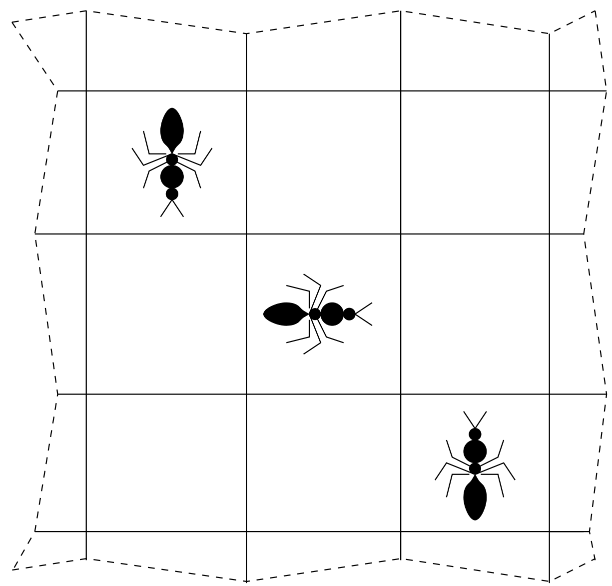
Cellular ANTomata: Principles and Progress

Arnold L. Rosenberg

Computer Science
Northeastern University
Boston, MA 02115, USA



$\boxed{\textbf{The Cellular ANTomaton Model}}$

The *parallel* component of the model:

An $n \times n$ *cellular AUTomaton:*

— the $n \times n$ mesh $\mathcal{M}_n$ with identical FSMs at each cell

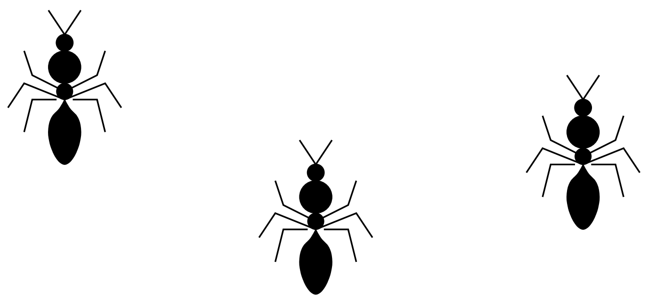
The Cellular ANTomaton Model

The *parallel* component of the model:

An $n \times n$ cellular AUTomaton

The *distributed* component of the model:

A (possibly heterogeneous) team of *mobile* FSMs
— which we call *ANTS*



The Cellular ANTomaton Model

The *parallel* component of the model:

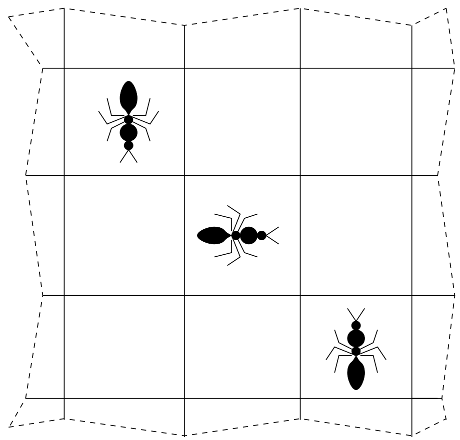
An $n \times n$ cellular AUTomaton

The *distributed* component of the model:

A (possibly heterogeneous) team of *mobile ANTS*

The Ants plus the cellular automaton equals —

A *Cellular ANTomaton*



Our Goals

We seek—

SCALABLE ALGORITHMS

for reality-inspired problems

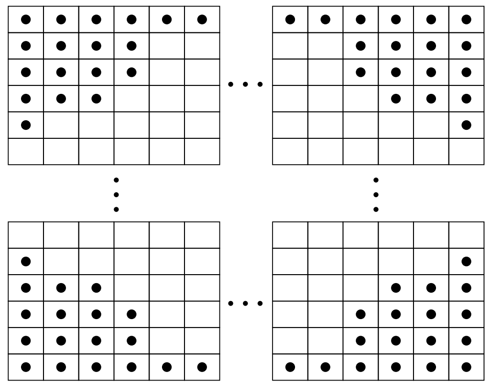
within the

Cellular ANTomaton model

A "Proof of Concept" Problem

THE *PARKING* PROBLEM FOR ROBOTIC ANTS

The *Parking* Problem for Robotic Ants



Informal

Have Ants congregate as compactly as possible in the closest corners of $\mathcal{M}_n$ (measured at moment of initiation)

Formal (for the *southwest quadrant* of $\mathcal{M}_n$)

Minimize the *parking potential function*:

$$\Pi(t) \ \stackrel{\text{def}}{=} \ \sum_{k=0}^{2n-2} (k+1) \ \times \ (\text{number of Ants on diagonal } k \text{ at step } t).$$

The Parking Problem for Robotic Ants

Theorem

A Cellular ANTomaton can park Ants in $\mathcal{M}_n$ in $\underline{O(n^2)}$ steps.

This is a *very* conservative bound: *Could $\underline{O(n)}$ steps be possible?*

For perspective:

Theorem

It is impossible for discrete Ants on an unintelligent floor to park.

The Parking Problem for Robotic Ants

Theorem

A Cellular ANTomaton can park Ants in $\mathcal{M}_n$ in $\underline{O(n^2)}$ steps.

1. The Cellular ANTomaton quadrisects $\mathcal{M}_n$ so each Ant knows its quadrant:
   Time: $O(n)$ steps

$\boxed{\textbf{The Parking Problem for Robotic Ants}}$

Theorem

A Cellular ANTomaton can park Ants in $\mathcal{M}_n$ in $\underline{O(n^2)}$ steps.

1. The Cellular ANTomaton quadrisects $\mathcal{M}_n$ so each Ant knows its quadrant:
   Time: $O(n)$ steps

2. Ants move in a "wavefront" toward the correct corner:
   Time: $O(n)$ steps

The Parking Problem for Robotic Ants

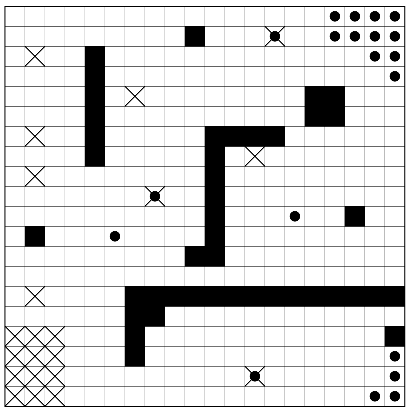
Theorem

A Cellular ANTomaton can park Ants in $\mathcal{M}_n$ in $\underline{O(n^2)}$ steps.

1. The Cellular ANTomaton quadrisects $\mathcal{M}_n$ so each Ant knows its quadrant:
   Time: $O(n)$ steps
2. Ants move in a "wavefront" toward the correct corner:
   Time: $O(n)$ steps
3. Ants wander around their corner, to achieve a *compact* configuration
   Time: our procedure takes $\Theta(n^2)$ steps

A “Proof of Concept” Problem

THE *FOOD-SEEKING* PROBLEM FOR ROBOTIC ANTS



CIRCLE	=	Ant
X	=	Food
Blackened cell	=	Obstacle

$\boxed{\text{The } \textbf{\textit{Food-Seeking}} \text{ Problem for Robotic Ants}}$

$\mathcal{M}_n$ contains $\underline{r}$ Ants and $\underline{s}$ food items.

Goal. *Match Ants and food items so that:*

- *if* $r \geq s$*, then some Ant will reach every food item;*
- *if* $s \geq r$*, then every Ant will get food.*

The ***Food-Seeking*** Problem for Robotic Ants

$\mathcal{M}_n$ contains $\underline{r}$ Ants and $\underline{s}$ food items.

Goal. *Match Ants and food items.*

We have developed two algorithms that achieve this goal:

1. The *Food-Initiated algorithm*.
   Time: $(r + O(1))n$ steps

The ***Food-Seeking*** Problem for Robotic Ants

$\mathcal{M}_n$ contains $\underline{r}$ Ants and $\underline{s}$ food items.

Goal. *Match Ants and food items.*

We have developed two algorithms that achieve this goal:

1. The *Food-Initiated algorithm*.
   Time: $(r + O(1))n$ steps

2. The *Active-Ant algorithm*.
   Time: $O(n\sqrt{s})$ steps

The ***Food-Seeking*** Problem for Robotic Ants

$\mathcal{M}_n$ contains $\underline{r}$ Ants and $\underline{s}$ food items.

Goal. *Match Ants and food items.*

We have developed two algorithms that achieve this goal:

1. The *Food-Initiated algorithm*.
   Time: $(r + O(1))n$ steps

2. The *Active-Ant algorithm*.
   Time: $O(n\sqrt{s})$ steps

For perspective:

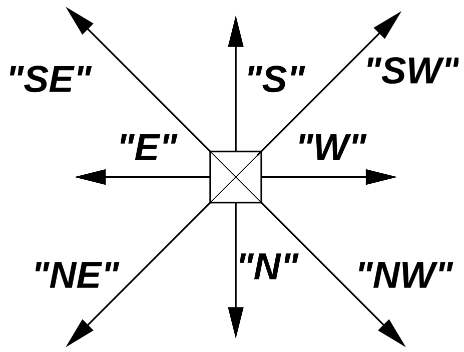
Theorem

A single intelligent Ant on an unintelligent floor requires, in the worst case, $\Omega(n^2)$ steps to find a single food item.

$\boxed{\textbf{The } \textbf{\textit{Food-Initiated}} \textbf{ Algorithm}}$

$\mathcal{M}_n$ contains $\underline{r}$ Ants and $\underline{s}$ food items.

1. FSMs with food send *food-announcing message* to all neighbors.



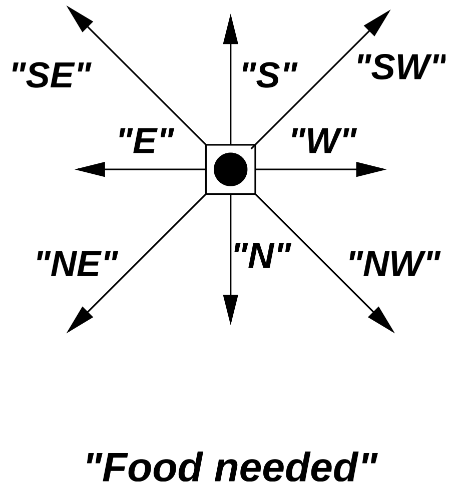
"Food available"

The ***Food-Initiated*** Algorithm

$\mathcal{M}_n$ contains $\underline{r}$ Ants and $\underline{s}$ food items.

1. FSMs with food send *food-announcing message* to all neighbors.

2. FSMs with Ants send *food-seeking message* to all neighbors.



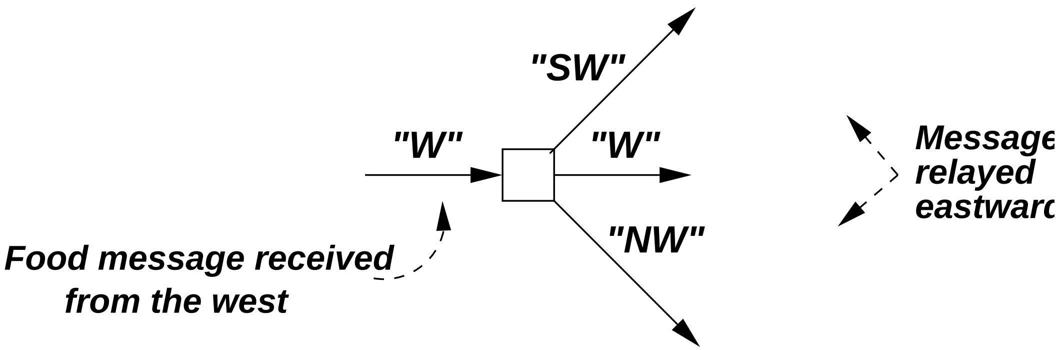
$\boxed{\textbf{The } \textbf{\textit{Food-Initiated}} \textbf{ Algorithm}}$

$\mathcal{M}_n$ contains $\underline{r}$ Ants and $\underline{s}$ food items.

1. FSMs with food send *food-announcing message* to all neighbors.

2. FSMs with Ants send *food-seeking message* to all neighbors.

3. All FSMs relay all messages (combining similar ones).



The ***Food-Initiated*** Algorithm

$\mathcal{M}_n$ contains r ants and s food items.

1. FSMs with food send *food-announcing message* to all neighbors.

2. FSMs with Ants send *food-seeking message* to all neighbors.

3. All FSMs relay all messages (combining similar ones).

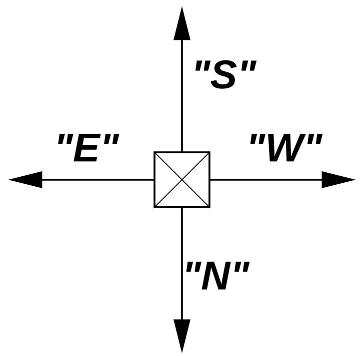
4. FSMs send Ants in the direction of a food-announcing message.

The ***Active-Ant*** Algorithm

$\mathcal{M}_n$ contains $\underline{r}$ Ants and $\underline{s}$ food items.

Goal. *Match Ants and food items.*

1. FSMs with food send *food-announcing message* on NEWS arcs.



"Food available'

The ***Active-Ant*** Algorithm

$\mathcal{M}_n$ contains $\underline{r}$ Ants and $\underline{s}$ food items.

Goal. *Match Ants and food items.*

1. FSMs with food send *food-announcing message* on NEWS arcs.

2. All FSMs relay food-announcing messages on NEWS arcs.

The ***Active-Ant*** Algorithm

$\mathcal{M}_n$ contains $\underline{r}$ Ants and $\underline{s}$ food items.

Goal. *Match Ants and food items.*

1. FSMs with food send *food-announcing message* on NEWS arcs.

2. All FSMs relay food-announcing messages on NEWS arcs.

3. FSMs send Ants clockwise around perimeter of $\mathcal{M}_n$.

$\boxed{\textbf{The } \textbf{\textit{Active-Ant}} \textbf{ Algorithm}}$

$\mathcal{M}_n$ contains $\underline{r}$ Ants and $\underline{s}$ food items.

Goal. *Match Ants and food items.*

1. FSMs with food send *food-announcing message* on NEWS arcs.

2. All FSMs relay food-announcing messages on NEWS arcs.

3. FSMs send Ants clockwise around perimeter of $\mathcal{M}_n$.

4. Ants (a) stay with food they encounter on way to perimeter
   (b) follow messages to food from perimeter
   —they rejoin the perimeter-walk if food has already been taken

A "Proof of Concept" Problem

THREE *PATTERN-MATCHING* PROBLEMS INSPIRED BY BIOINFORMATICS

Bio-Inspired Pattern-Matching

Problem 1. A CA $\mathcal{C}$ has:

- length-n *master pattern* Π along row 0
- length-$(m \leq n)$ *input pattern* π left-justified along row $n-1$

$\mathcal{C}$ identifies all positions in Π where a copy of π begins.

$\boxed{\textbf{Bio-Inspired Pattern-Matching}}$

<u>Problem 1</u>. A CA $\mathcal{C}$ has:

- length-n *master pattern* Π along row 0
- length-$(m \leq n)$ *input pattern* π left-justified along row $n-1$

$\mathcal{C}$ identifies all positions in Π where a copy of π begins.

Time: $n+m$ steps

$\boxed{\textbf{Bio-Inspired Pattern-Matching}}$

<u>Problem 1</u>. A CA $\mathcal{C}$ has:

- length-n *master pattern* Π along row 0
- length-$(m \le n)$ *input pattern* π left-justified along row $n-1$

$\mathcal{C}$ identifies all positions in Π where a copy of π begins.

Time: $n+m$ steps

<u>Problem 2</u>. $\mathcal{C}$ solves a sequence of instances of **Problem 1** for pattern Π and a sequence of input patterns, $\pi_1, \ldots, \pi_r$ of lengths $n \ge m_1 \ge \cdots \ge m_r$.

$\boxed{\textbf{Bio-Inspired Pattern-Matching}}$

<u>Problem 1</u>. A CA $\mathcal{C}$ has:

- length-n *master pattern* Π along row 0
- length-$(m \leq n)$ *input pattern* π left-justified along row $n-1$

$\mathcal{C}$ identifies all positions in Π where a copy of π begins.

Time: $n+m$ steps

<u>Problem 2</u>. $\mathcal{C}$ solves a sequence of instances of **Problem 1** for pattern Π and a sequence of input patterns, $\pi_1, \ldots, \pi_r$ of lengths $n \geq m_1 \geq \cdots \geq m_r$.

Time: $n + m_1 + \cdots + m_r$ steps.

Bio-Inspired Pattern-Matching

Problem 1. A CA $\mathcal{C}$ has:

- length-n *master pattern* Π along row 0
- length-$(m \leq n)$ *input pattern* π left-justified along row $n-1$

$\mathcal{C}$ identifies all positions in Π where a copy of π begins.

Time: $n + m$ steps

Problem 2. $\mathcal{C}$ solves a sequence of instances of **Problem 1** for pattern Π and a sequence of input patterns, $\pi_1, \ldots, \pi_r$ of lengths $n \geq m_1 \geq \cdots \geq m_r$.

Time: $n + m_1 + \cdots + m_r$ steps.

Problem 3. Expand **Problem 1**: Allow occurrences of π to *wrap around* Π.

Bio-Inspired Pattern-Matching

Problem 1. A CA $\mathcal{C}$ has:

- length-n *master pattern* Π along row 0
- length-$(m \le n)$ *input pattern* π left-justified along row $n-1$

$\mathcal{C}$ identifies all positions in Π where a copy of π begins.

Time: $n+m$ steps

Problem 2. $\mathcal{C}$ solves a sequence of instances of **Problem 1** for pattern Π and a sequence of input patterns, $\pi_1, \ldots, \pi_r$ of lengths $n \ge m_1 \ge \cdots \ge m_r$.

Time: $n + m_1 + \cdots + m_r$ steps.

Problem 3. Expand **Problem 1**: Allow occurrences of π to *wrap around* Π (as if Π were a *ring of symbols*).

Bio-Inspired Pattern-Matching

Problem 1. A CA $\mathcal{C}$ has:

- length-n *master pattern* Π along row 0
- length-$(m \leq n)$ *input pattern* π left-justified along row $n-1$

$\mathcal{C}$ identifies all positions in Π where a copy of π begins.

Time: $n+m$ steps

Problem 2. $\mathcal{C}$ solves a sequence of instances of **Problem 1** for pattern Π and a sequence of input patterns, $\pi_1, \ldots, \pi_r$ of lengths $n \geq m_1 \geq \cdots \geq m_r$.

Time: $n + m_1 + \cdots + m_r$ steps.

Problem 3. Expand **Problem 1**: Allow occurrences of π to *wrap around* Π. (as if Π were a *ring of symbols*).

Time: $O(n)$ steps.

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching

The motivating challenge:
As $\mathcal{C}$ searches for all positions in Π where π begins, partial data "piles up."

σ_0	σ_1	σ_2	σ_3	$\cdots$
τ_0	τ_1	$\cdots$		
	τ_0	τ_1	$\cdots$	
		τ_0	τ_1	$\cdots$

Each column represents one parallel comparison-step.

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching

The motivating challenge:

As $\mathcal{C}$ searches for all positions in Π where π occurs, partial data "piles up."

Zip-matching avoids this congestion

σ_0	σ_1	σ_2	σ_3	σ_4	σ_5
$\tau_0^{(0)}$	$\tau_1^{(0)}$	$\tau_2^{(0)}$			

$\Longrightarrow \cdots \Longrightarrow$

σ_0	σ_1	σ_2	σ_3	σ_4	σ_5
		$\tau_2^{(0)}$	$\tau_2^{(1)}$	$\tau_2^{(2)}$	$\tau_2^{(3)}$
		$\tau_1^{(0)}$	$\tau_1^{(1)}$	$\tau_1^{(2)}$	$\tau_1^{(3)}$
		$\tau_0^{(0)}$	$\tau_0^{(1)}$	$\tau_0^{(2)}$	$\tau_0^{(3)}$

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching

The motivating challenge:

As $\mathcal{C}$ searches for all positions in Π where π occurs, partial data "piles up"

Zip-matching avoids this congestion

σ_0	σ_1	σ_2	σ_3	σ_4	σ_5
		$\tau_2^{(0)}$	$\tau_2^{(1)}$	$\tau_2^{(2)}$	$\tau_2^{(3)}$
		$\tau_1^{(0)}$	$\tau_1^{(1)}$	$\tau_1^{(2)}$	$\tau_1^{(3)}$
		$\tau_0^{(0)}$	$\tau_0^{(1)}$	$\tau_0^{(2)}$	$\tau_0^{(3)}$

$\Longrightarrow$

σ_0	σ_1	$()$ σ_2 $\tau_2^{(0)}$	$()$ σ_3 $\tau_2^{(1)}$	$()$ σ_4 $\tau_2^{(2)}$	$()$ σ_5 $\tau_2^{(3)}$
		$\tau_1^{(0)}$	$\tau_1^{(1)}$	$\tau_1^{(2)}$	$\tau_1^{(3)}$
		$\tau_0^{(0)}$	$\tau_0^{(1)}$	$\tau_0^{(2)}$	$\tau_0^{(3)}$

Bio-Inspired Pattern-Matching Underlying Algorithmic Ideas

Idea 1. Zip-matching

The motivating challenge:

As $\mathcal{C}$ searches for all positions in Π where π occurs, partial data "piles up"

Zip-matching avoids this congestion

σ_0	σ_1	$()$ σ_2 $\tau_2^{(0)}$	$()$ σ_3 $\tau_2^{(1)}$	$()$ σ_4 $\tau_2^{(2)}$	$()$ σ_5 $\tau_2^{(3)}$
		$\tau_1^{(0)}$	$\tau_1^{(1)}$	$\tau_1^{(2)}$	$\tau_1^{(3)}$
		$\tau_0^{(0)}$	$\tau_0^{(1)}$	$\tau_0^{(2)}$	$\tau_0^{(3)}$

$\Longrightarrow$

σ_0	$\underline{\varepsilon_{220}}$ σ_1 $\tau_1^{(0)}$	$\underline{\varepsilon_{321}}$ σ_2 $\tau_1^{(1)}$	$\underline{\varepsilon_{422}}$ σ_3 $\tau_1^{(2)}$	$\underline{\varepsilon_{523}}$ σ_4 $\tau_1^{(3)}$	σ_5
	$\tau_0^{(0)}$	$\tau_0^{(1)}$	$\tau_0^{(2)}$	$\tau_0^{(3)}$	

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. <u>Zip-matching</u>

The motivating challenge:

As $\mathcal{C}$ searches for all positions in Π where π occurs, partial data "piles up"

<u>Zip-matching avoids this congestion</u>

σ_0	$\underline{\varepsilon_{220}}$ σ_1 $\tau_1^{(0)}$	$\underline{\varepsilon_{321}}$ σ_2 $\tau_1^{(1)}$	$\underline{\varepsilon_{422}}$ σ_3 $\tau_1^{(2)}$	$\underline{\varepsilon_{523}}$ σ_4 $\tau_1^{(3)}$	σ_5
	$\tau_0^{(0)}$	$\tau_0^{(1)}$	$\tau_0^{(2)}$	$\tau_0^{(3)}$	

$\Longrightarrow$

$\underline{\varepsilon_{220} \wedge \varepsilon_{110}}$ σ_0 $\tau_0^{(0)}$	$\underline{\varepsilon_{321} \wedge \varepsilon_{211}}$ σ_1 $\tau_0^{(1)}$	$\underline{\varepsilon_{422} \wedge \varepsilon_{312}}$ σ_2 $\tau_0^{(2)}$	$\underline{\varepsilon_{523} \wedge \varepsilon_{413}}$ σ_3 $\tau_0^{(3)}$	σ_4	σ_5

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching

The motivating challenge:

As $\mathcal{C}$ searches for all positions in Π where π occurs, partial data "piles up"

Zip-matching avoids this congestion

$\underline{\varepsilon_{220} \wedge \varepsilon_{110}}$ σ_0 $\tau_0^{(0)}$	$\underline{\varepsilon_{321} \wedge \varepsilon_{211}}$ σ_1 $\tau_0^{(1)}$	$\underline{\varepsilon_{422} \wedge \varepsilon_{312}}$ σ_2 $\tau_0^{(2)}$	$\underline{\varepsilon_{523} \wedge \varepsilon_{413}}$ σ_3 $\tau_0^{(3)}$	σ_4	σ_5

$\Longrightarrow$

$\underline{\varepsilon_{220} \wedge \varepsilon_{110} \wedge \varepsilon_{000}}$ σ_0	$\underline{\varepsilon_{321} \wedge \varepsilon_{211} \wedge \varepsilon_{101}}$ σ_1	$\underline{\varepsilon_{422} \wedge \varepsilon_{312} \wedge \varepsilon_{202}}$ σ_2	$\underline{\varepsilon_{523} \wedge \varepsilon_{413} \wedge \varepsilon_{303}}$ σ_3	σ_4	σ_5

Bio-Inspired Pattern-Matching Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Zip-matching on length-n Π and length-$(m \leq n)$ π can be done in $n + m$ steps.

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n+m$ steps

Idea 2. Tracks and Layers

The symbols within FSMs' memories can have *structure*

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers

The symbols within FSMs' memories can have *structure*
e.g., they can be *tuples* of atomic symbols: $\alpha = \langle \beta_1, \beta_2, \beta_3 \rangle$

Bio-Inspired Pattern-Matching Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers

The symbols within FSMs' memories can have *structure*
e.g., they can be *tuples* of atomic symbols: $\alpha = \langle \beta_1, \beta_2, \beta_3 \rangle$

Thereby, we can endow a mesh's rows and columns with *tracks*

Bio-Inspired Pattern-Matching Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers

The symbols within FSMs' memories can have *structure*
e.g., they can be *tuples* of atomic symbols: $\alpha = \langle \beta_1, \beta_2, \beta_3 \rangle$

Thereby, we can endow a mesh's rows and columns with *tracks*
—even to the point of endowing $\mathcal{M}_n$ with complete *layers*:

$$\begin{array}{|c|}\hline \langle \alpha_0, \beta_0 \rangle \\ \hline \langle \alpha_1, \beta_1 \rangle \\ \hline \end{array} \Longrightarrow \begin{array}{|c|}\hline \alpha_0 \\ \alpha_1 \\ \hline \end{array} \quad \ddots \quad \begin{array}{|c|}\hline \beta_0 \\ \beta_1 \\ \hline \end{array} \quad \ddots$$

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers: **Time:** $O(n)$ steps

$\mathcal{C}$ can implement Tracks and Layers on rows, columns, or all of $\mathcal{M}_n$ by performing a barrier synchronization. The *Firing Squad Protocol* achieves this in time $O(n)$.

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers: **Time:** $O(n)$ steps

Idea 3. The L-C transformation: Fast Cyclic Rotations

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers: **Time:** $O(n)$ steps

Idea 3. The L-C transformation: Fast Cyclic Rotations

When one cyclically rotates a pattern, intersymbol distances can change a lot:

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers: **Time:** $O(n)$ steps

Idea 3. The L-C transformation: Fast Cyclic Rotations

When one cyclically rotates a pattern, intersymbol distances can change a lot: cf., σ_{n-1} and σ_{n-2} within $\underline{\Pi = \sigma_0 \cdots \sigma_{n-2}\sigma_{n-1}}$ and $\underline{\rho(\Pi) = \sigma_{n-1}\sigma_0 \cdots \sigma_{n-2}}$

Bio-Inspired Pattern-Matching Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers: **Time:** $O(n)$ steps

Idea 3. The L-C transformation: Fast Cyclic Rotations

When one cyclically rotates a pattern, inter-symbol distance can change a lot:

The Linear-Cyclic (L-C) transformation avoids this: distances stay small!

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers: **Time:** $O(n)$ steps

Idea 3. The L-C transformation: Fast Cyclic Rotations

When one cyclically rotates a pattern, inter-symbol distance can change a lot:

The L-C transformation:

Write $\sigma_0 \cdots \sigma_{n-1}$ by selecting symbols from alternating ends:

$$\lambda(\sigma_0\sigma_1 \cdots \sigma_{n-2}\sigma_{n-1}) \ = \ \sigma_0\sigma_{n-1}\sigma_1\sigma_{n-2} \cdots$$

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers: **Time:** $O(n)$ steps

Idea 3. The L-C transformation: Fast Cyclic Rotations

When one cyclically rotates a pattern, inter-symbol distance can change a lot:

The L-C transformation:
Write $\sigma_0 \cdots \sigma_{n-1}$ by selecting symbols from alternating ends.

The interplay between the *rotation operator* ρ and the L-C operator λ
(for both odd- and even- n):

ξ	$=$	$\sigma_0\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5$	η	$=$	$\tau_0\tau_1\tau_2\tau_3\tau_4\tau_5\tau_6$
$\lambda(\xi)$	$=$	$\sigma_0\sigma_5\sigma_1\sigma_4\sigma_2\sigma_3$	$\lambda(\eta)$	$=$	$\tau_0\tau_6\tau_1\tau_5\tau_2\tau_4\tau_3$
$\rho(\xi)$	$=$	$\sigma_5\sigma_0\sigma_1\sigma_2\sigma_3\sigma_4$	$\rho(\eta)$	$=$	$\tau_6\tau_0\tau_1\tau_2\tau_3\tau_4\tau_5$
$\lambda(\rho(\xi))$	$=$	$\sigma_5\sigma_4\sigma_0\sigma_3\sigma_1\sigma_2$	$\lambda(\rho(\eta))$	$=$	$\tau_6\tau_5\tau_0\tau_4\tau_1\tau_3\tau_2$

Bio-Inspired Pattern-Matching
Underlying Algorithmic Ideas

Idea 1. Zip-matching: **Time:** $n + m$ steps

Idea 2. Tracks and Layers: **Time:** $O(n)$ steps

Idea 3. The L-C transformation: Fast Cyclic Rotations: **Time:** $O(n)$ steps

The L-C transformation can be done and undone to a pattern Π in linear time.

Bio-Inspired Pattern-Matching Solving the Three Problems

Problem 1.

The enabling tool is *Zip-Matching*

Bio-Inspired Pattern-Matching Solving the Three Problems

Problem 1.

The enabling tool is *Zip-Matching*

Turning π "on its head" and replicating it along row 0 is "tailor-made" for cellular automata.

Bio-Inspired Pattern-Matching
Solving the Three Problems

Problem 1. **Enabling tool:** *Zip-Matching*; **Time:** $n + m$ steps

Bio-Inspired Pattern-Matching
Solving the Three Problems

Problem 1. **Enabling tool:** *Zip-Matching*; **Time:** $n + m$ steps

Problem 2.

The enabling tool is *Pipelined Zip-Matching*

Bio-Inspired Pattern-Matching
Solving the Three Problems

Problem 1. **Enabling tool:** *Zip-Matching*; **Time:** $n + m$ steps

Problem 2.

The enabling tool is *Pipelined Zip-Matching*

Time the pipeline must be done carefully — a straightforward challenge for cellular automata

Bio-Inspired Pattern-Matching
Solving the Three Problems

Problem 1. **Enabling tool:** *Zip-Matching*; **Time:** $n + m$ steps

Problem 2. **Enabling tool:** *Pipelined Zip-Matching*; **Time:** $n + m_1 + \cdots + m_r$ steps

Bio-Inspired Pattern-Matching Solving the Three Problems

Problem 1. **Enabling tool:** *Zip-Matching*; **Time:** $n + m$ steps

Problem 2. **Enabling tool:** *Pipelined Zip-Matching*; **Time:** $n + \Sigma_i m_i$ steps

Problem 3. (the most complex problem)

1. Establish 3 layers in $\mathcal{M}_n$: π-layer, Π-layer, flow-layer

Bio-Inspired Pattern-Matching Solving the Three Problems

Problem 1. **Enabling tool:** *Zip-Matching*; **Time:** $n + m$ steps

Problem 2. **Enabling tool:** *Pipelined Zip-Matching*; **Time:** $n + \Sigma_i m_i$ steps

Problem 3. (the most complex problem)

1. Establish 3 layers in $\mathcal{M}_n$: π-layer, Π-layer, flow-layer
2. Populate the Π-layer with all cyclic rotations of Π, one per row

Bio-Inspired Pattern-Matching
Solving the Three Problems

Problem 1. **Enabling tool:** *Zip-Matching*; **Time:** $n + m$ steps

Problem 2. **Enabling tool:** *Pipelined Zip-Matching*; **Time:** $n + \Sigma_i m_i$ steps

Problem 3. (the most complex problem)

1. Establish 3 layers in $\mathcal{M}_n$: π-layer, Π-layer, flow-layer
2. Populate the Π-layer with all cyclic rotations of Π, one per row
3. Prepare π for Zip-Matching at row 0 within the π-layer

Bio-Inspired Pattern-Matching Solving the Three Problems

Problem 1. **Enabling tool:** *Zip-Matching*; **Time:** $n + m$ steps

Problem 2. **Enabling tool:** *Pipelined Zip-Matching*; **Time:** $n + \Sigma_i m_i$ steps

Problem 3. (the most complex problem)

1. Establish 3 layers in $\mathcal{M}_n$: π-layer, Π-layer, flow-layer
2. Populate the Π-layer with all cyclic rotations of Π, one per row
3. Prepare π for Zip-Matching at row 0 within the π-layer
4. Inductively:
   (a) as $n - m$ copies of π (in the π-layer) get Zip-Matched to a cyclic rotation of Π (say at row k of the Π-layer),
   (b) $\mathcal{C}$ sends a replica of the $n - m$ copies of π (in the flow layer) southward to the cyclic rotation of Π at row $k + 1$

Bio-Inspired Pattern-Matching
Solving the Three Problems

Problem 1. **Enabling tool:** *Zip-Matching*; **Time:** $n + m$ steps

Problem 2. **Enabling tool:** *Pipelined Zip-Matching*; **Time:** $n + \Sigma_i m_i$ steps

Problem 3. (the most complex problem)

1. Establish 3 layers in $\mathcal{M}_n$: π-layer, Π-layer, flow-layer
2. Populate the Π-layer with all cyclic rotations of Π, one per row
3. Prepare π for Zip-Matching at row 0 within the π-layer
4. Inductively:
   (a) as $n-m$ copies of π (in the π-layer) get Zip-Matched to a cyclic rotation of Π (say at row k of the Π-layer),
   (b) $\mathcal{C}$ sends a replica of the $n-m$ copies of π (in the flow layer) southward to the cyclic rotation of Π at row $k+1$

$\mathcal{C}$ enhances parallelism by orchestrating step 4 in the manner of a *Systolic Array*.

Therefore, the entire solution of Problem 3 is accomplished within $O(n)$ steps.

Bio-Inspired Pattern-Matching
Summing Up

Problem 1. **Enabling tool:** *Zip-Matching*;
Time: $n + m$ steps

Problem 2. **Enabling tool:** *Pipelined Zip-Matching*;
Time: $n + \Sigma_i m_i$ steps

Problem 3. *Several tools*; **Time:** $O(n)$ steps