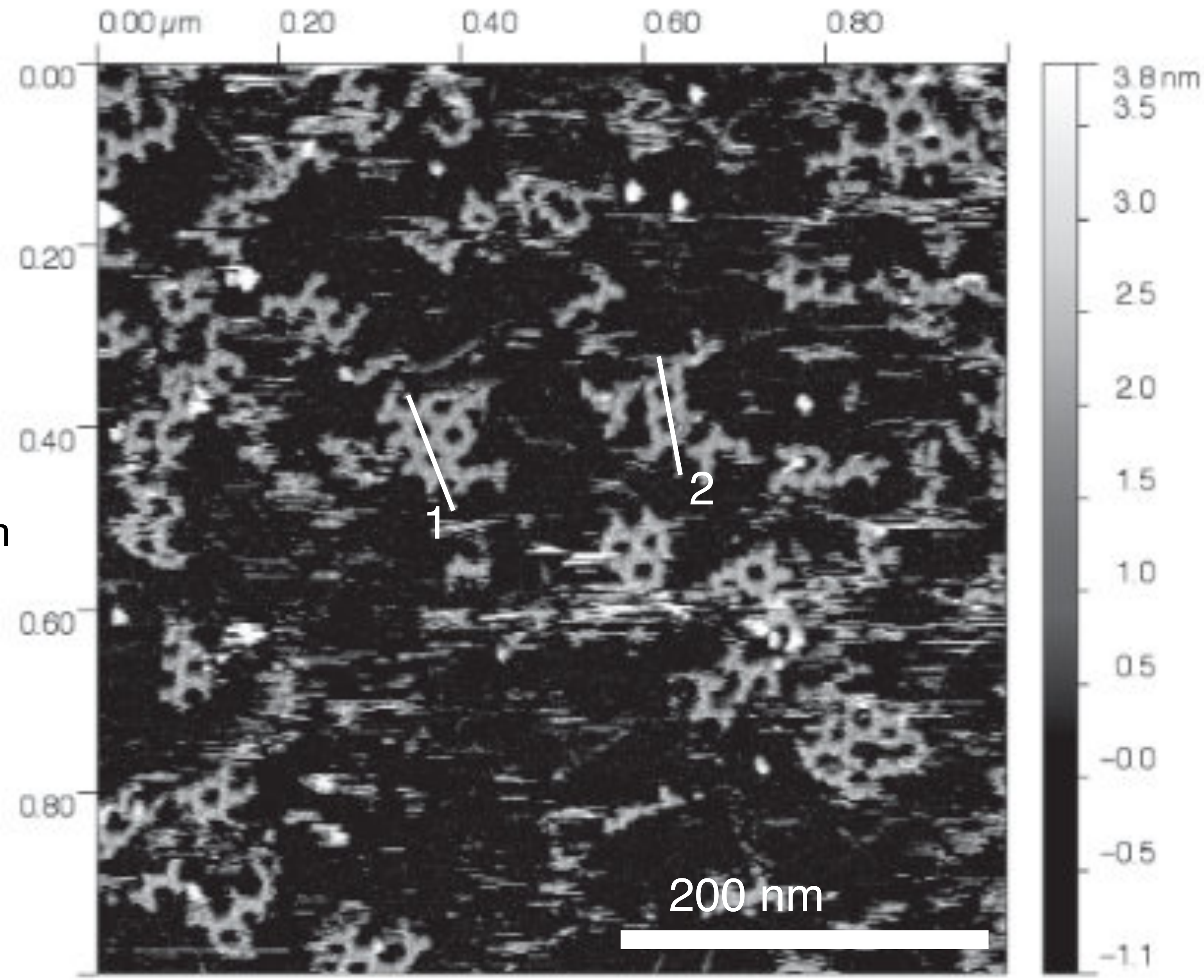
*Westhof and Leontis (Science, 2014)*

# AFM imaging of 4H-AE co-transcriptional assembly



Note that the modeled spacing was 33.5nm

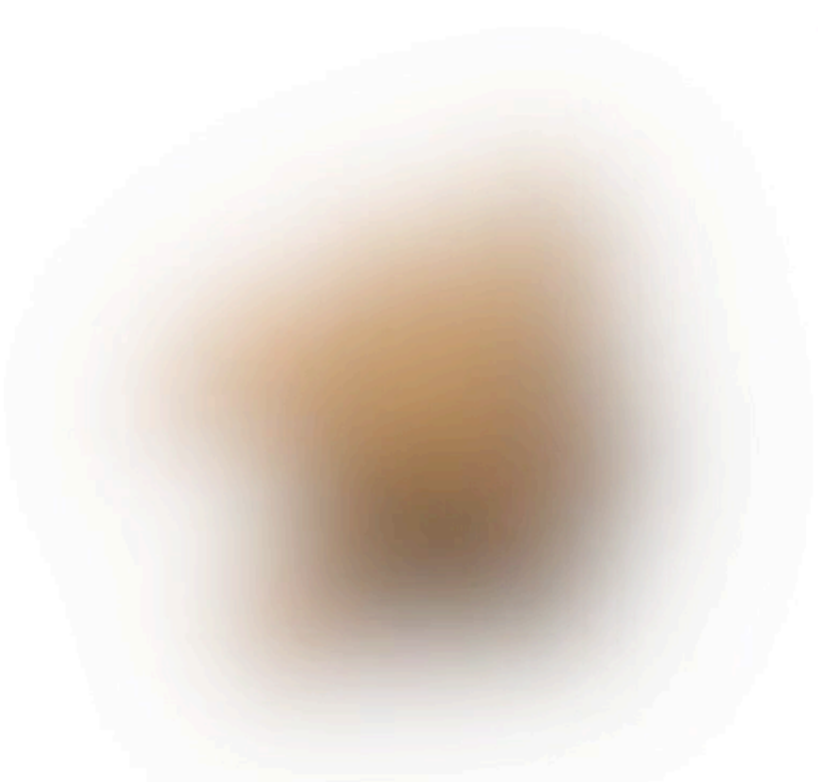



*Geary et al (2014) Science*

# RNA Folding

## (Real time: ~1 second)



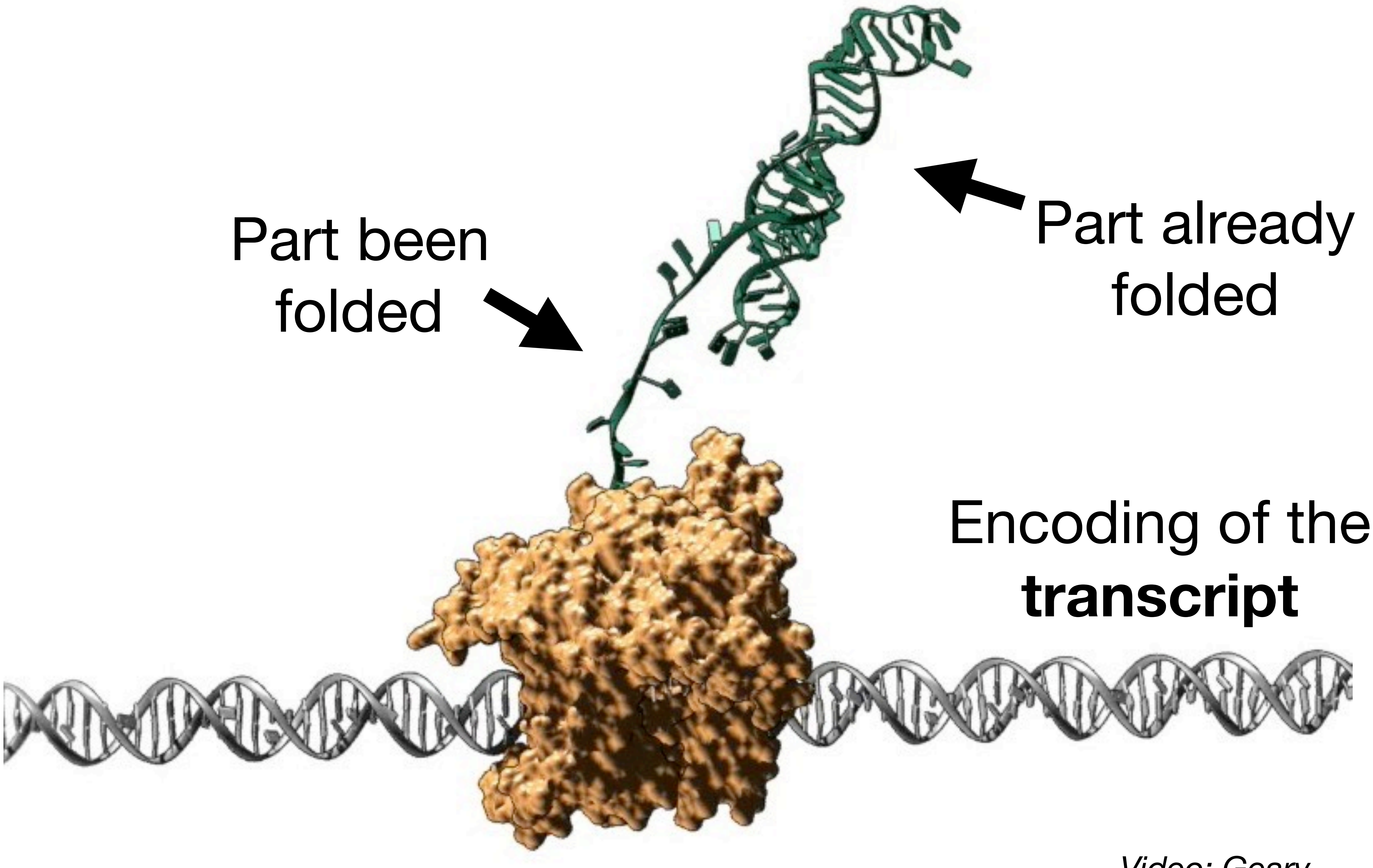
*Video: Geary*

# **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:**

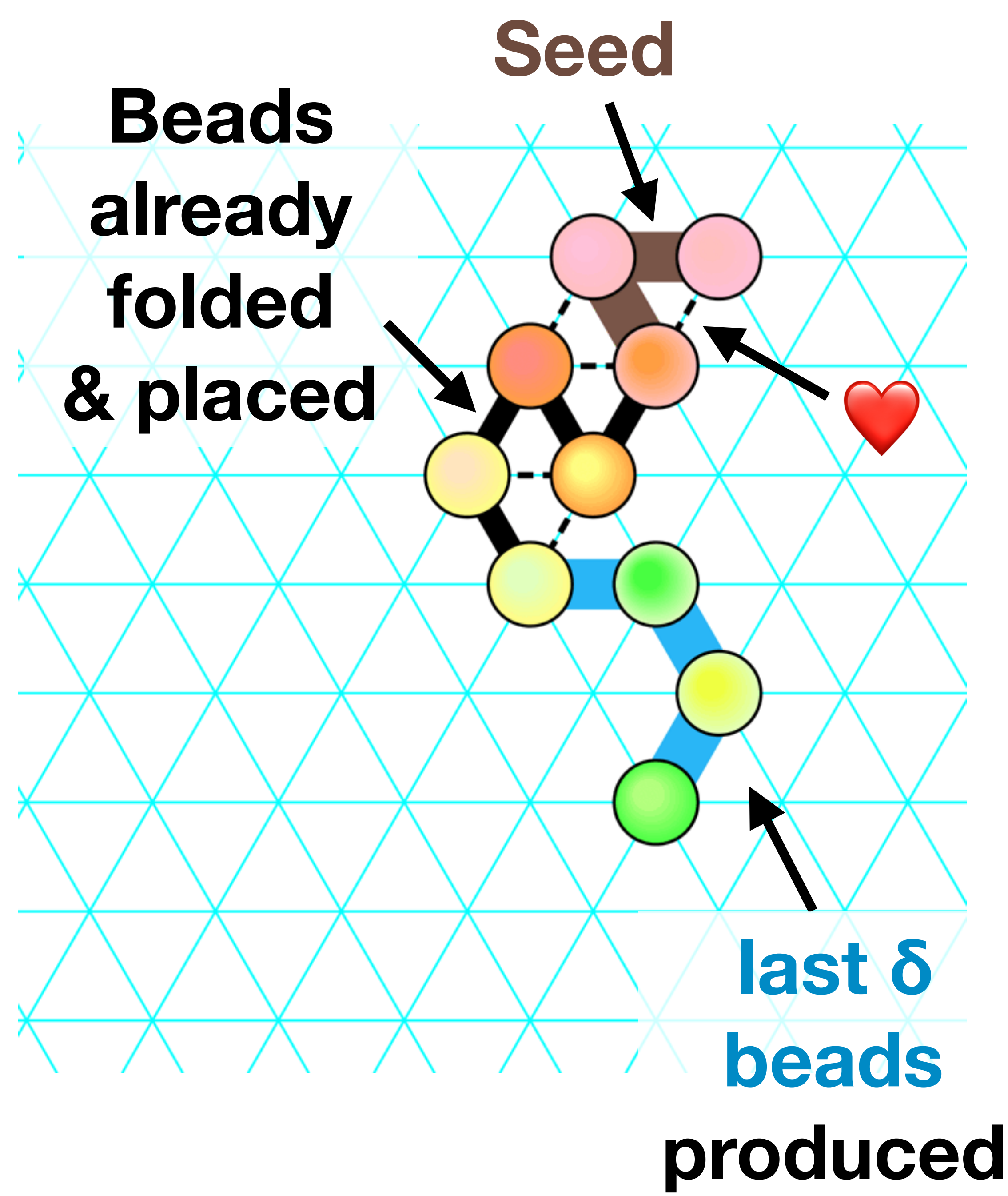
- a **sequence** of **bead types** (the **transcript**)

**The instructions:**

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

**The input configuration:**

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





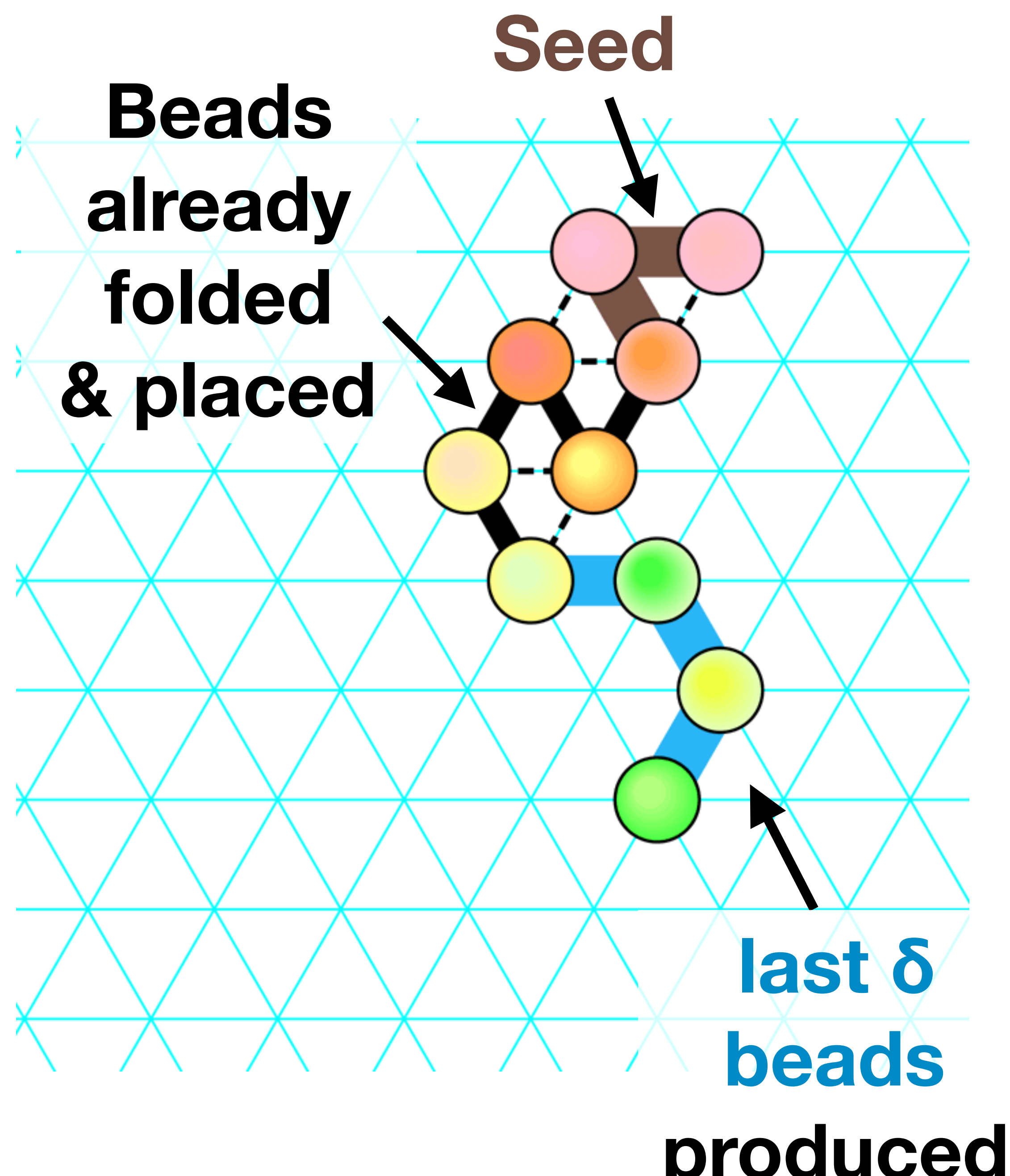

*Geary, Meunier, Schabanel, Seki MFCS 2016*

# 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**

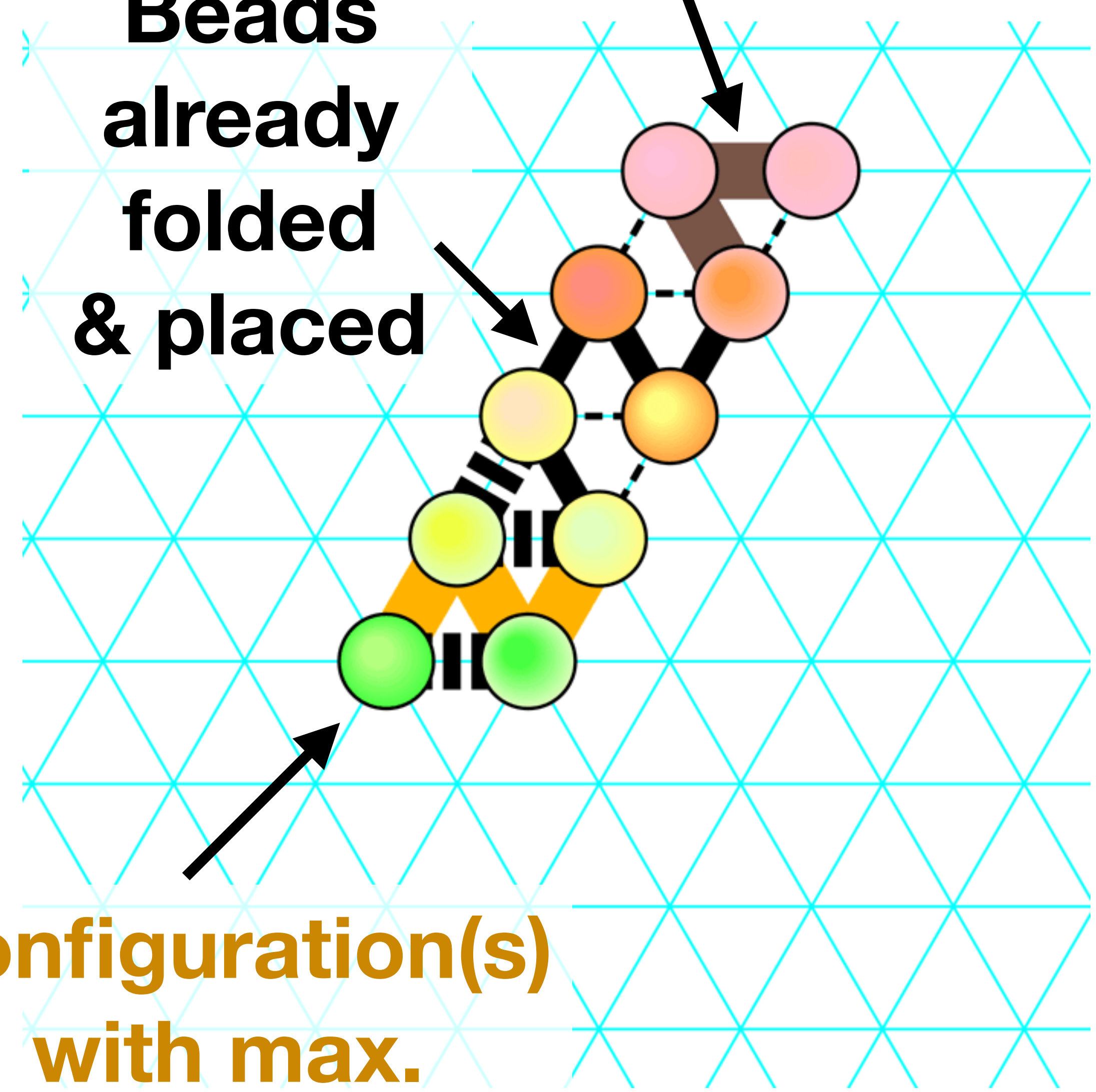



*Geary, Meunier, Schabanel, Seki MFCS 2016*

# 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



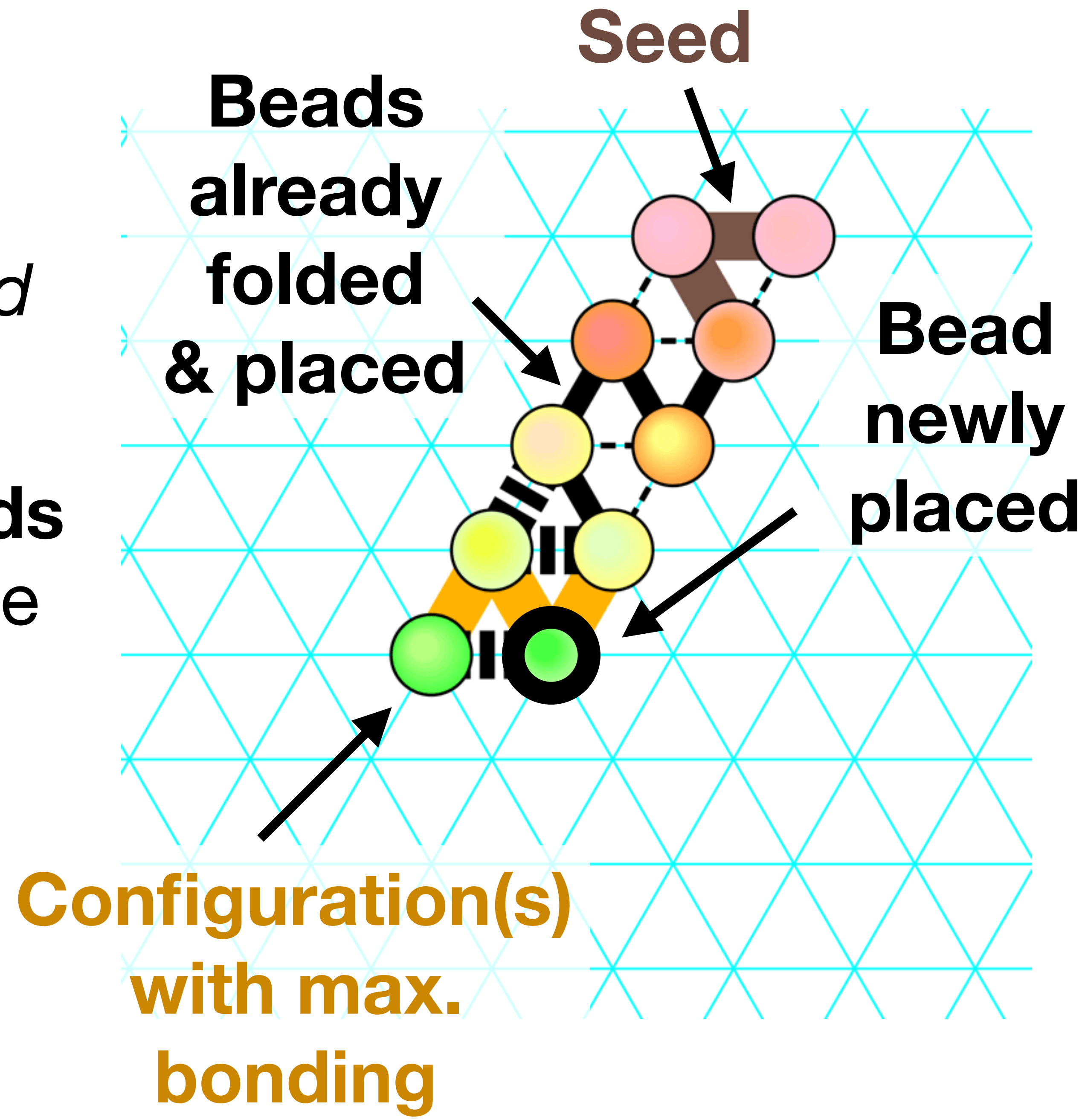



*Geary, Meunier, Schabanel, Seki MFCS 2016*

# 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



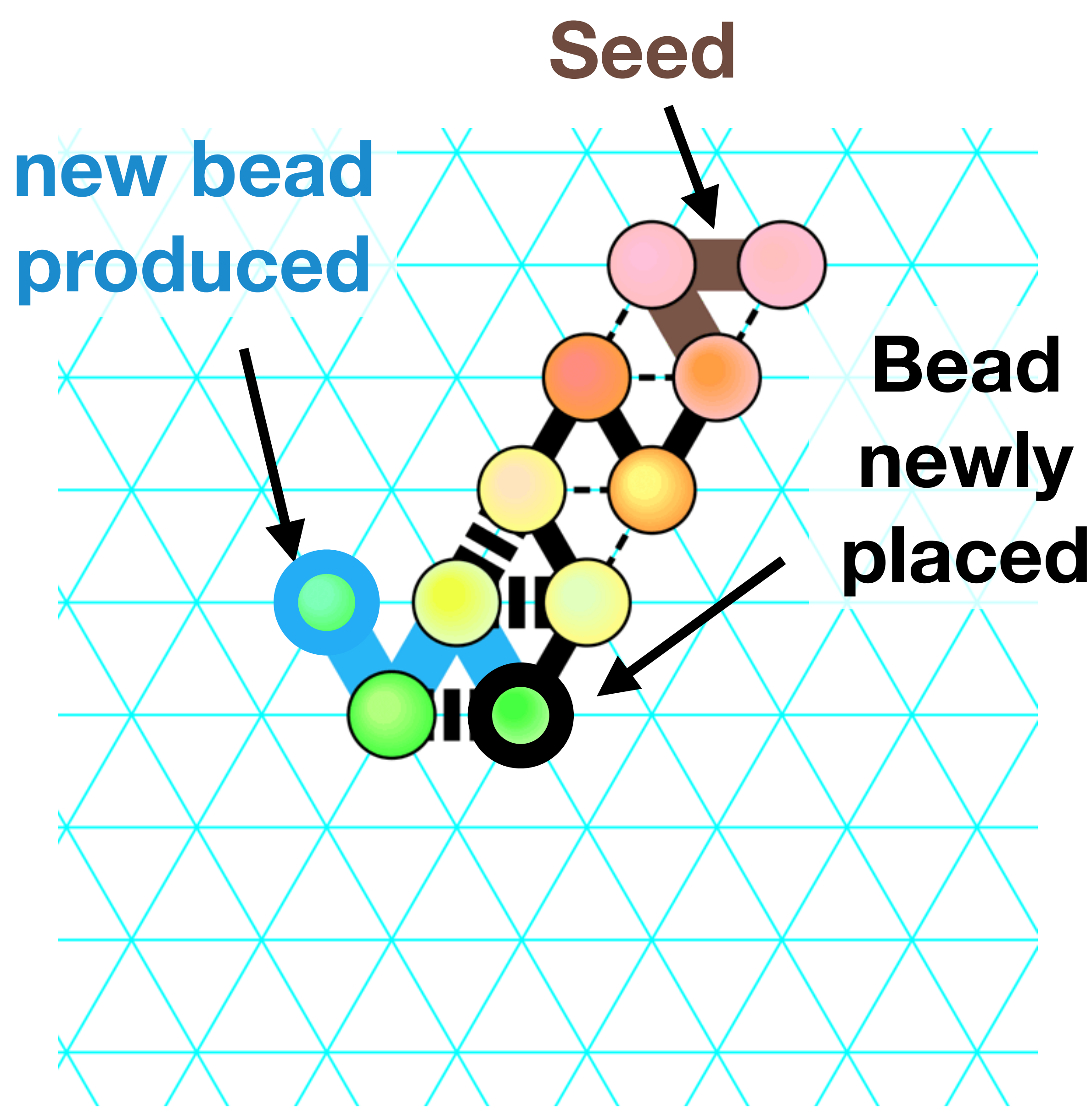



*Geary, Meunier, Schabanel, Seki MFCS 2016*

# 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



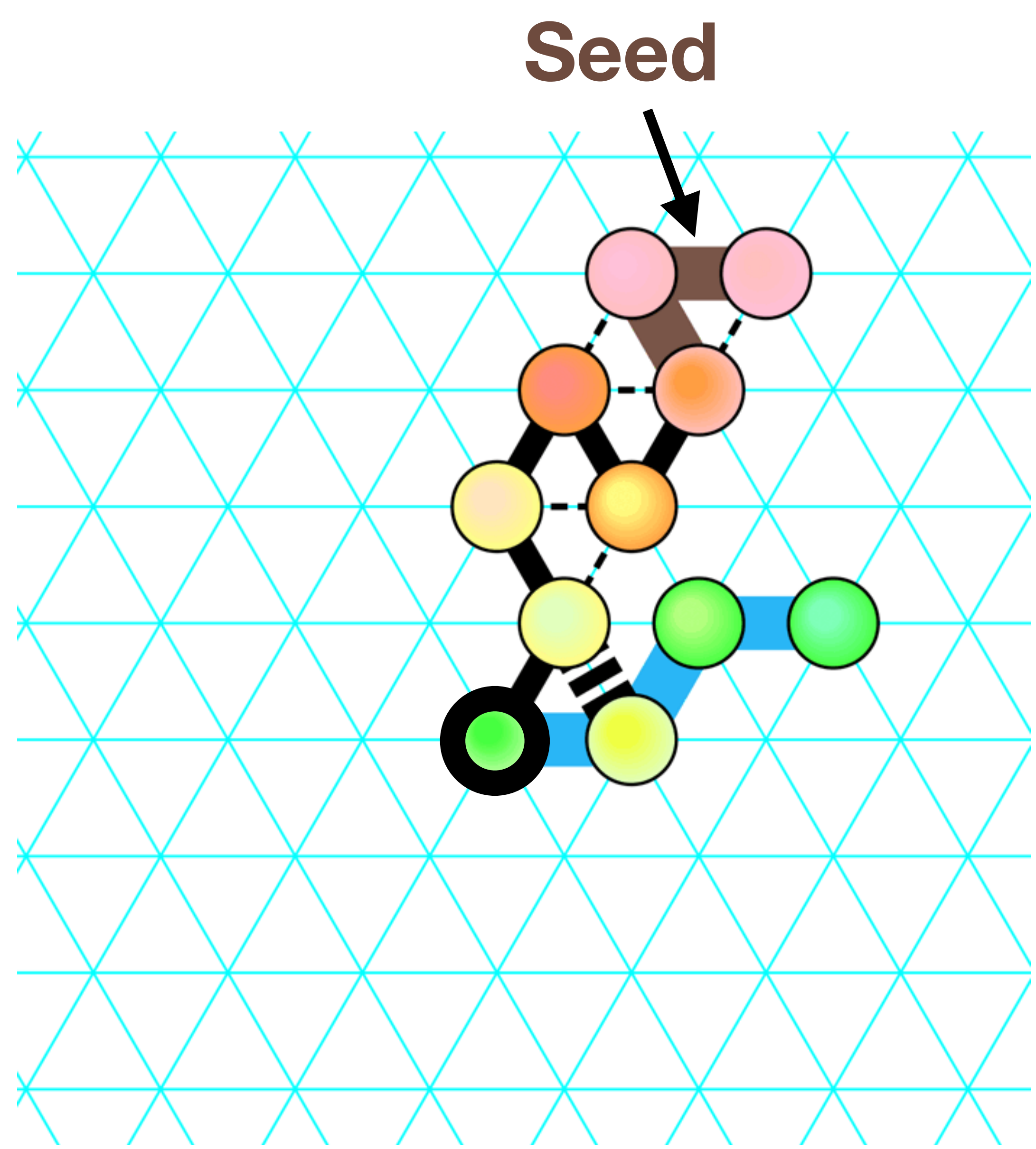



*Geary, Meunier, Schabanel, Seki MFCS 2016*

# 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



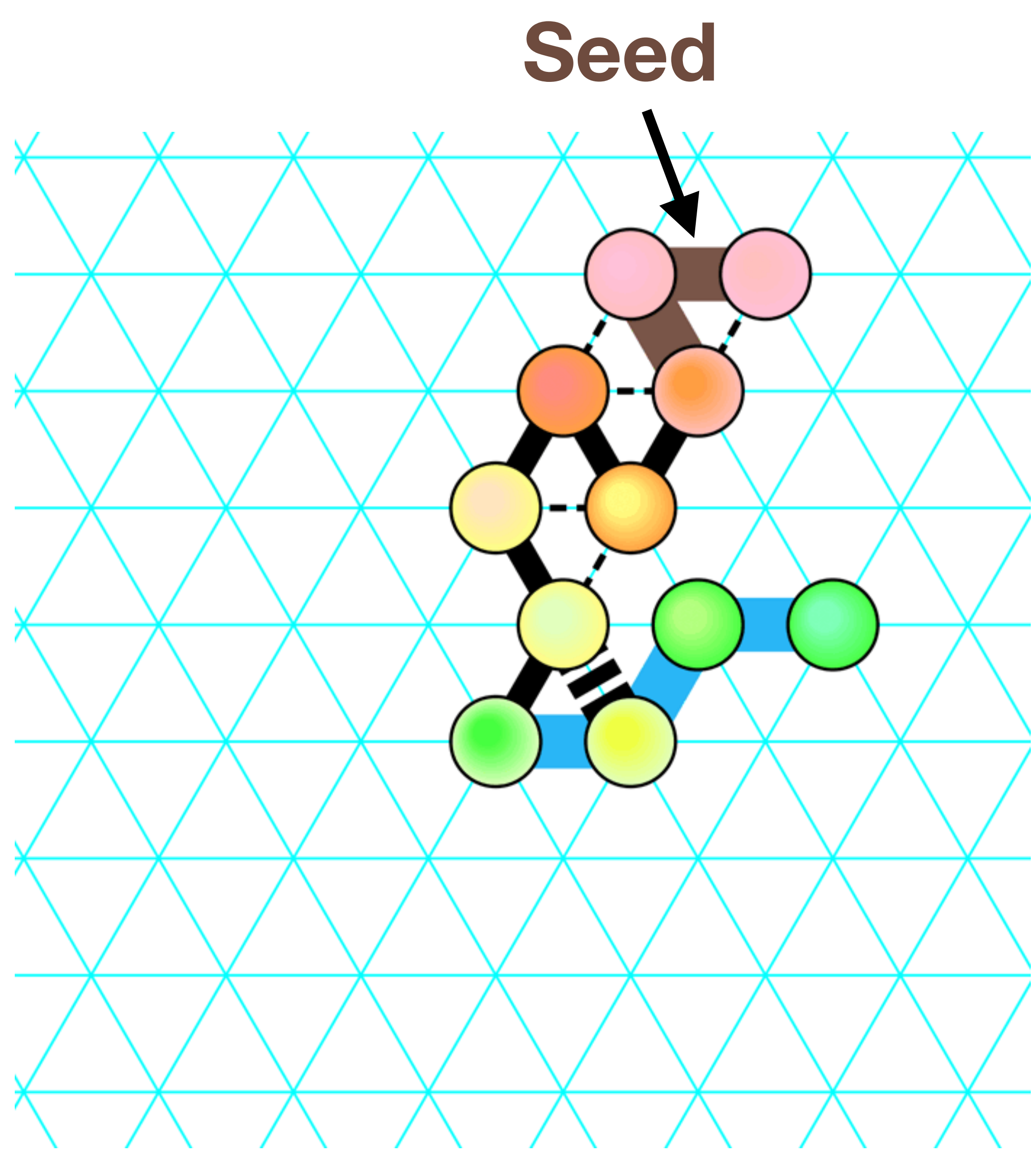



*Geary, Meunier, Schabanel, Seki MFCS 2016*

# 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

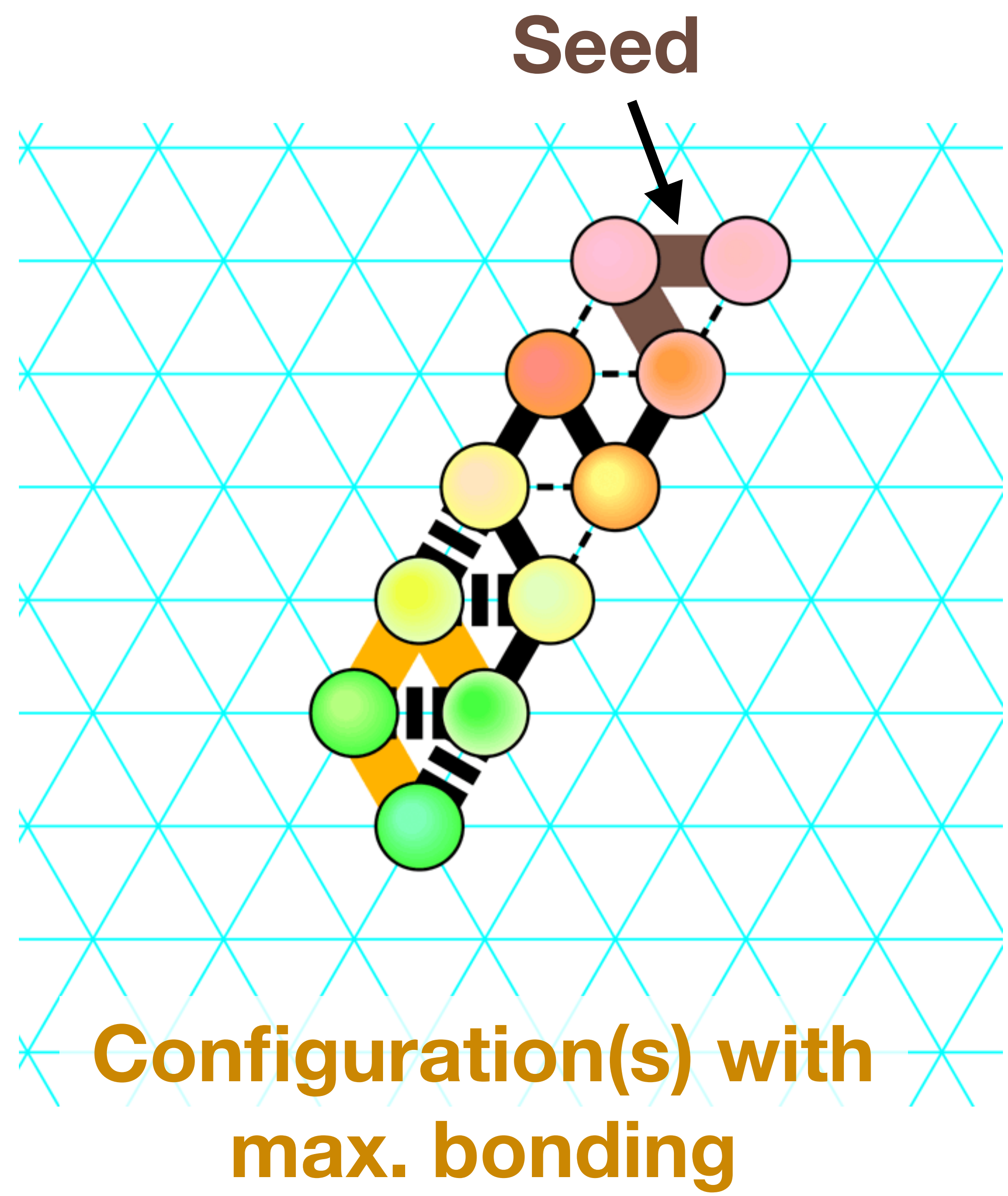



*Geary, Meunier, Schabanel, Seki MFCS 2016*

# 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



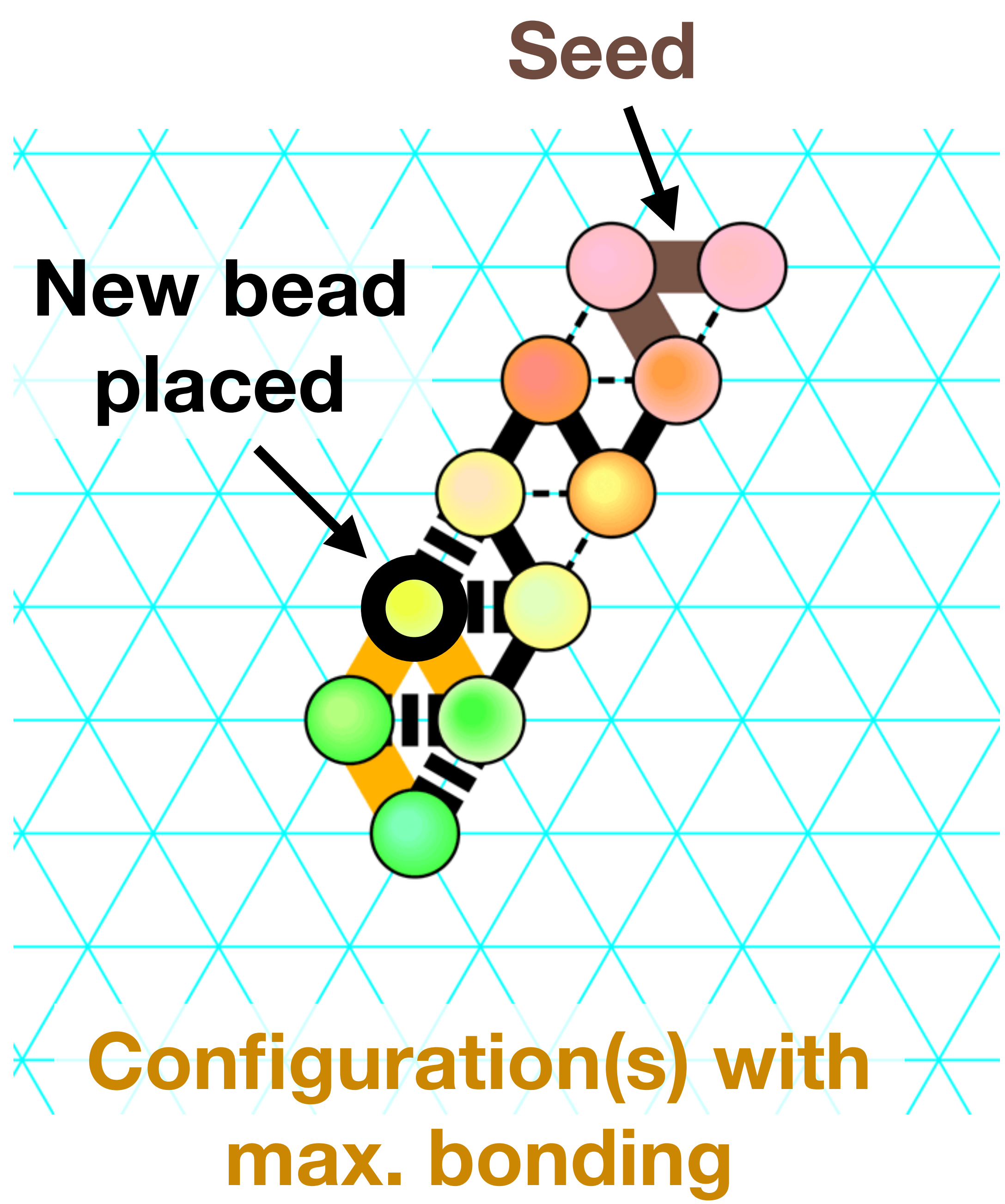



*Geary, Meunier, Schabanel, Seki MFCS 2016*

# 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 <u>several</u> configurations with max. bonding**

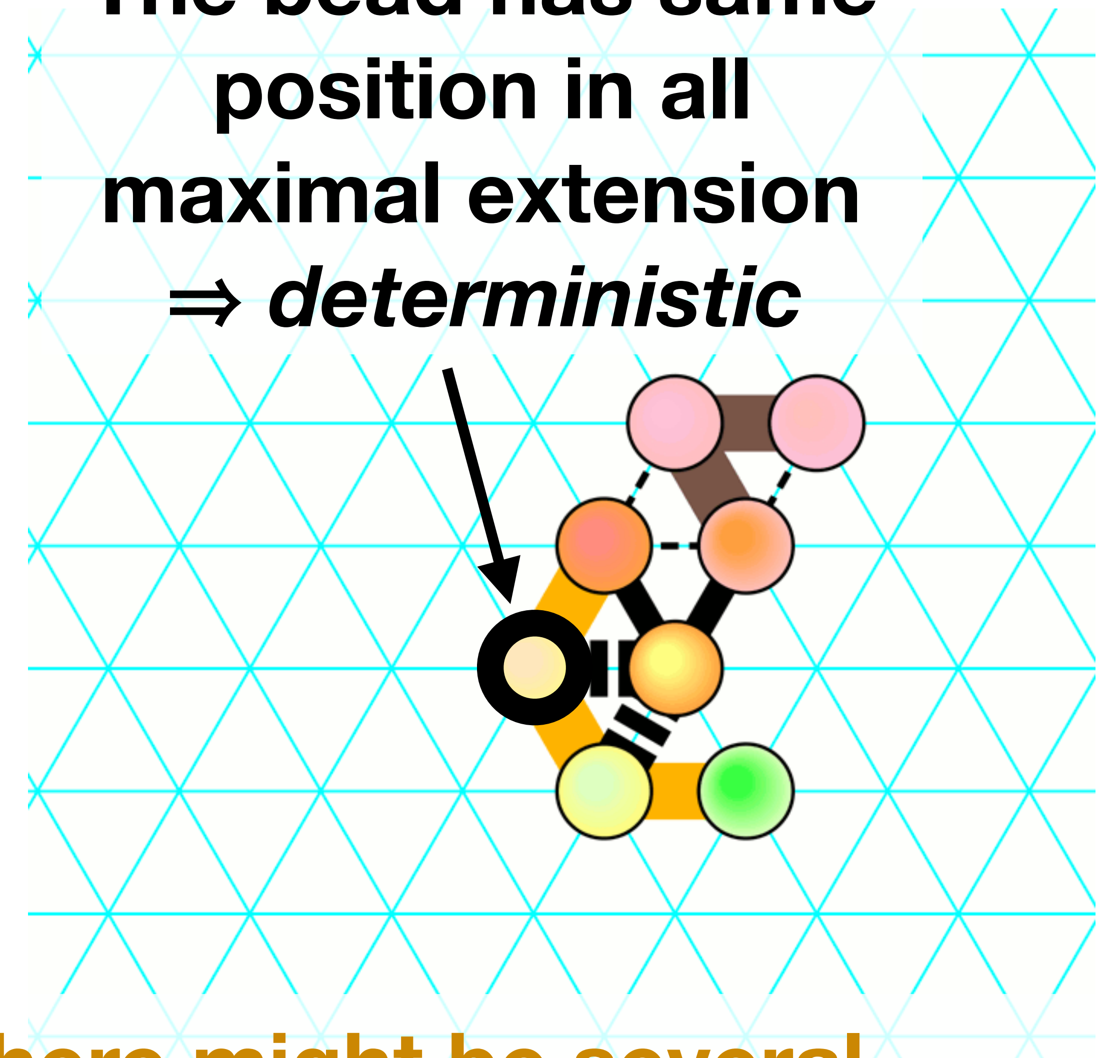
*Geary, Meunier, Schabanel, Seki MFCS 2016*

# 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***

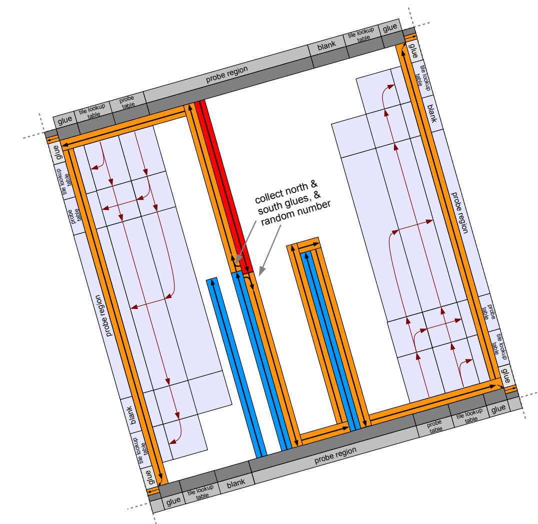


**There might be several configurations with max. bonding**



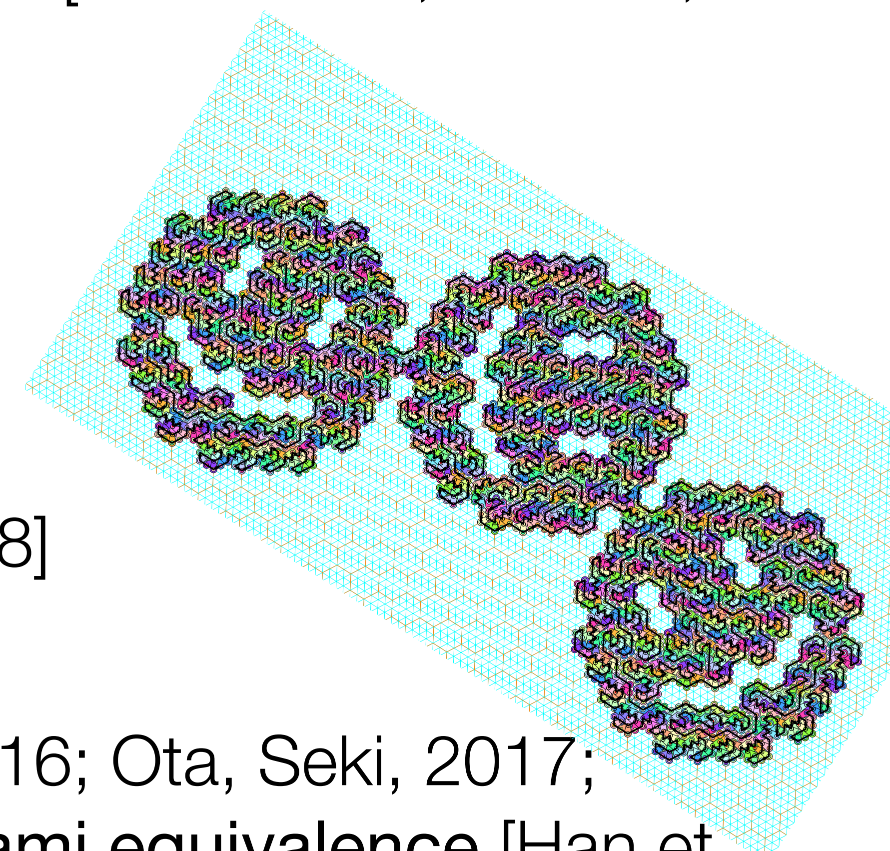
*Geary, Meunier, Schabanel, Seki MFCS 2016*

# Previous work



**Some self-assembly seminal work**

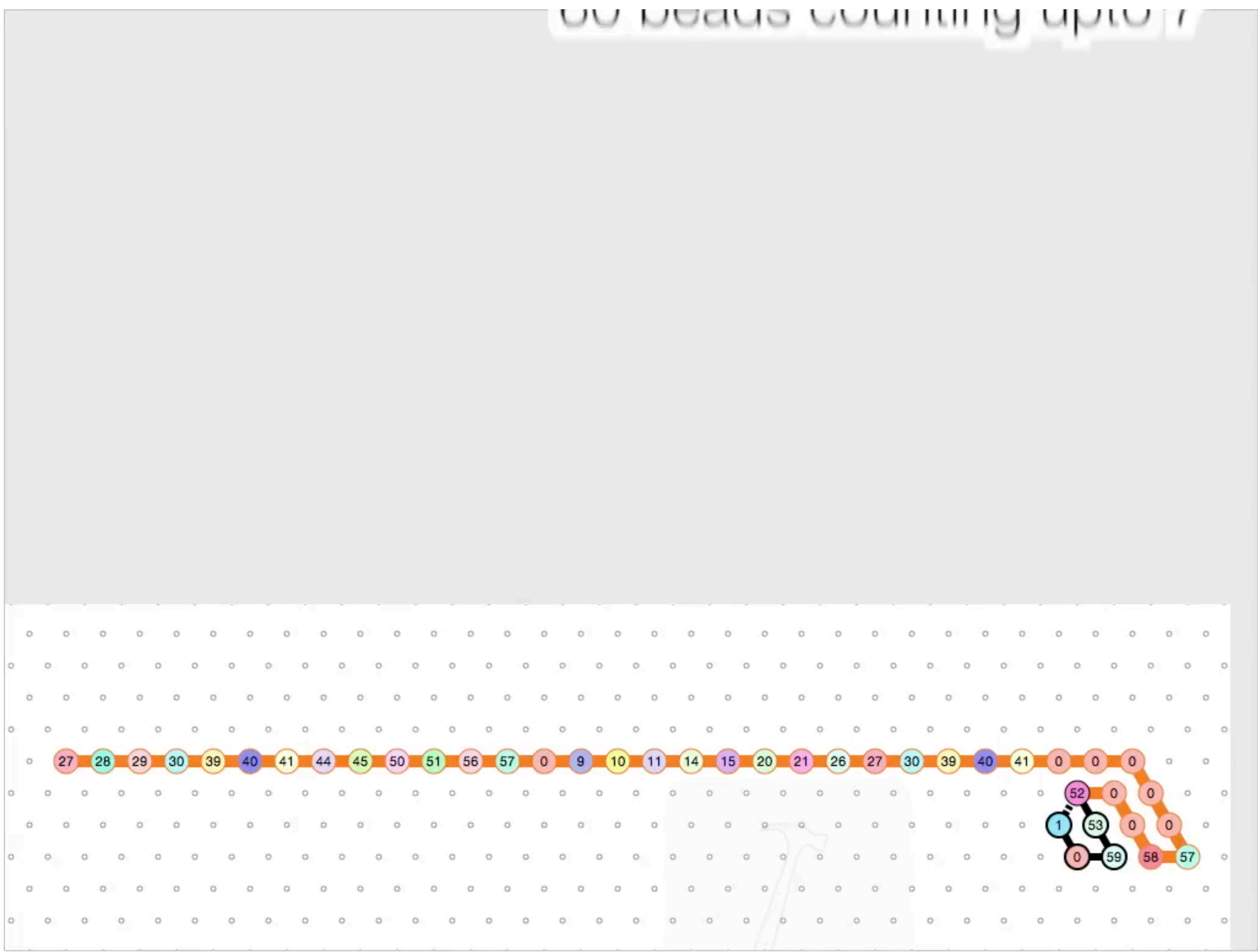
- **Tile assembly systems are Turing universal** [Winfree, 1998]
- **Arbitrary shape assembly with optimal tile set size** [Soloveichik, Winfree, 2007]
- **Intrinsic universality** [Doty et al, 2012]



**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]

# Oritatami
## A first example



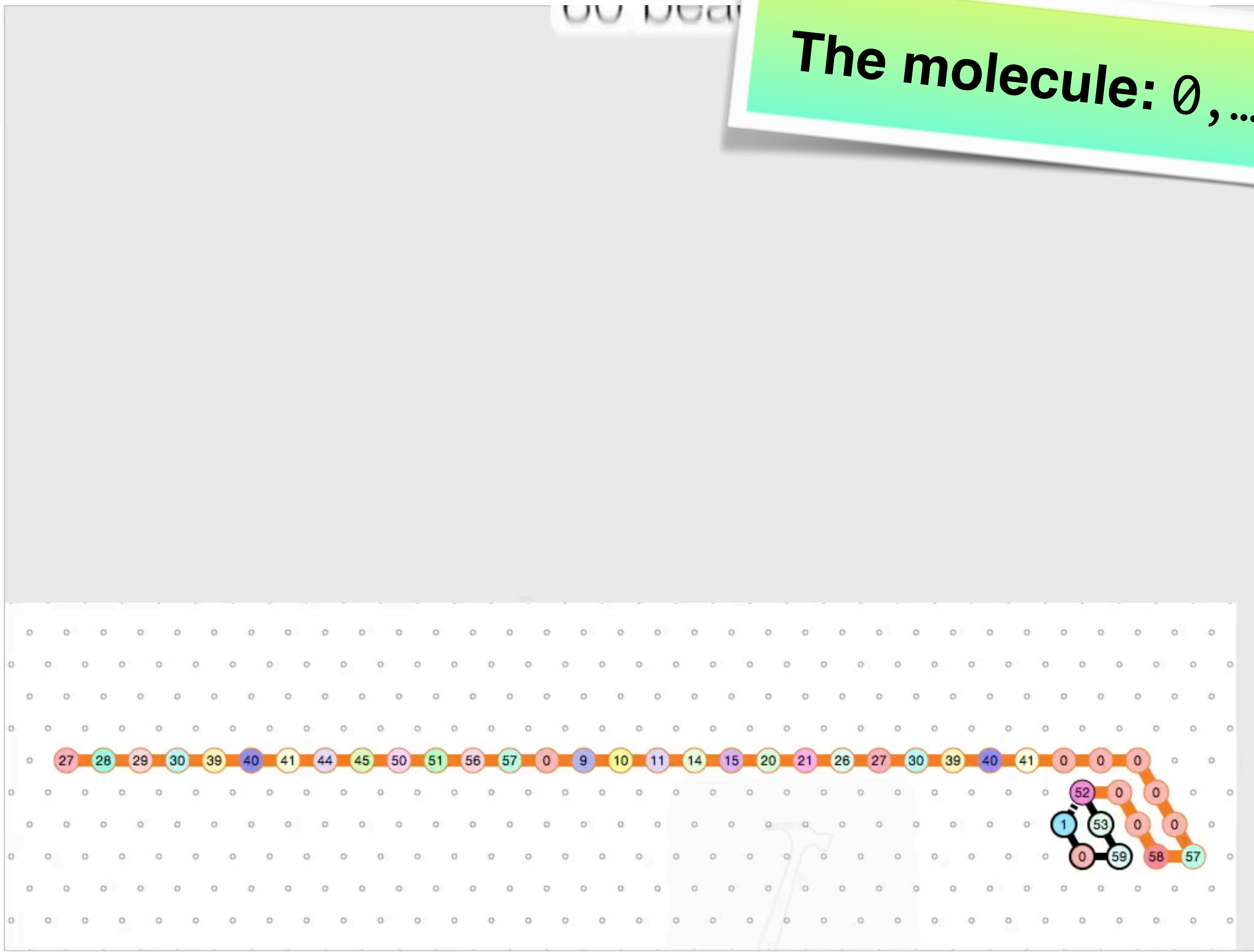

A binary counter made of a **single** 60-beads periodic molecule folding upon itself

*Geary Meunier Seki Schabanel 2015*

# Oritatami

## A first example

**The molecule:** `0,…,59,0,…,59,0,…`



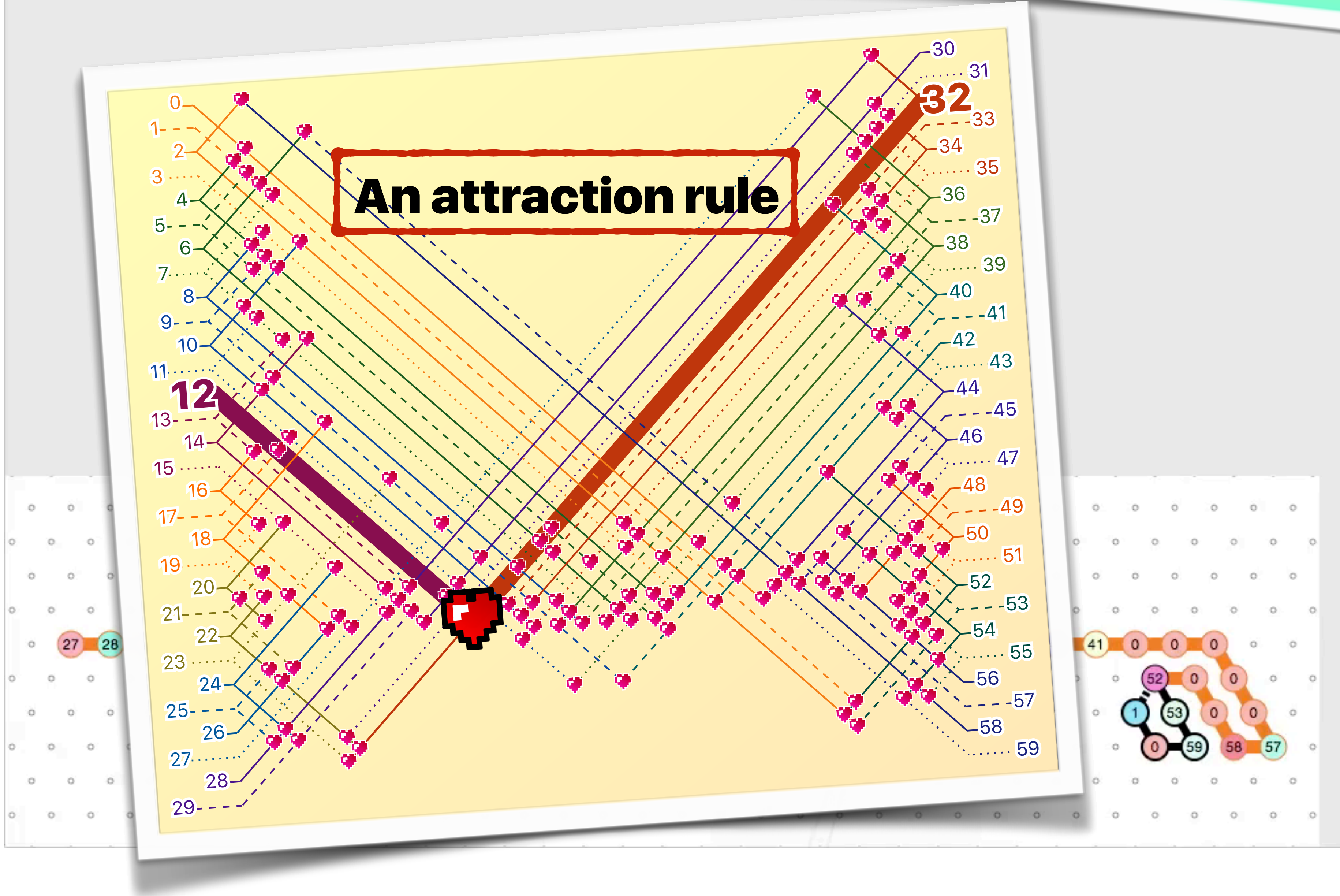

of a **single** 6…
beads
periodic
molecule
folding upon
itself

*Geary Meunier Seki Schabanel 2015*

# Oritatami

## A first example

**The molecule:** `0,…,59,0,…,59,0,…`

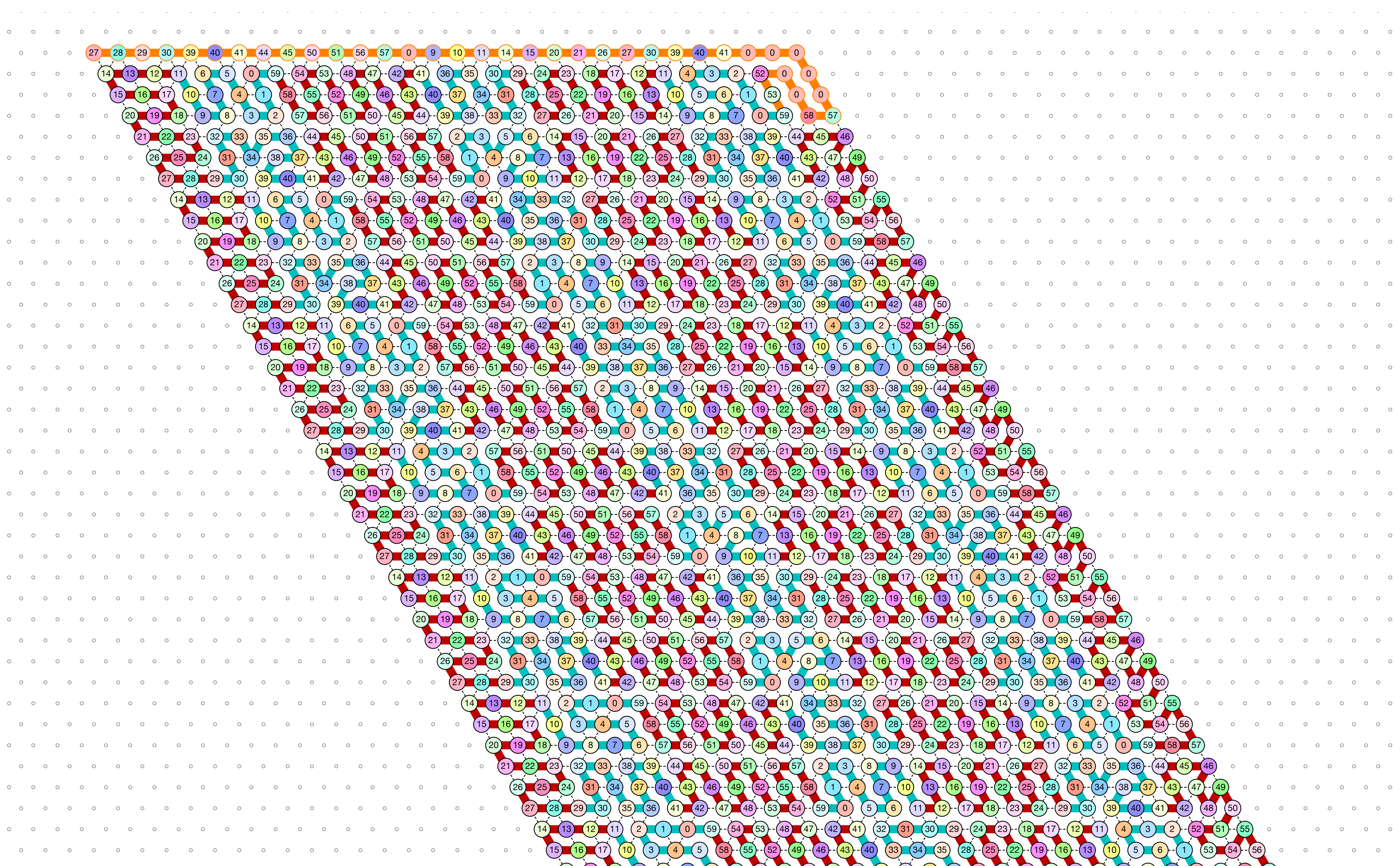



of a **single** 60-
beads
periodic
molecule
folding upon
itself

*Geary Meunier Seki Schabanel 2015*

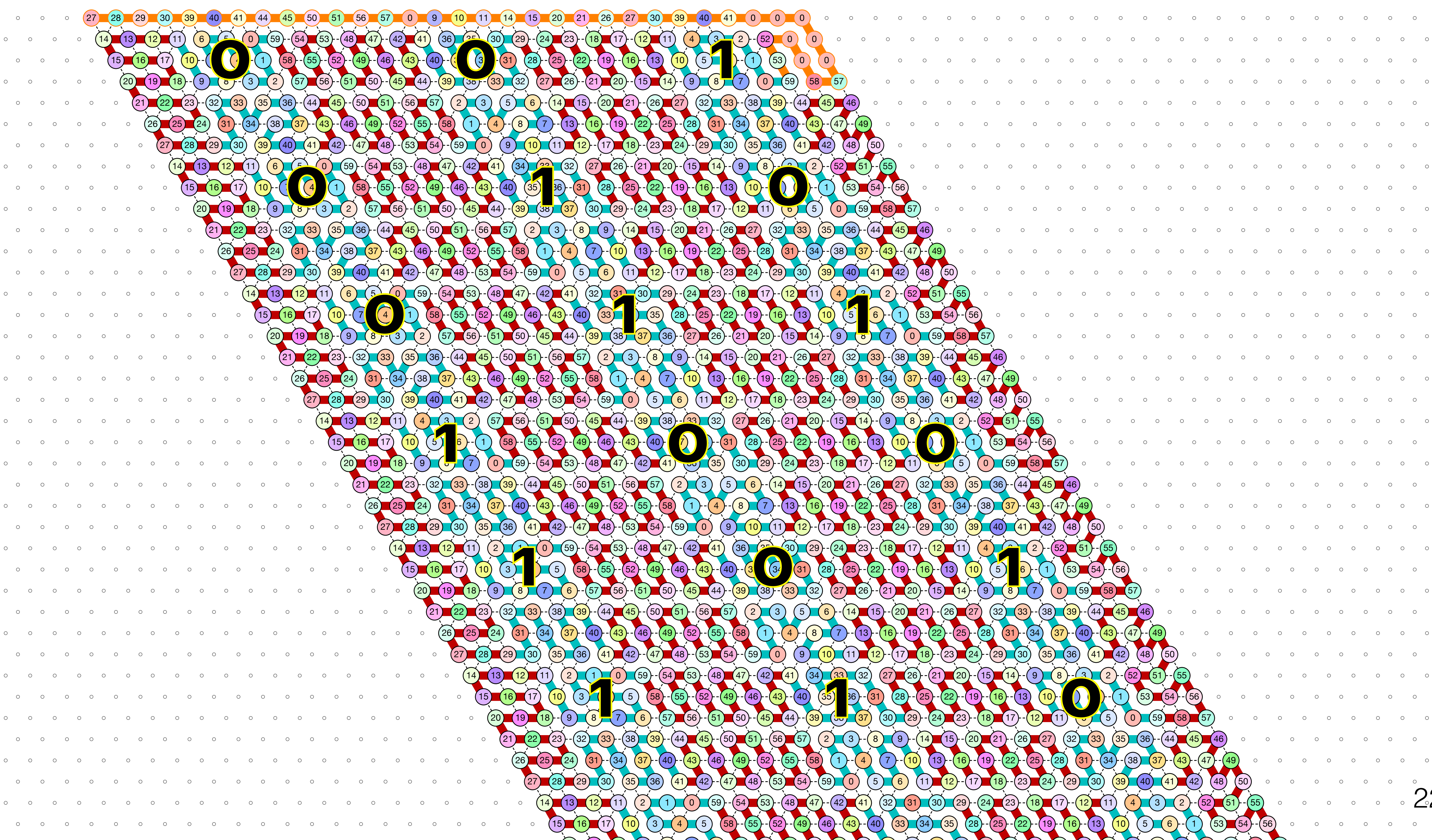
# **Oritatami.** A binary counter

Information is encoded in the geometry



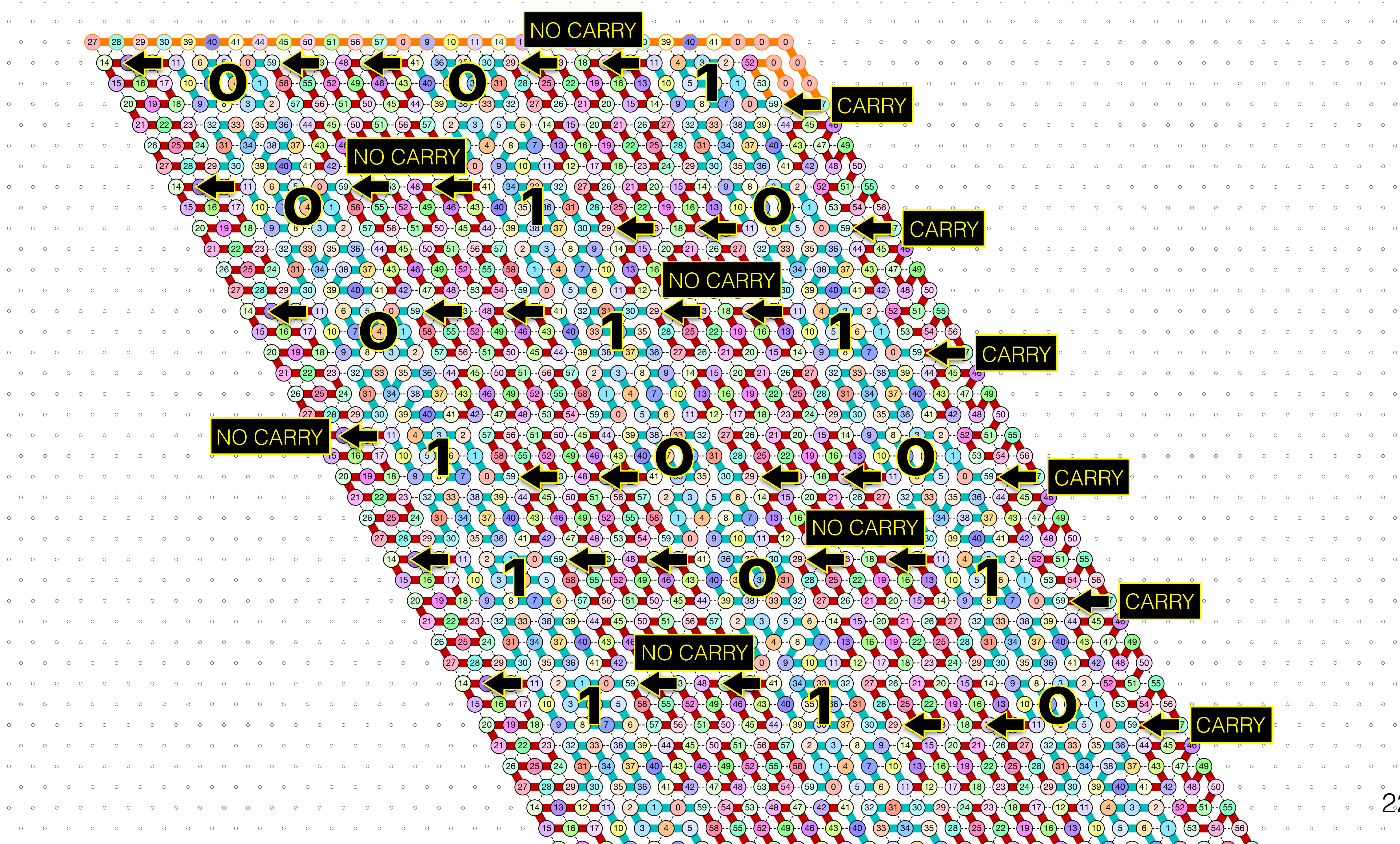
# **Oritatami.** A binary counter

Information is encoded in the geometry



# **Oritatami.** A binary counter

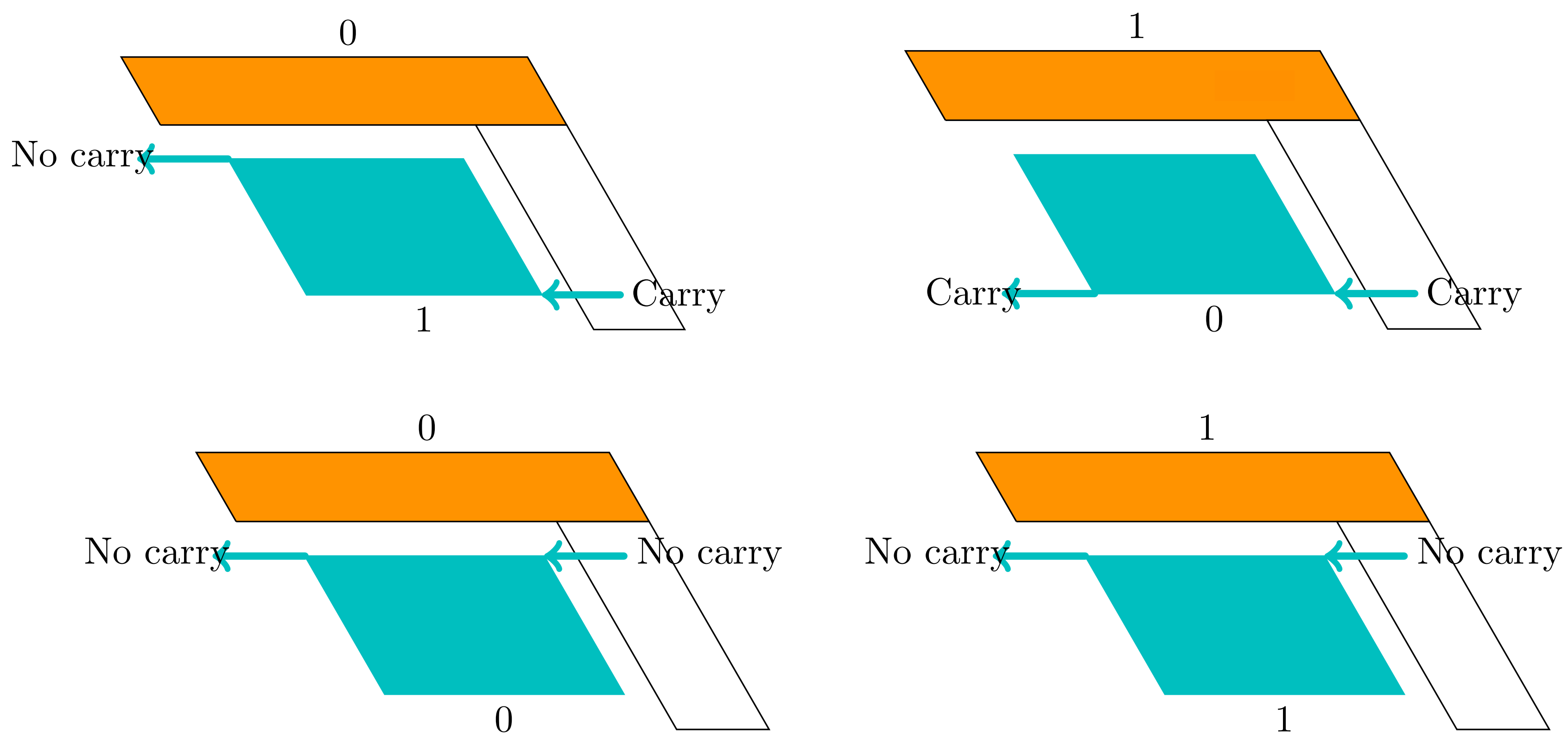
Information is encoded in the geometry



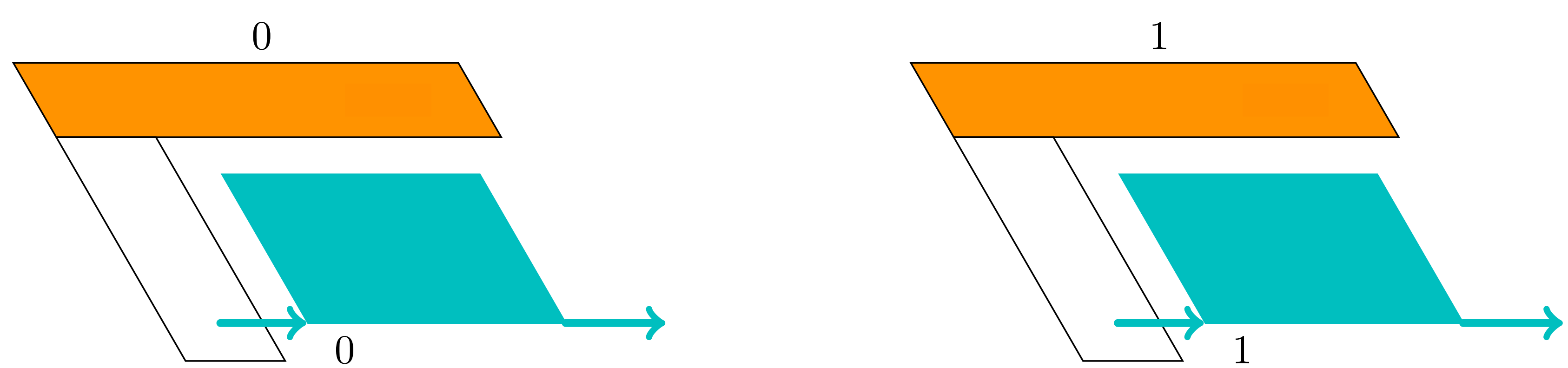
# **Oritatami.** A binary counter

## Information is encoded in the geometry

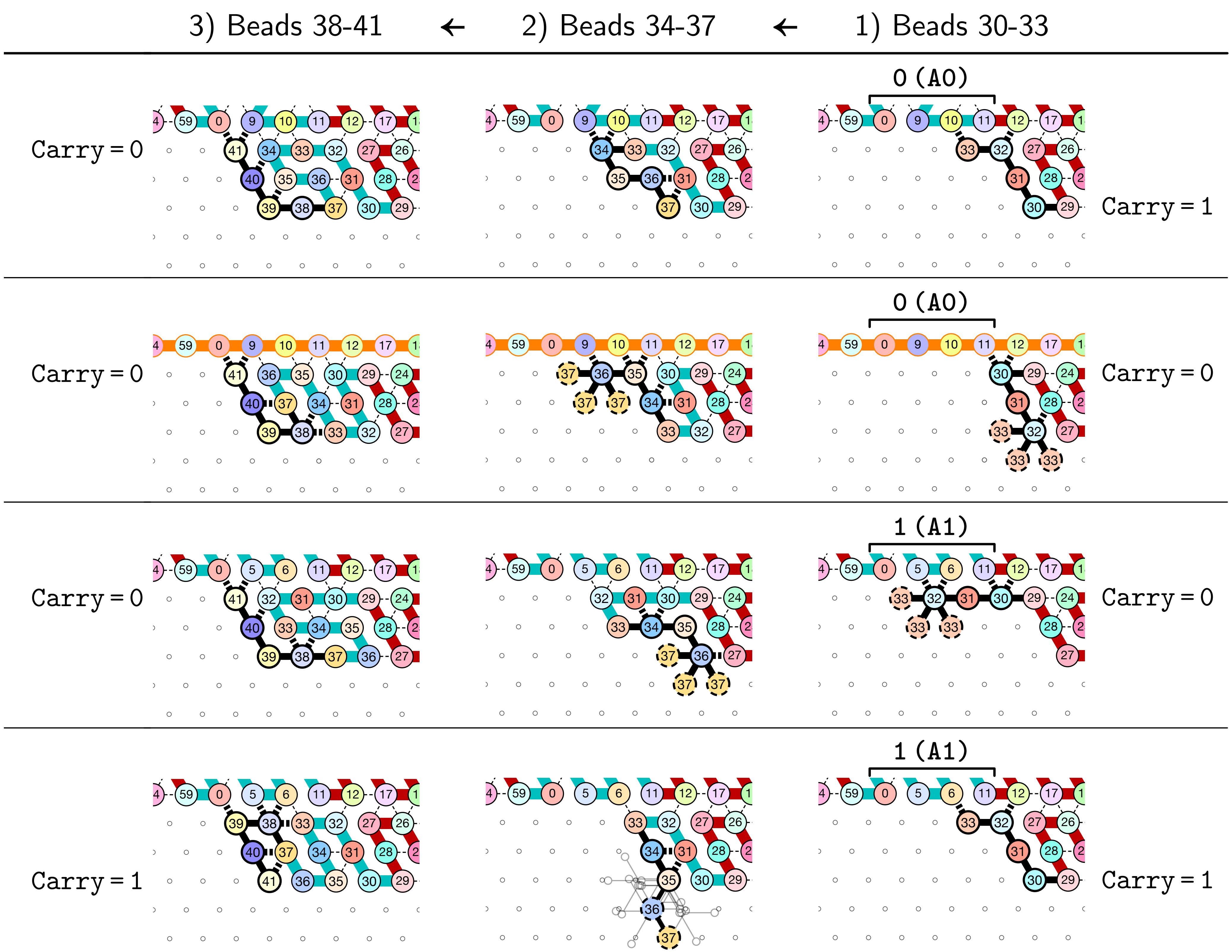
Carry propagation




Line feed



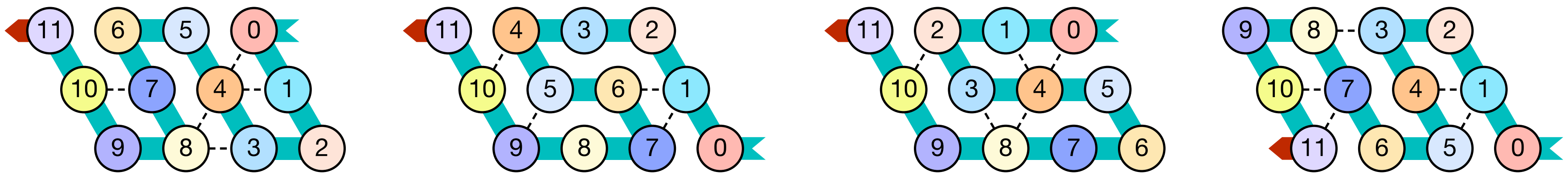
# How does computation work?



# **Proving the binary counter:**
# First, define the *bricks*

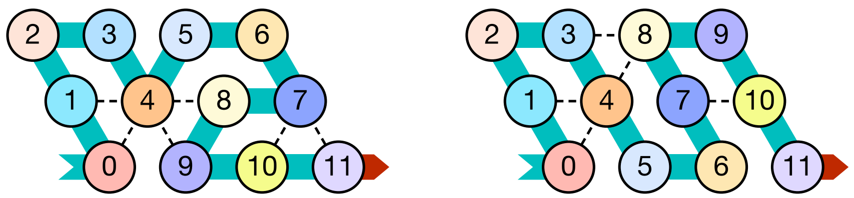
- Module $A$, First Half-Adder (beads 0–11):

Zig Direction ⟵



Brick A00 Brick A01 Brick A10 Brick A11

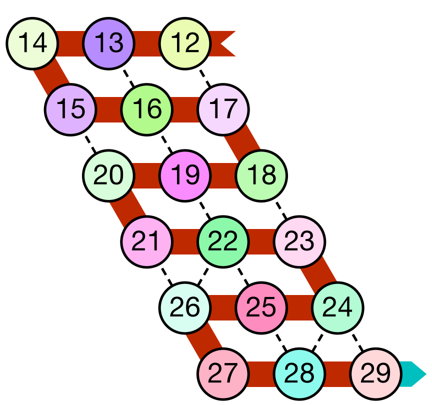
Zag Direction ⟶



Brick A0 Brick A1

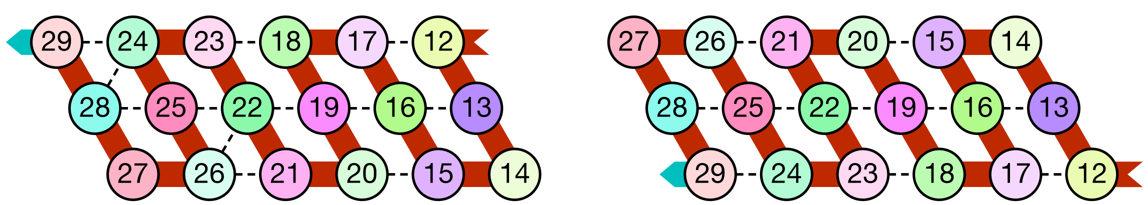
- Module $B$, Left-Turn module (beads 12–29)

Left Turn ↪ (Zig-to-Zag turn)



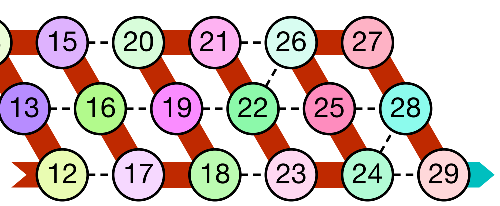
Brick BT

Zig Direction ⟵



Brick B0 Brick B1

Zag Direction ⟶

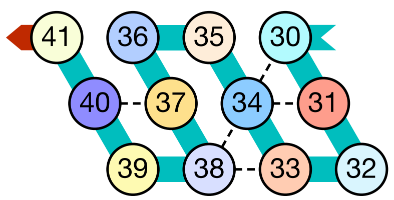


Brick B2

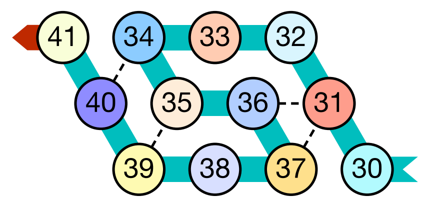
# **Proving the binary counter:**
# First, define the *bricks*

- Module $C$, Second Half-Adder (beads 30–41)

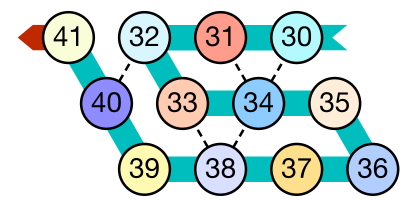
Zig Direction ⟵



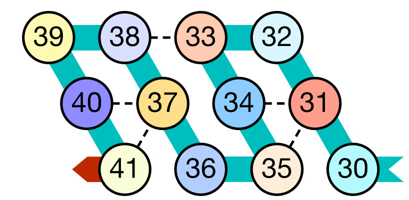
Brick C00



Brick C01

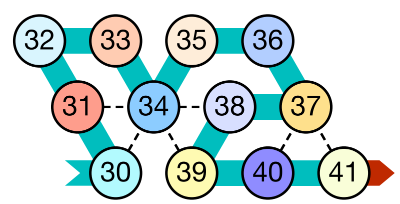


Brick C10

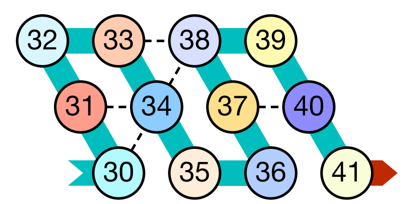


Brick C11

Zag Direction ⟶



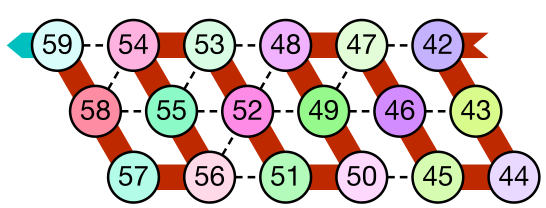
Brick C0



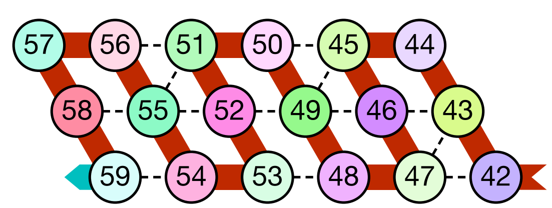
Brick C1

- Module $D$, Right-Turn module (beads 42–59)

Zig Direction ⟵

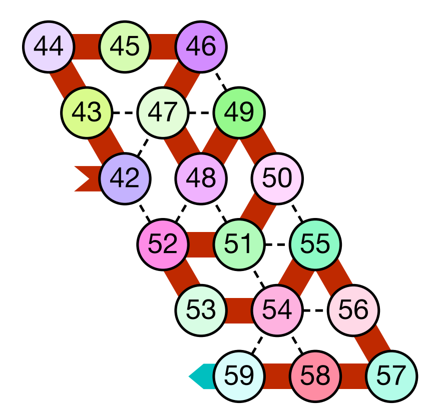


Brick D0



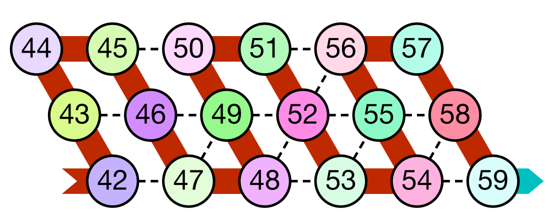
Brick D1

Right Turn ⟵ (Zag-to-Zig turn)



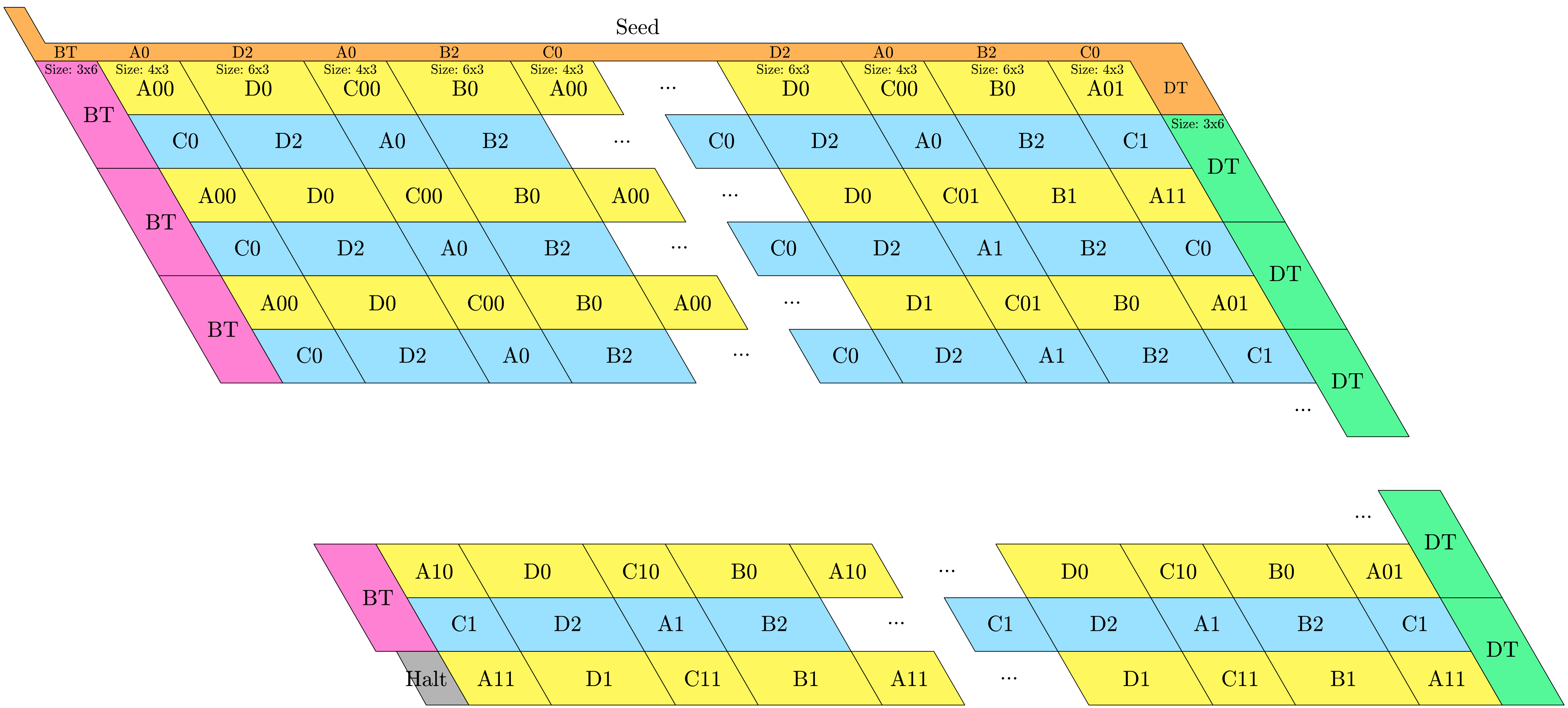
Brick DT

Zag Direction ⟶



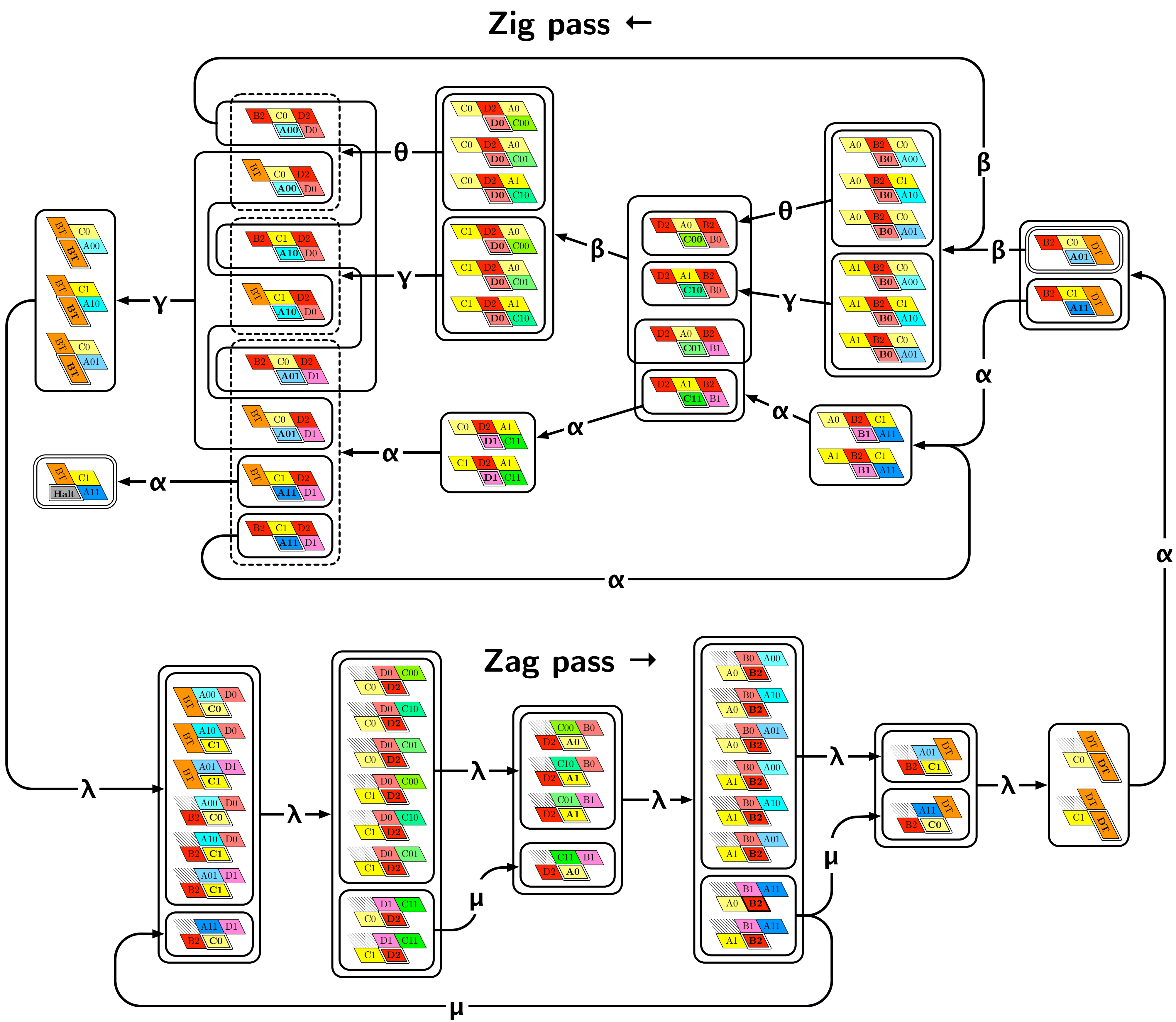
Brick D2

# 2nd, describe the final folding

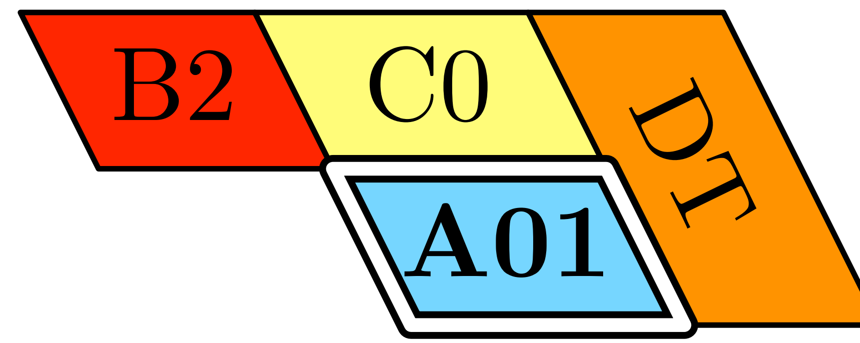


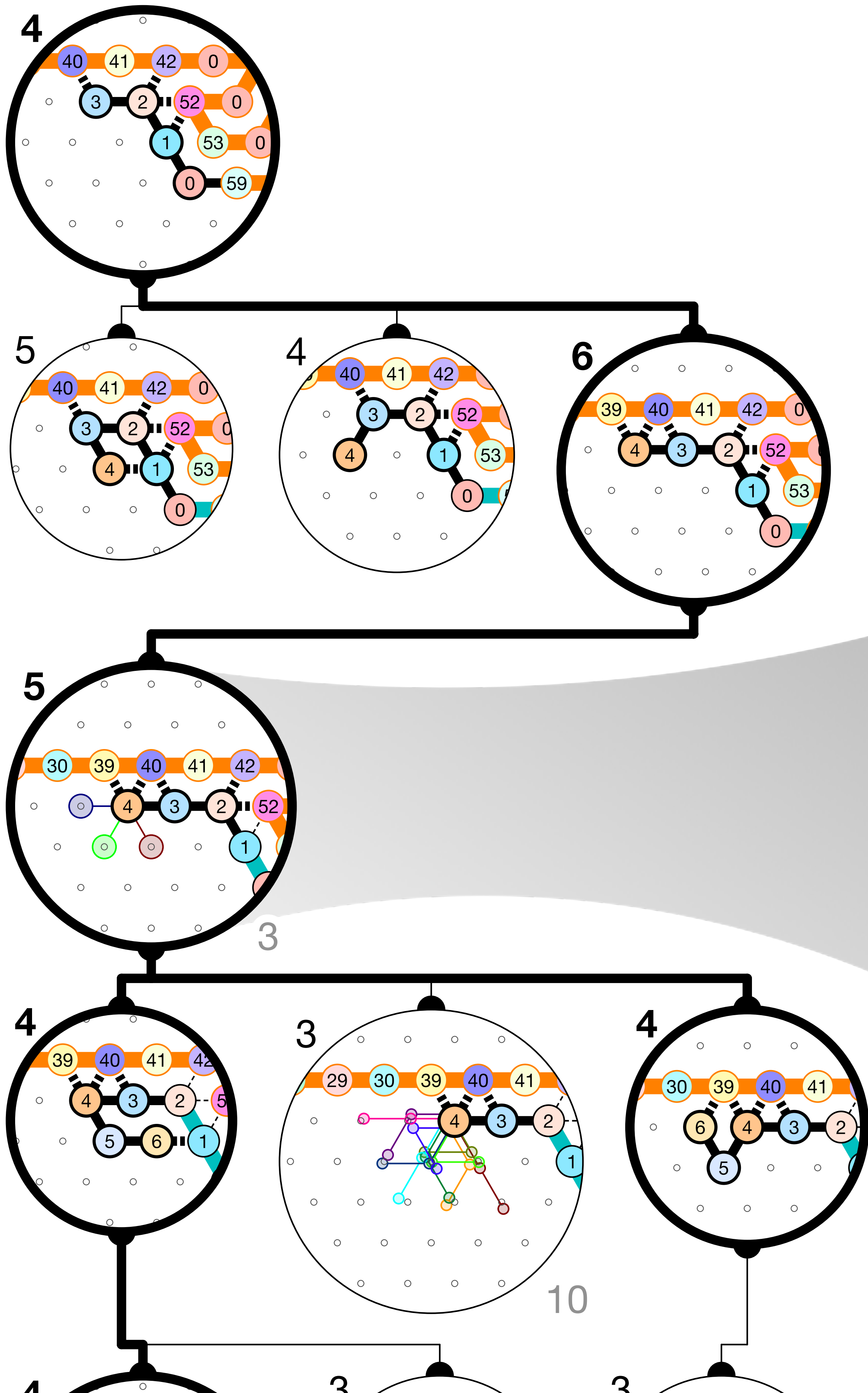

**We prove that the molecule folds like this by induction**

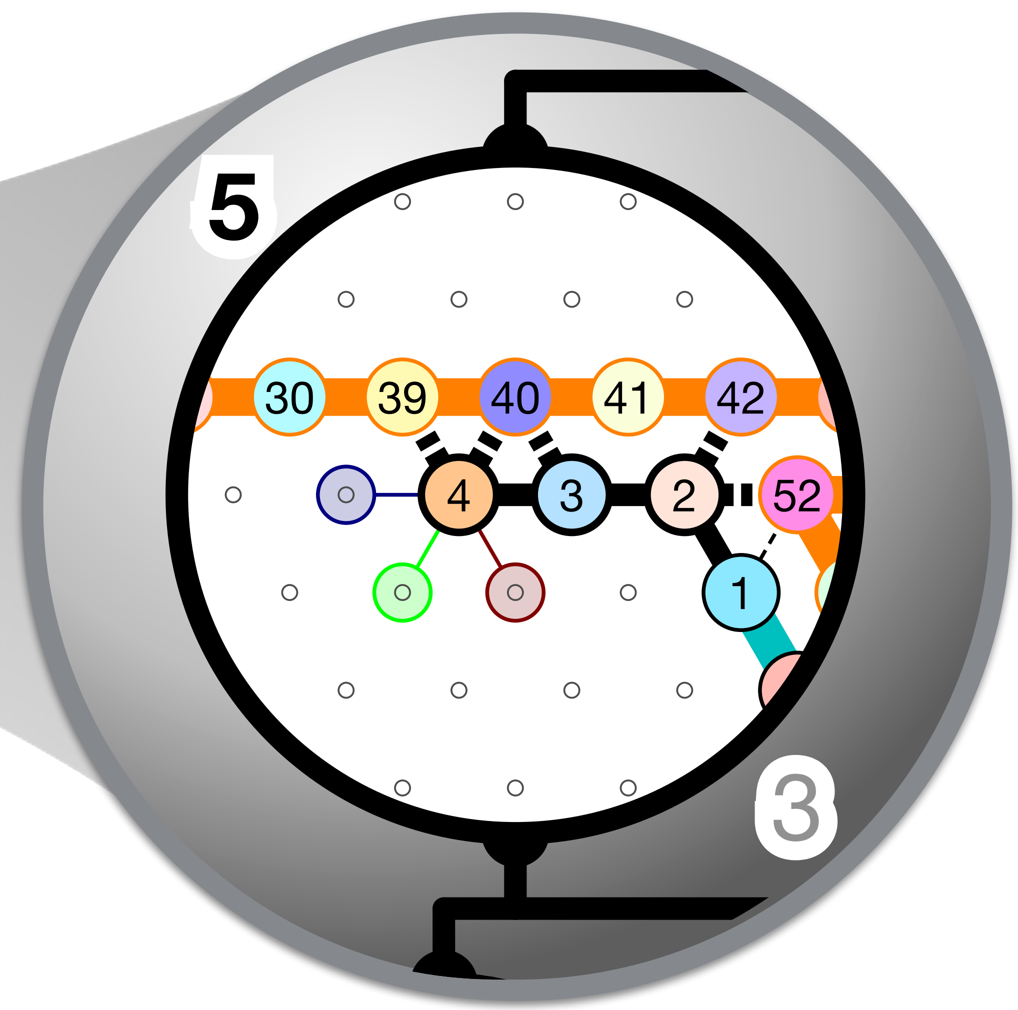
# 3rd, enumerate all the environments for each brick

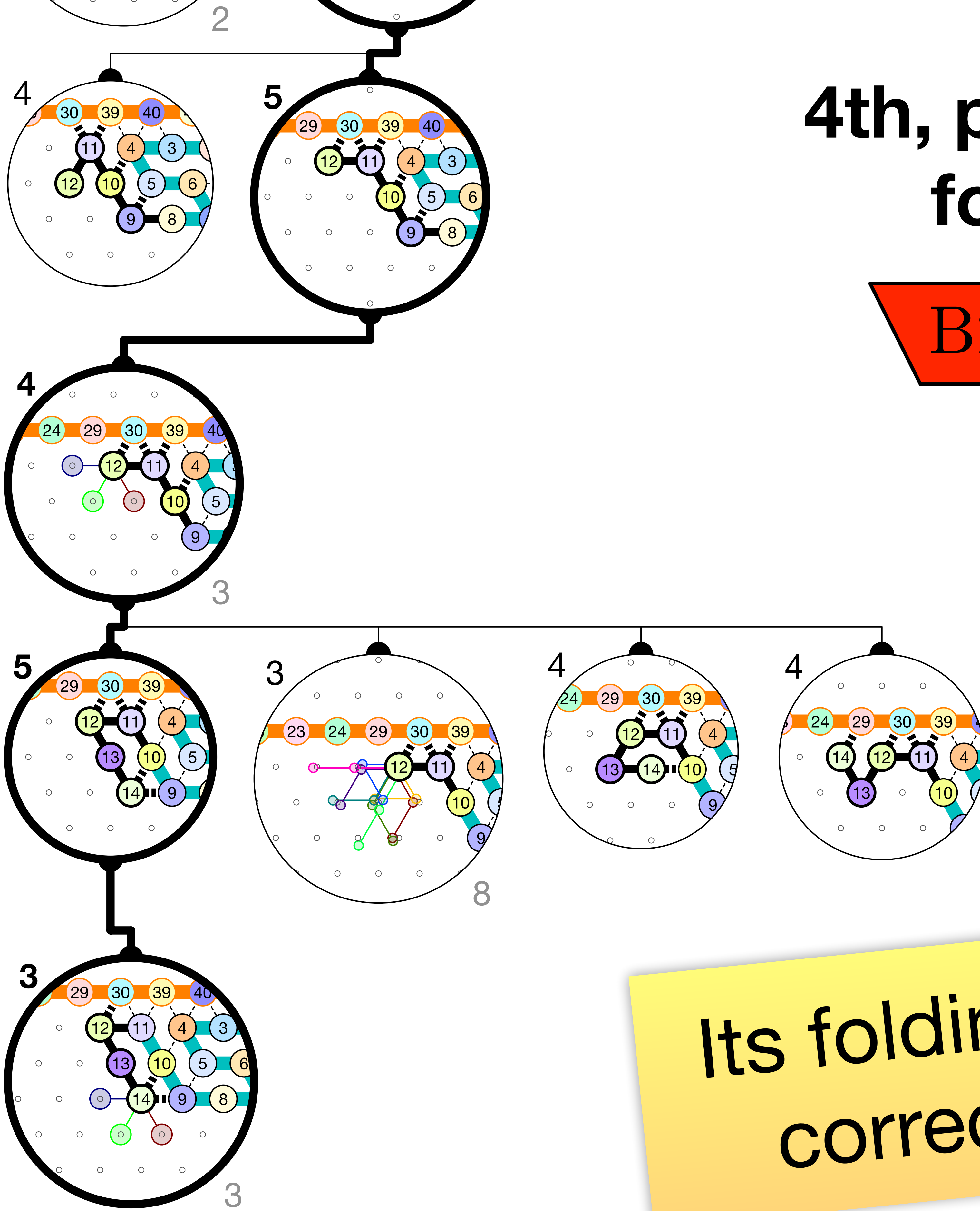


# 4th, prove the folding for each brick

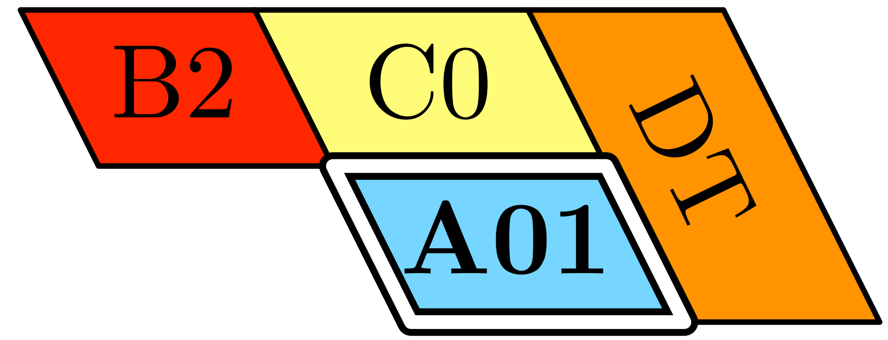


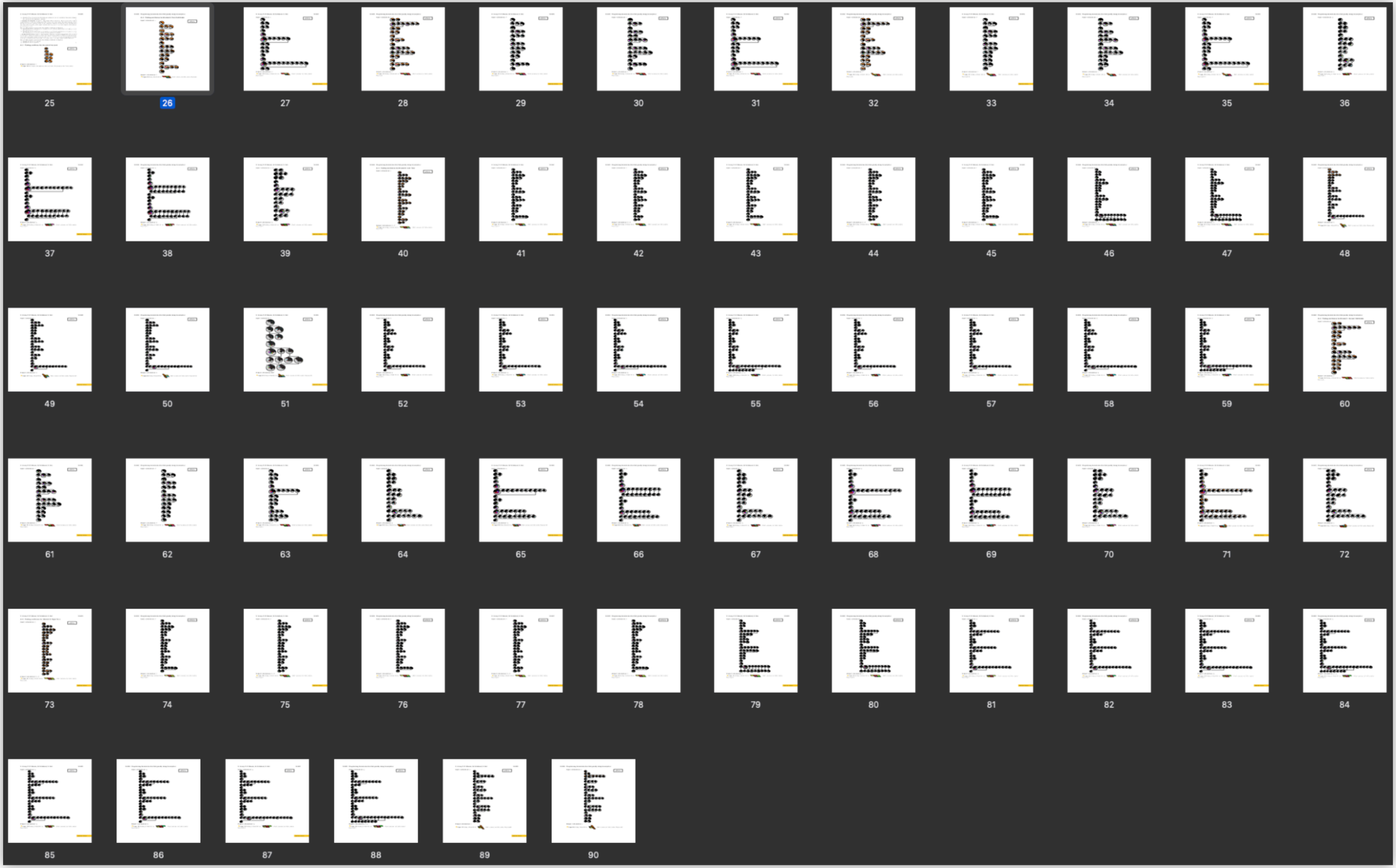


# 4th, prove the folding for each brick

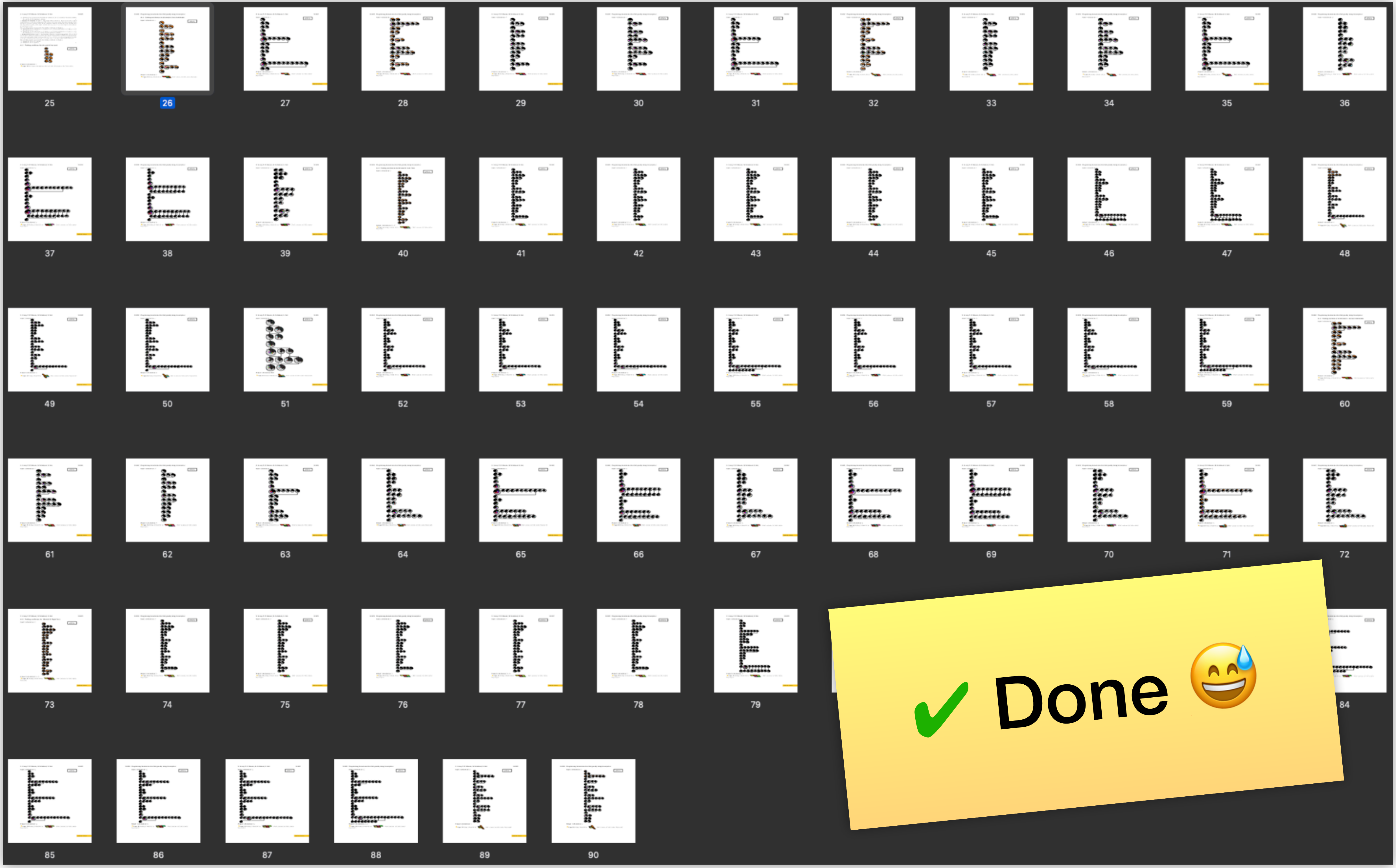



Its folding is correct !

# Repeat for each brick in each environment



# Repeat for each brick in each environment



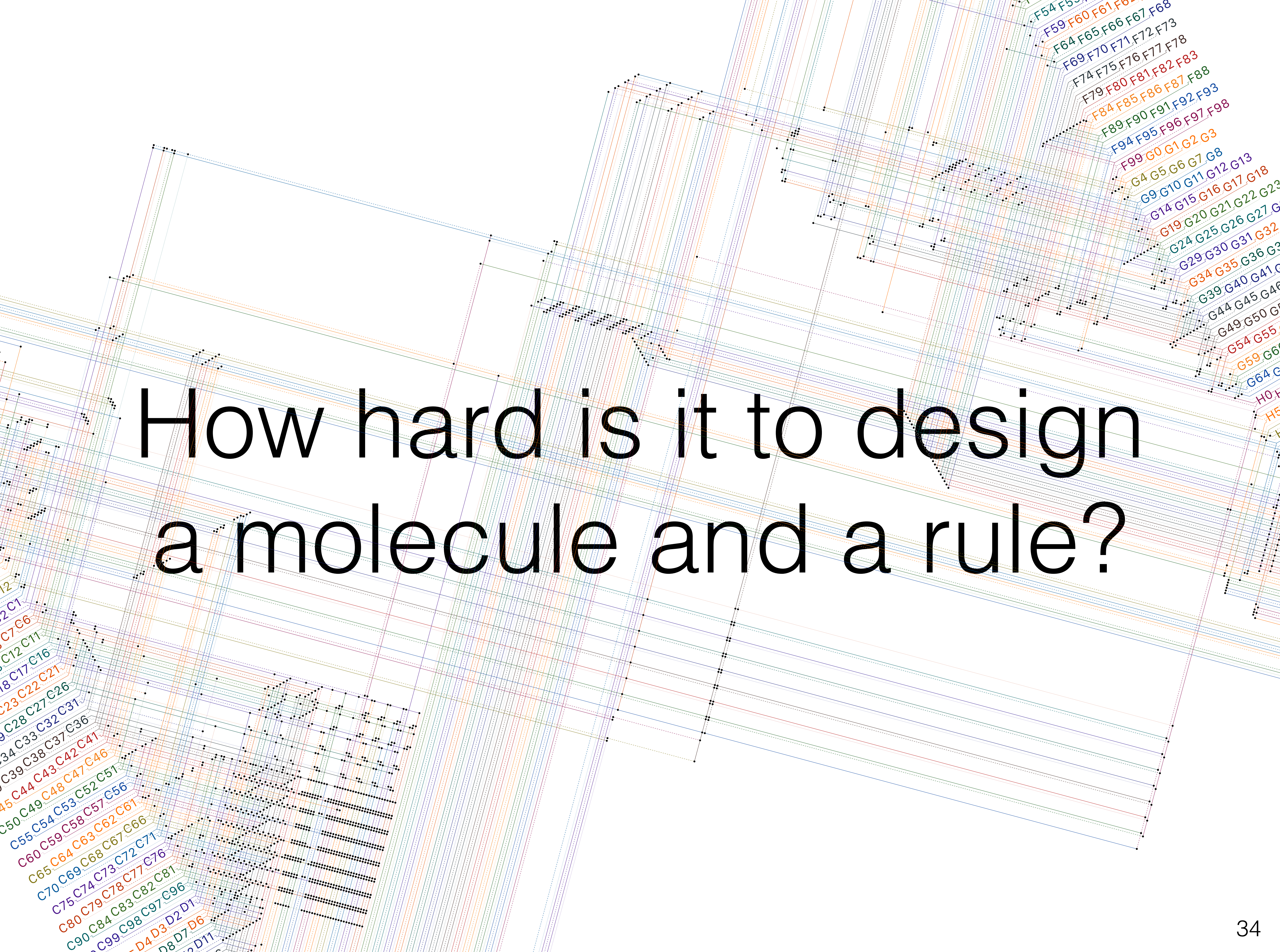
# Binary counter: conclusion

- **Theorem.** There is a 60-periodic molecule that simulates a binary counter using 60 bead types and delay 3.

# Back to general oritatami

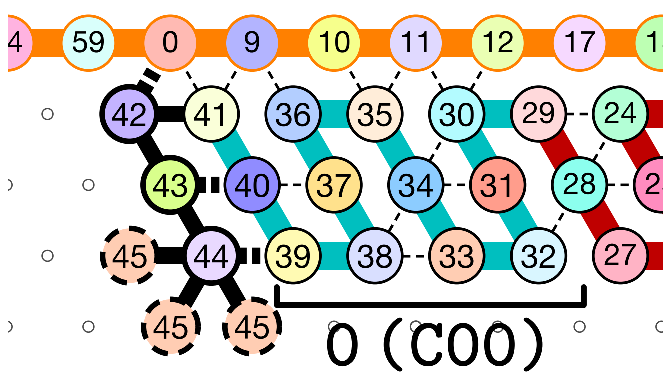
- How hard is it to design a rule?
  - NP-hard… but FPT, thus feasible!
- What can it compute?
  - Simulates any Turing Machine… *efficiently!*

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

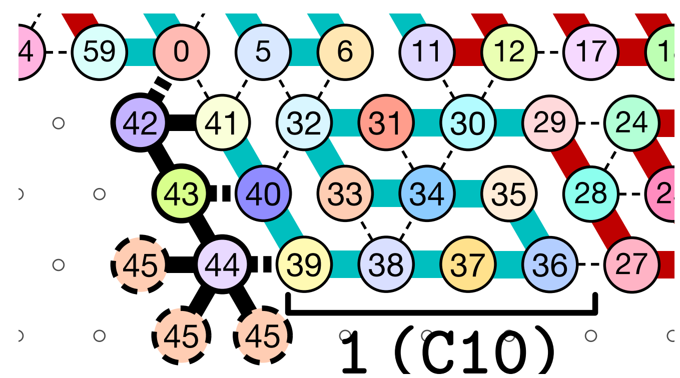


# The first challenge: Designing the desired shapes

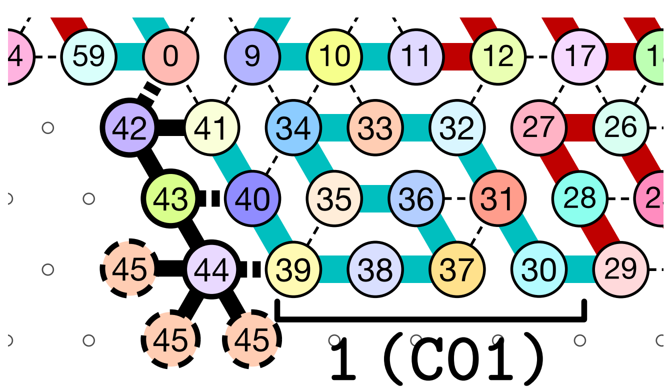
- Design shapes for which a **common** rule ❤️ exists



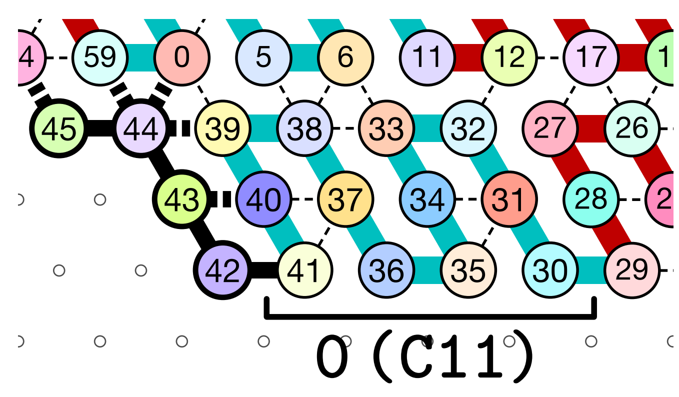

0+0 = 0 + no C




1+0 = 1 + no C



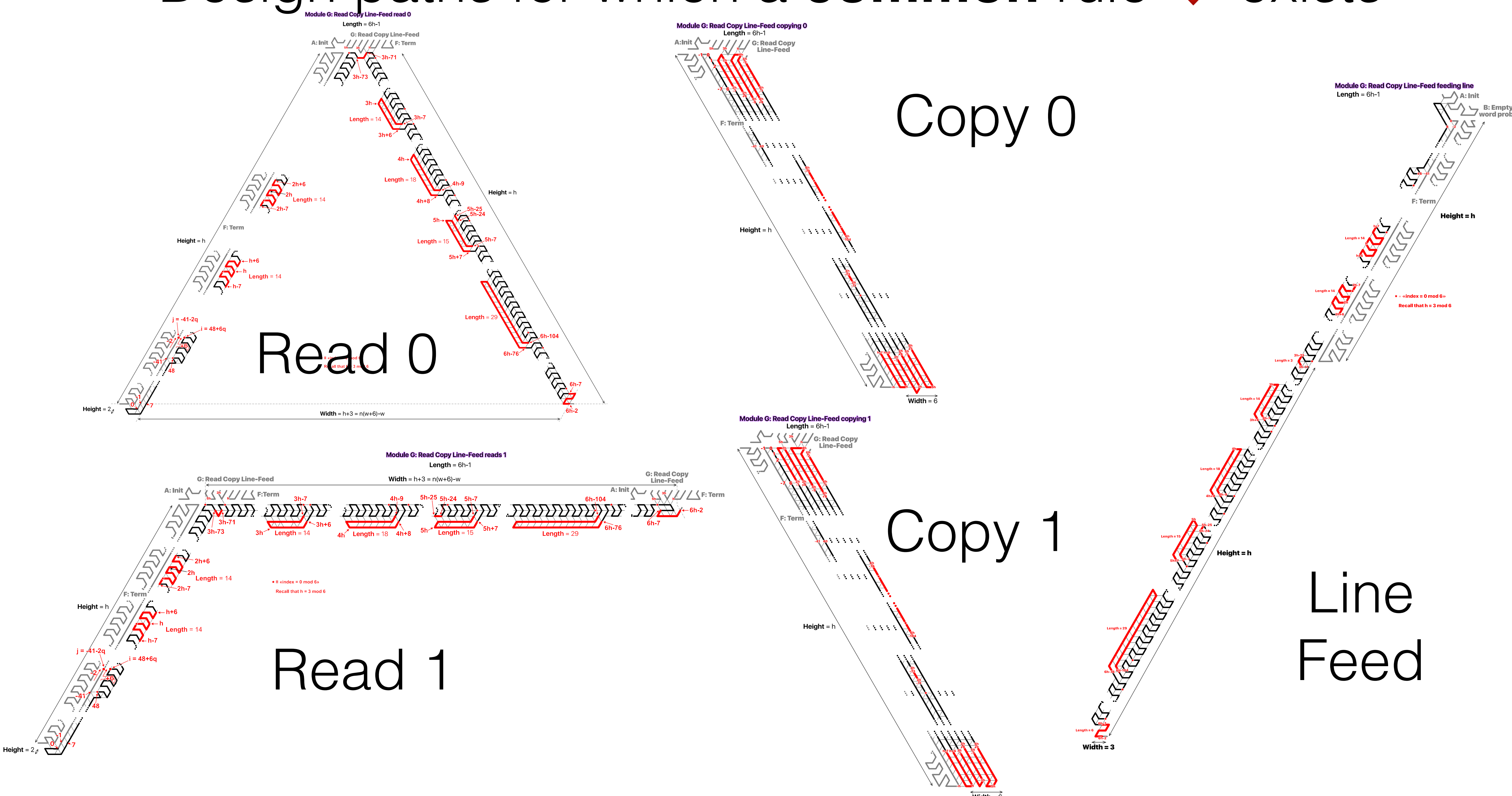

0+1 = 1 + no C




1+1 = 0 + C

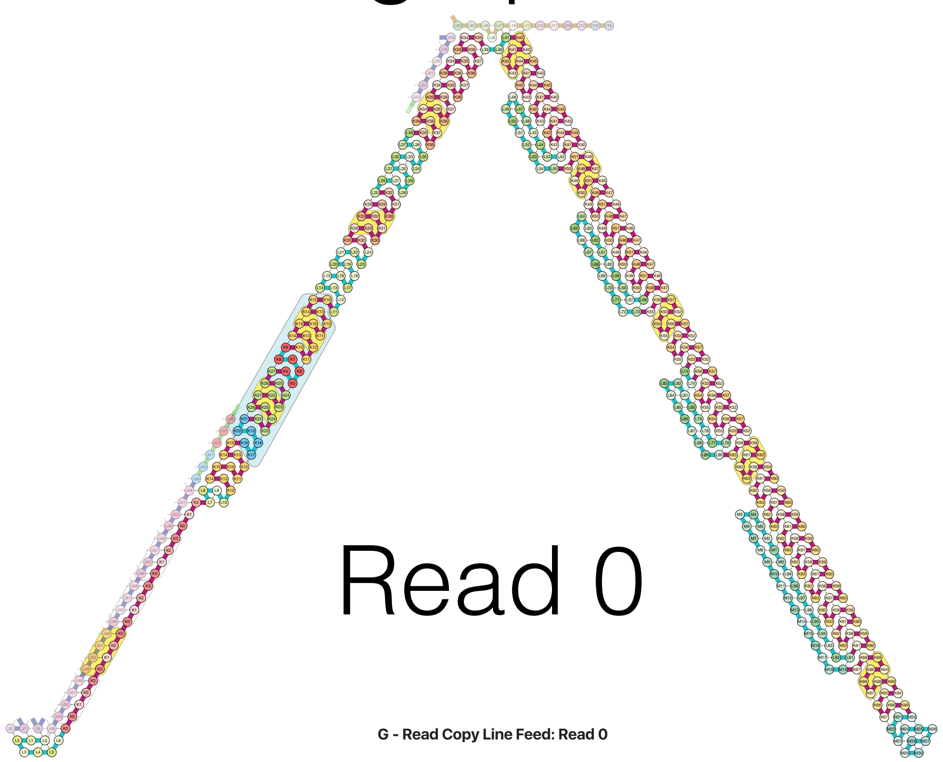
# The first challenge: Designing the desired paths

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

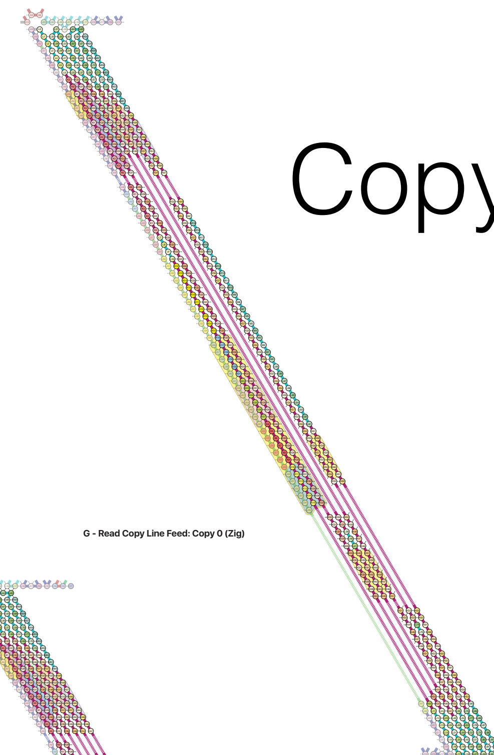


# The first challenge: Designing the paths

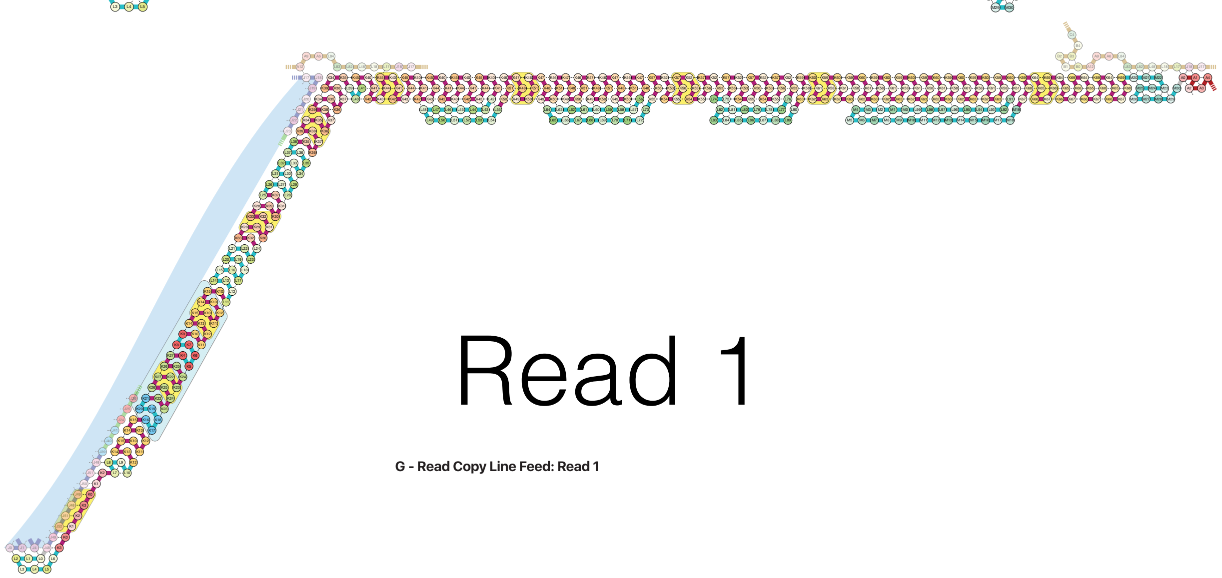
- Design paths for which a common rule ❤️ exists



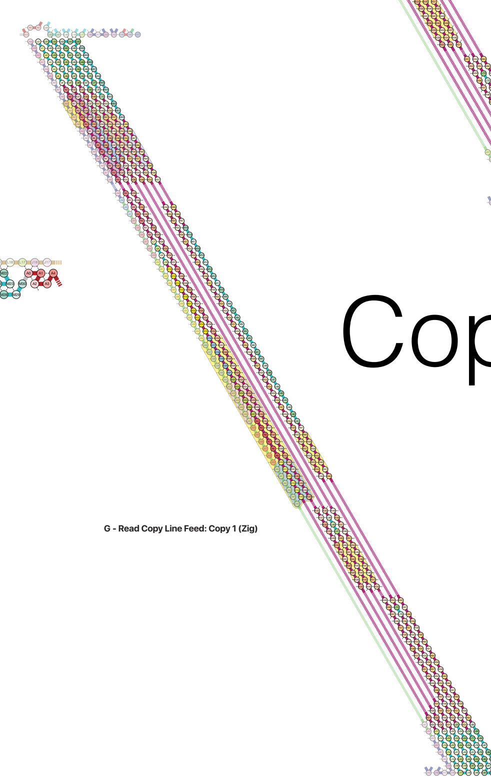

Read 0



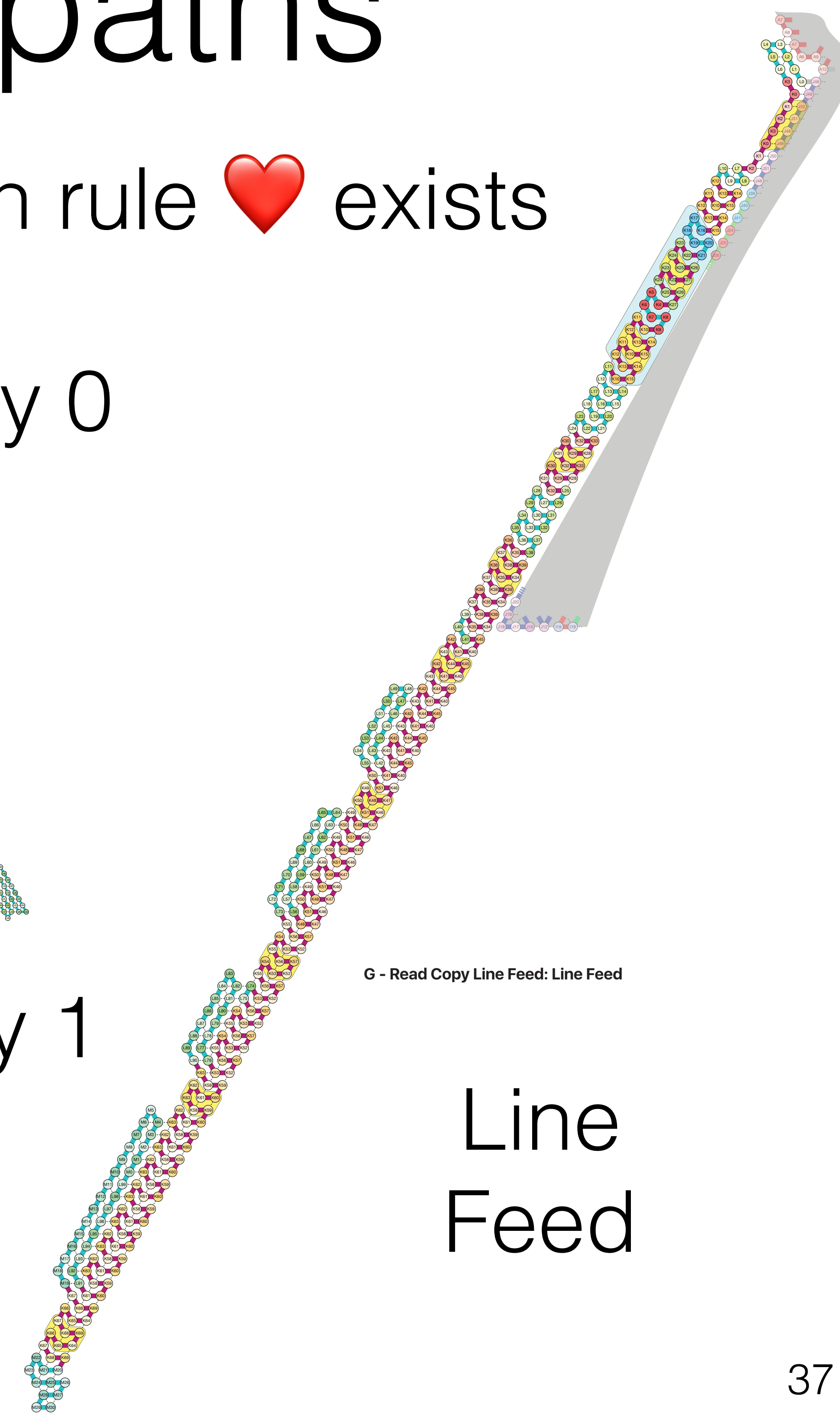

Copy 0




Read 1




Copy 1



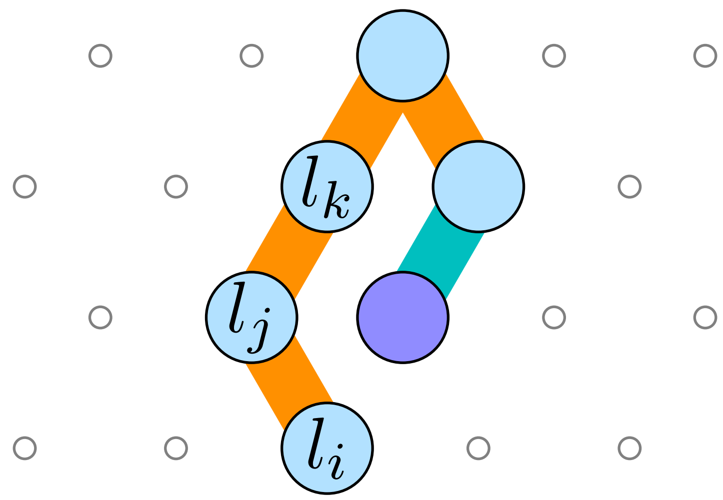

Line Feed

# Oritatami design is NP-hard

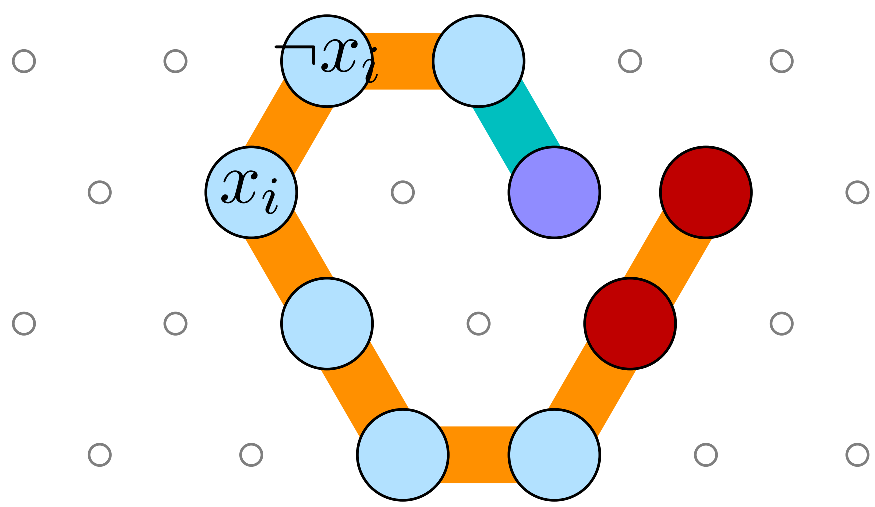
| | |
|---|---|
| INPUT: | a delay time $\delta$, a list of $n > 0$ seeds $\sigma_1, \sigma_2, \ldots, \sigma_n$, and a list of $n$ conformations $c_1, c_2, \ldots, c_n$ of the same length $l$ |
| OUTPUT: | an attraction rule $\heartsuit$ such that for all $i \in \{1, 2, \ldots, n\}$, Oritatami system $\mathcal{O}_i = (s, \sigma_i, \heartsuit, \delta)$ 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 $l_i \vee l_j \vee l_k$




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.

Previous result:

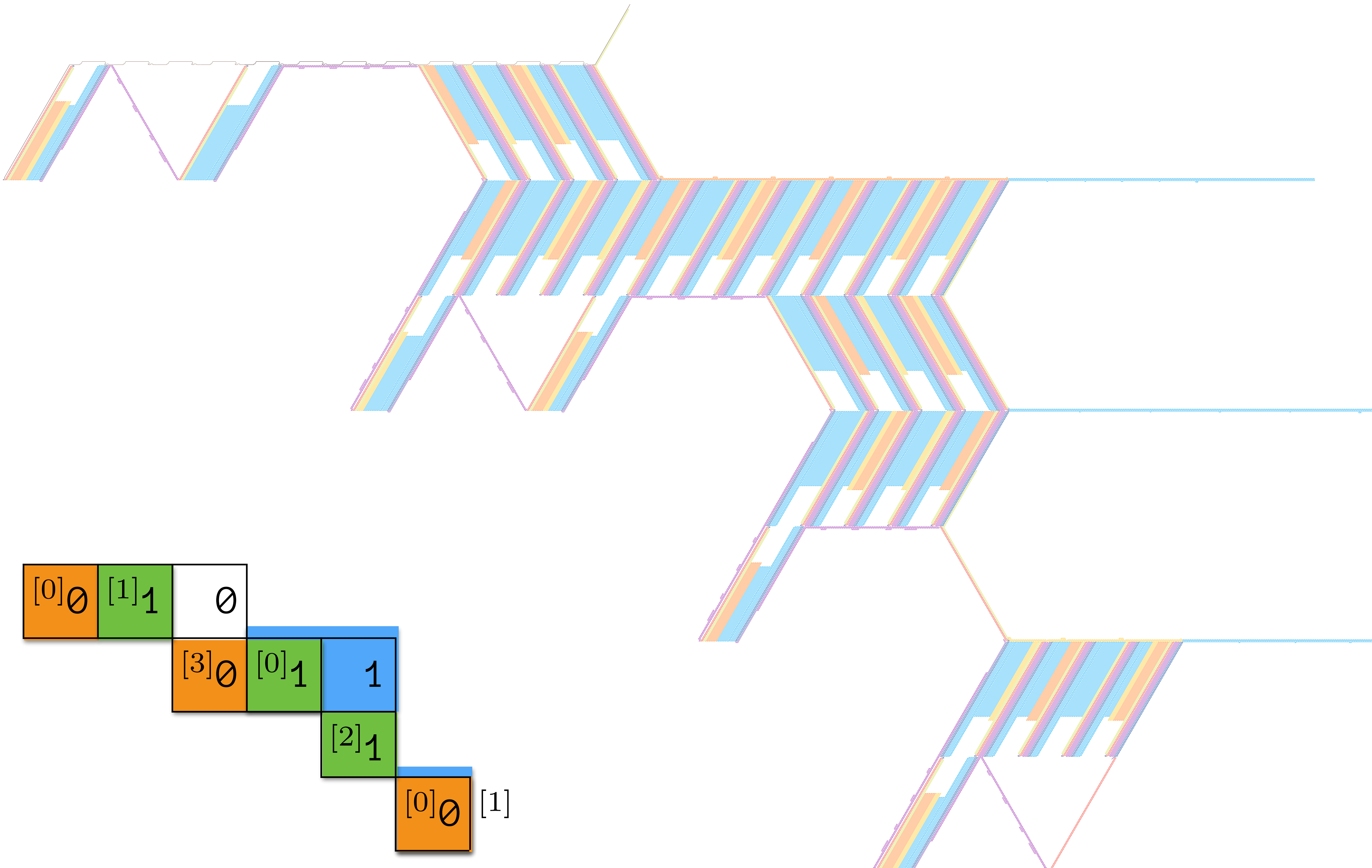
**Oritatami is Turing complete**

# Trimmed space-time diagram

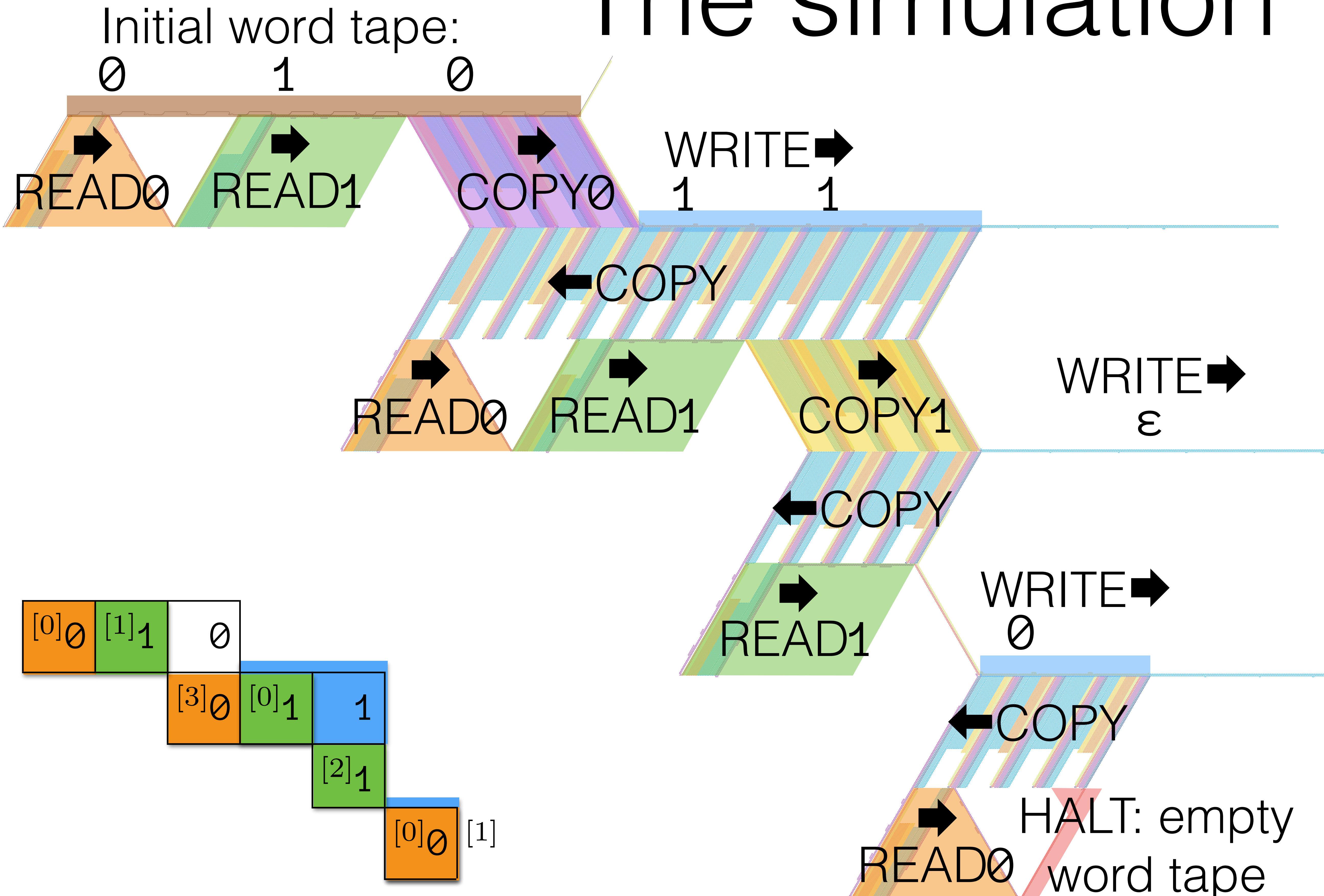
Consider the following productions: $p = \langle \overset{[0]}{\texttt{110}}, \overset{[1]}{\epsilon}, \overset{[2]}{\texttt{11}}, \overset{[3]}{\texttt{0}} \rangle$

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

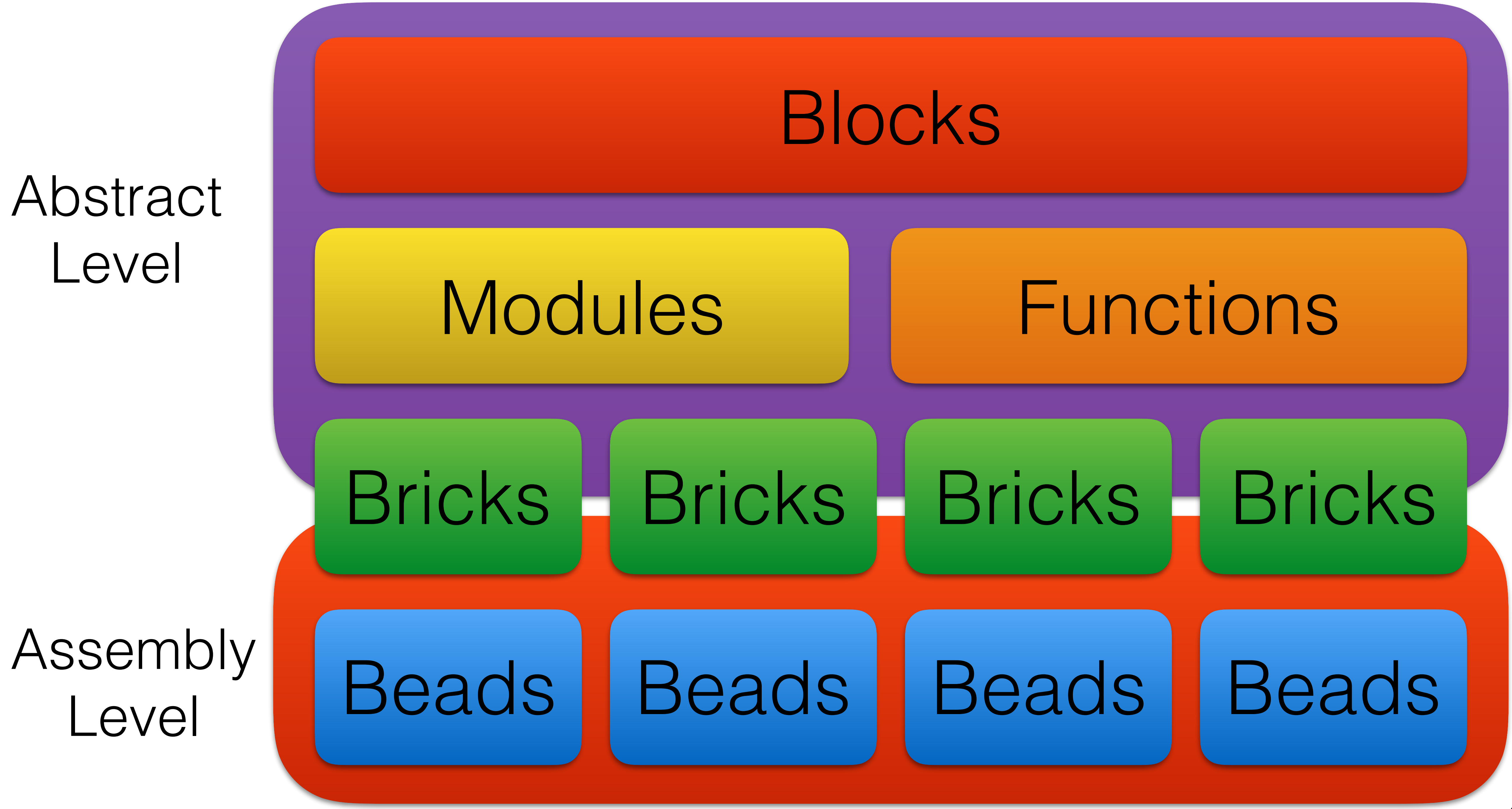
# The simulation



# The simulation



# A general programming framework



# General programming tools

State
- Area entry point
- Position in Molecule

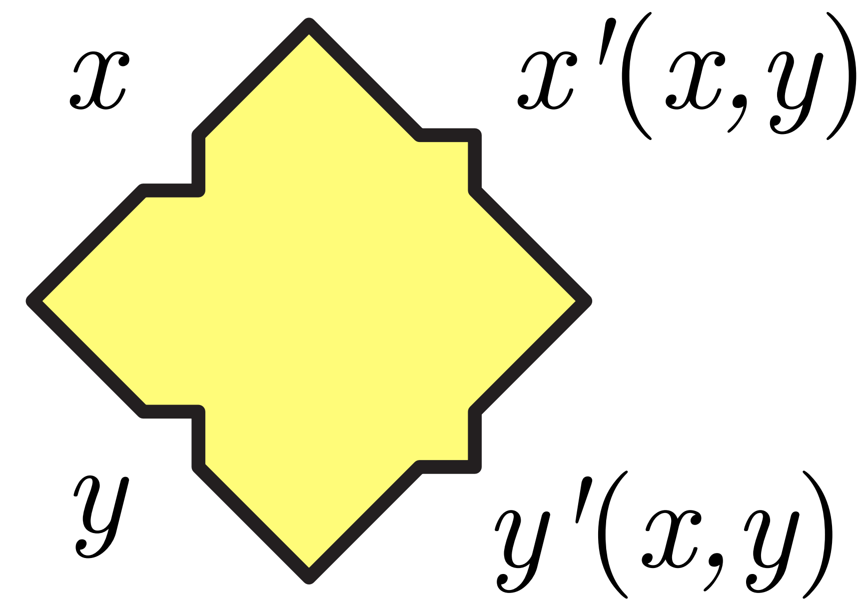
Logic
- Sliding shapes
- Bouncing gliders
- Geometry
- Expanding shapes

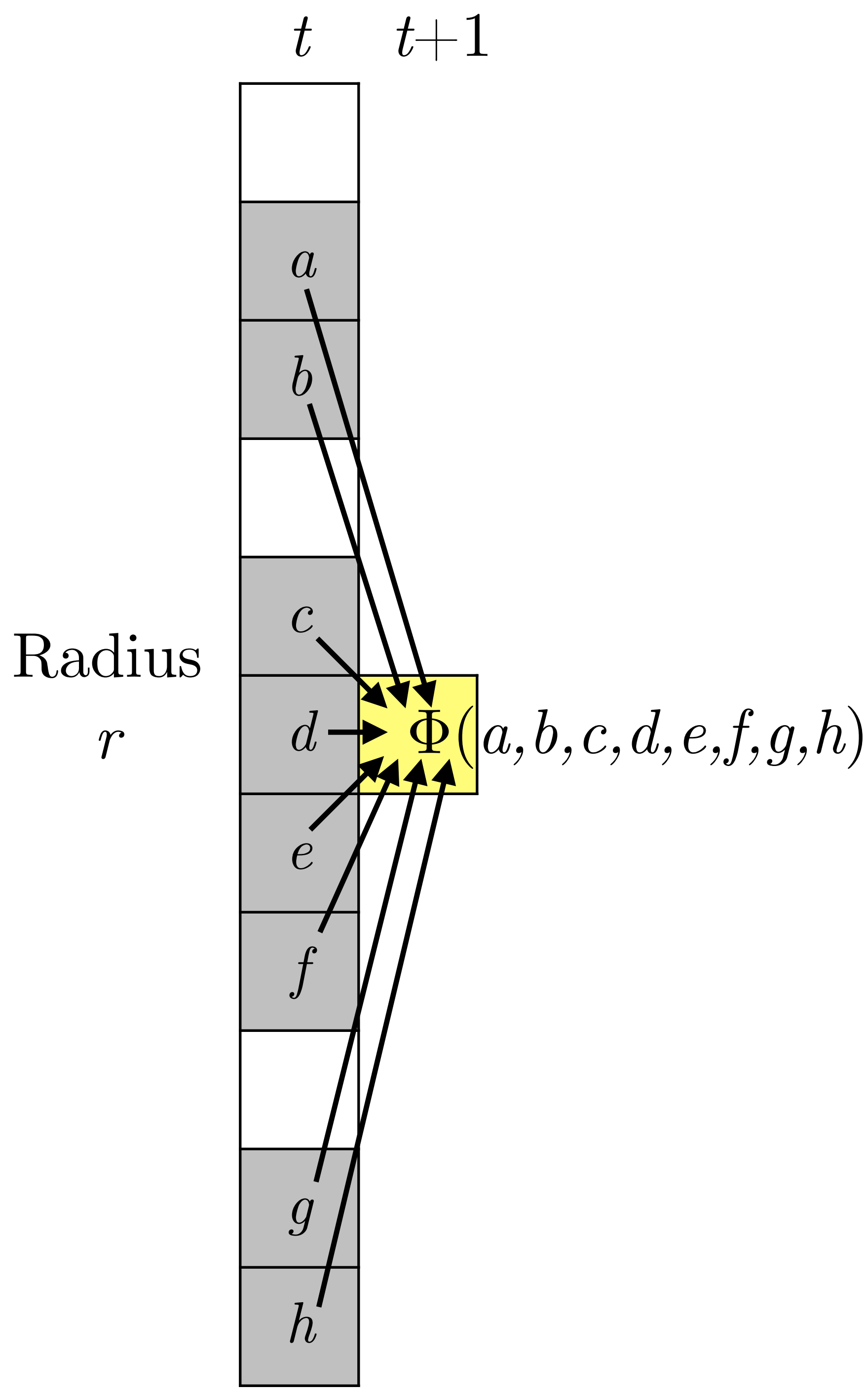
Goto
- Offsets
- Socks
- Exponential coloring
- Hiding

Our new result:

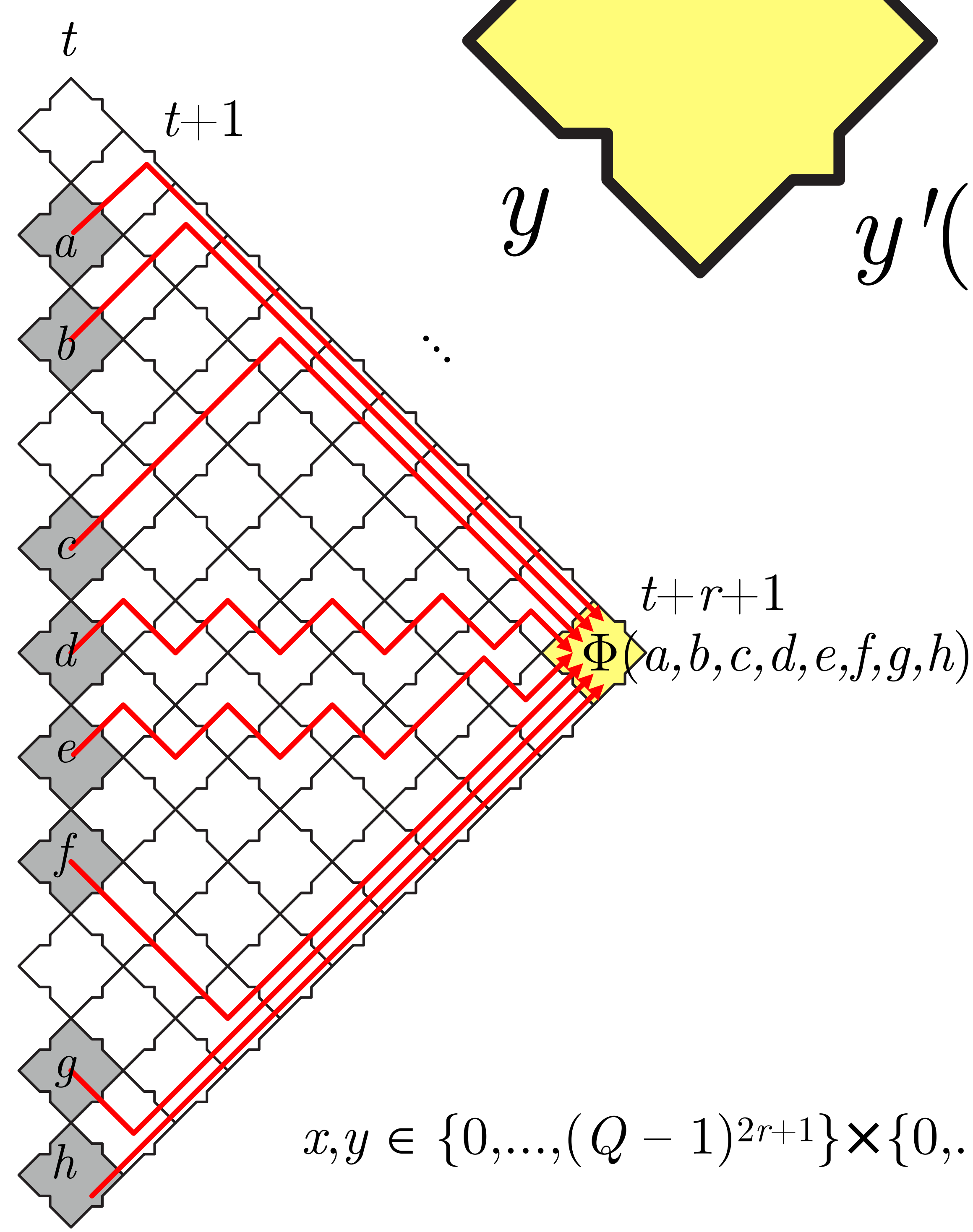
*Intrinsic Simulation of 1D Cellular Automata*

# from 1D CA to 2-in 2-out Gates



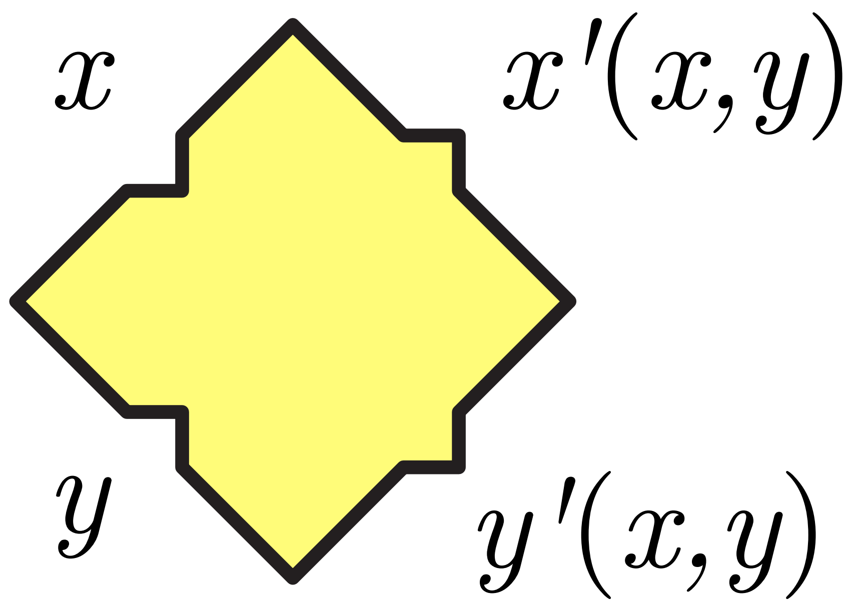




$a,b,c,d,e,f,g,h \in \{0,\ldots,Q-1\}$



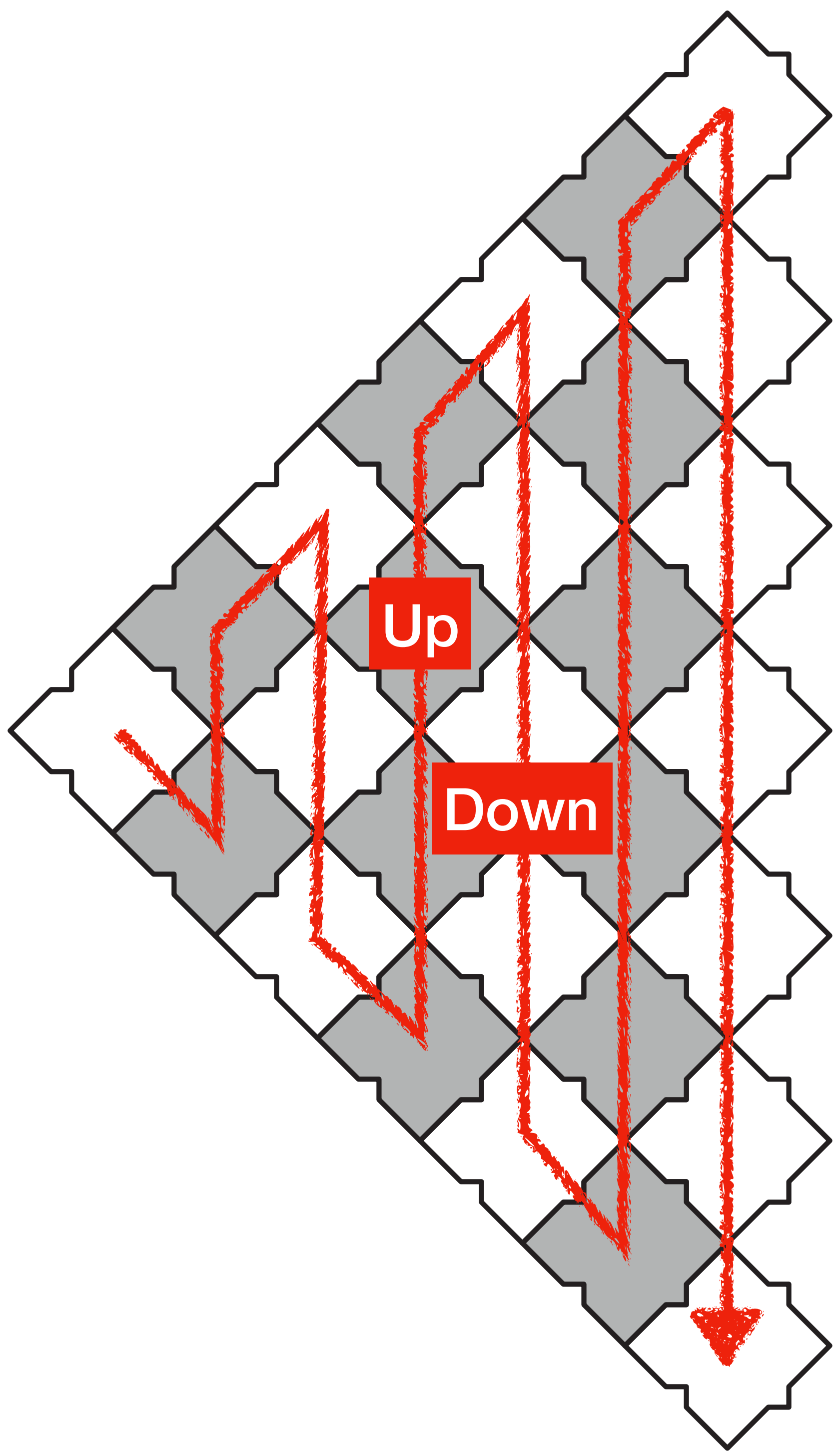

$x,y \in \{0,\ldots,(Q-1)^{2r+1}\} \times \{0,\ldots,r\}$

# Oritatami system simulating 2-in 2-out gates




where $x, y \in \{0, \ldots, Q-1\}$

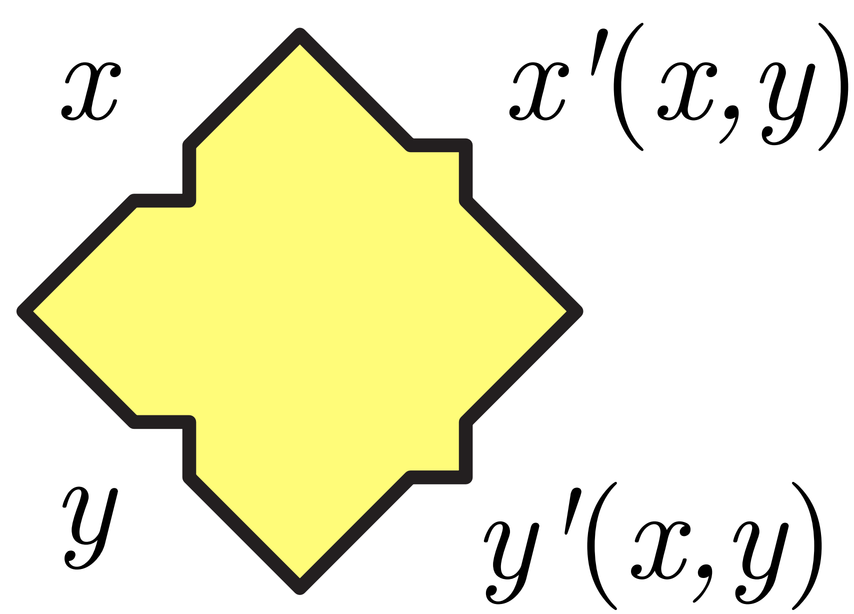
# Oritatami system simulating 2-in 2-out gates



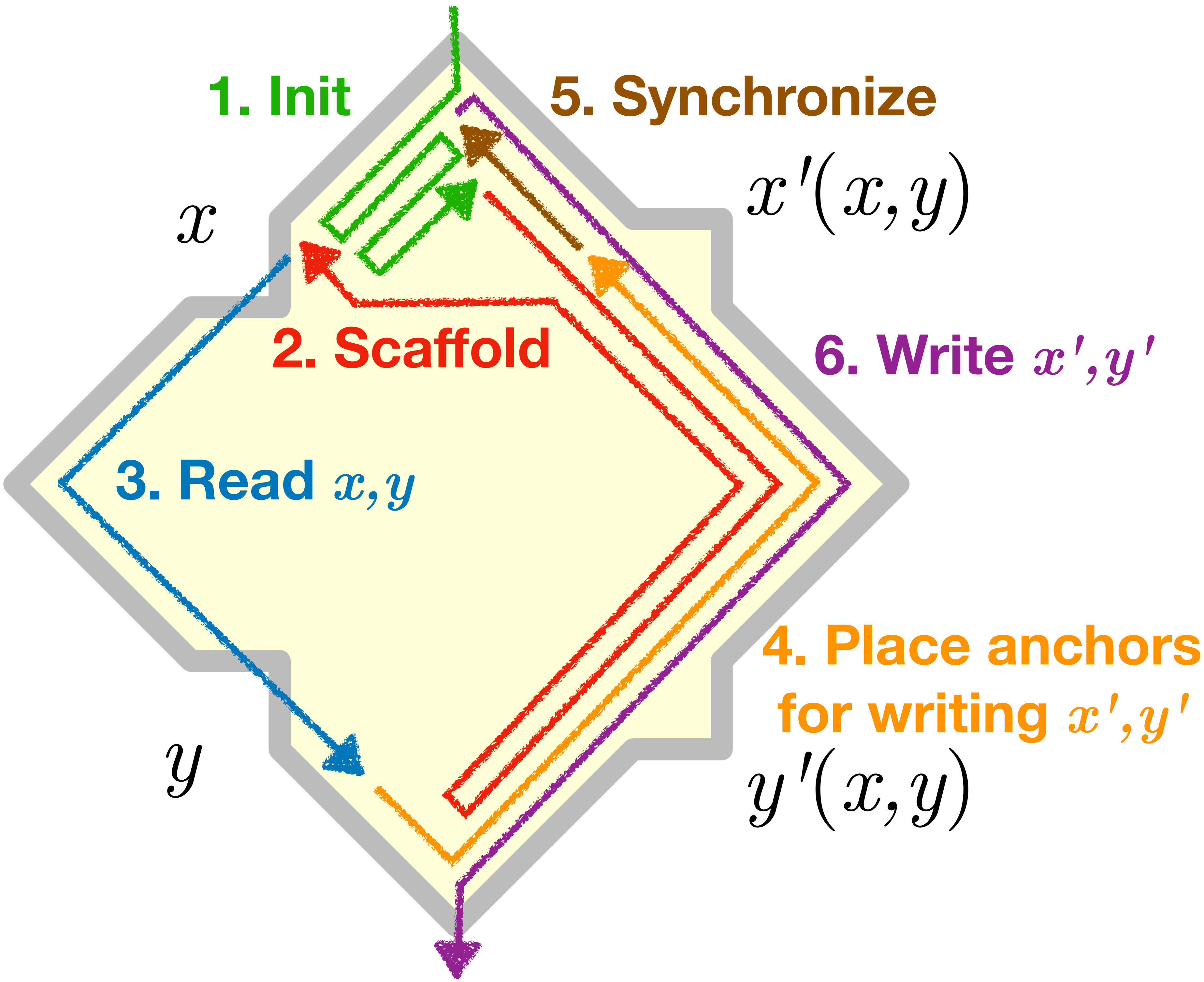

**Up & Down paths are just mirrored of each other**

**→ Just need to add an extra Up/Down-state and to mirror the transition function:**

$$X'(x,y,\downarrow) := (x'(x,y),\uparrow)$$
$$X'(x,y,\uparrow) := (y'(y,x),\downarrow)$$



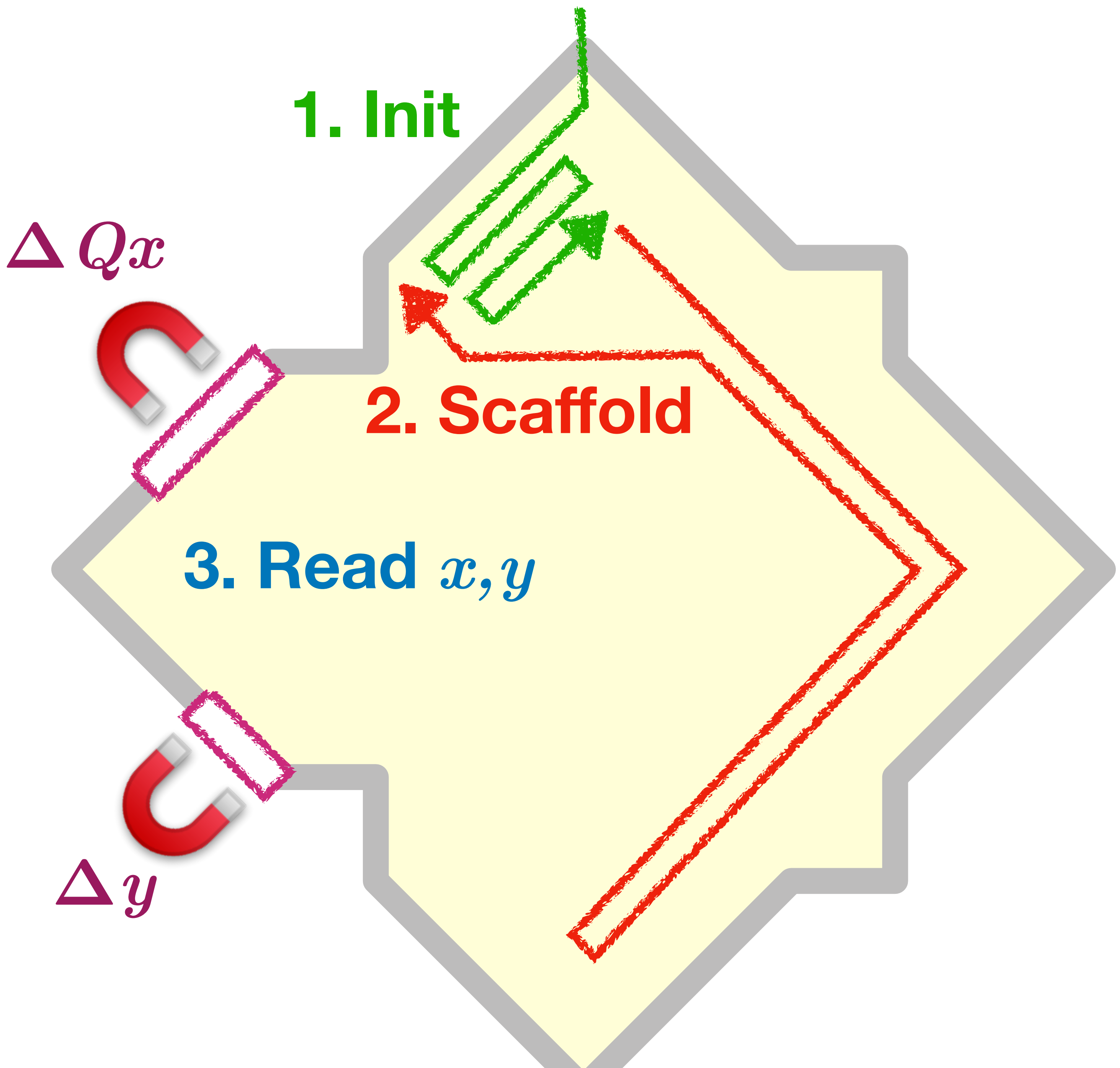
# Let's simulate one 2-in 2-out gate



# Read-Write mechanism

Writing $x, y$ = Placing magnets of length $\Delta\, Qx$ and $\Delta\, y$

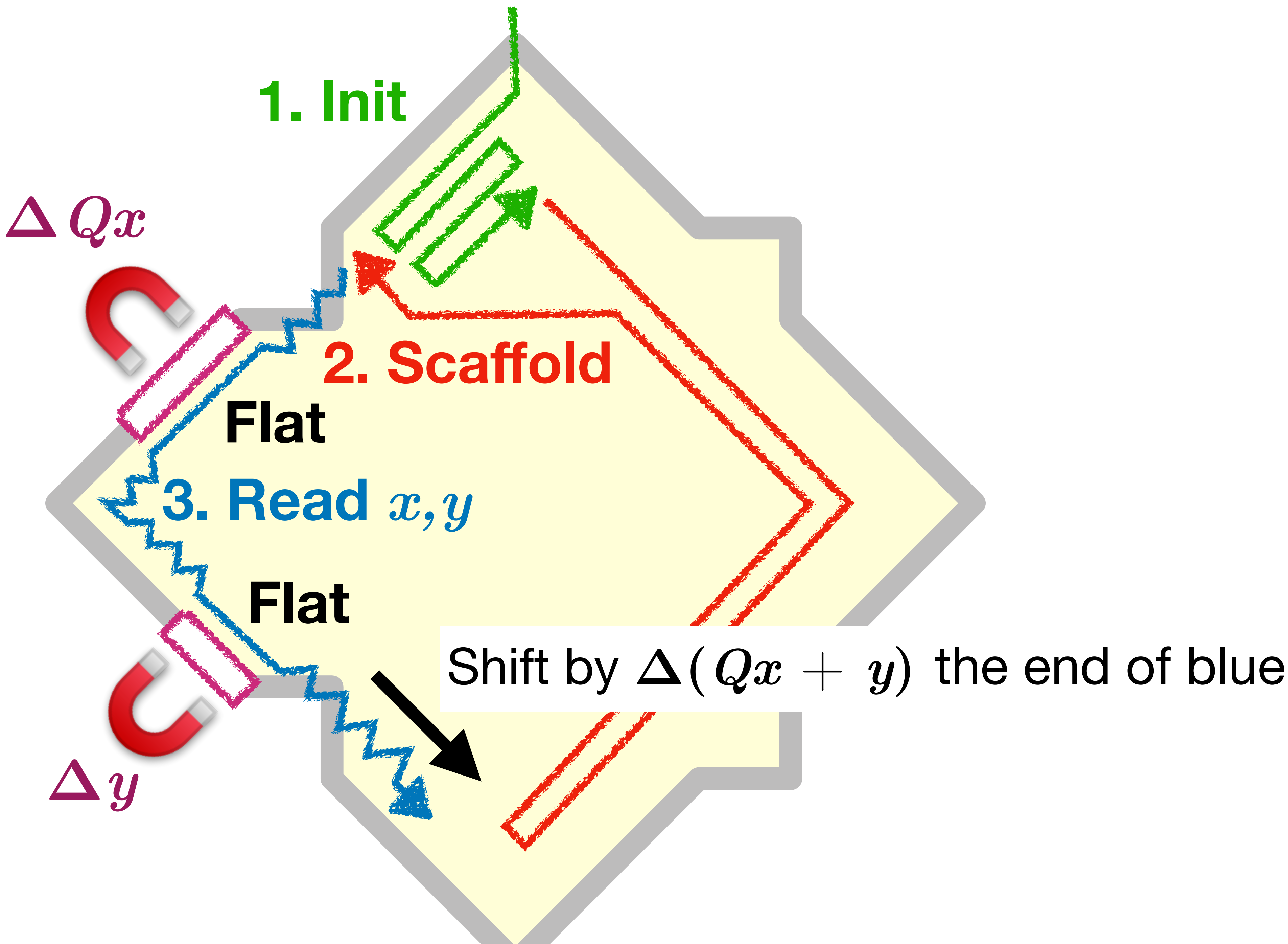
Reading $x, y$ = Create an offset of $\Delta(Qx + y)$



# Read-Write mechanism

Writing $x,y$ = Placing magnets of length $\Delta Qx$ and $\Delta y$

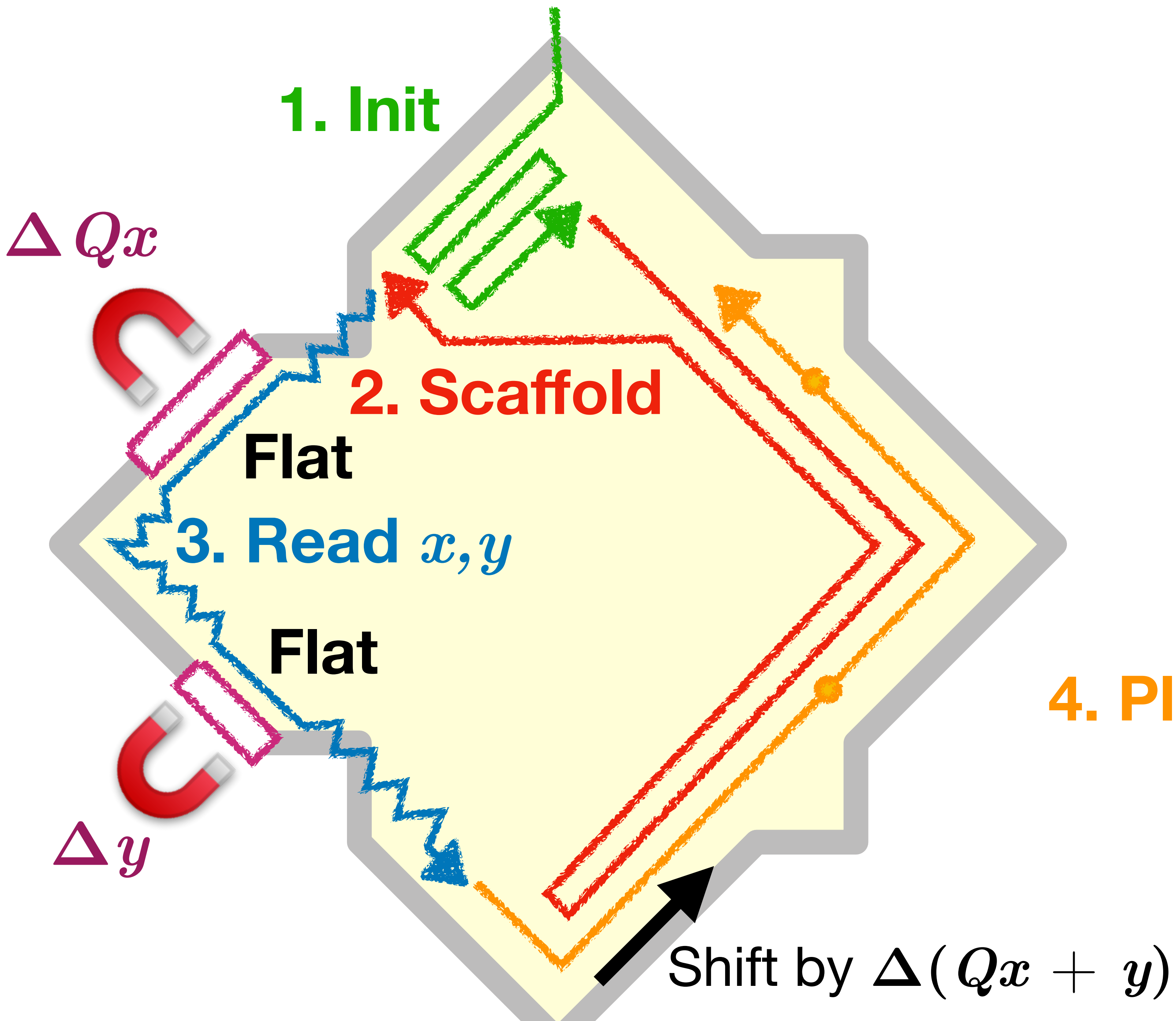
Reading $x,y$ = Create an offset of $\Delta(Qx + y)$



# Read-Write mechanism

Writing $x, y$ = Placing magnets of length $\Delta\, Qx$ and $\Delta\, y$

Reading $x, y$ = Create an offset of $\Delta(Qx + y)$

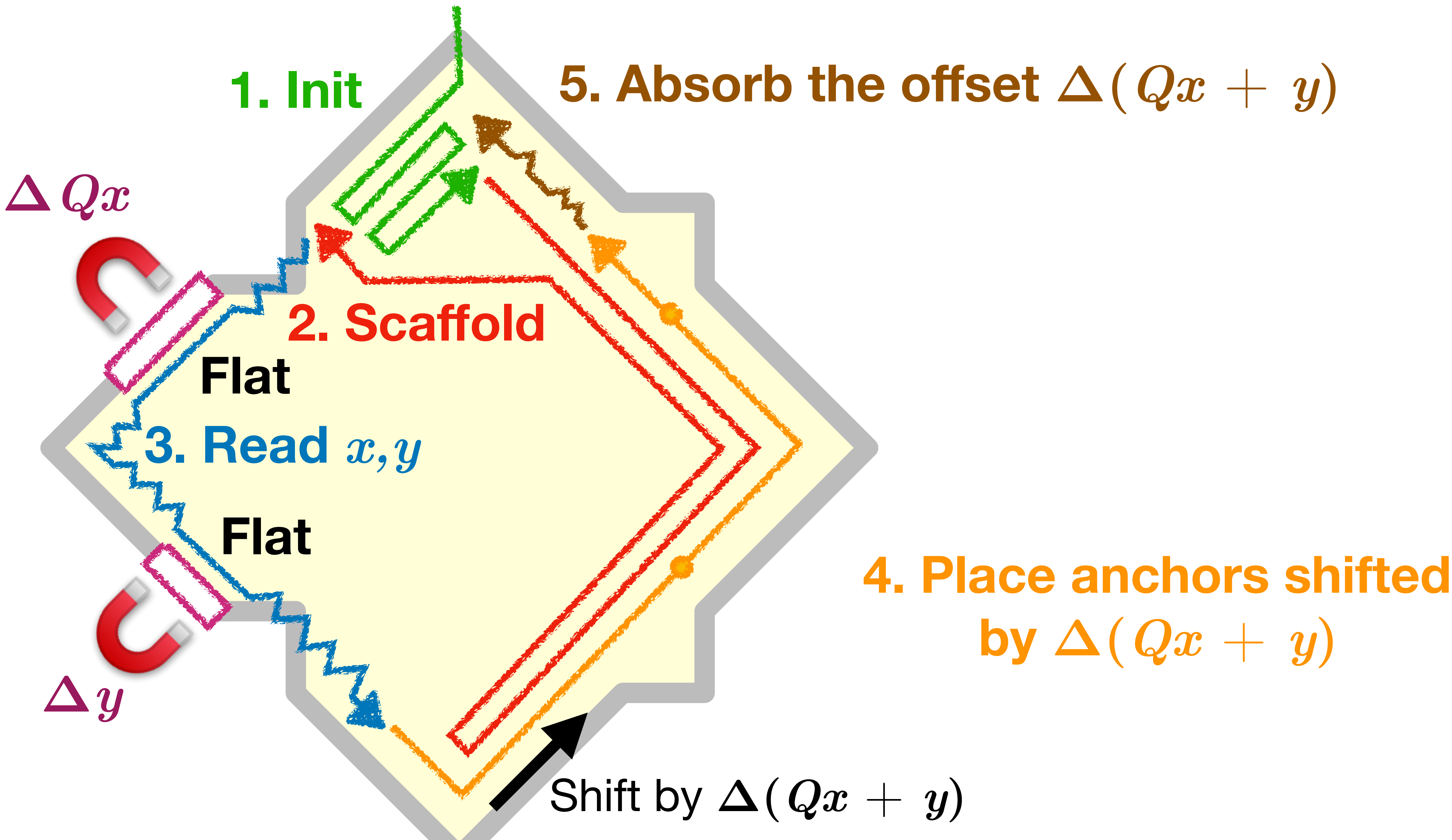



4. Place anchors shifted by $\Delta(Qx + y)$

# Read-Write mechanism

Writing $x, y$ = Placing magnets of length $\Delta Qx$ and $\Delta y$

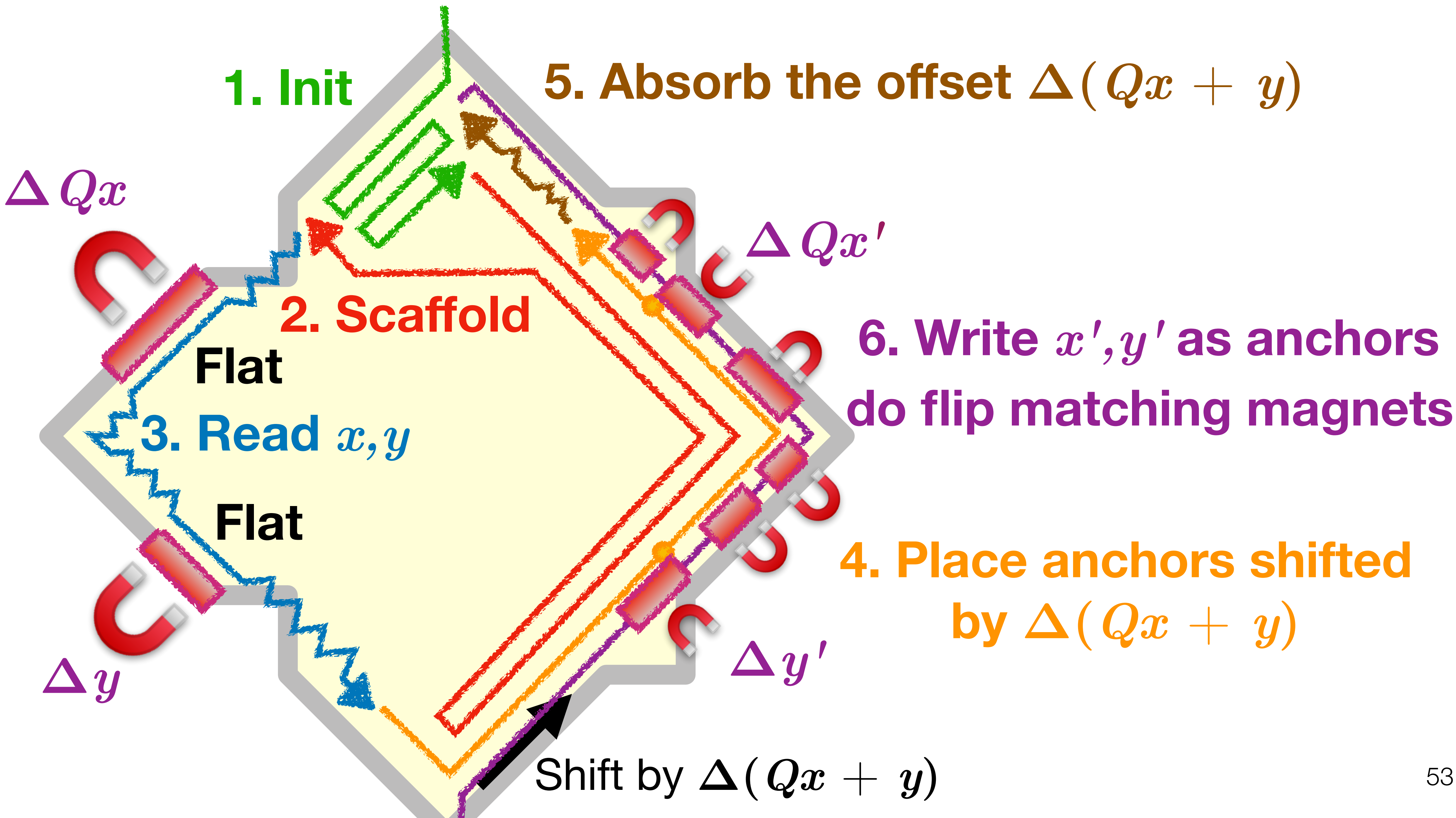
Reading $x, y$ = Create an offset of $\Delta(Qx + y)$



# Read-Write mechanism

Writing $x, y$ = Placing magnets of length $\Delta Qx$ and $\Delta y$

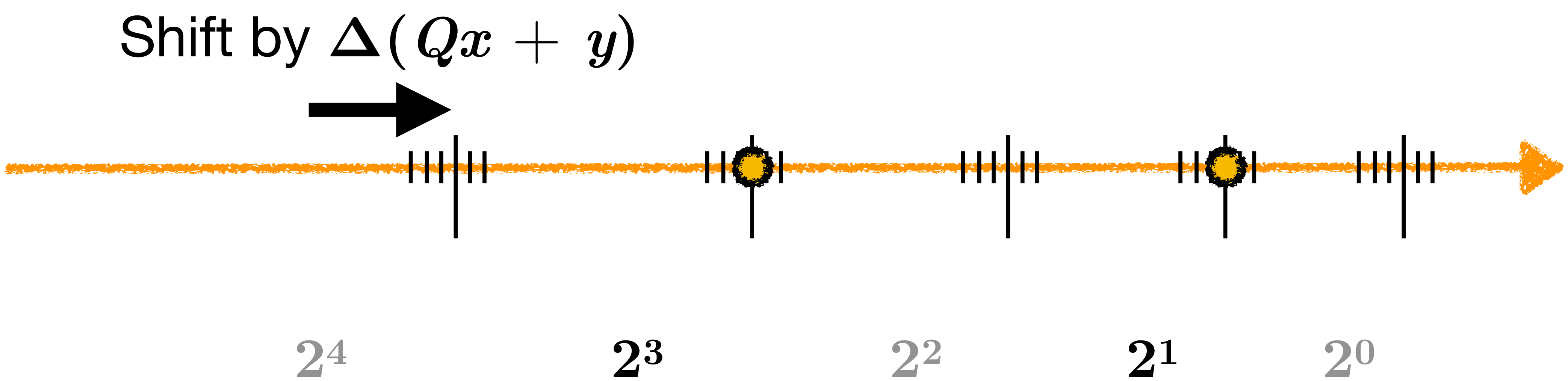
Reading $x, y$ = Create an offset of $\Delta(Qx + y)$



# Read-Write mechanism

Place the anchors as follows:

Consider $x'(x,y) = \mathbf{10} = \mathbf{2^3} + \mathbf{2^1}$

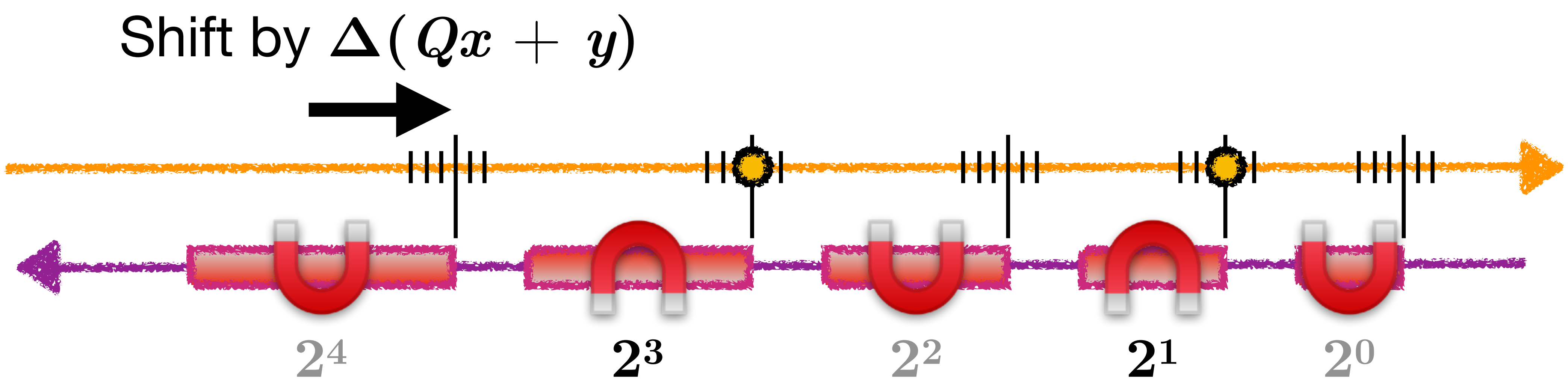



**Place anchors at $2^3$ and $2^1$ at positions shifted by by $\Delta(Qx + y)$**

# Read-Write mechanism

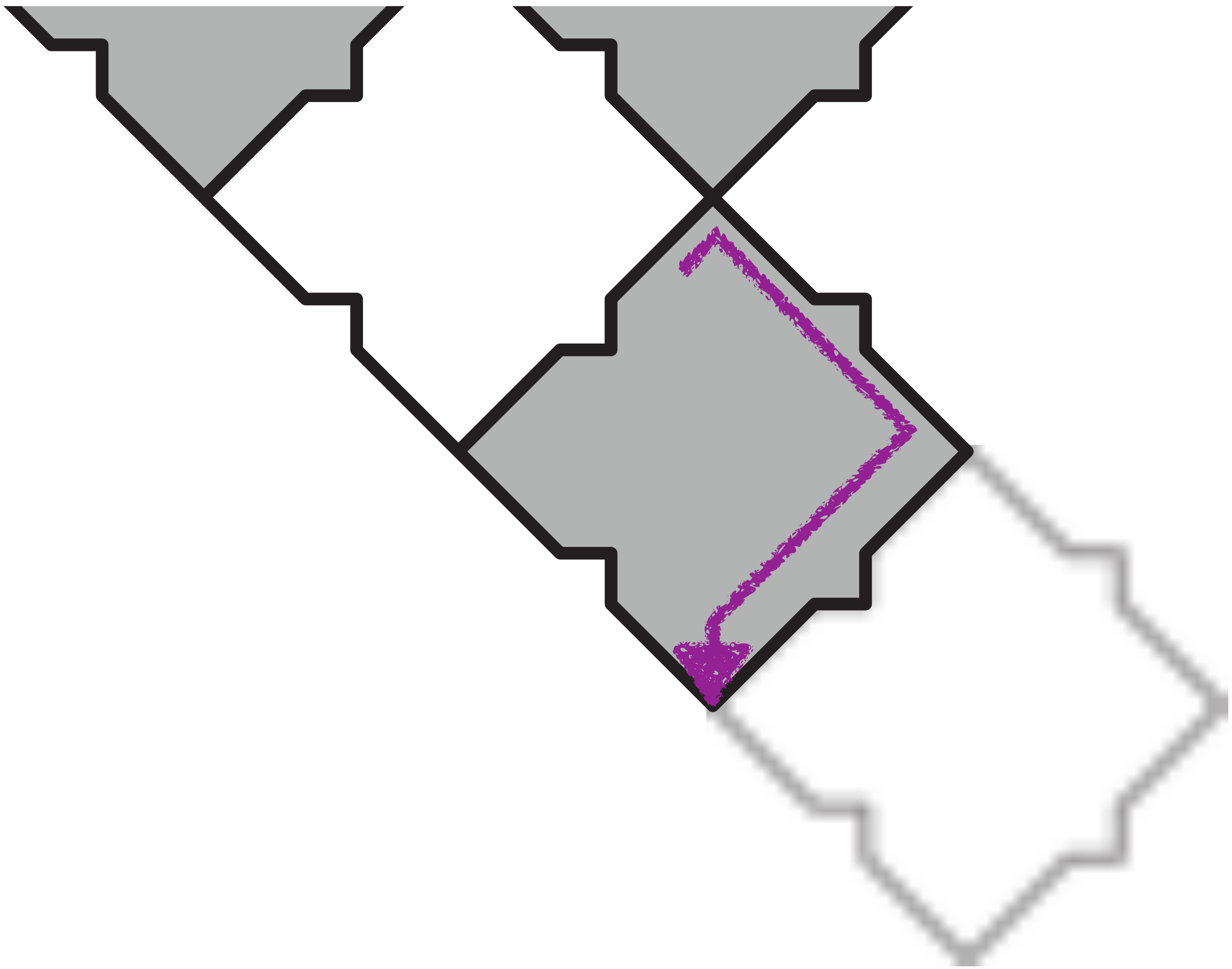
Place the anchors as follows:

Consider $x'(x, y) = \mathbf{10} = \mathbf{2^3} + \mathbf{2^1}$




**Place anchors at $2^3$ and $2^1$ at positions shifted by by $\Delta(Qx + y)$**

# Expanding the configuration

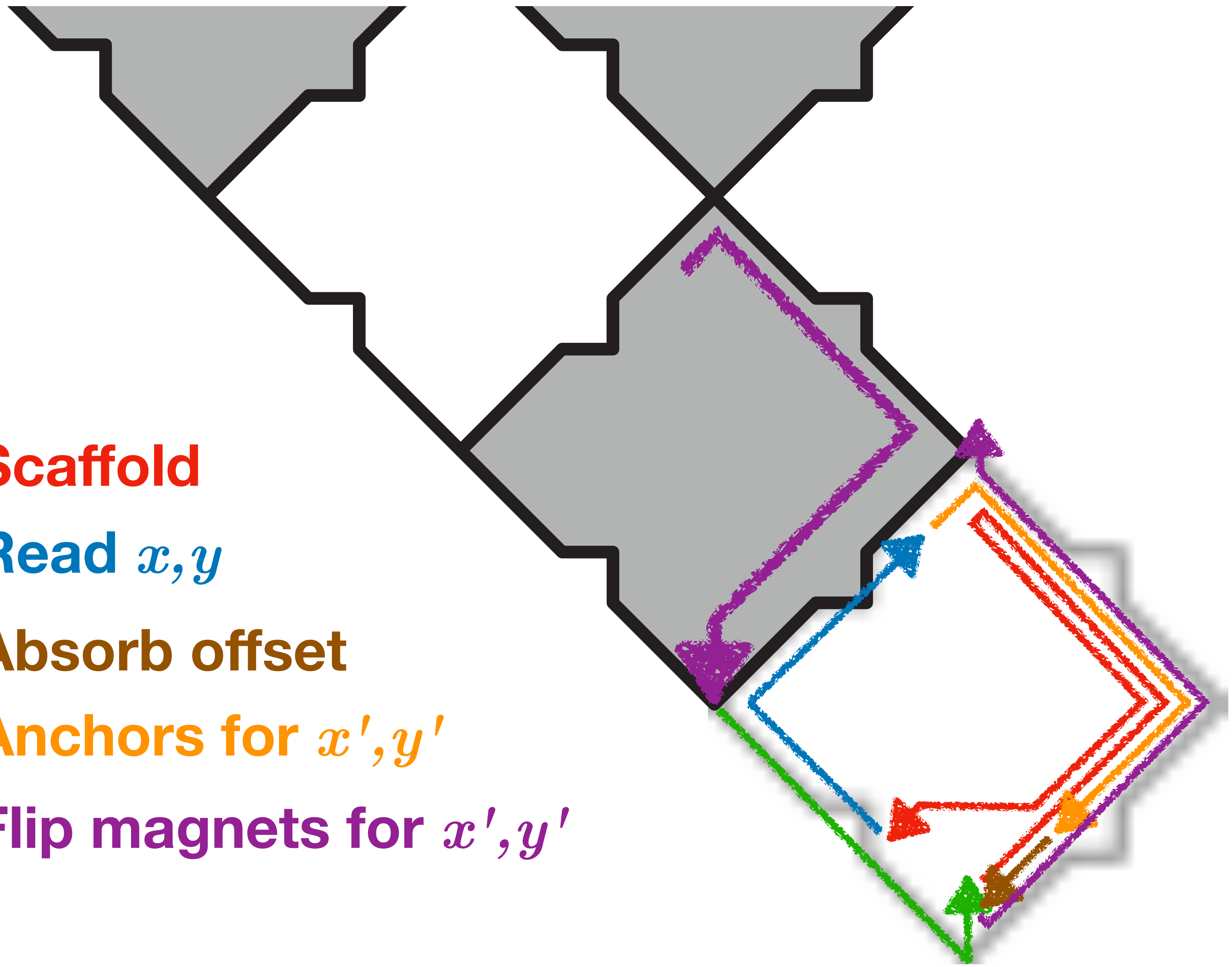


**1. "Init" unfolds to build the new cell and miror direction**

# Expanding the configuration

**1. "Init" unfolds to build the new cell and miror direction**

# Expanding the configuration



2. Scaffold

3. Read $x, y$

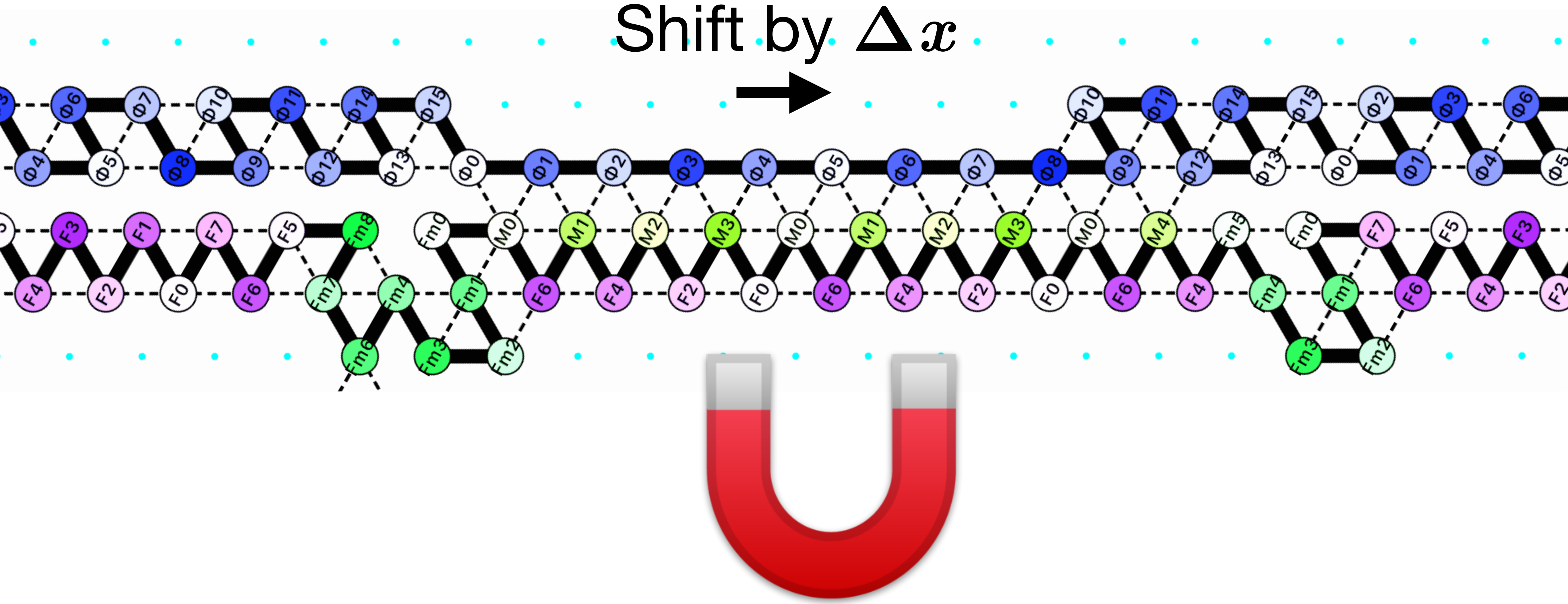
5. Absorb offset

4. Anchors for $x', y'$

6. Flip magnets for $x', y'$

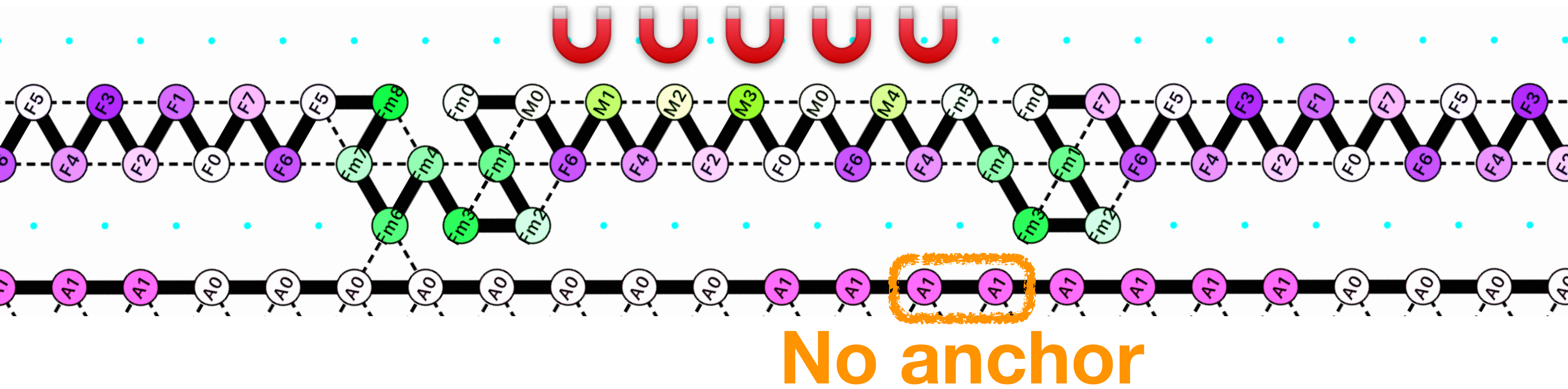
1. "Init" unfolds to build the new cell and miror direction

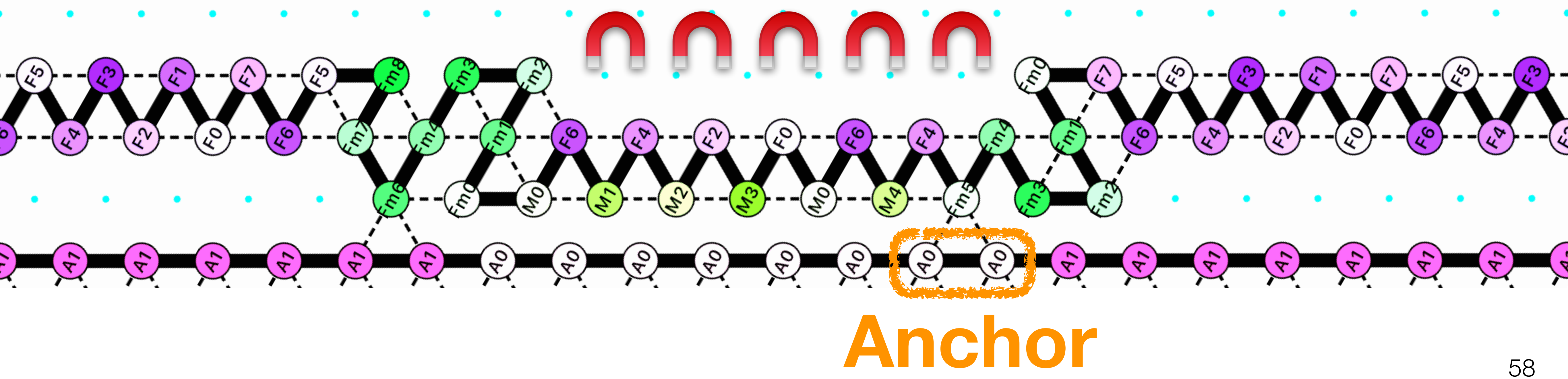
# Getting hands dirty:
# Read > Offset



# Getting hands dirty:

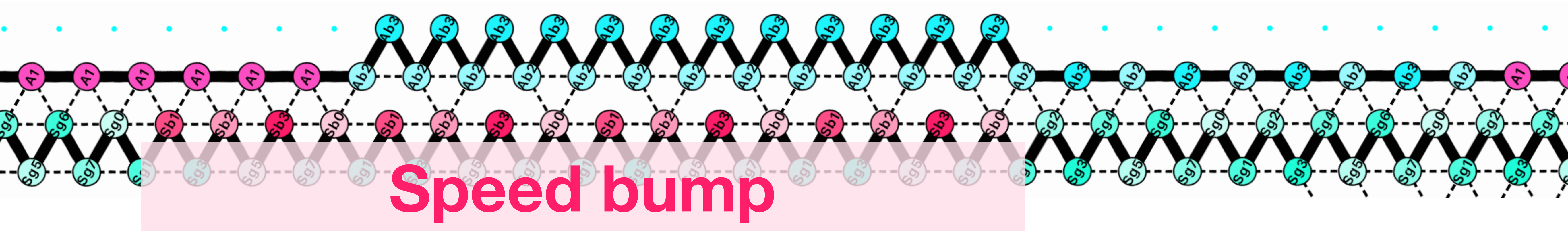
## Write: A0-Anchor flips the magnet



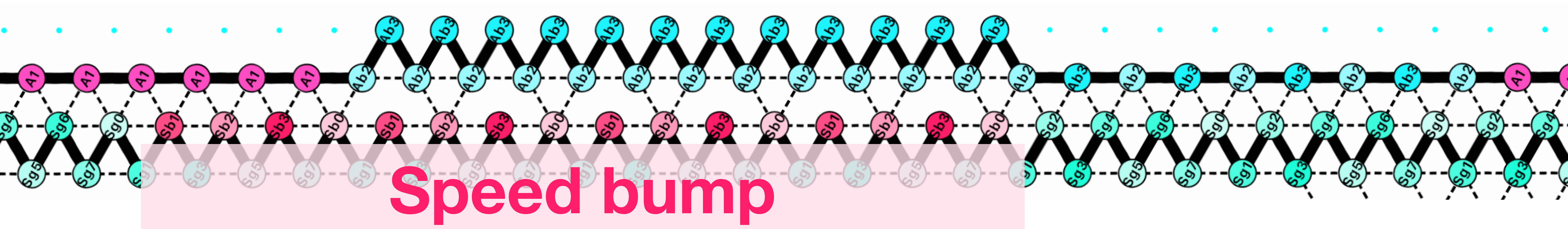
# Getting hands dirty: Absorbing Offset

Offset divided by 2

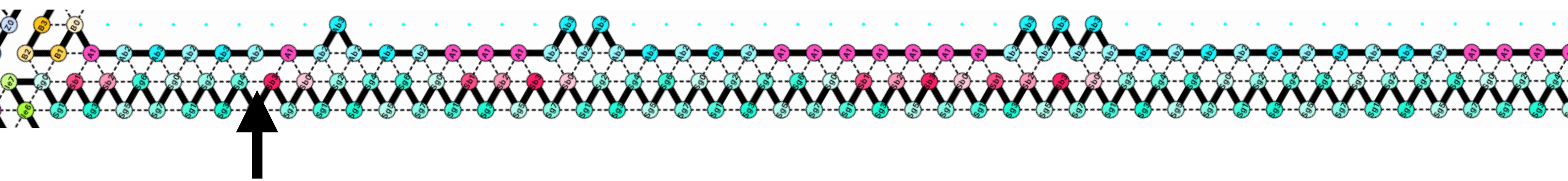


# Getting hands dirty:
# Absorbing Offset

**Offset divided by 2**




**Repeat log(Max offset) times!**

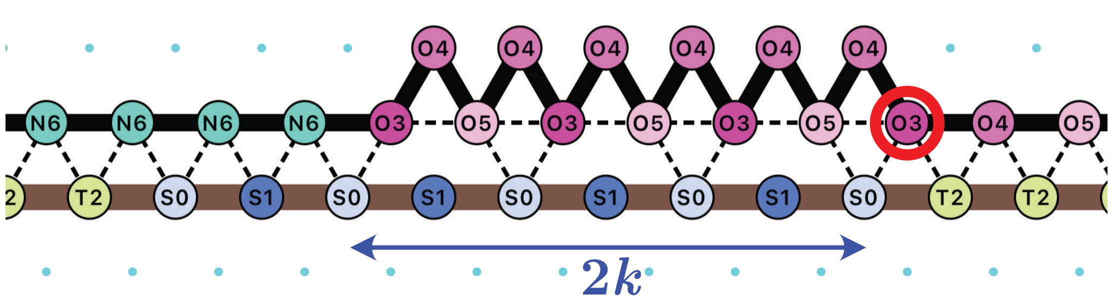


**Synchronized!**

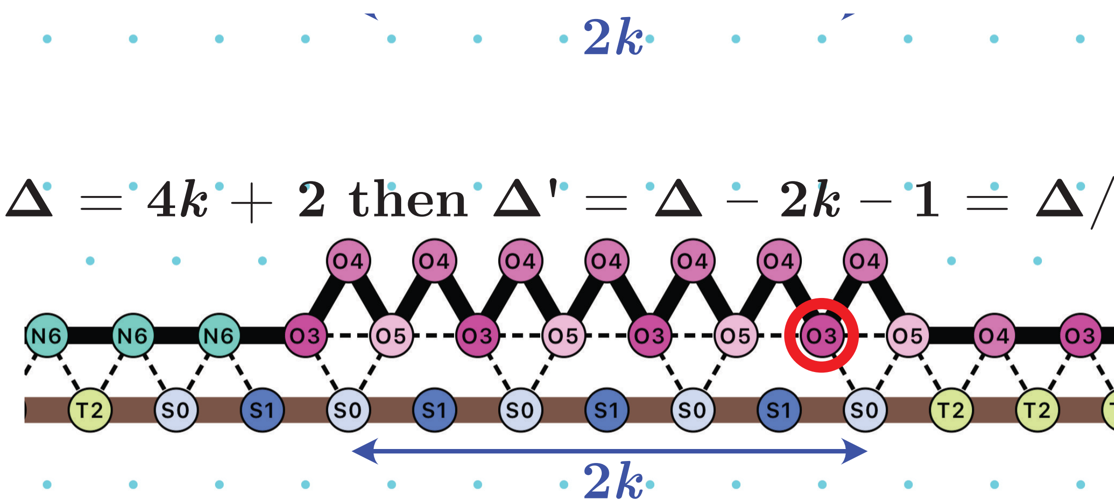
# Getting hands dirty:
# Absorbing Offset

Here, $k = 3$

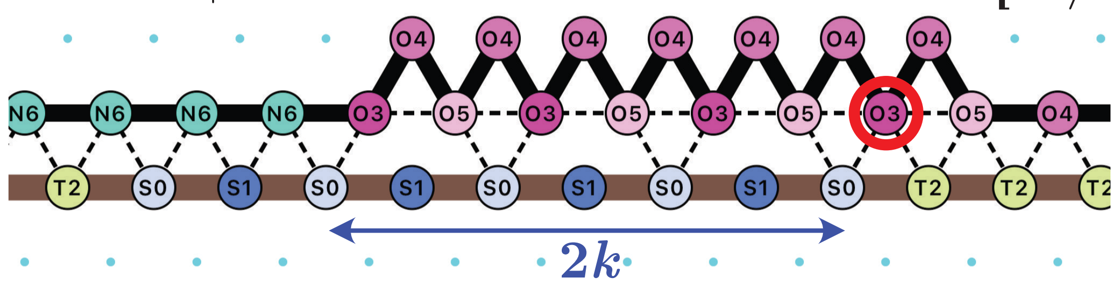
$\Delta = 4k + 0$ then $\Delta' = \Delta - 2k = \Delta/2$



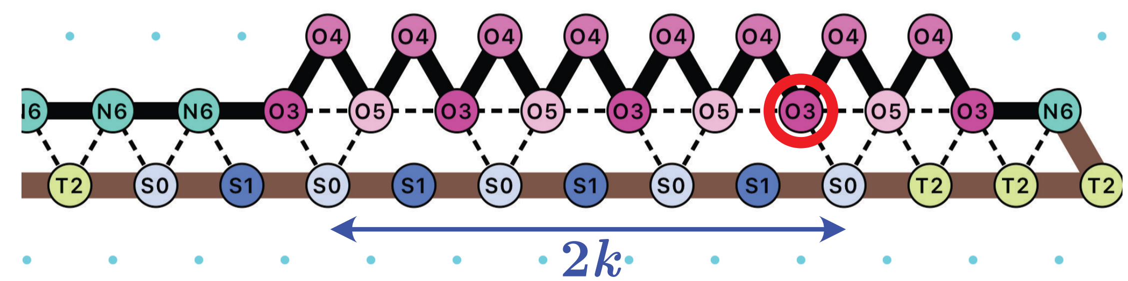

$\Delta = 4k + 2$ then $\Delta' = \Delta - 2k - 1 = \Delta/2$



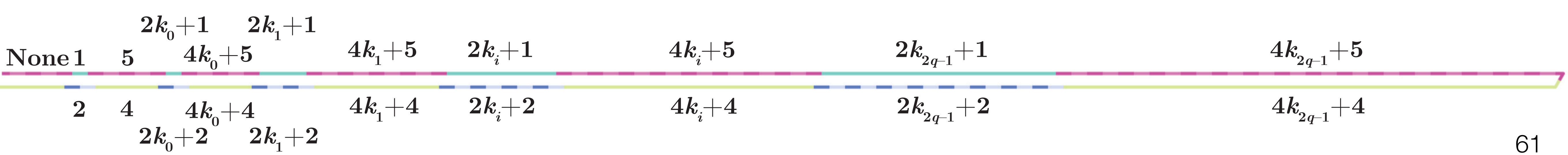

$\Delta = 4k + 1$ then $\Delta' = \Delta - 2k - 1 = \lceil\Delta/2\rceil$



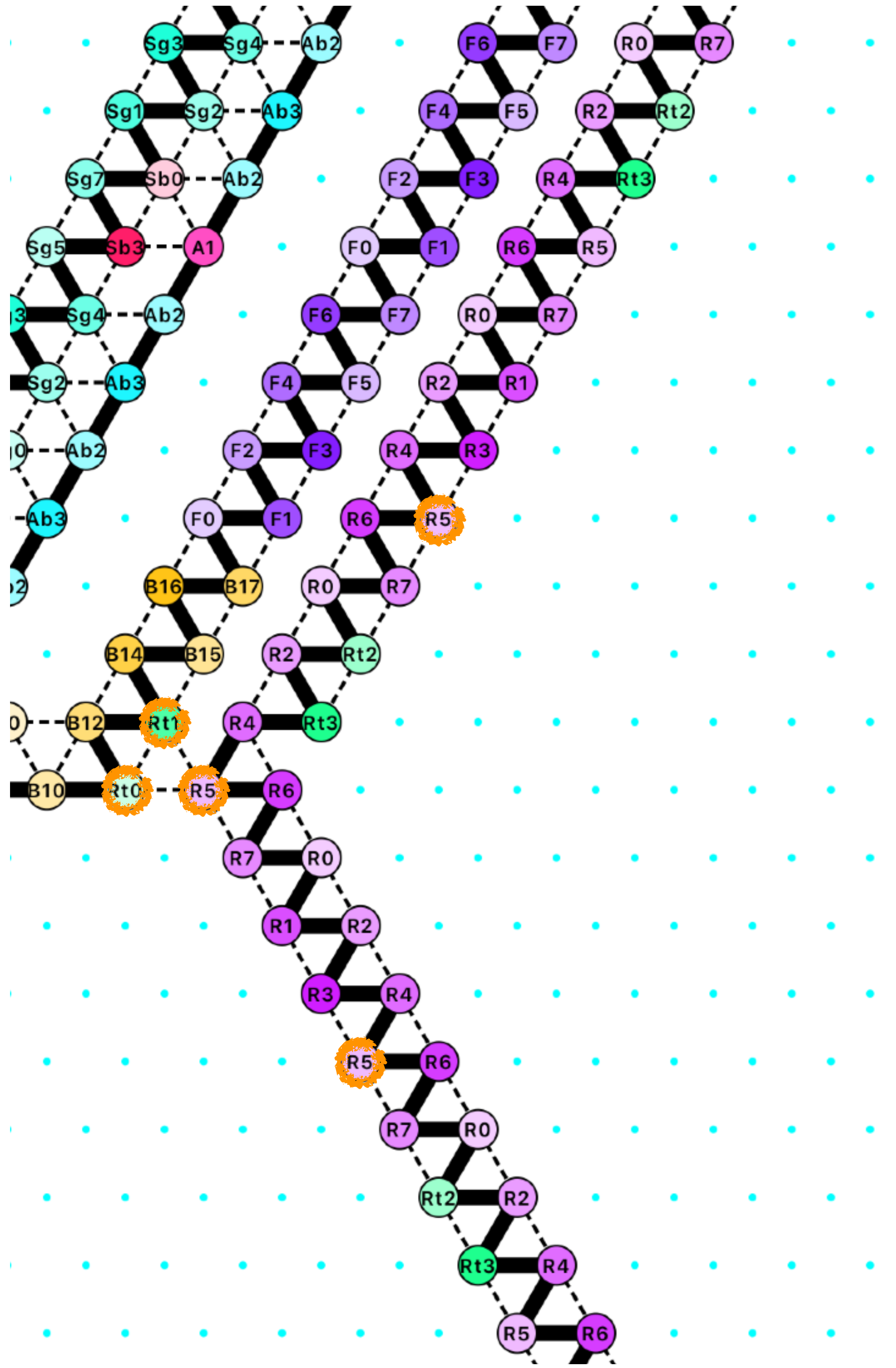

$\Delta = 4k + 3$ then $\Delta' = \Delta - 2k - 2 = \lceil\Delta/2\rceil$



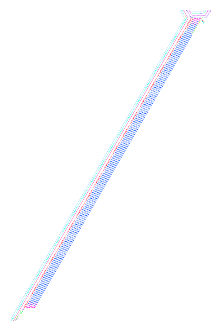

$$k_0 = 0 \text{ and } k_{i+1} = 2k_i + 1 = 2^i - 1$$



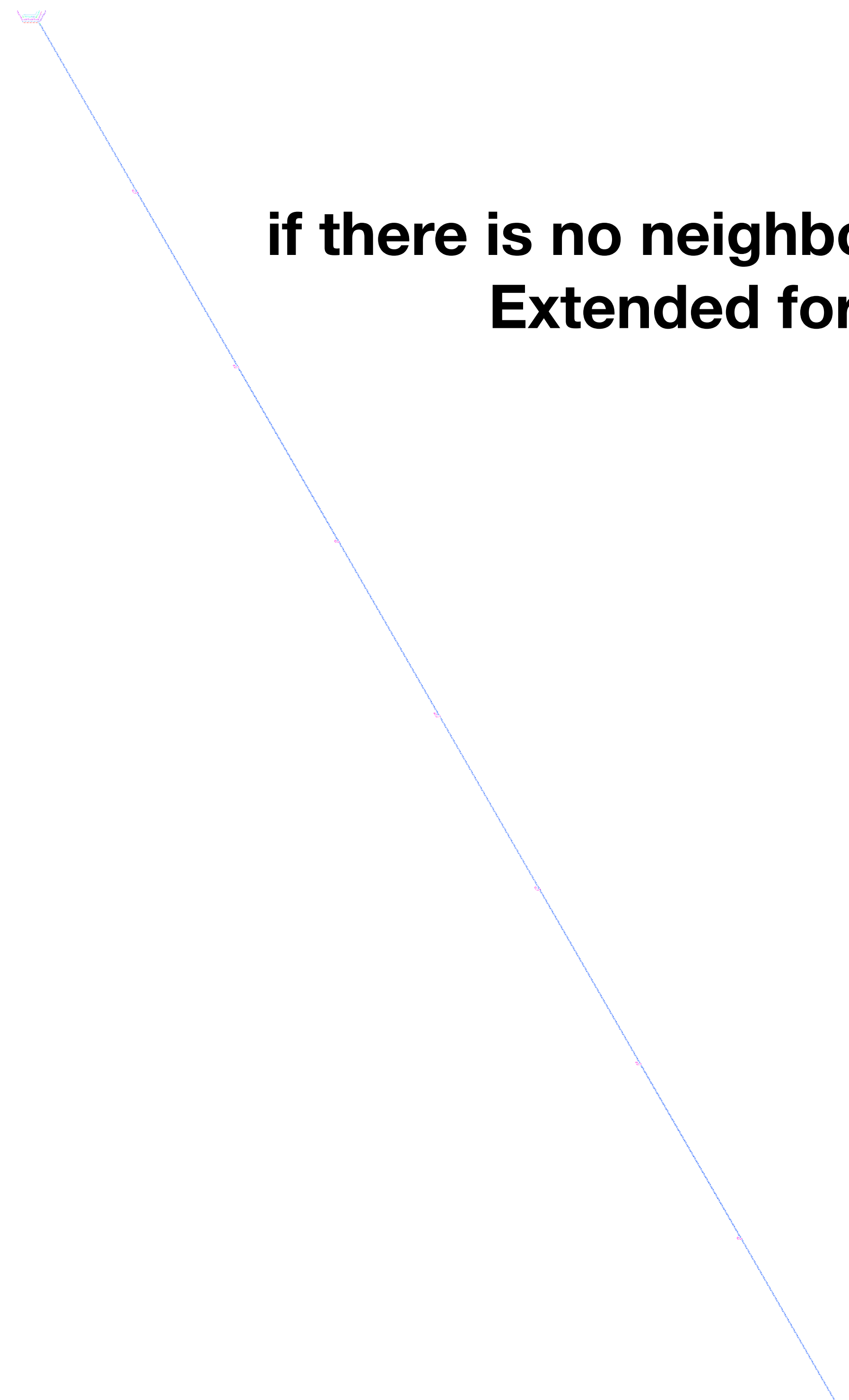
# Getting hands dirty:
# Turning Scaffold



# Getting hands dirty:
# Init: unfolding glider

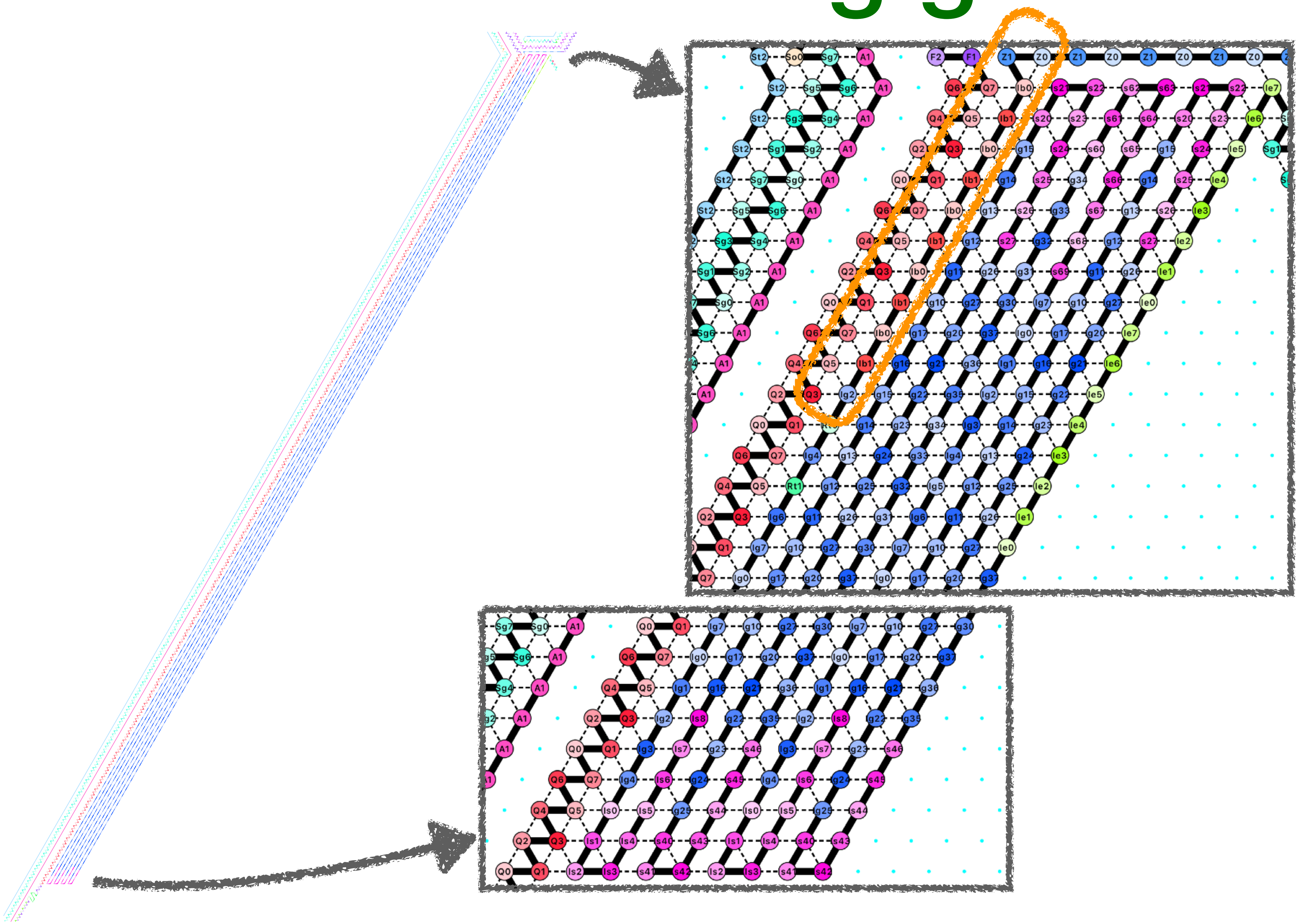


if there is a neighboring cell :
**Compact form**

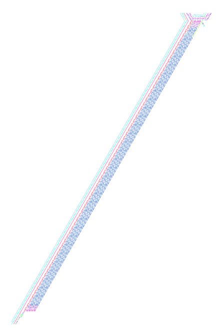


**if there is no neighboring cell :**
**Extended form**

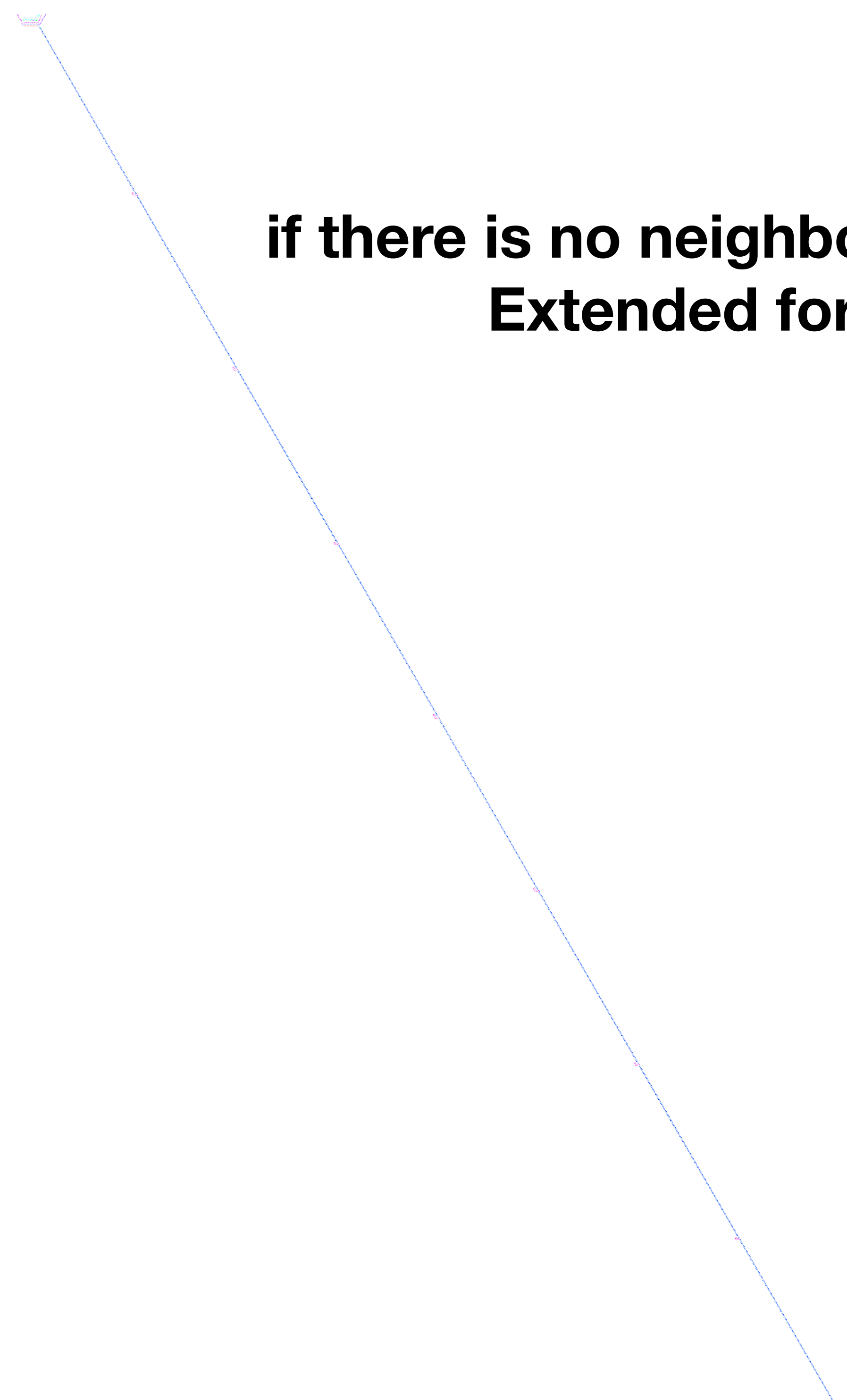
# Getting hands dirty:
## Init: unfolding glider



# Getting hands dirty:
# Init: unfolding glider

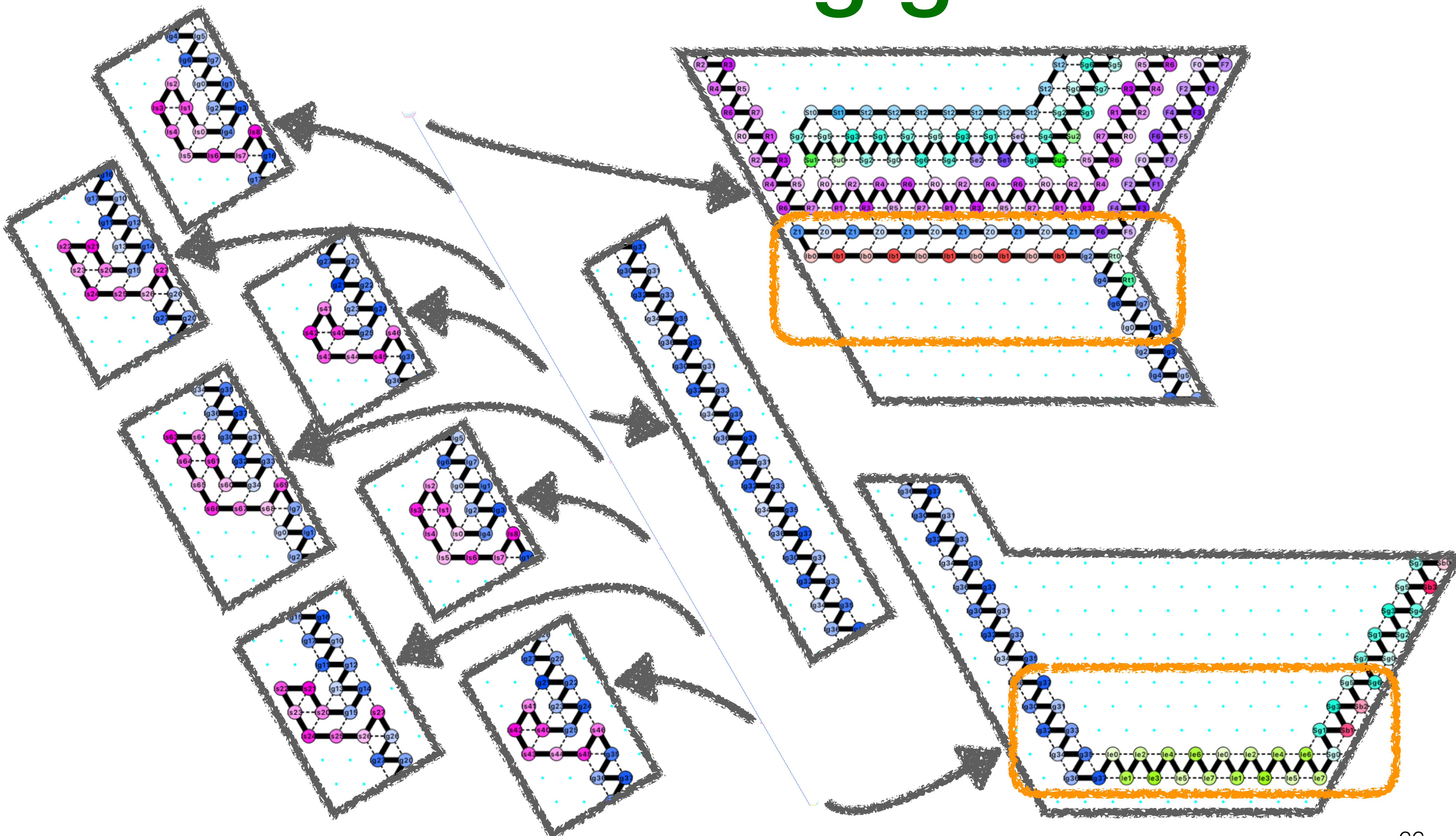


if there is a neighboring cell :
Compact form



if there is no neighboring cell :
Extended form

# Getting hands dirty:
# Init: unfolding glider



# Conclusion

**Our results**

- Oritatami system can simulate intrinsically any 1D cellular automata
- "Mechanical" tools for designing simpler oritatami system

**Next…**

- An oritatami programming language?
- How to implement RAM? Loops? Concatenation? Subroutine call?
- Design a program simple enough to be *implemented in wet-lab?*