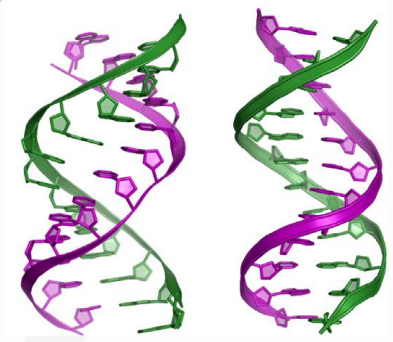
RNA Co-transcriptionality

Shinnosuke Seki
@ FLAT seminar Nov. 20, 2024

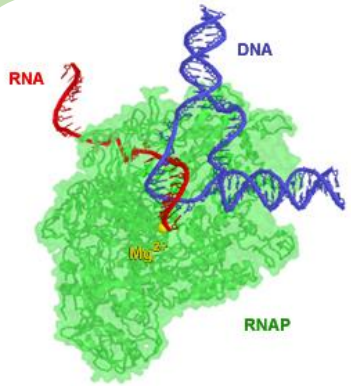
ORITATAMI LAB
@ENS Lyon and UEC Tokyo


Building blocks for life.
LEGO



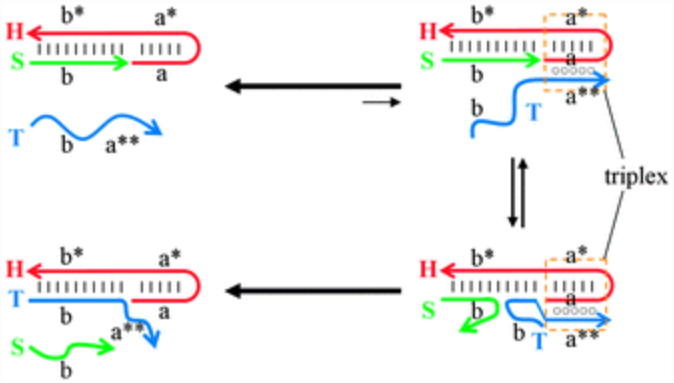
Substrate

+

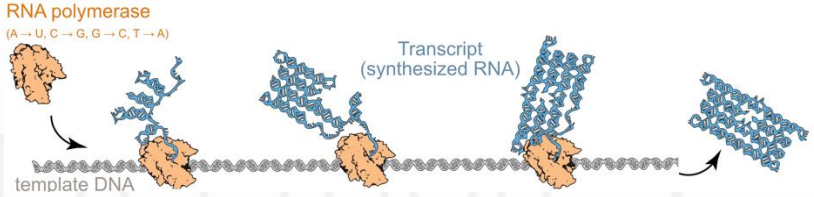


RNA polymerase

Natural Computers (enzymes) if needed



(DNA triplex) toehold-mediated strand displacement

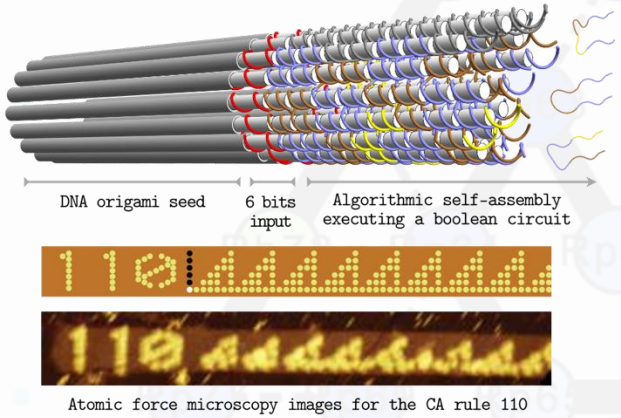


RNA co-transcriptional folding

⋮

Programming Platforms

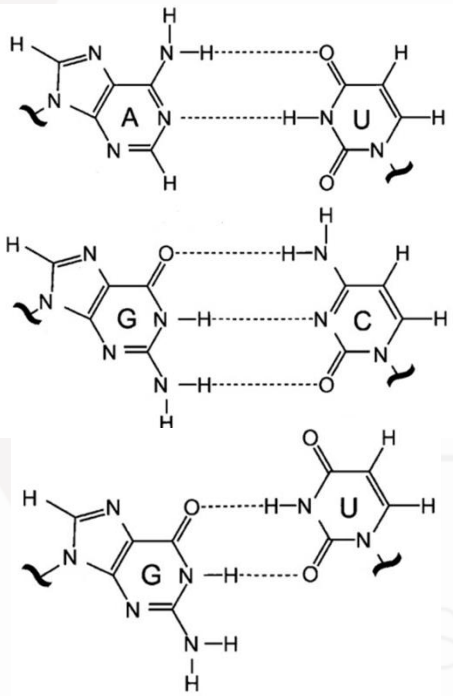
=



DNA-based rule-110 cellular automata (Woods et al. 2019)

(Re-)programmable Molecular Computers

Substrate



DNA/RNA are a chain of 4 kinds of bases A, G, C, T/U.

They may *hybridize* with each other primarily by 3 types of base pairs (shown left) between purines (A, G) and pyrimidines (C, T/U).

-- AcGucAu ->

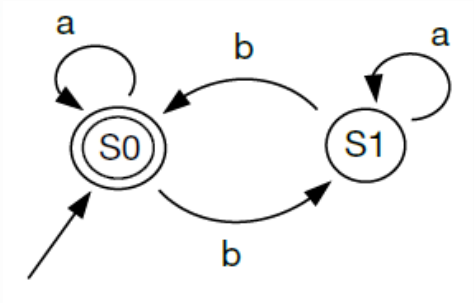
<- uGcGGcA --

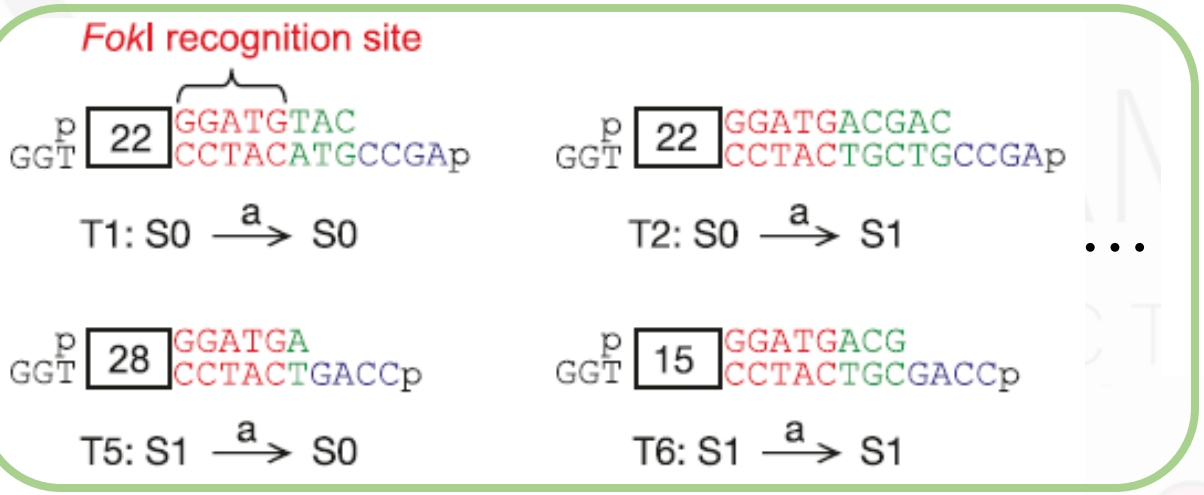
⇔

-- AcGucAu ->

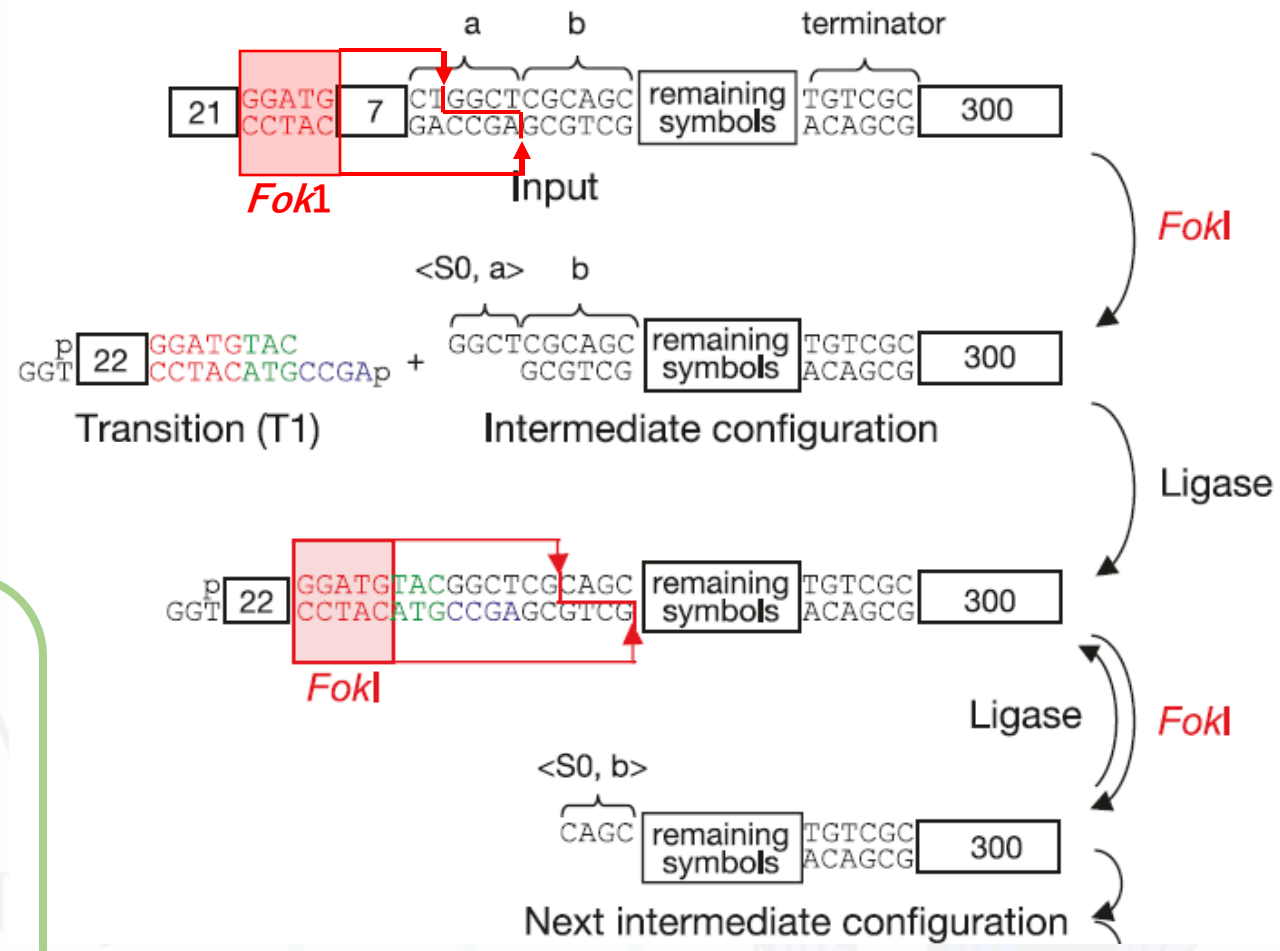
<- uGcGGcA --

*Fok*1-driven FA





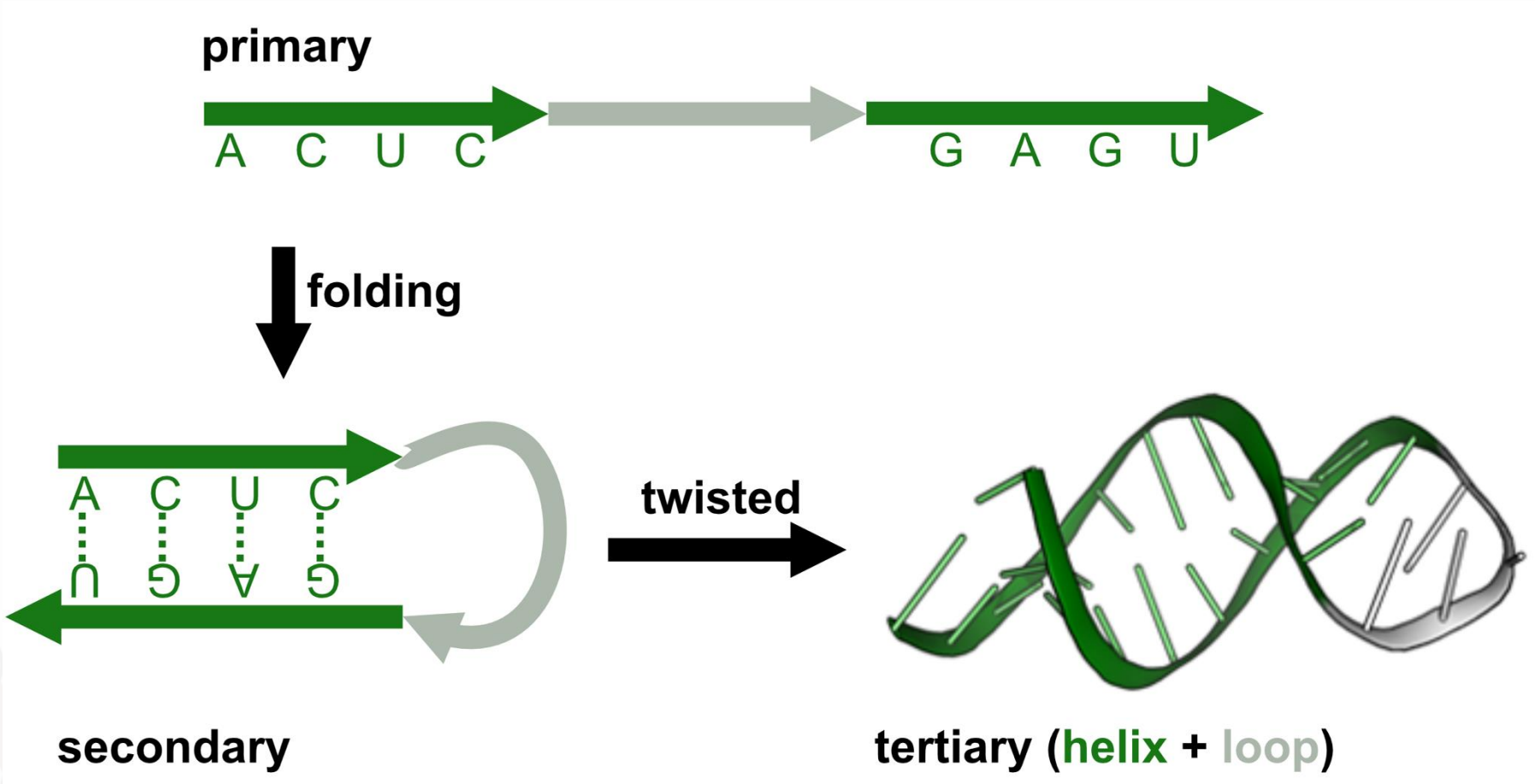
Transition molecules



Benenson et al. *Nature* 414, pp.430-434, 2001

Helix

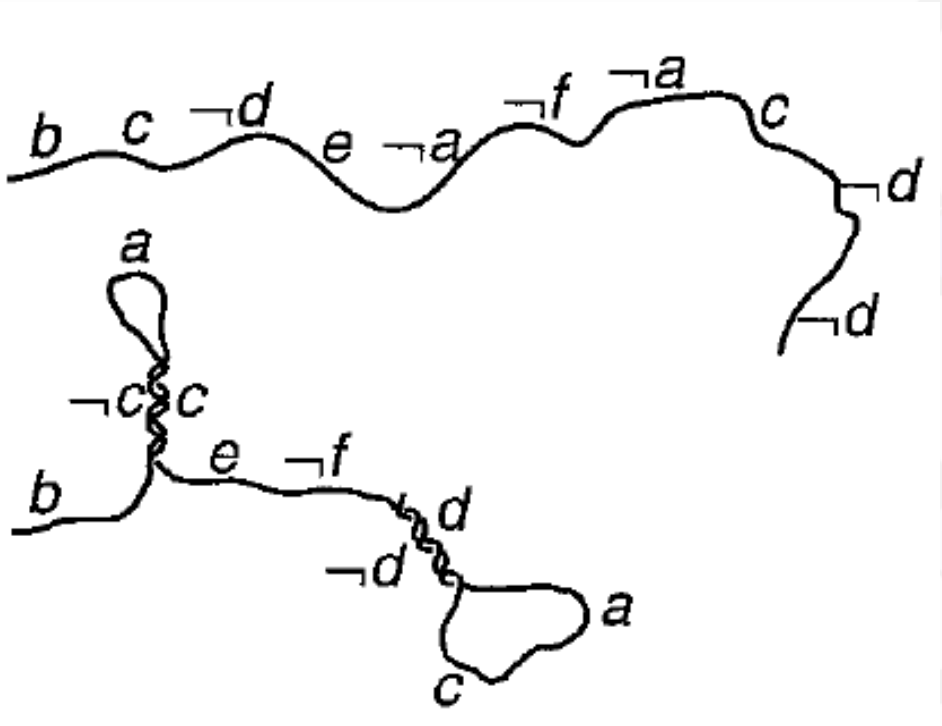
Atoms of single-stranded RNA structures



Enzyme-driven single-stranded computation

A molecular SAT solver.

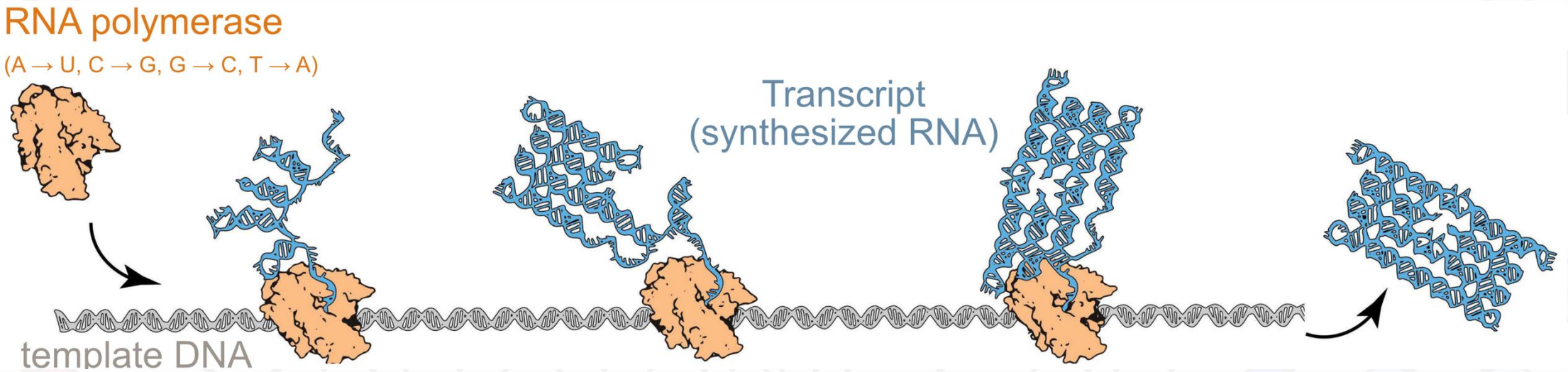
1. Literals are assigned with DNA sequences in such a way that
   - x and ¬y hybridize with each other iff x = y
   - x and y never hybridize, or neither do ¬x and ¬y
2. A SAT instance is programmed as a pool made of DNA sequences obtained by choosing one literal from each clause and catenating the corresponding DNA sequences.
3. Such a DNA sequence forms a hairpin iff both x and ¬x of a variable are involved.
4. A restriction enzyme cuts a hairpin.
5. It suffices to check if some DNA sequence has "survived."

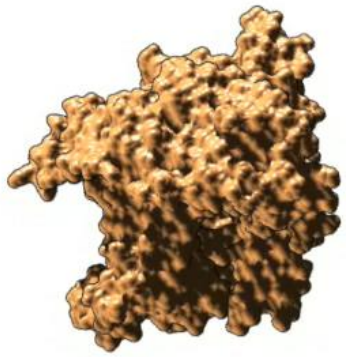


Sakamoto et al. *Science* 288, pp.1223-1226, 2000

RNA co-transcriptionality

Programmable platform for in vitro/vivo computations


RNA polymerase
(A → U, C → G, G → C, T → A)
Transcript
(synthesized RNA)
template DNA



Geary, Rothemund, and Andersen, *Science* **345** (6198): 799-804, 2014

RNA polymerase

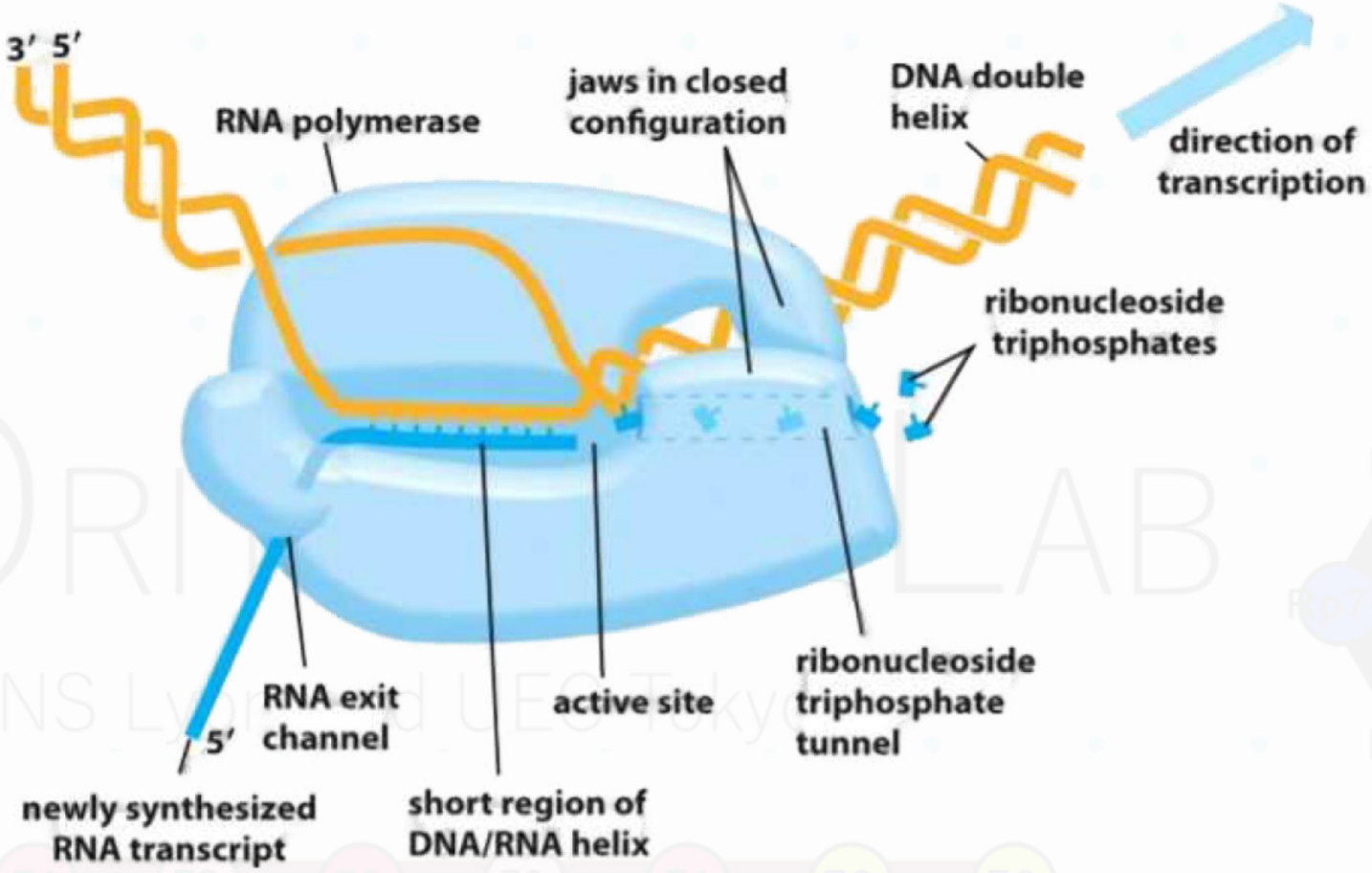
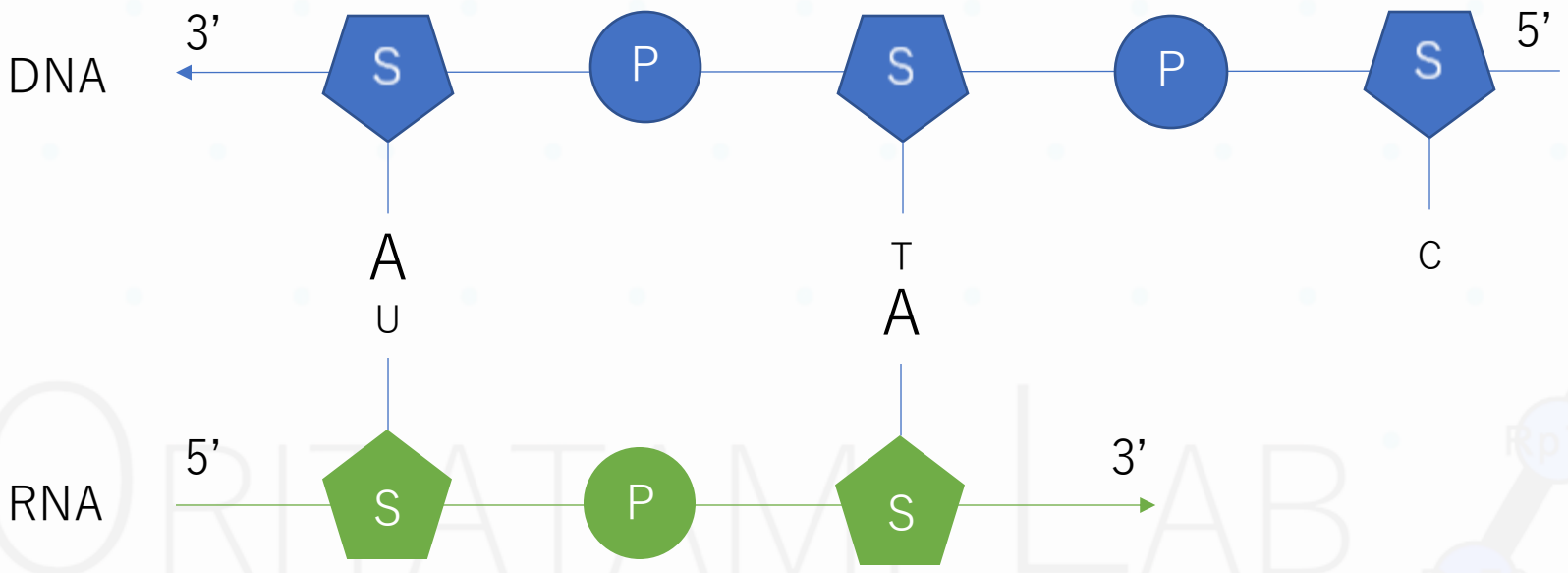
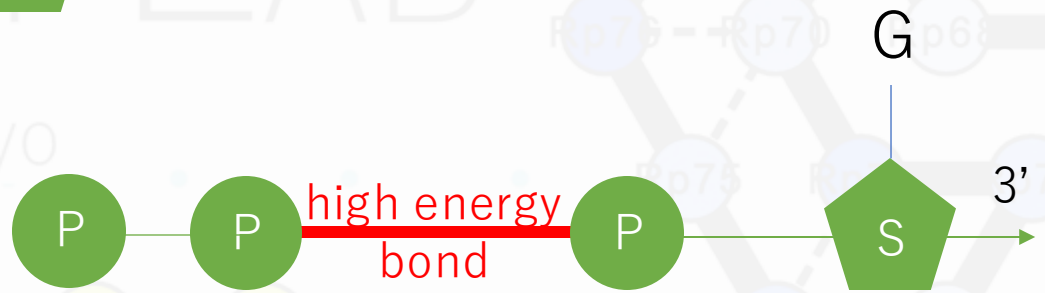


Figure 7-7 *Essential Cell Biology* (©Garland Science 2010)

RNA polymerase

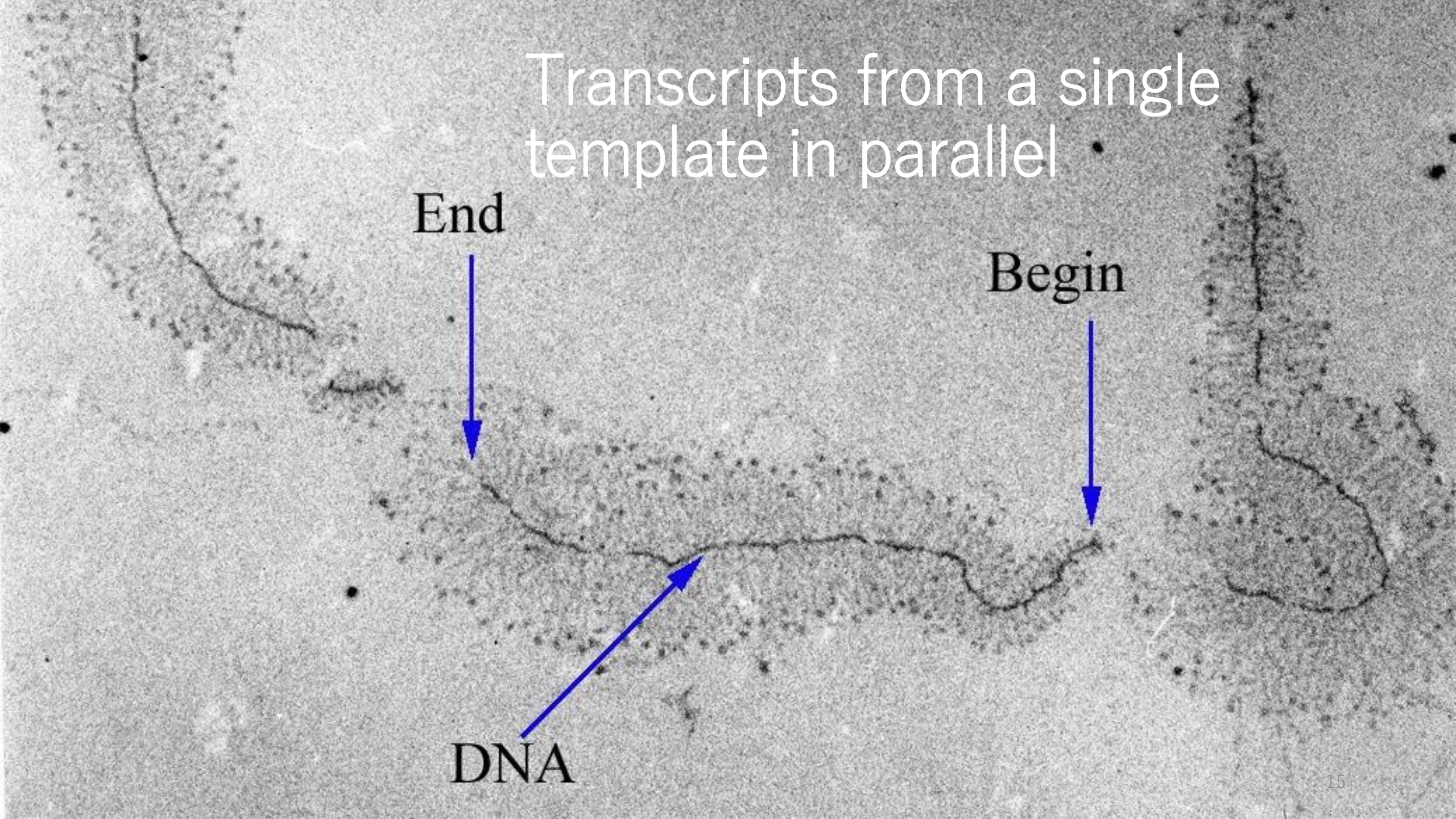


A → U, C → G, G → C, T → A



The idea of this diagram is from Feynman Lectures on Computation, 1996

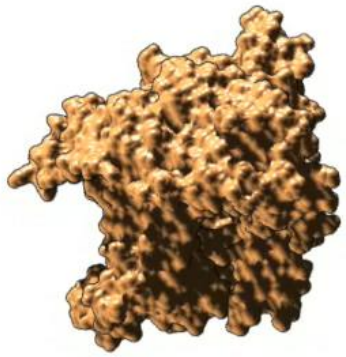
Transcripts from a single template in parallel



RNA origami

architecture for hard-coding a structure into CF

```
int main() {
    while (1) {
        std:cout << “Hello RNA World!!”;
    }
    return 0;
}
```



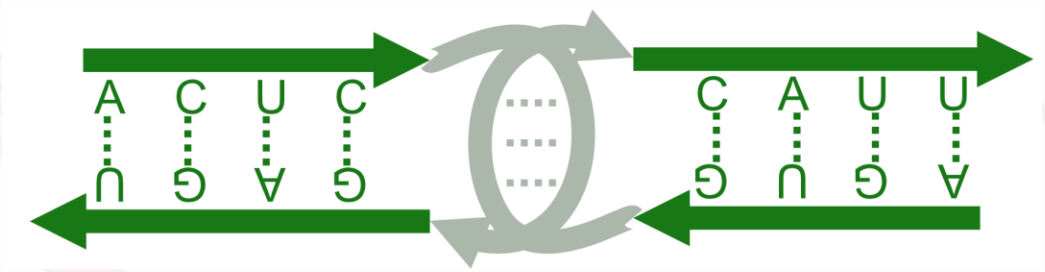
Geary, Rothemund, and Andersen, *Science* **345** (6198): 799-804, 2014

Helix

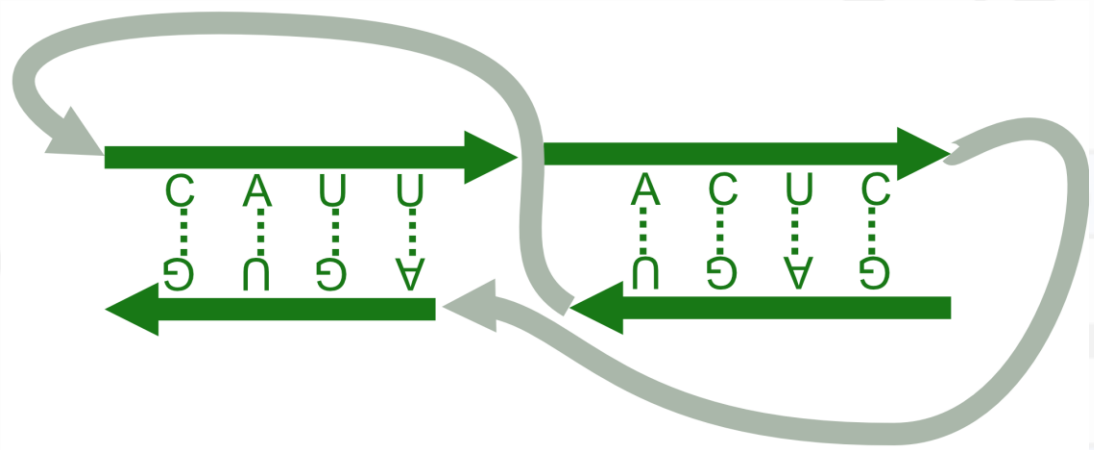
co-axial stacking

Helices are stabilized **co-axially** via base-stacking at their interface.

Two common motifs involving co-axial stacking are:



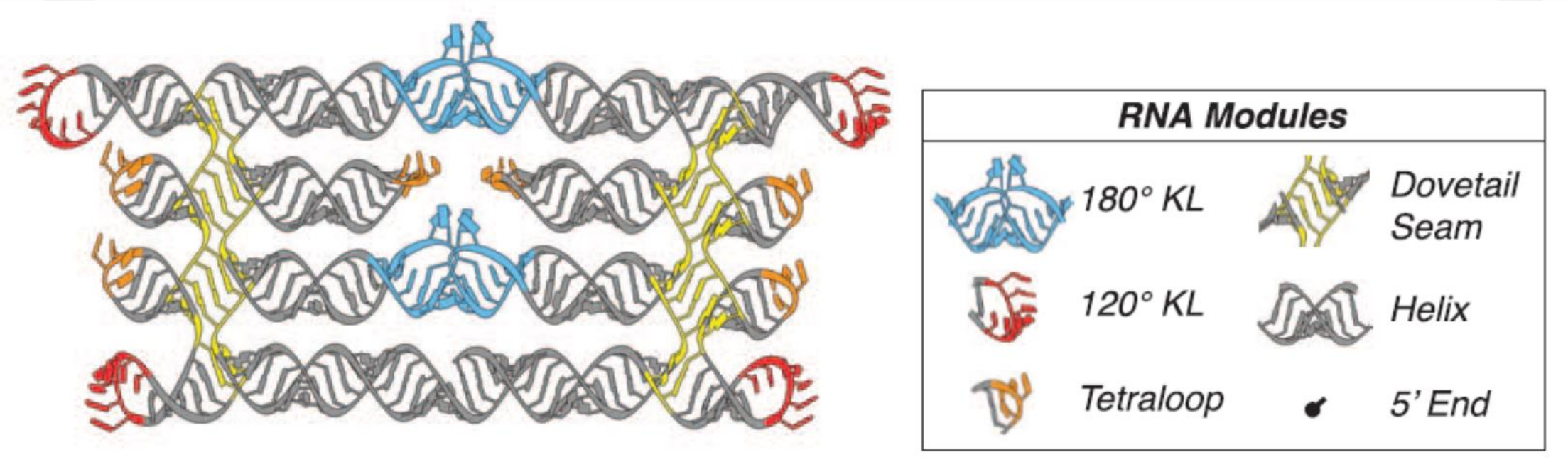
Kissing loop



Pseudoknot

RNA origami

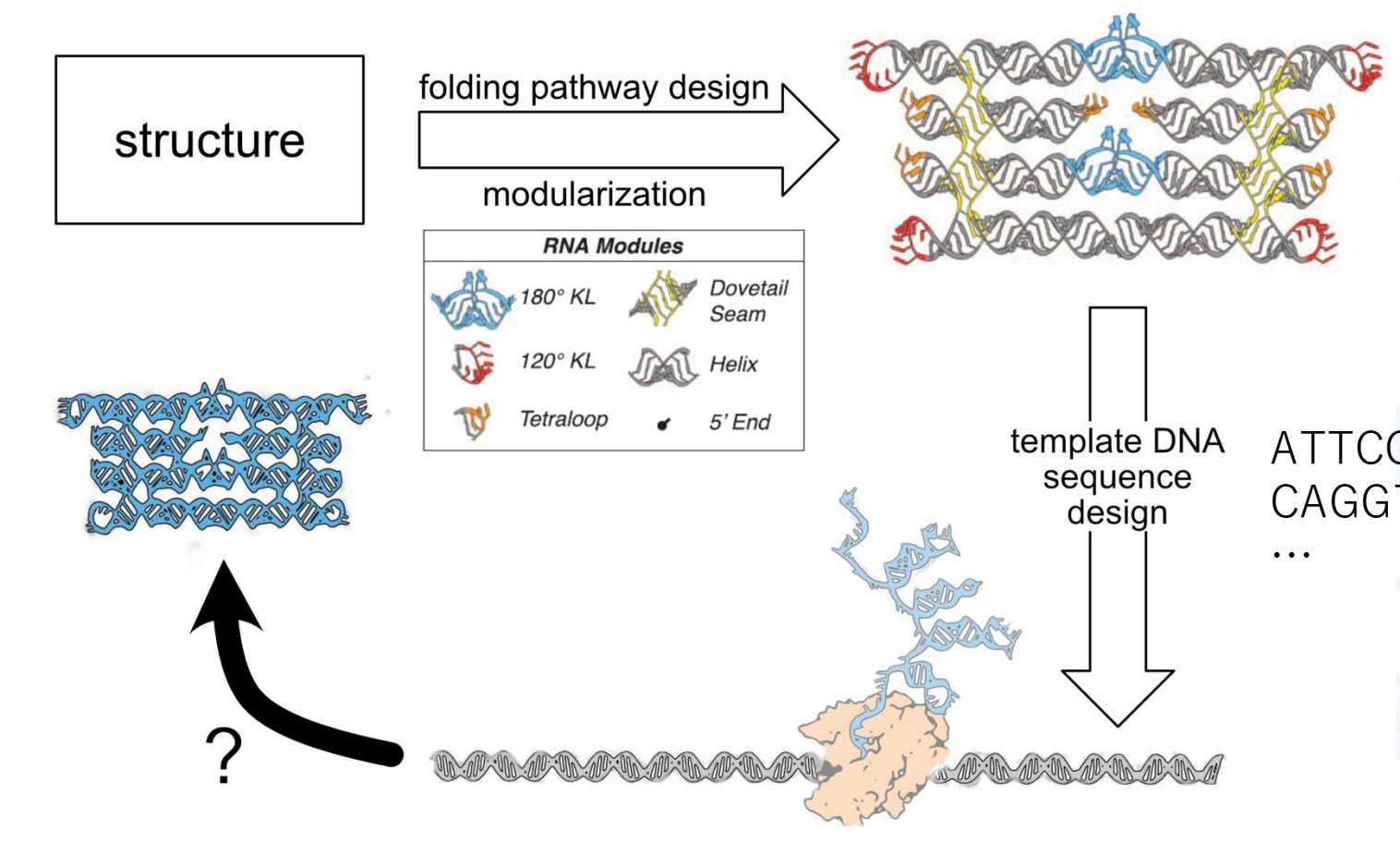
modular design of an RNA tile



Geary, Rothemund, and Andersen, *Science* **345** (6198): 799-804, 2014

RNA origami

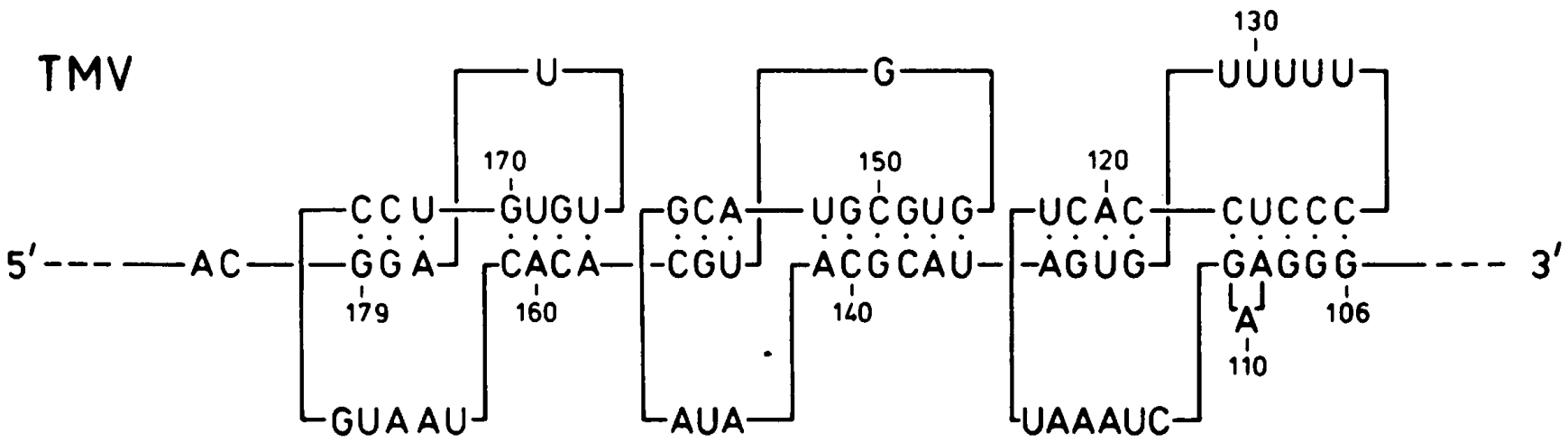
hard-coded CF



Geary, Rothemund, and Andersen, *Science* **345** (6198): 799-804, 2014

Helix

co-axially stacked into a viral backbone

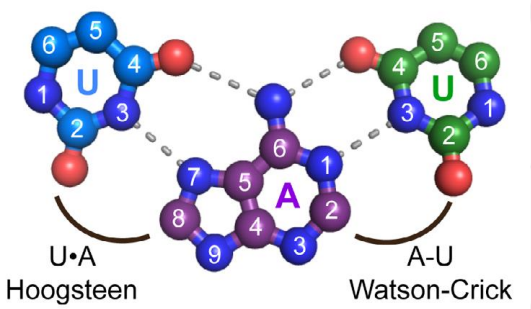


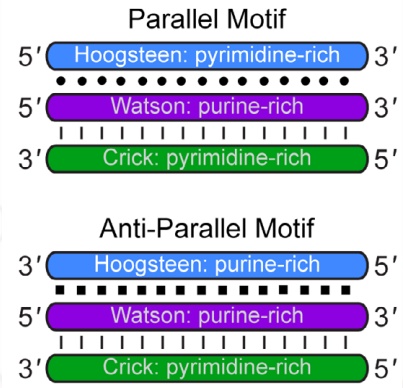
Helix

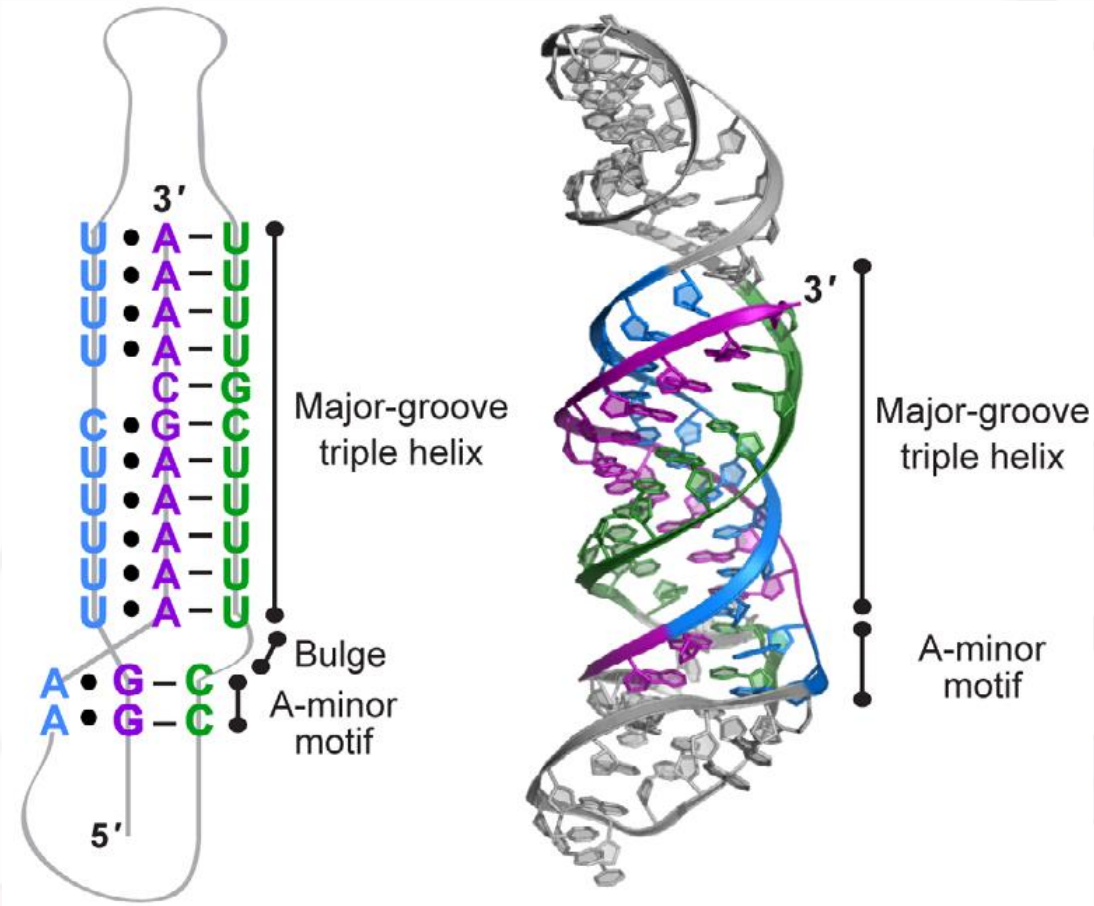
RNA triple helix

RNA sequences are capable of keeping their 3'-end away from their 5'-end, thus,

- folding into non-tree structures, transcending the bound of CFL;
- avoiding to be degraded by ribonuclease (RNase).



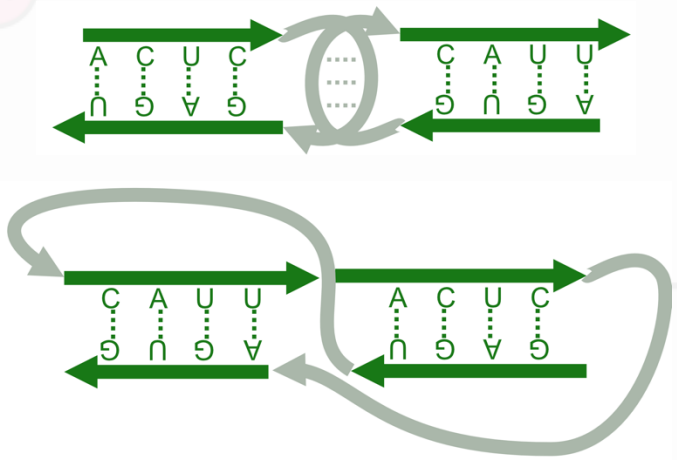




J. A. Brown. *WIREs RNA*. 2020;11:e1598

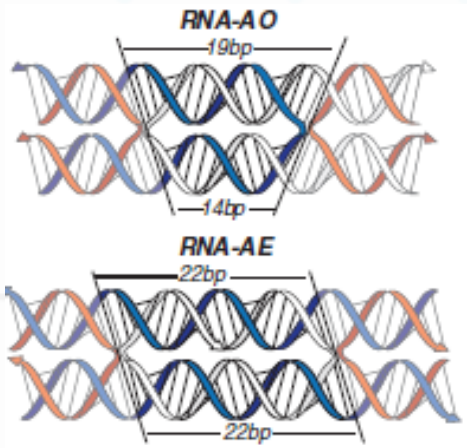
RNA Origami to Oritatami

Co-axial stackings



+

Cross-over motifs

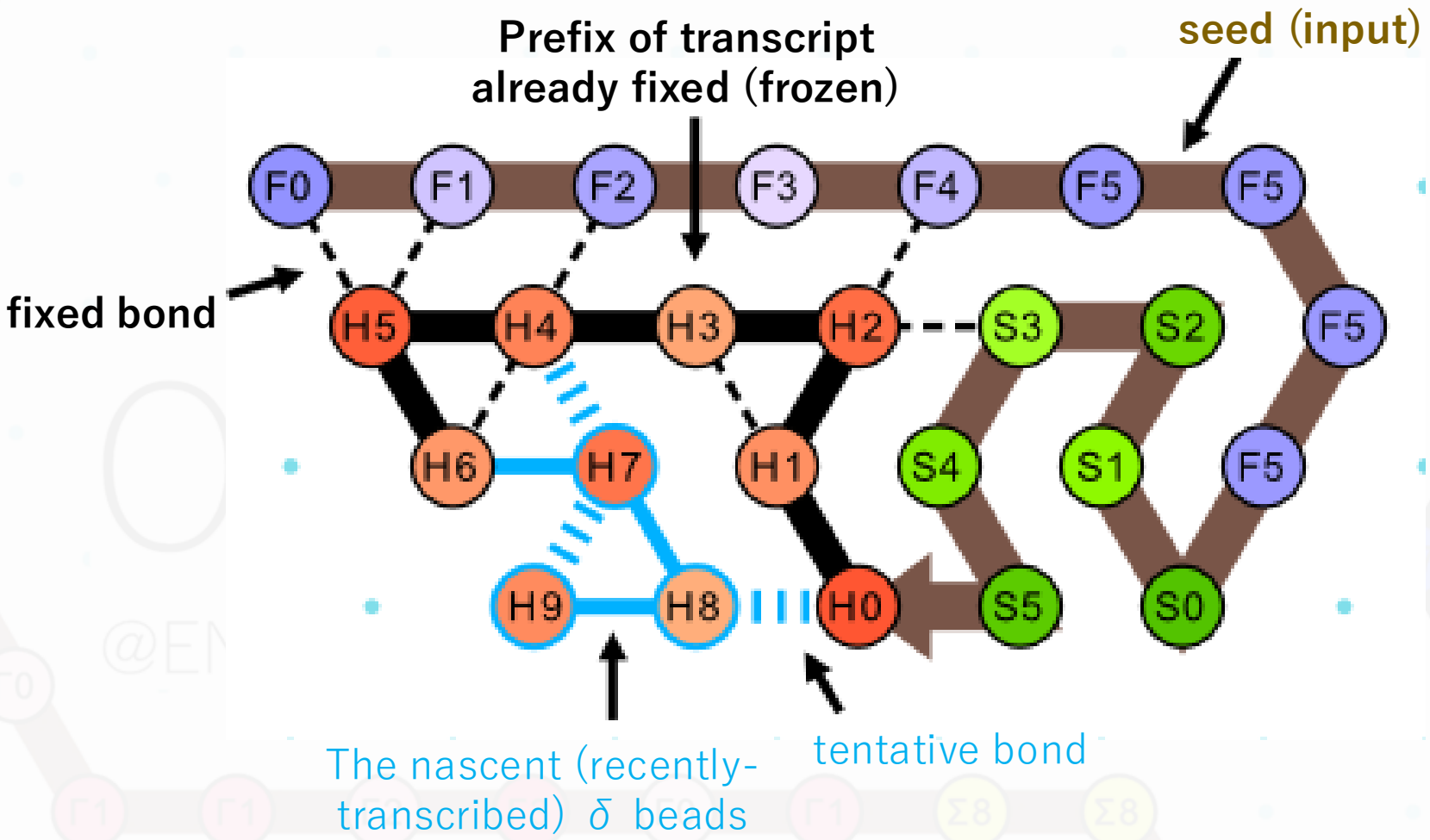


=

Quasi-2D engineering of co-transcriptional folding

Oritatami

model of CF-driven computing

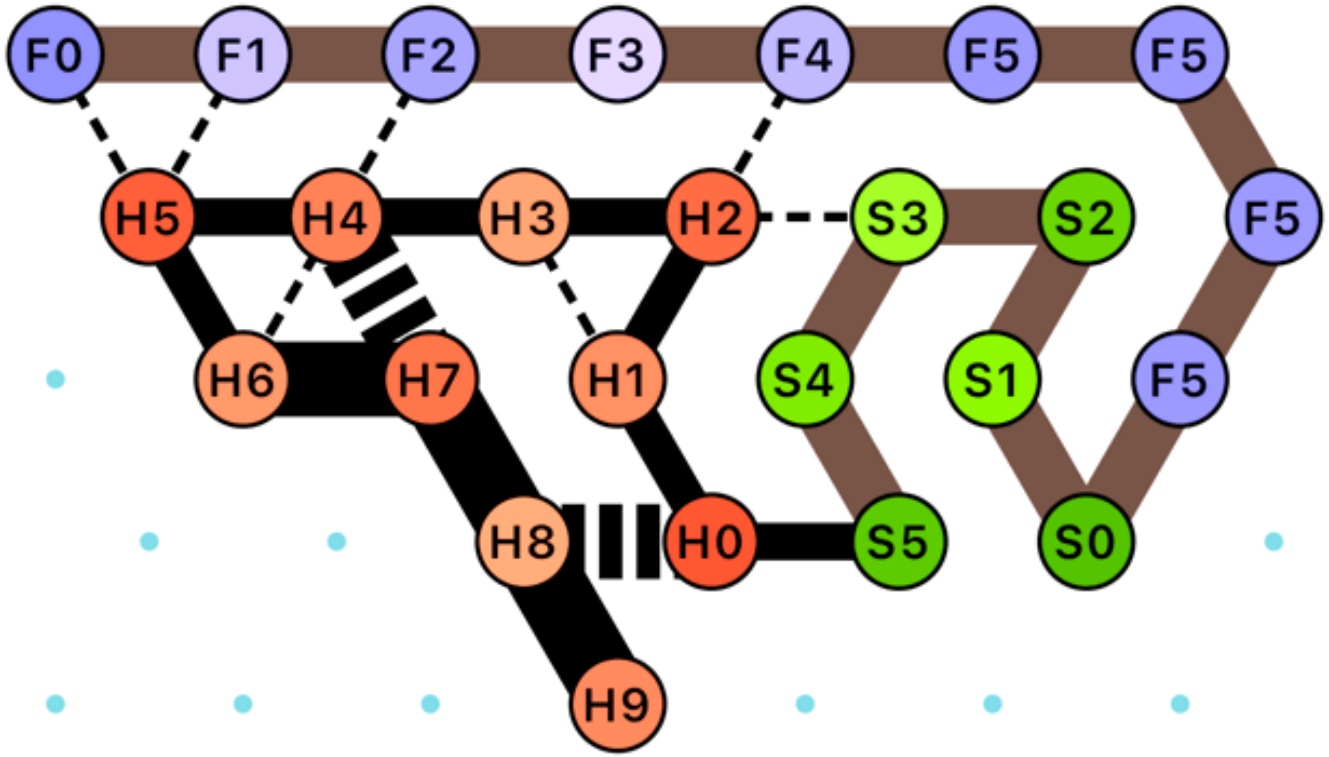


Geary et al. **MFCS2016**: 43:1-43:14

Oritatami

model of CF-driven computing

The nascent fragment tries to fold with as many bonds as possible.

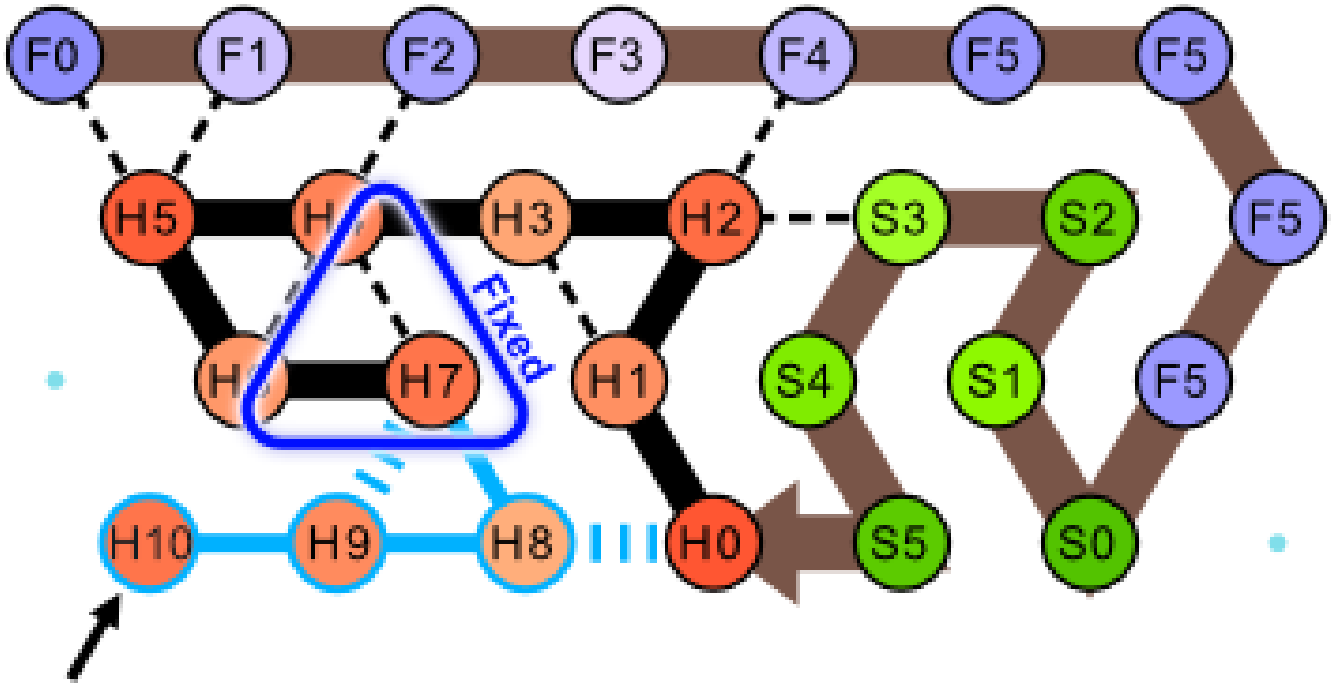


Geary et al. **MFCS2016**: 43:1-43:14

Oritatami

model of CF-driven computing

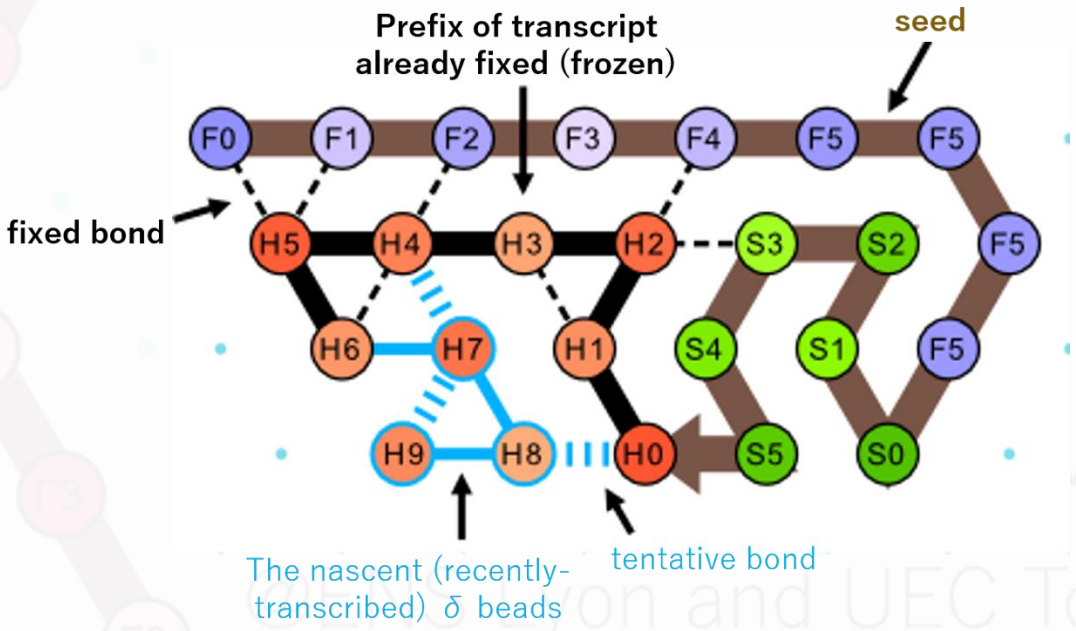
If there are more than one way of fixing H7 point-wise or bond-wise, then **nondeterminism**.



Transcribed! (according to the **preprogrammed** transcript)

Oritatami

model of CF-driven computing



An oritatami system consists of

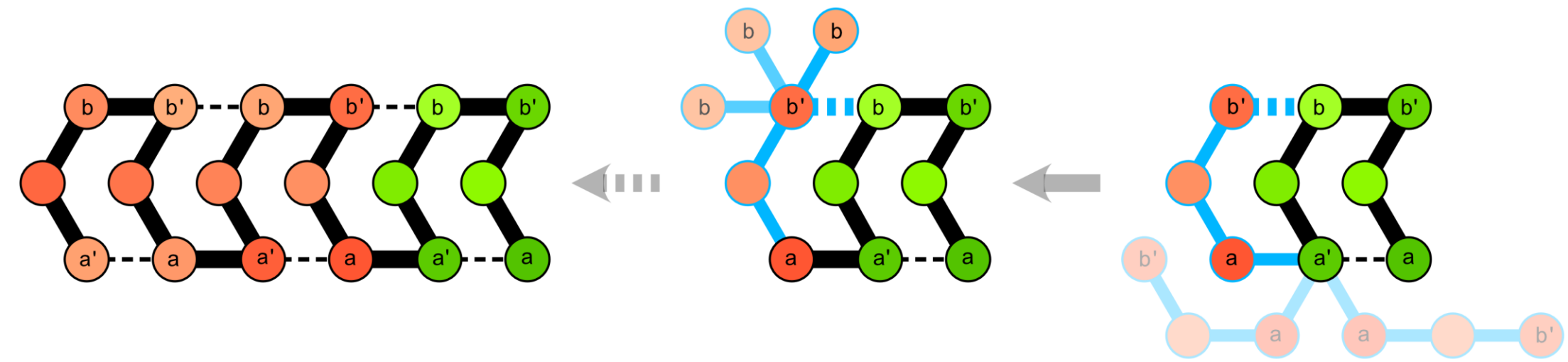
- A finite set Σ of types of abstract molecule (*bead*).
- $w \in \Sigma^*$ (*transcript*)
- $R \subseteq \Sigma \times \Sigma$ (*affinity/ binding rule*)
- δ (*delay*)
- α (*arity*), max # of bonds per bead, formed on the first-come-first-served basis

States by definition!? No! Implement them if needed.

Oritatami

Glider, a self-standing motif

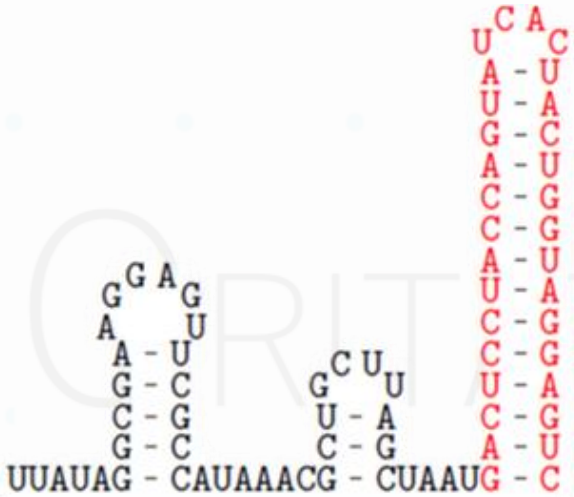
With $\delta = 3$, arbitrary arity $\alpha \geqq 1$, $R = \{(a, a'), (b, b')\}$, the periodic transcript a-•-b'-b-•-a'-a-•-⋯ folds into the self-standing *glider* motif.



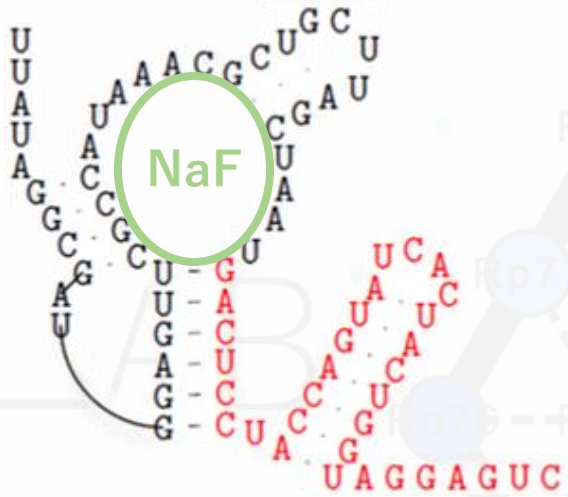
Oritatami

context-sensitive folding

A transcript can fold differently, depending on what are around (environment).



Terminator stem folded
(0mM NaF)



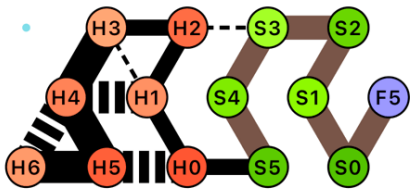
Terminator stem formation
failed (10mM NaF)

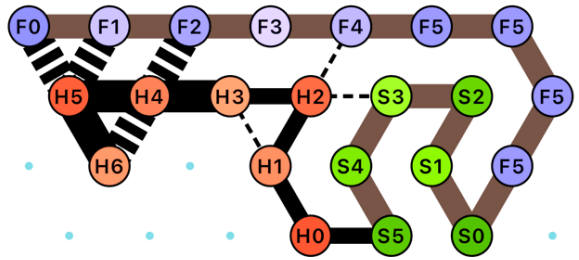
Watters et al. *Nat. Struct. Mol. Biol.* 23(12), pp.1124-1131, 2016

Oritatami

context-sensitive folding

A transcript can fold differently, depending on what are around (environment).

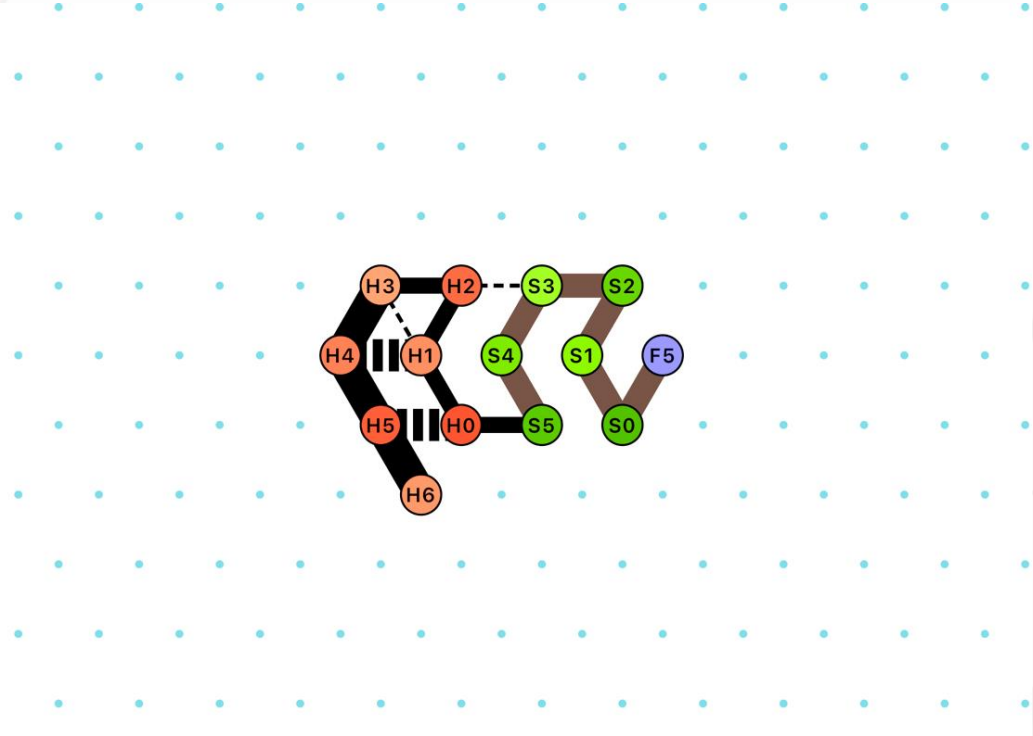


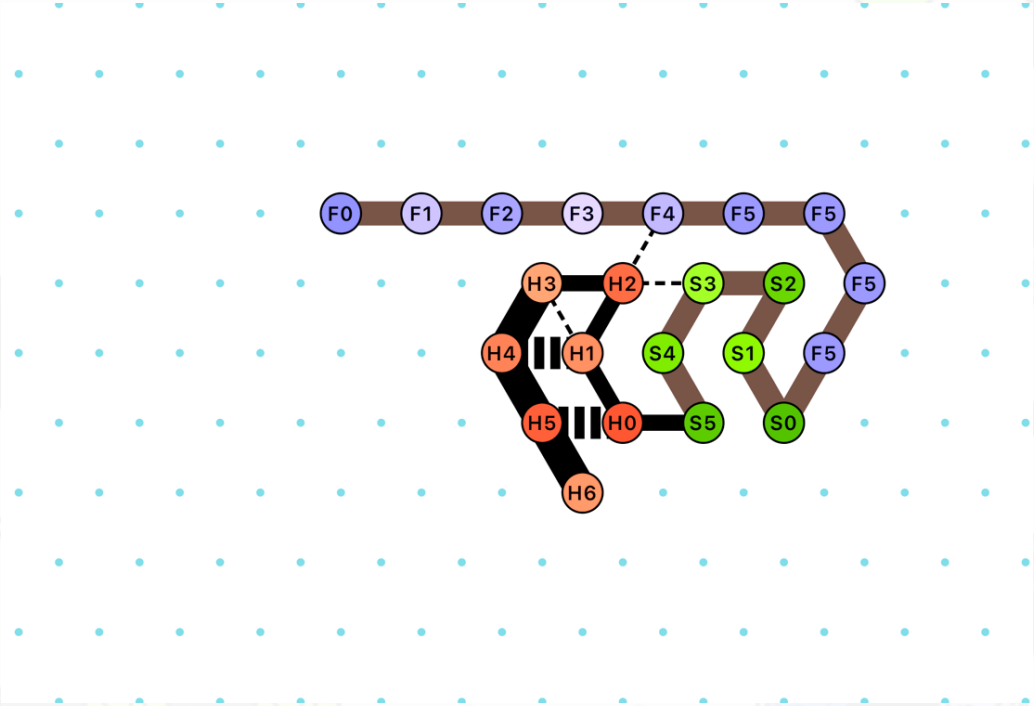


Oritatami

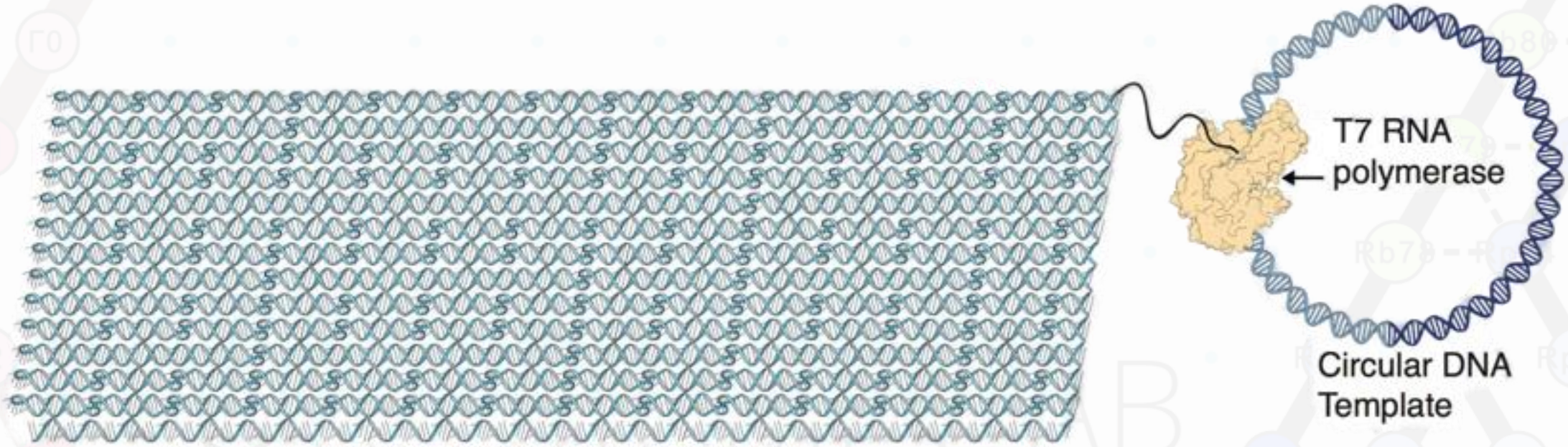
context-sensitive folding

A transcript can fold differently, depending on what are around (environment).





Homo-polymeric CF-driven computing



Complex structures are often made of simple identical (homo-) units (polymers) in nature.

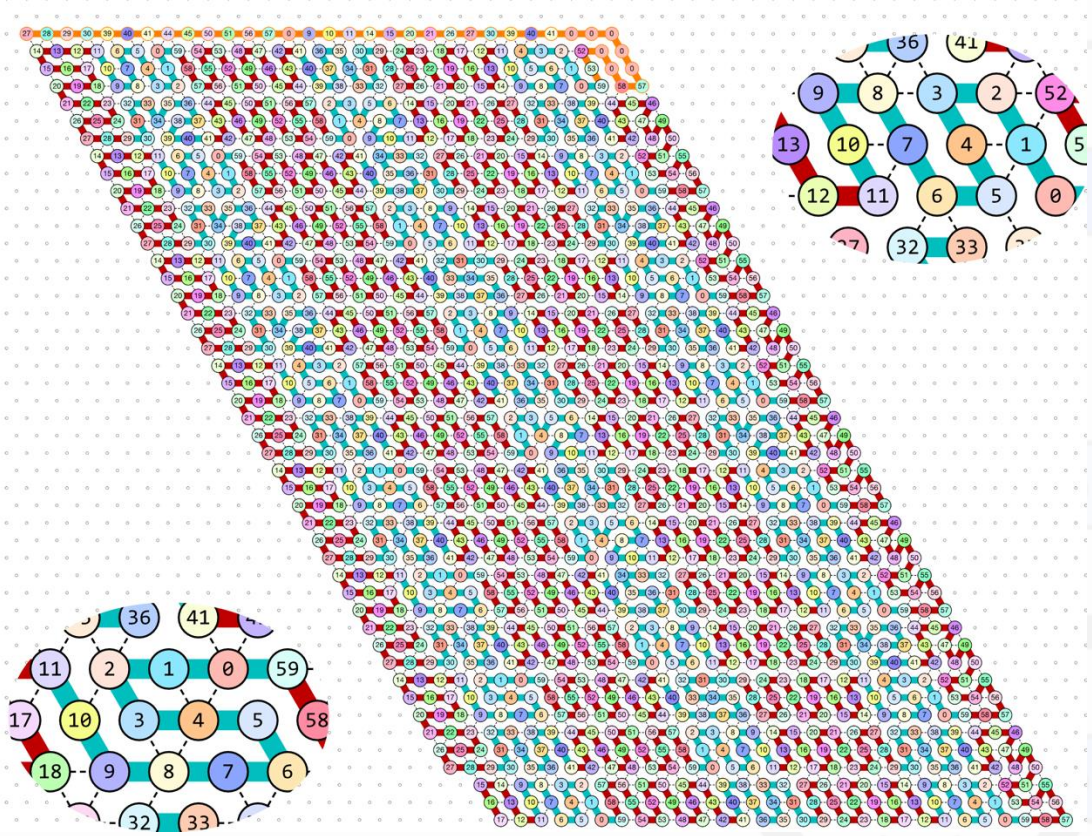
A periodic RNA transcript (homo-polymer) can be transcribed from a circular DNA.

Geary & Andersen, **DNA2014**: 1-19

Homo-polymeric CF-driven computing

Zigzag binary counter

- The first oritatami implementation
- Fixed bit-width (3 in the right figure), but later endowed with capability to widen by 1 bit at every overflow
- A transcript is of period 60 as **0-1-⋯-11-12-⋯-29-30-⋯-41-42-⋯-59-0-1-⋯**.
- Increment by 1 per zigzag.
- The factors **0-⋯-11** and **30-⋯-41** serve as a half-adder by folding into one of 4 possible conformations, depending on what are around.

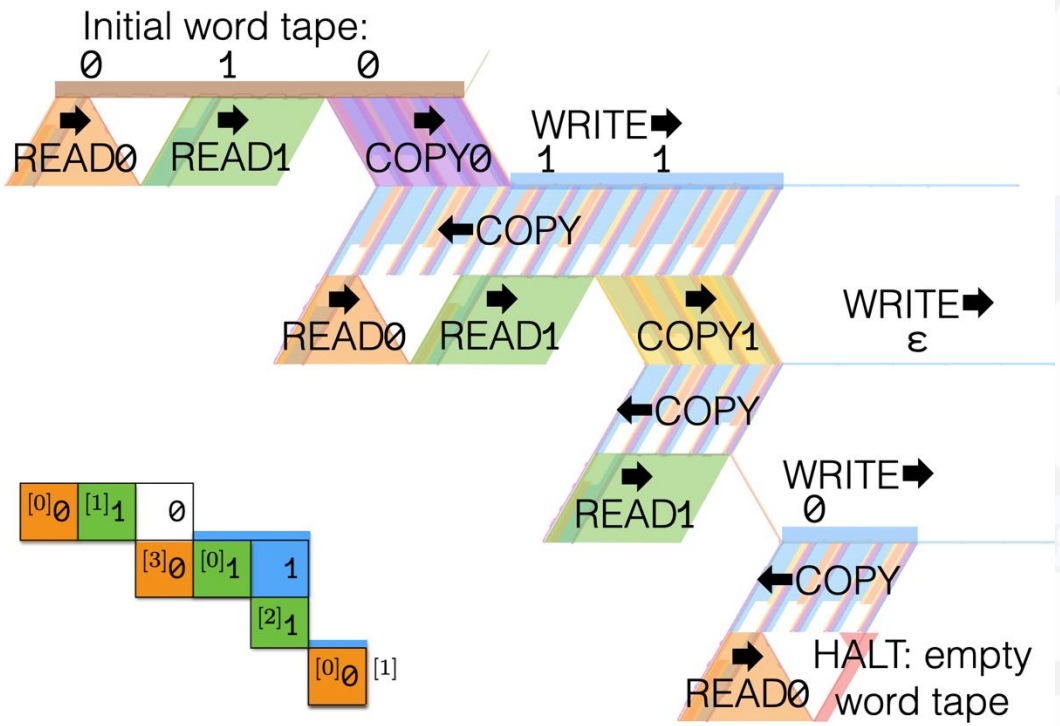


Geary et al. **MFCS 2016**. 43:1-43:14

Homo-polymeric CF-driven computing

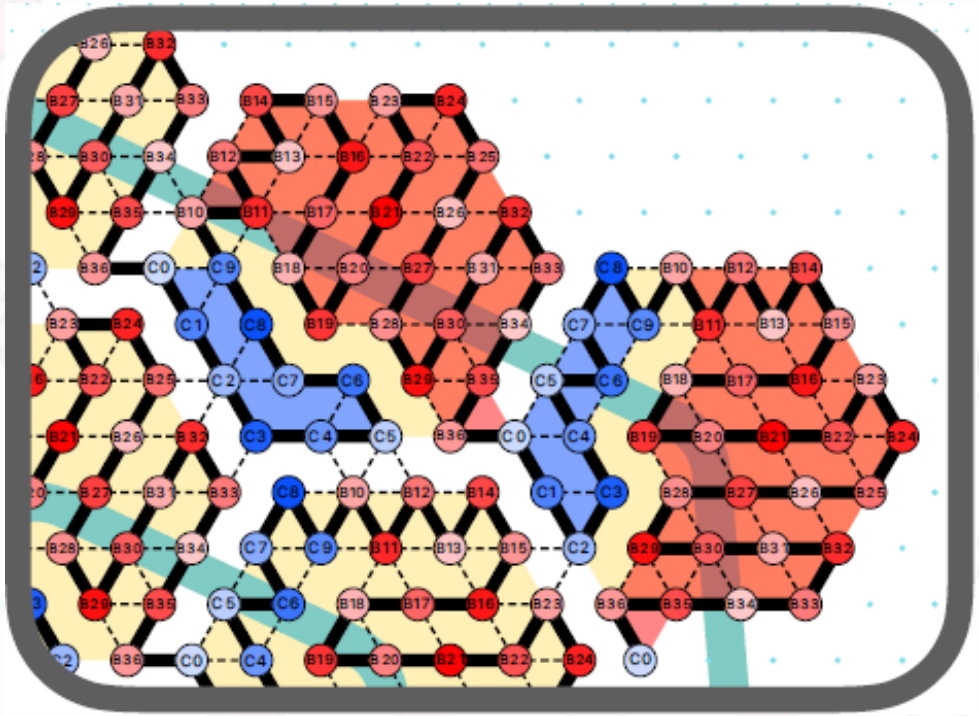
Turing-universality

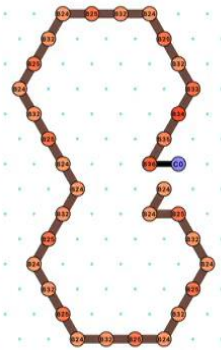
- Cyclic tag system (cts) is a binary rewriting system made of a cyclic list of $u_0, u_1, \cdots, u_{k-1} \in \{0, 1\}^*$ and a pointer $0 \leqq p < k$. It rewrites a word $w = a_1 a_2 \cdots a_n$ as:
  1. temp $= a_2 \cdots a_n$ (a_1 is deleted from w)
  2. If $a_1 = 1$, temp = temp $\cdot\, u_p$
  3. $w =$ temp
  4. p++ (mod k)



Homo-polymeric CF-driven computing

Tessellation by a transcript of period 37 at delay 3





Turedo

(Tur[-ing] + [Ter-]edo [navilis])

Programming language for CF

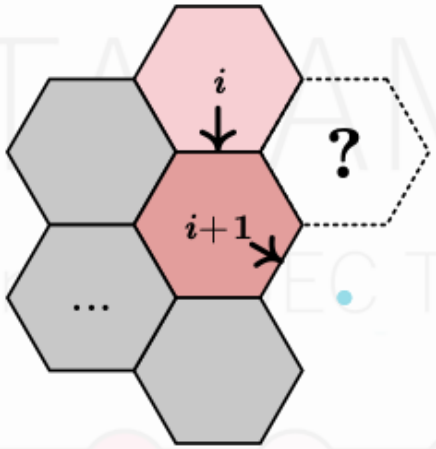
Turedo

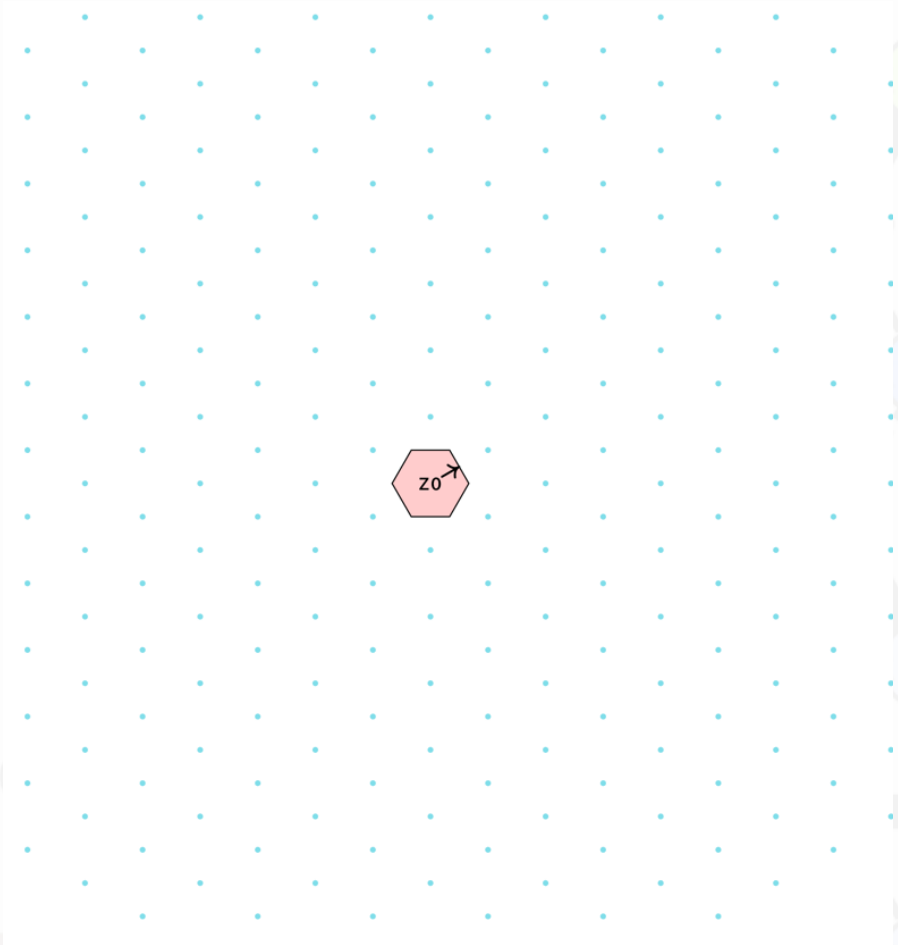
2D Turing machine on the hex grid that is *self-avoiding*, that is,

- Once visited, a cell won't be visited again.
- According to the configuration within the radius-r from its head, it colors the current cell and decides which neighbor to visit next, where r is a system parameter.

Example of radius-1 Turedo

(mod-4 clockwise walker)





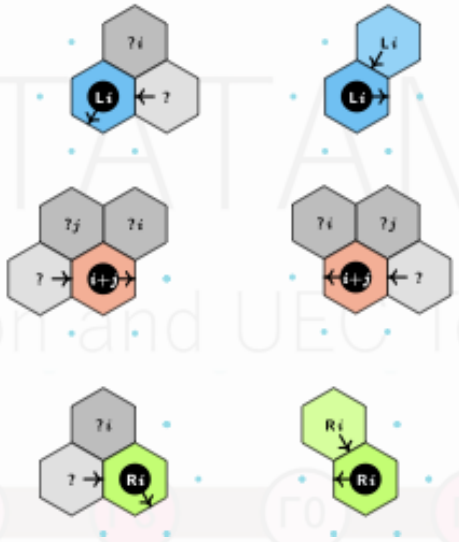
Turedo

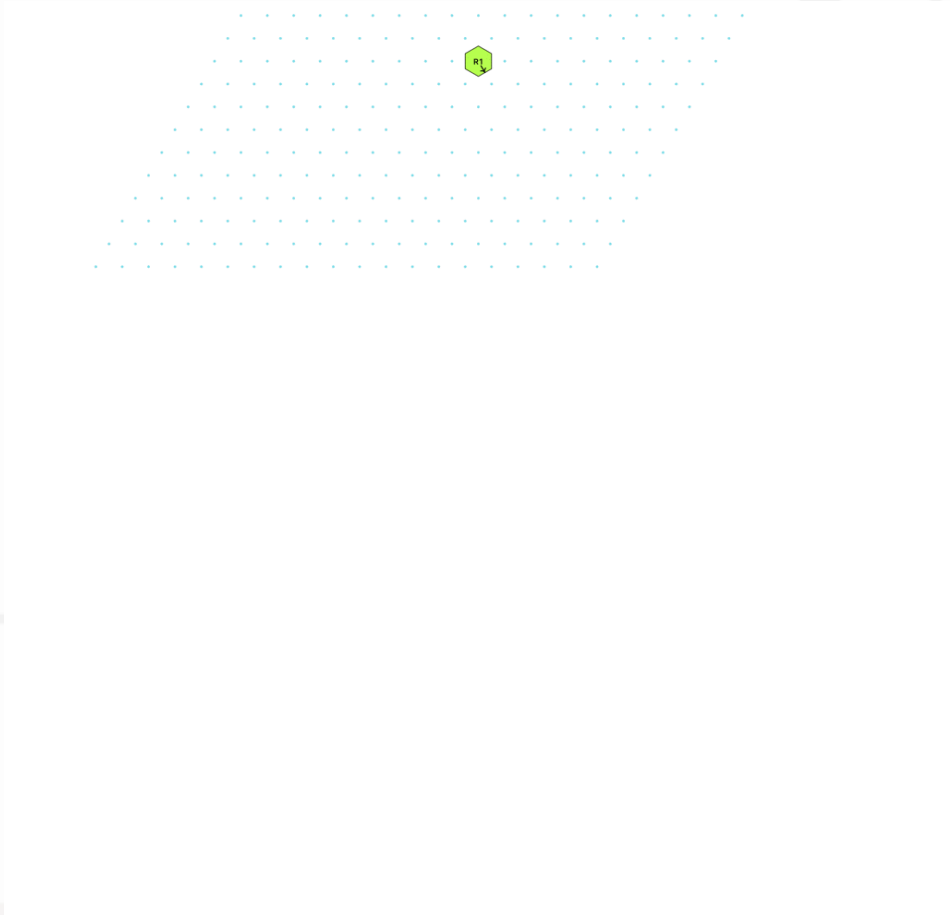
2D Turing machine on the hex grid that is *self-avoiding*, that is,

- Once visited, a cell won't be visited again.
- According to the configuration within the radius-*r* from its head, it colors the current cell and decides which neighbor to visit next, where *r* is a system parameter.

Example of radius-1 Turedo

(Sierpinski triangle)



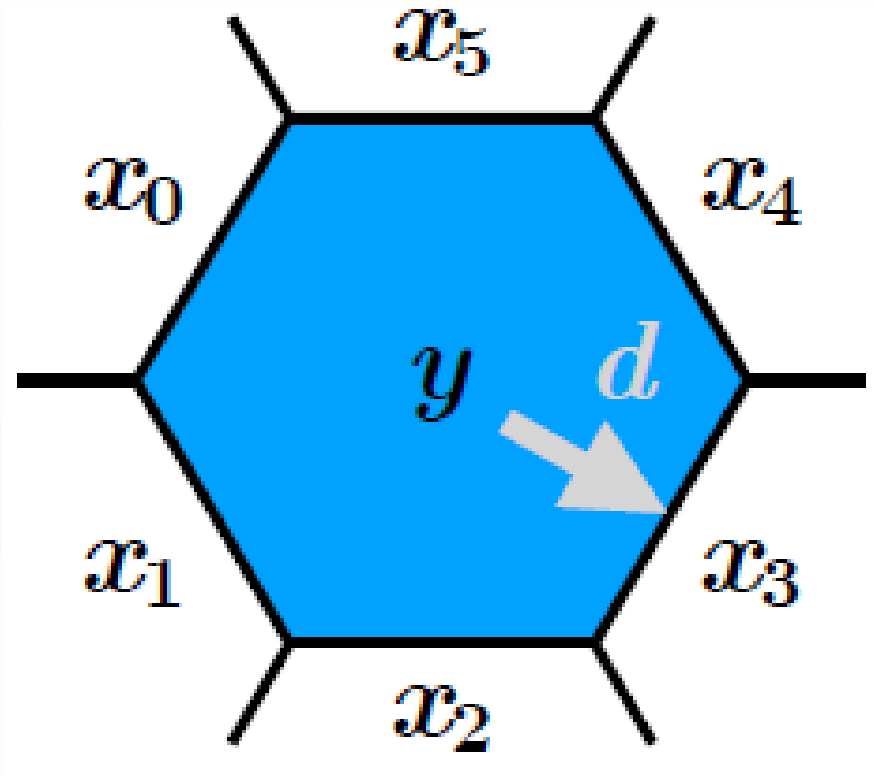


Pchelina et al. **STACS2022**: 51:1-51:23

Turedo

- A radius-1 stateless Turedo $T = (A, \delta)$ is a pair of a tape alphabet A including the blank $\perp$ and a transition function $\delta\colon A^6 \to A \times \{\mathrm{BR}, \mathrm{FR}, \mathrm{S}, \mathrm{FL}, \mathrm{BL}\}$.

- Suppose T has come from north, where x_5 is written. If $\delta(q, x_0, x_1, x_2, x_3, x_4, x_5) = (a, d)$, then it
  1. writes $a \in A$ at the current cell, and
  2. moves to the neighbor cell in the direction d, <u>as long as the cell is still blank</u>; otherwise, it halts.



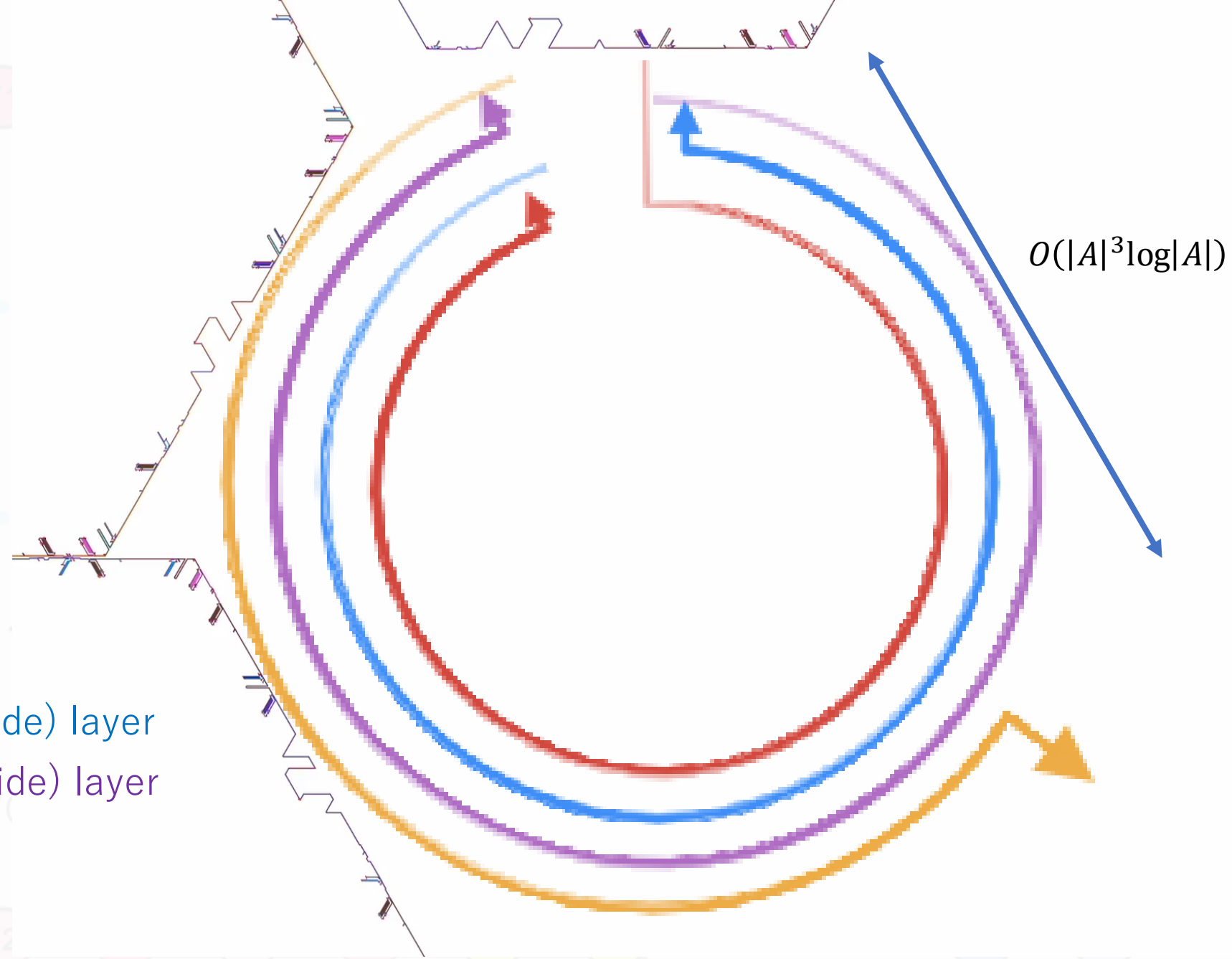
Turedo

Turedo-to-Oritatami Compiler Theorem.

A radius-1 stateless Turedo $T = (A, \delta)$ can be programmed into a transcript w of period $O(|A|^6 \log|A|)$ with which the deterministic delay-3 oritatami system $(\Sigma, w, R, 3, 6)$ simulates T intrinsically, where

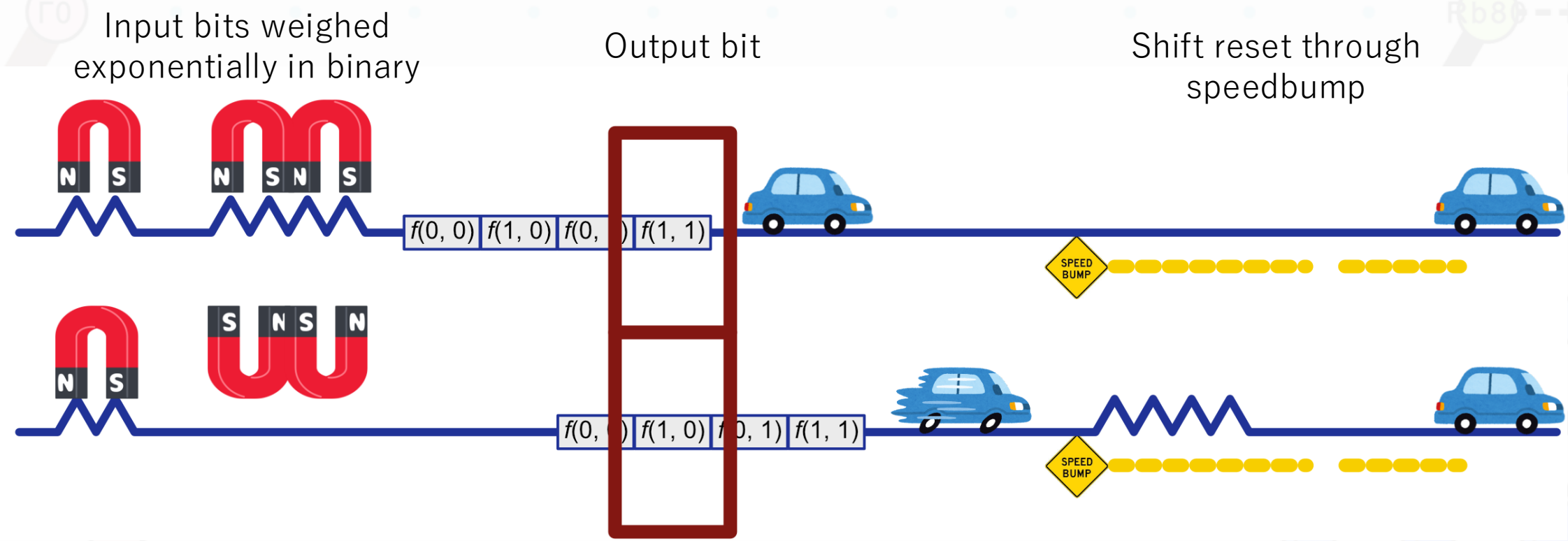
- Σ is universal, that is, independent of T, and consists of 1753 bead types.
- R is also universal.
- Each period of w folds into a **macrocell** of side length $O(|A|^3 \log|A|)$.

Macrocell



1. Scaffold layer
2. Read ($\log|A|$ bits/side) layer
3. Write ($\log|A|$ bits/side) layer
4. Exit layer

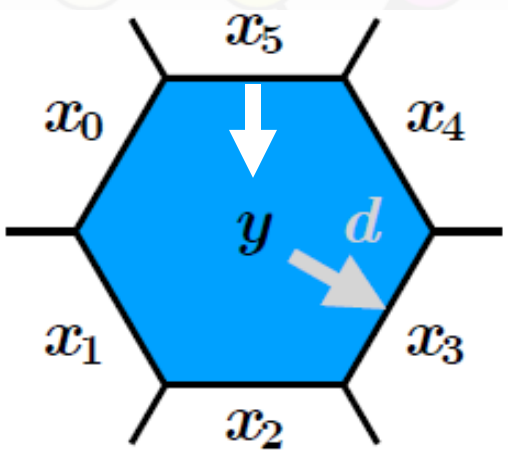
Shift-driven computing



$6 \log|A|$ bits → Max offset $O(|A|^6)$ → Length $O(|A|^6)$

Pchelina et al., **LATIN2020**, pp.425-436.

Period of transcript



A Turedo $T = (A, \delta)$ is encoded in the period of a transcript as:

Scaffold → **Read** → **Write** → **Speedbump** → **Exit** → Scaffold → ⋯

Scaffold hardcodes the macrocell's skeleton

Read $\boldsymbol{x} = (\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4}, \boldsymbol{x_5})$ with $x_s = (b_{s,0}, b_{s,1}, \dots, b_{s,\log|A|-1})$

- Weigh-sums the bits in $x_0, x_1, \dots\ x_5$ in this order as:

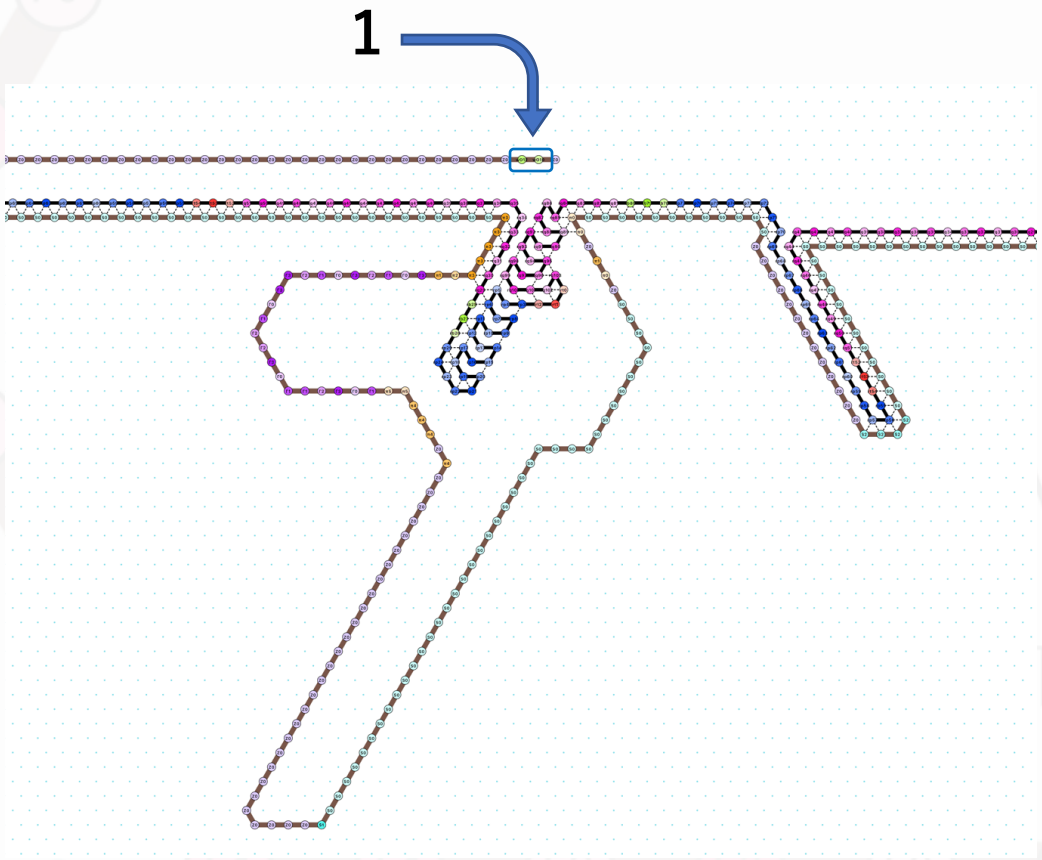
$$\Delta(x) = \sum_{s=0}^{5} \sum_{i=0}^{\log|A|-1} b_{s,i} 2^{s\log|A|+i}$$

- Pushes the remaining transcript forward by this offset.

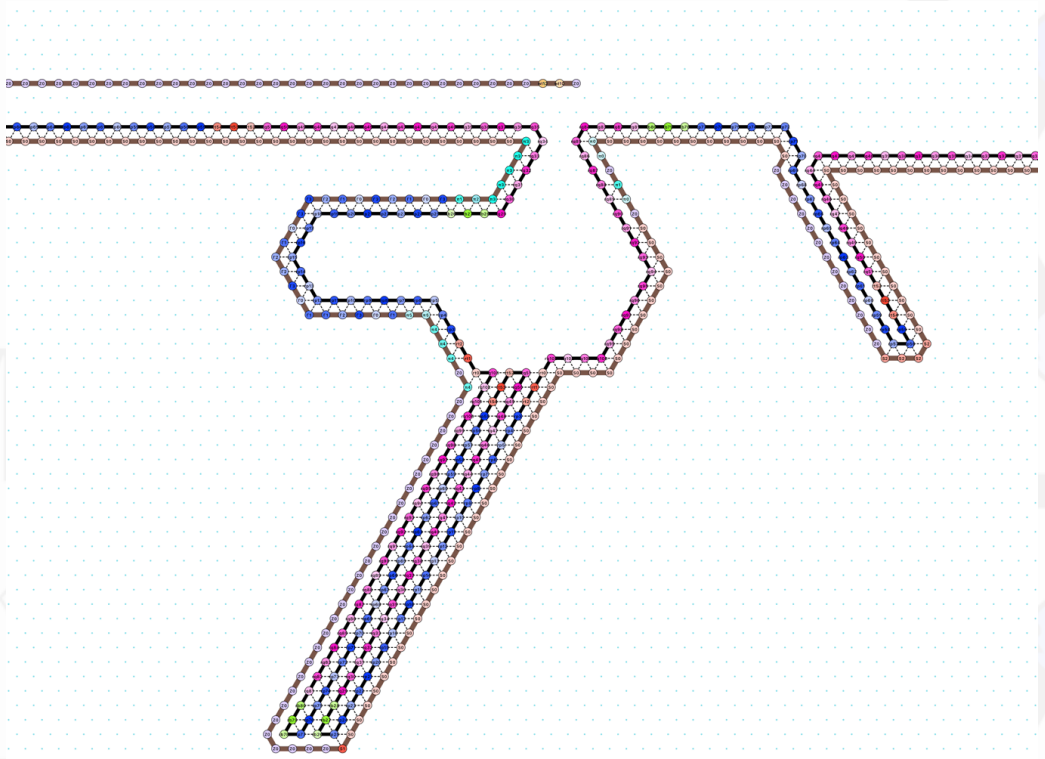
NOTE. these bits must have been written in a **uniform** format. It is uncomputable whether a cell will be visited; let alone from which direction it will be entered.

Read ↺: Reading pockets

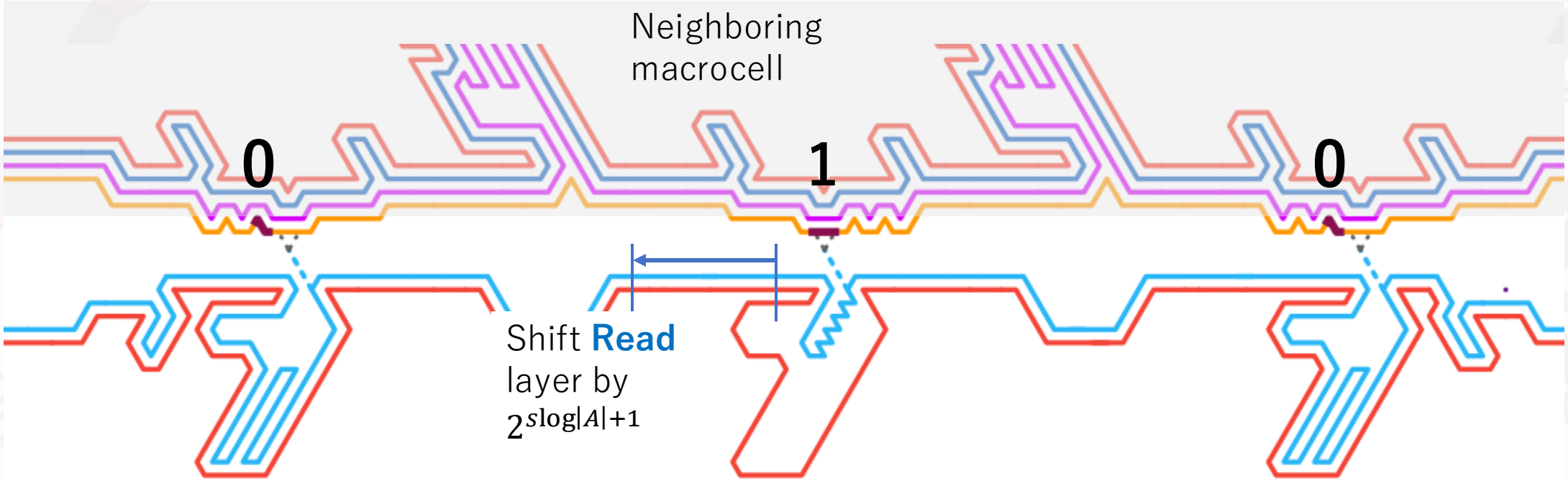
OK, readers, it's you who weighs bits!!



Otherwise



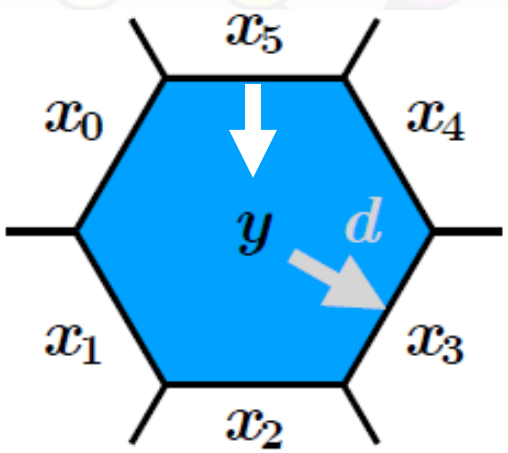
Read ↺: Reading pockets



Capacities (Pocket size) : $2^{s\log|A|+2}$ > $2^{s\log|A|+1}$ > $2^{s\log|A|+0}$

If the j-th bit is 1, then $\Delta(x) \mathrel{+}= 2^{s\log|A|+j}$

Period of transcript



A Turedo $T = (A, \delta)$ is encoded in the period of a transcript as:

Scaffold → **Read** → **Write** → **Speedbump** → **Exit** → Scaffold → ⋯

Write a and d (letter & direction)

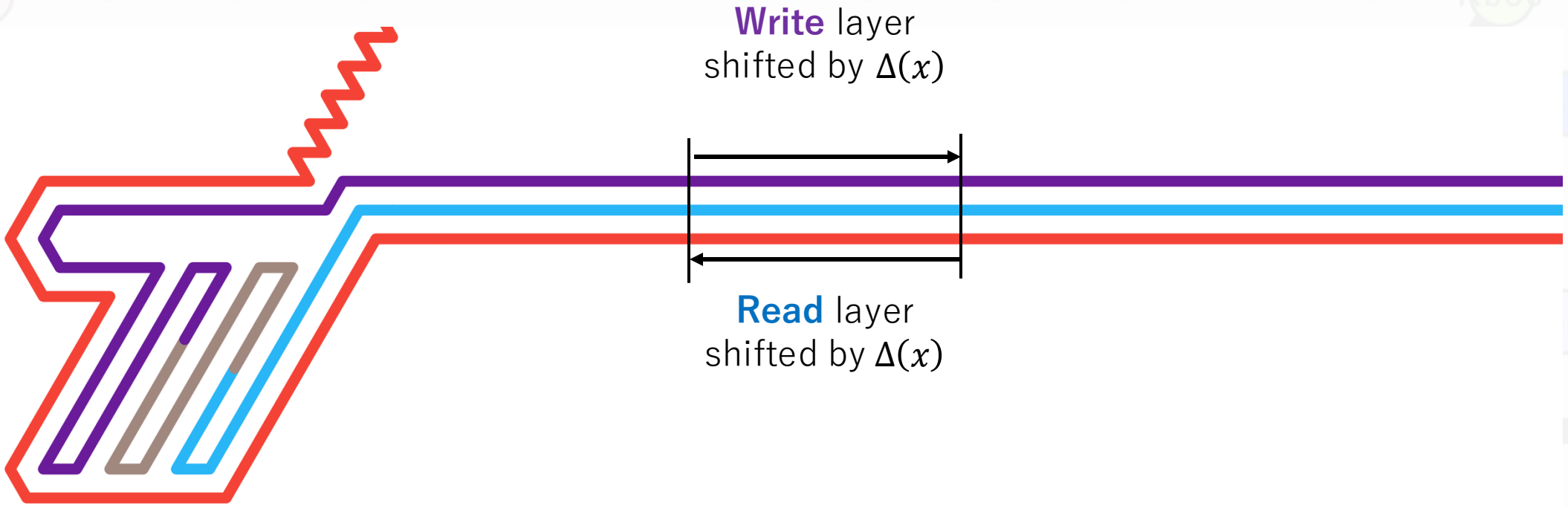
- All the transition tables for each bit and for exit-direction are encoded: $|A|^6$ entries per table.
- This layer is shifted by $\Delta(x)$ so that only the referred entry by x of each table is exposed at a position readable later while the others are "hidden."
- Outputs must be in a uniform format along all the sides.

Speedbump absorbs $\Delta(x)$

Exit at the side $d(x)$ specified by **Write**

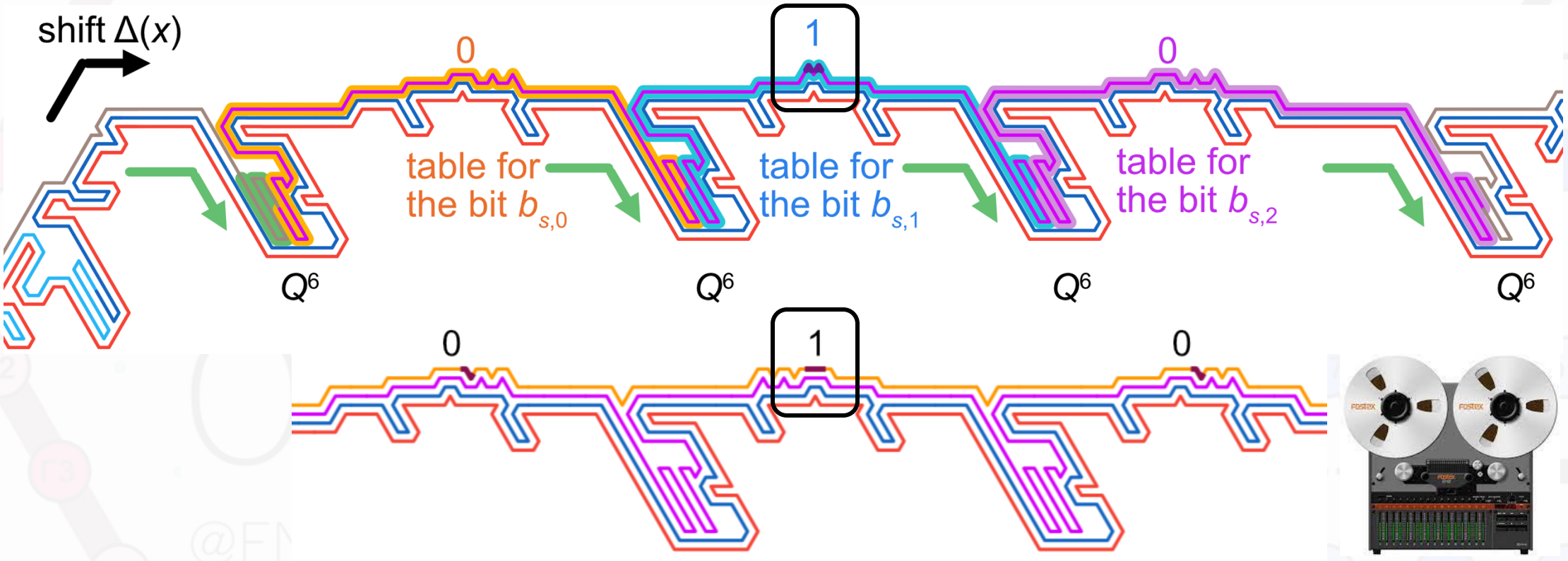
Read ↺ to Write ↻: U-turn pocket

The shift is transferred from **Read** to **Write**.



Capacity: $|A|^6$

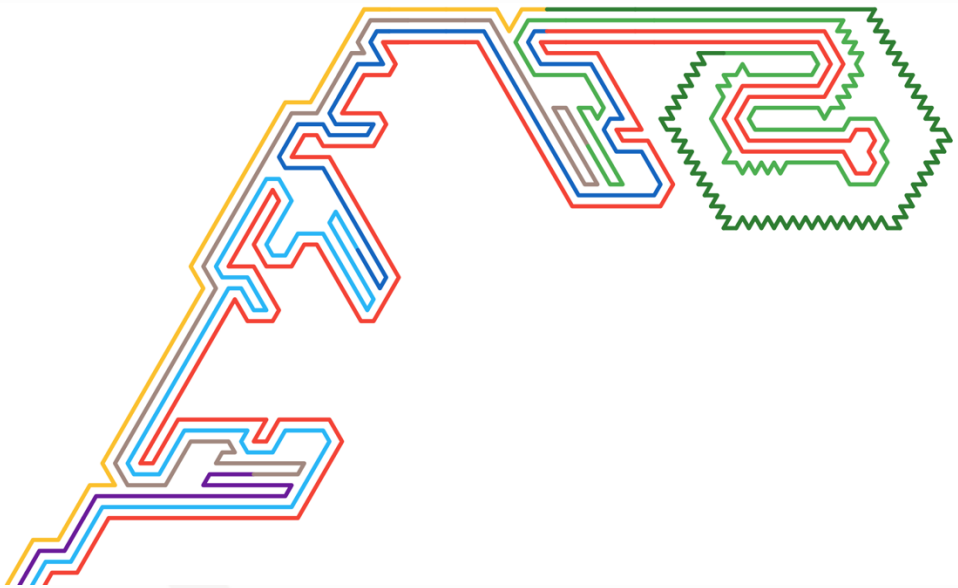
Write ↻: tables shifted by $\Delta(x)$

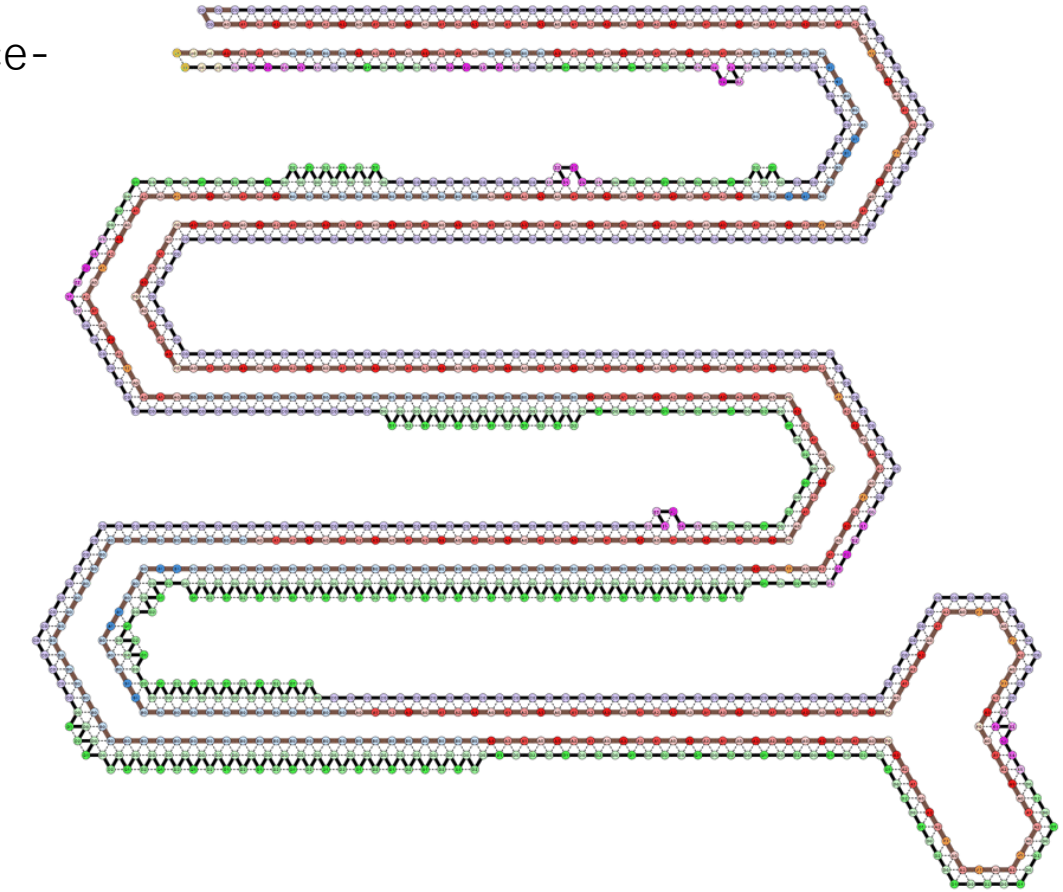


NOTE: The system knows in the stage of programming tables for each x, whether each bit will be covered by **Exit** according to the exit direction $d(x)$.

“Foldable” Speedbump: absorbing $\Delta(x)$

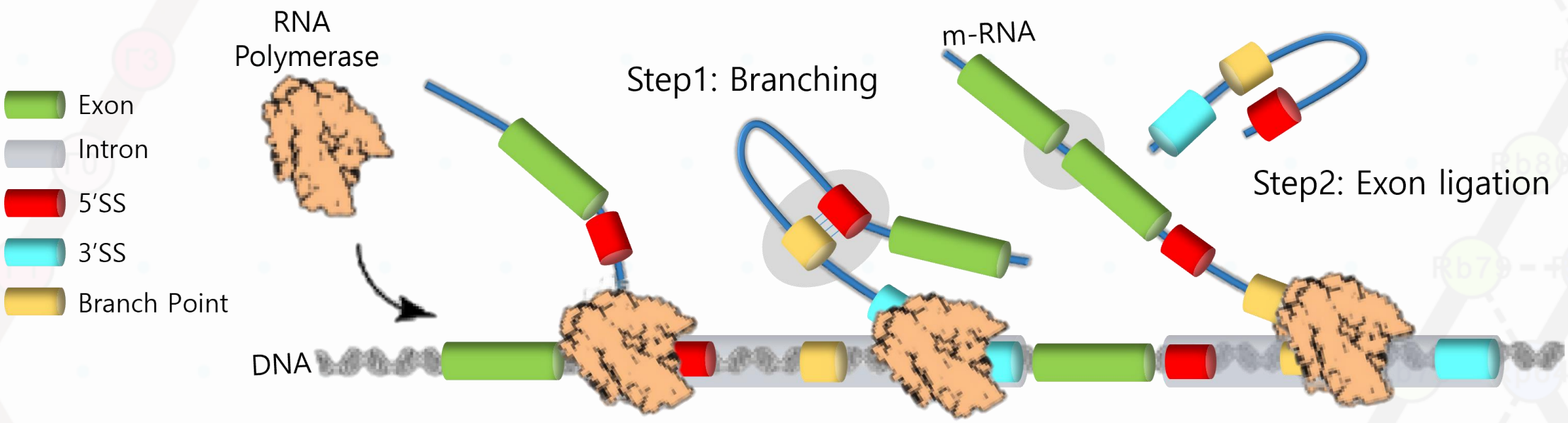
Based on the straight speedbump [PchelinaSSU20] but quadratically more space-efficient.





RNA Spinner

In vitro/vivo auto-synthesis of RNA components by NFA



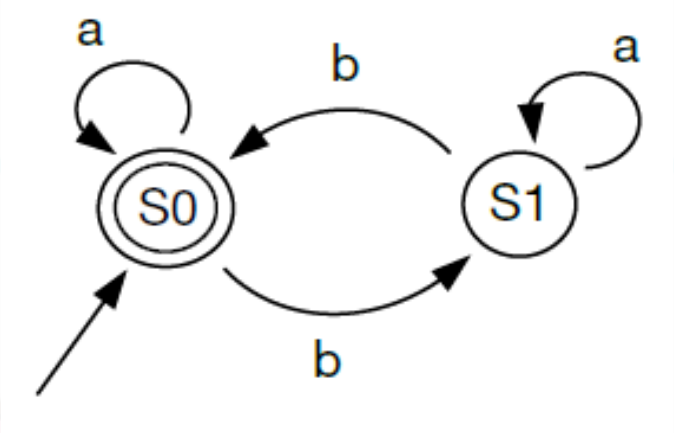
Co-transcriptional splicing

$uxs\ell\theta(s)yv \to uv$ if

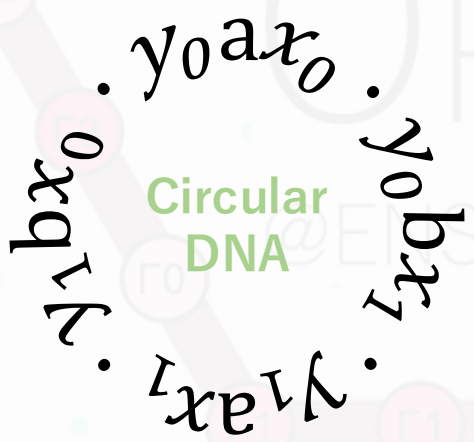
1. (x, y) is an enzymatically-recognizable context and
2. $s\ell\theta(s)$ is a *stable* hairpin, where θ is an antimorphic involution.

RNA spinner

An NFA for a *superset* of R.



encoding



Co-transcriptional splicing

Any molecular system consists of *finitely many* (*kinds*) of DNA and RNA sequences. Let R be the *finite* set of RNA sequences it involves.

Recall that RNAs are naturally degraded.

$$y_0 a\overrightarrow{x_0} \cdot y_0 b x_1 \cdot y_1 a x_1 \cdot y_1 b x_0 - \overleftarrow{y_0} a x_0 \cdot y_0 b x_1 \cdots$$

$$\rightarrow \quad y_0 a a\overrightarrow{x_0} \cdot \overleftarrow{y_0} b x_1 \cdot y_1 a x_1 \cdots$$

$$\rightarrow \quad y_0 a a b x_1 \cdot y_1 a x_1 \cdots$$

Problems to be solved for RNA Spinner

Problem 1.

Given R and a finite set D of domains (via which sequences in R interact with each other or with other molecules), construct an NFA A with as *few transitions* as possible s.t.

$$R \subseteq L(A) \subseteq R \cup \overline{\Sigma^* D \Sigma^*}$$

By setting $D = \Sigma$, this problem is reduced to the NP-hard problem of finding a transition-minimal NFA for finite languages [GruberH07]. Such a ubiquitous domain however turns any system into a chaotic soup.

Problems to be solved for RNA Spinner

A hairpin gets less stable with a longer loop and a shorter stem.

Contribution by stem is linear, while it remains open how a loop is penalized.

Problem 2

Propose a proper energy model for RNA hairpin stability, and study hairpin-related operations by considering only stable hairpins in the model.

Table 6. Free-energy increments for loops

Loop size	Internal loop*†	Bulge loop*‡	Hairpin loop*§
1	—	+3.3	—
2	+0.8	+5.2	—
3	+1.3	+6.0	+7.4
4	+1.7	+6.7	+5.9
5	+2.1	+7.4	+4.4
6	+2.5	+8.2	+4.3
7	+2.6	+9.1	+4.1
8	+2.8	+10.0	+4.1
9	+3.1	+10.5	+4.2
10	+3.6	+11.0	+4.3
12	+4.4	+11.8	+4.9
14	+5.1	+12.5	+5.6
16	+5.6	+13.0	+6.1
18	+6.2	+13.6	+6.7
20	+6.6	+14.0	+7.1
25	+7.6	+15.0	+8.1
30	+8.4	+15.8	+8.9

Freier et al., *PNAS* 83, pp.9373-9377, 1986

Thanks

Da-Jung Cho
(Suwon)

Szilard Zsolt
Fazekas (Akita)

Cody W. Geary
(Aarhus)

Hwee Kim
(Incheon)

Daria Pchelina
(Lyon)

Nicolas Schabanel (Lyon)

Guillaume Theyssier
(Marseille)

Max Wiedenhöft
(Kiel)

ENS DE LYON

科研費
KAKENHI

国立大学法人
電気通信大学
The University of Electro-Communications

References

- [FazekasIKMST24] S. Fazekas, N. Iwano, Y. Kihara, R. Matsuoka, S. Seki, and H. Takeuchi. *Theoretical Computer Science* 999: 114550, 2024
- [FazekasKMST22] S. Fazekas, H. Kim, R. Matsuoka, S. Seki, and H. Takeuchi. **ISAAC2022**, LIPIcs 248, 37:1-37:15
- [GearyMSS16] C. Geary, P-E. Meunier, N. Schabanel, and S. Seki. **MFCS2016**, LIPIcs 58, 43:1-43:14
- [GearyMSS18] C. Geary, P-E. Meunier, N. Schabanel, and S. Seki. **ISAAC2018**, LIPIcs 123, 23:1-23:13
- [GruberH07] H. Gruber and M. Holzer, **LATA2007**, 261-272, 2007
- [MaruyamaS21] K. Maruyama and S. Seki. *Natural Computing*, 20(2), pp.329-340, 2021
- [PchelinaSST22] D. Pchelina, N. Schabanel, S. Seki, and G. Theyssier. **STACS2022**, LIPIcs 219, pp.51:1-51:23
- [PchelinaSSU20] D. Pchelina, N. Schabanel, S. Seki, and Y. Ubukata. **LATIN2020**, LNCS 12118, pp.425-436