Fundamental properties of cellular automata and computation ability 2

Katsunobu Imai
Hiroshima University

IEC laboratory

Quasi periodic tile and language theory (Kyoto) Jun. 8-10 2009

How to Play with Cellular Programming?

- Signal based algorithm
  - Composition of various signals are frequently used.
- Reducing the number of used states is always wanted. (**It needs craftsmanship training.**)
- Programming of rules and programming of Initial configurations
  - In some case, rules and the number of state are given and programming is just the problem of giving a proper initial configuration. (cf. the game of Life)

Signals on Cellular Automata (Example)

rules

$(s, s, s) \to s$

Speed 1 signal:

time ↓

s	s	R	s	s	s	s	s	s	s
s	s	s	R	s	s	s	s	s	s
s	s	s	s	R	s	s	s	s	s

$(s, s, R) \to s$

$(s, R, s) \to s$

$(R, s, s) \to R$

Speed 1/3 signal:

time ↓

s	s	A	s	s	s	s	s	s	s
s	s	B	s	s	s	s	s	s	s
s	s	C	s	s	s	s	s	s	s
s	s	s	A	s	s	s	s	s	s
s	s	s	B	s	s	s	s	s	s
s	s	s	C	s	s	s	s	s	s

$(s, s, A) \to s$

$(s, s, B) \to s$

$(s, s, C) \to s$

$(s, A, s) \to B$

$(s, B, s) \to C$

$(s, C, s) \to s$

$(A, s, s) \to s$

$(B, s, s) \to s$

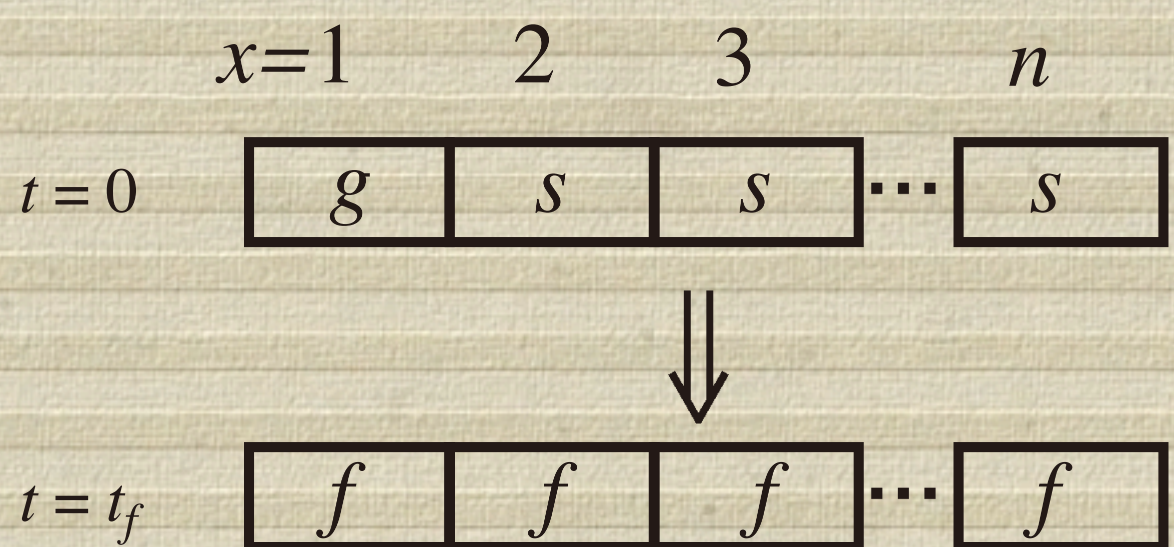
$(C, s, s) \to A$

$(s, s, s) \to s$

Firing Squad Synchronization Problem (FSSP)

Moore 1964

To construct an one-dimensional radius 1 CA of arbitrary finite length such that one of the end cell (general) makes all the other cells (soldiers) be in a particular state (firing state) at a certain time.



1964〜 Minsky & McCathy 3n time

1966 Goto, optimal time

1967 Balzar 8-state, optimal time

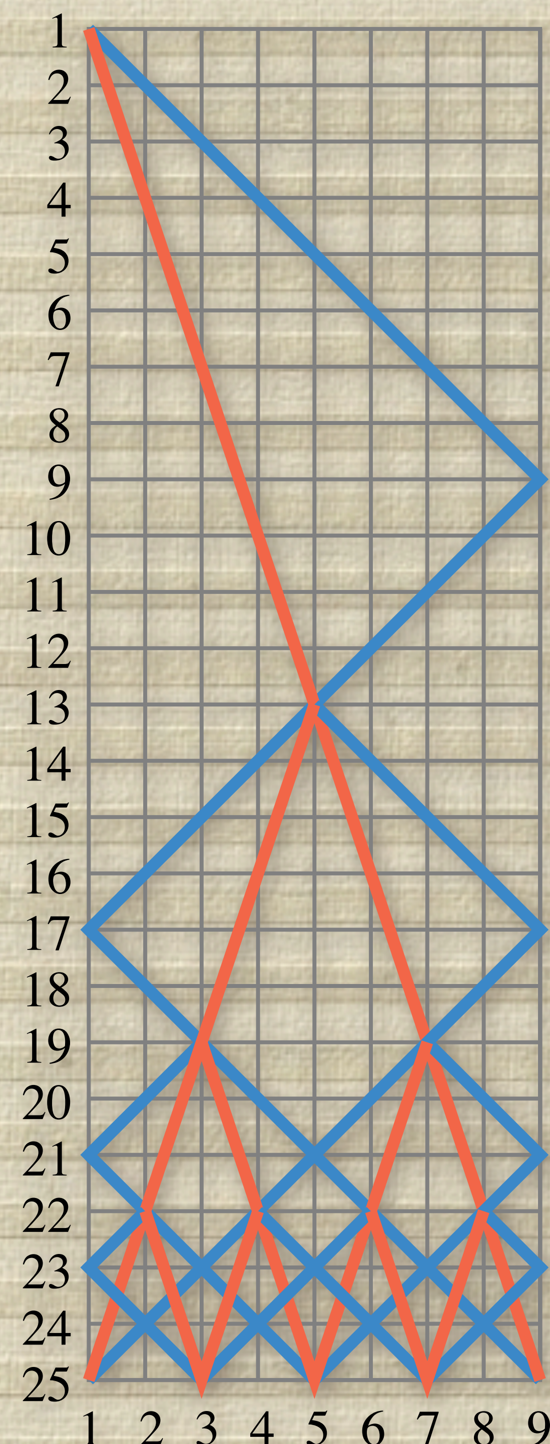
1967 Mazoyer 6-state, optimal time

g: general, *s*: soldier, *f*: firing state

4-state is known to be impossible.
5-state is still remained.

Firing Squad Synchronization Problem (FSSP)

7-state 3n-time solution



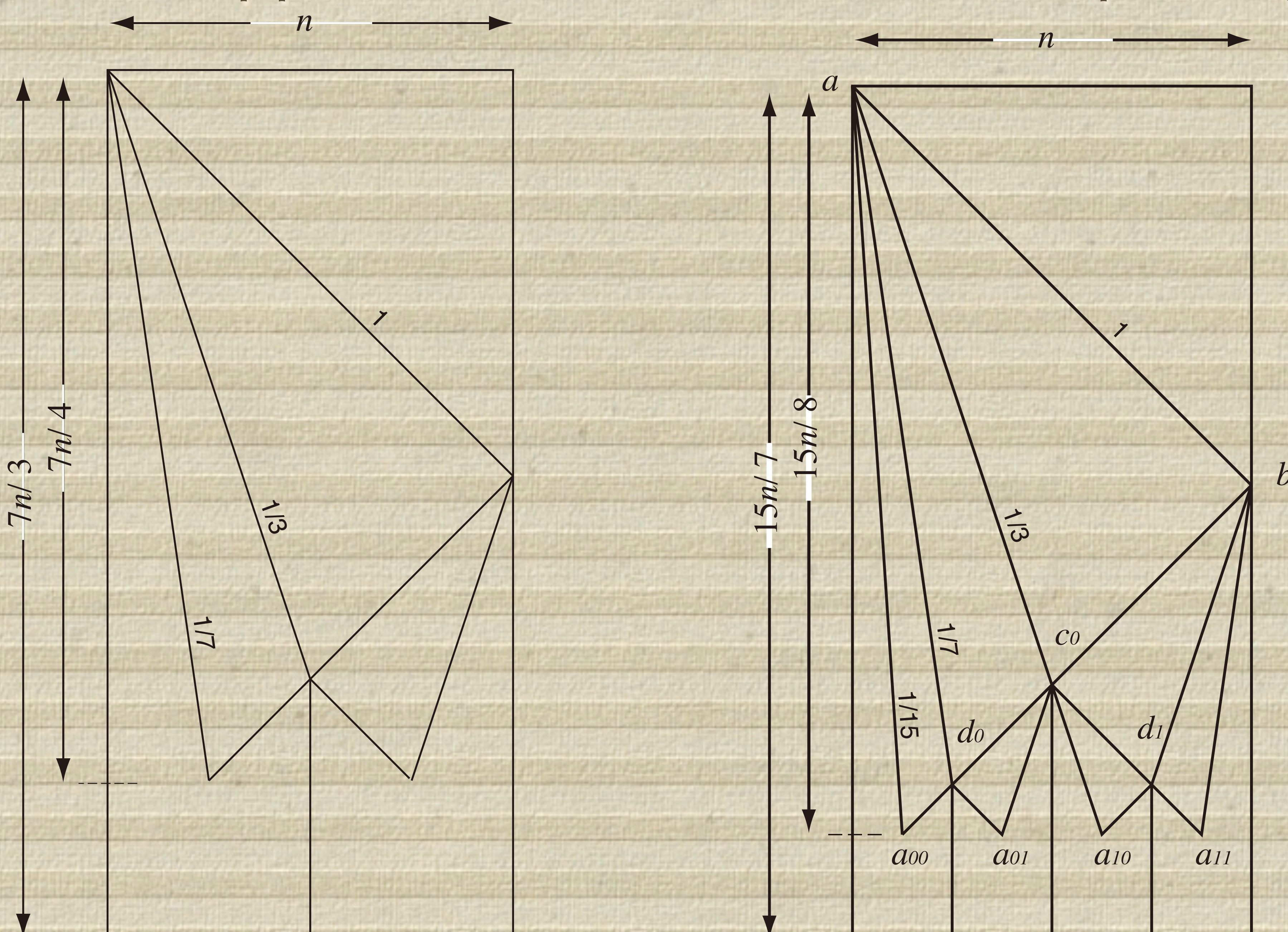
G	s	s	s	s	s	s
X	R	s	s	s	s	s
X	s	R	s	s	s	s
s	A	s	R	s	s	s
s	B	s	s	R	s	s
s	C	s	s	s	R	s
s	s	A	s	s	s	R
s	s	B	s	s	s	L
s	s	C	s	s	L	s
s	s	s	A	L	s	s
s	s	s	G	s	s	s
s	s	L	G	R	s	s
s	L	s	G	s	R	s
L	s	a	G	A	s	R
R	s	b	s	B	s	L
s	R	c	s	C	L	s
s	G	G	s	G	G	s
L	G	G	X	G	G	R
R	G	G	Y	G	G	L
F	F	F	F	F	F	F

divide and conquer algorithm

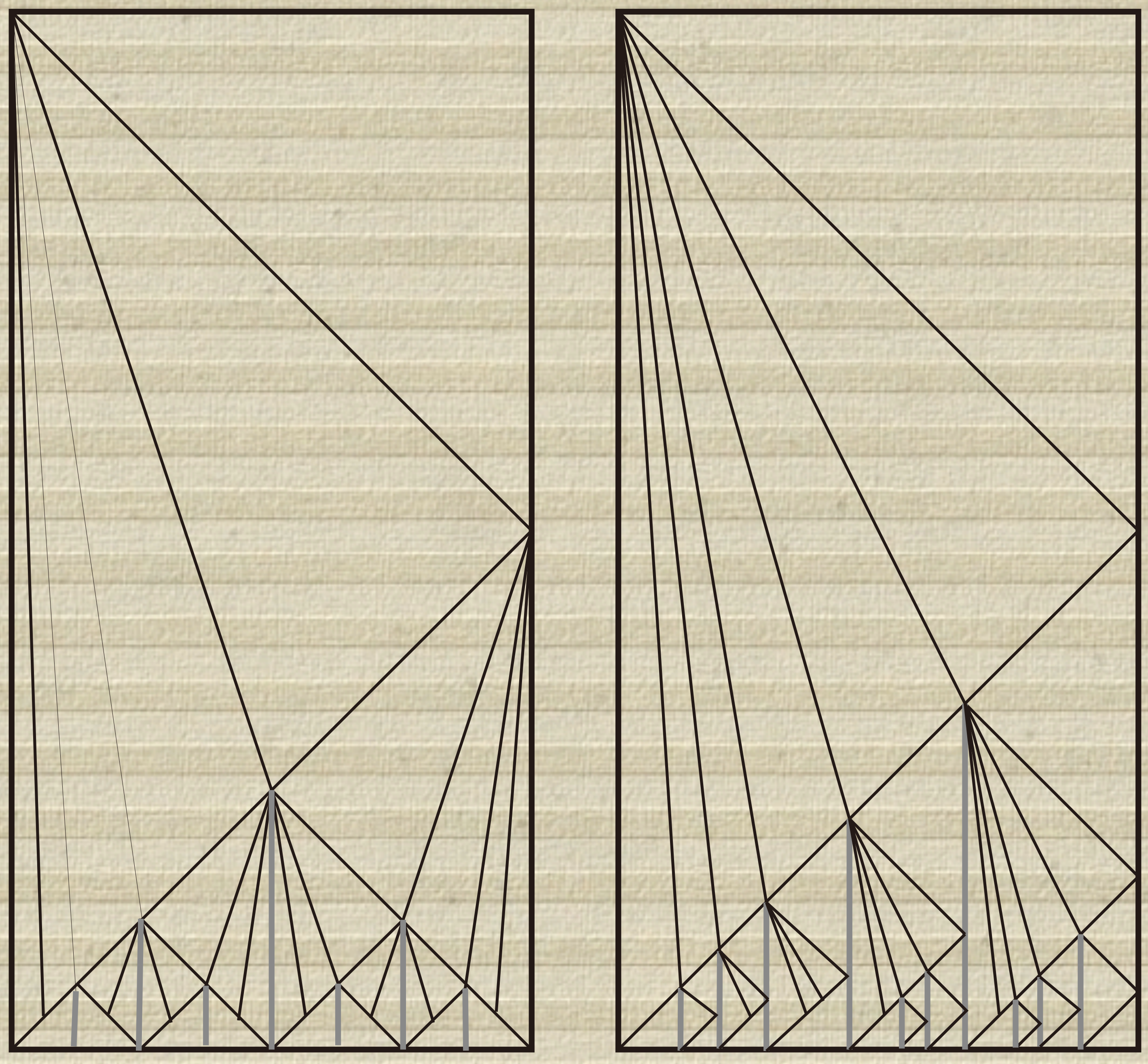
Yunes 1997

Optimization of Firing Time

Recursive application of divide and conquer algorithm



Outline of Signals in Optimal Time Solution



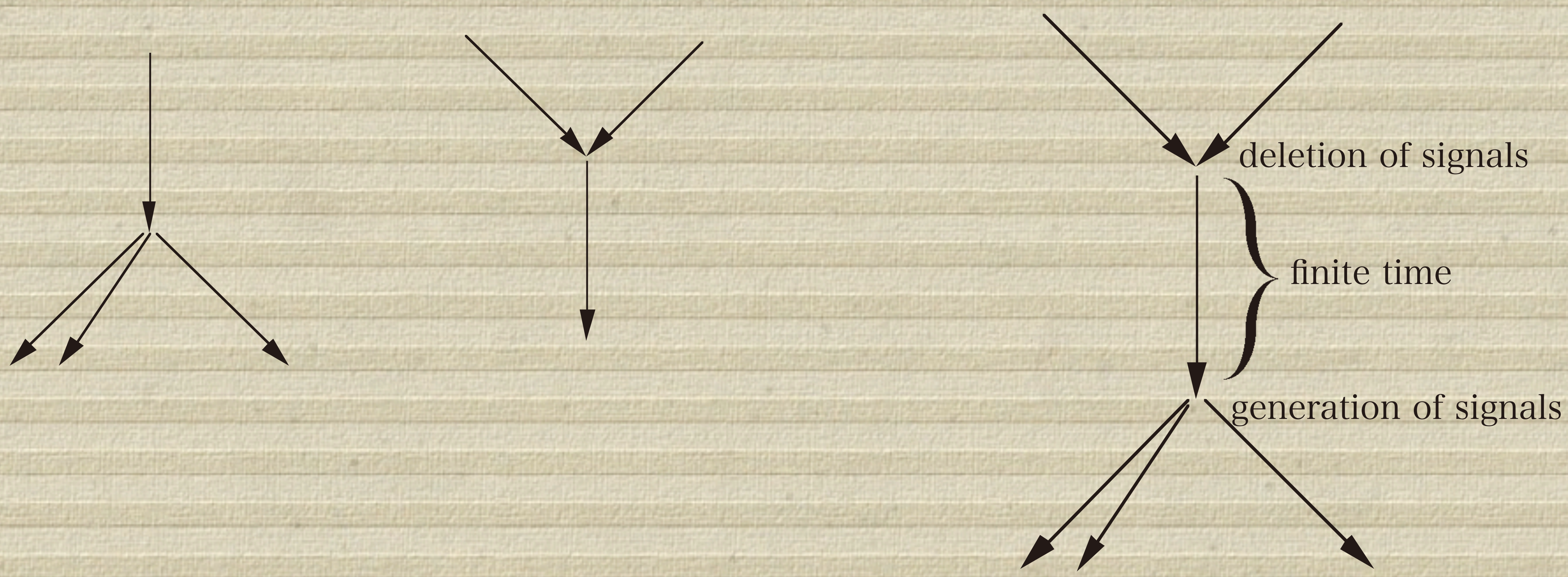
(a) Waksman-Balzer type solution (b) Mazoyer type solution

Limit of recursive application of divide and conquer algorithm attains optimal 2n-2 time solution.

Designing Reversible Cellular Automata

Signals and Reversibility

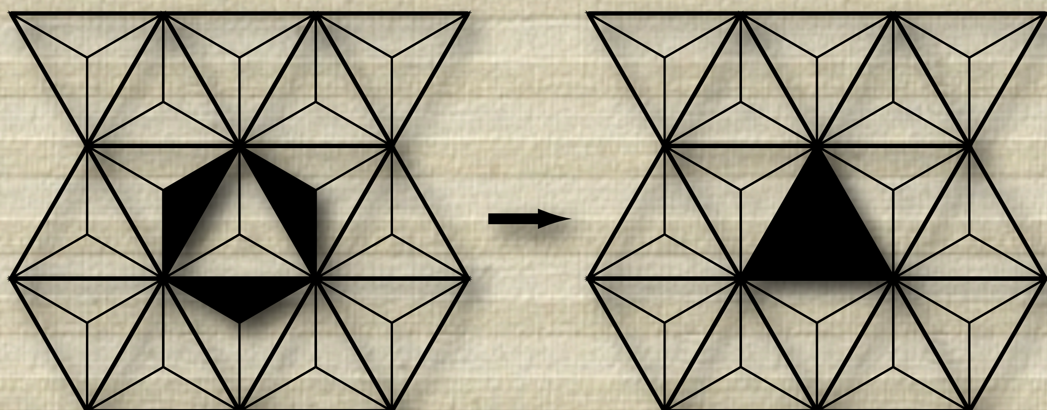
You can't generate or annihilate signals.

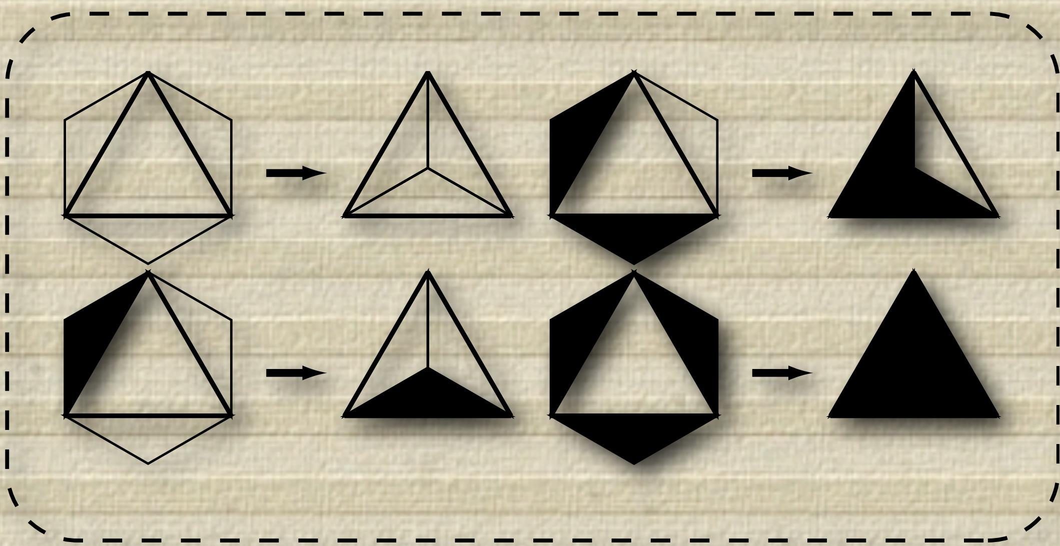


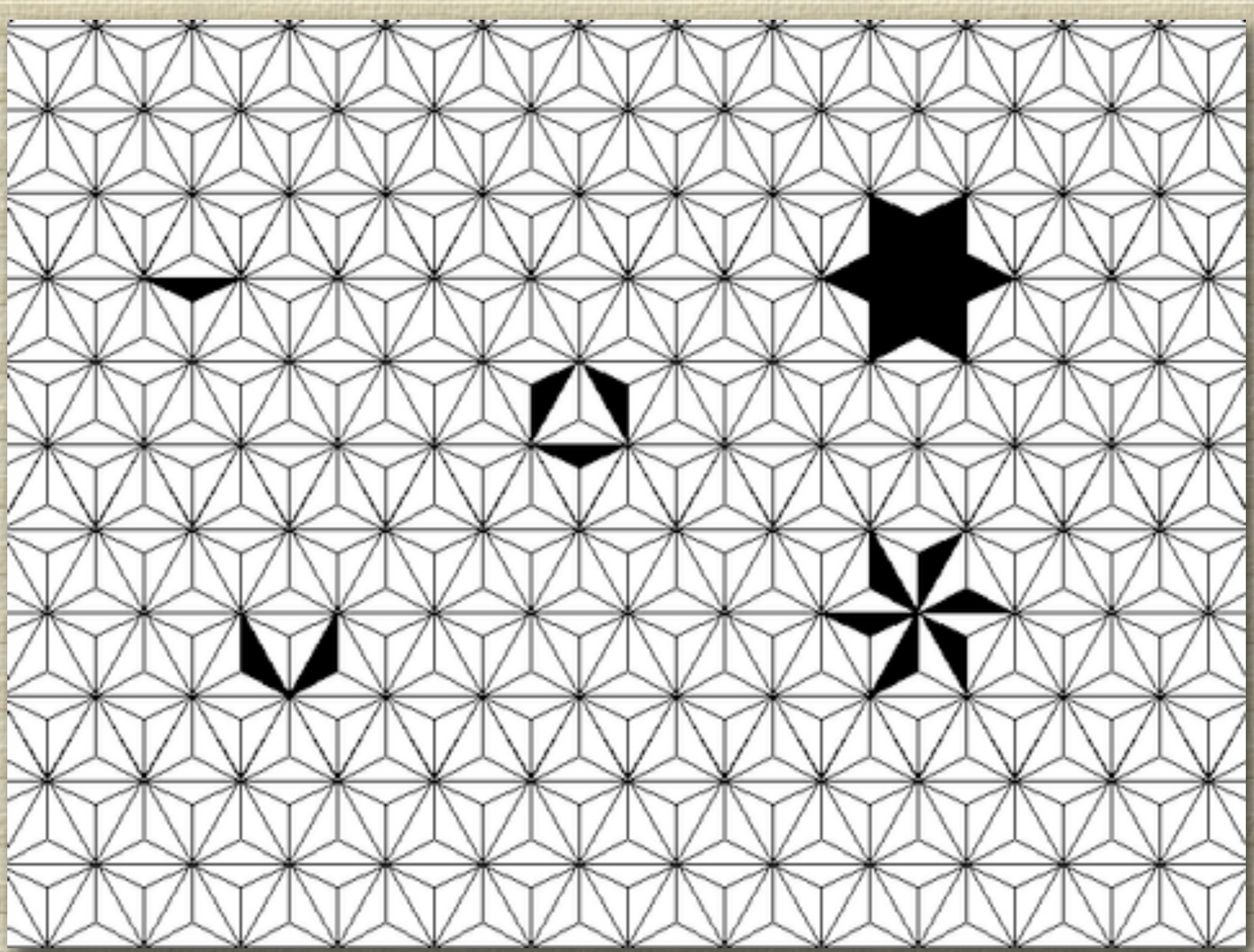
Generation and annihilation of signals must be coupled.

8-state Logically-Universal Reversible CA

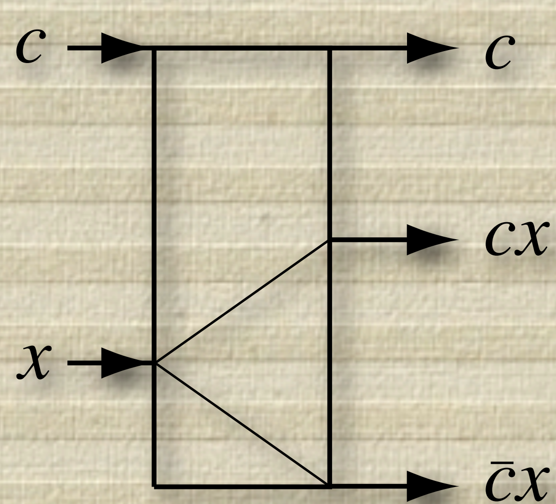
Imai, Morita 1998

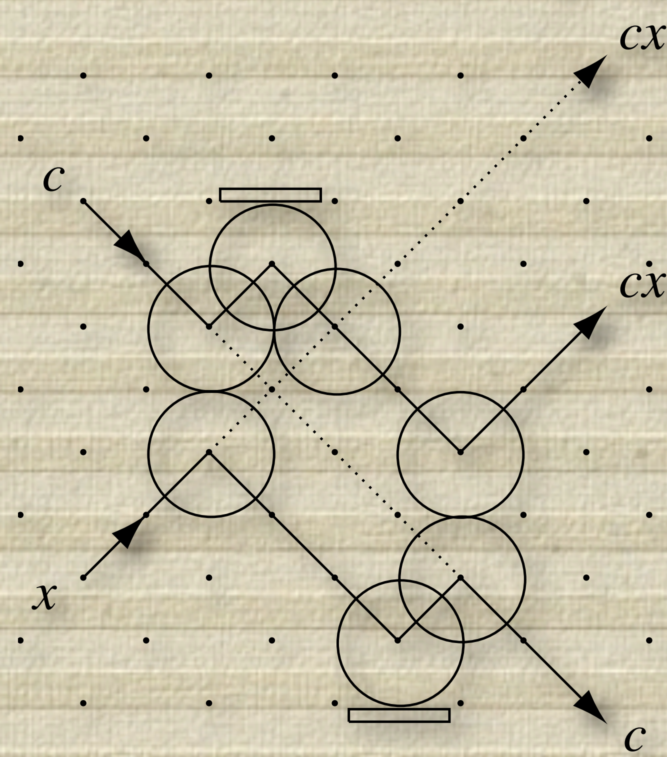


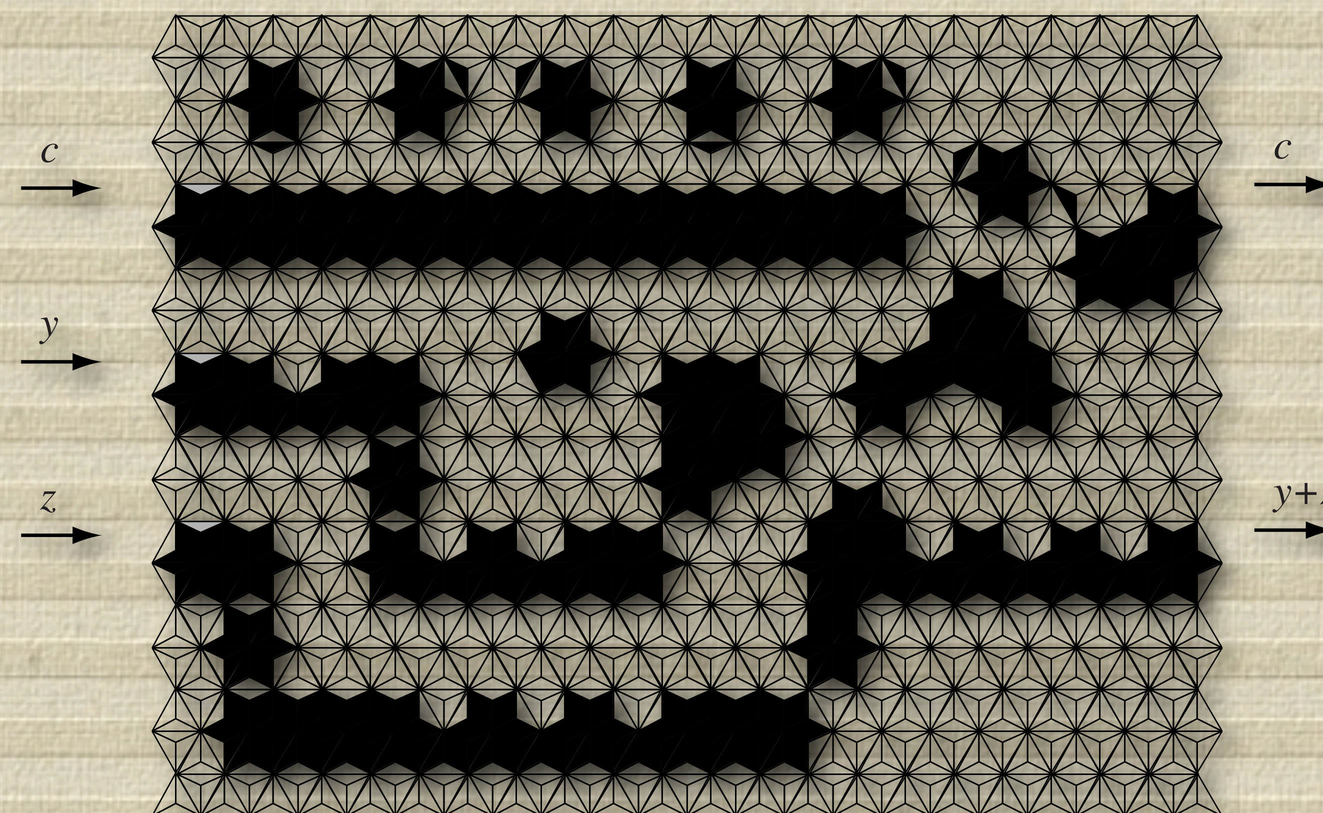




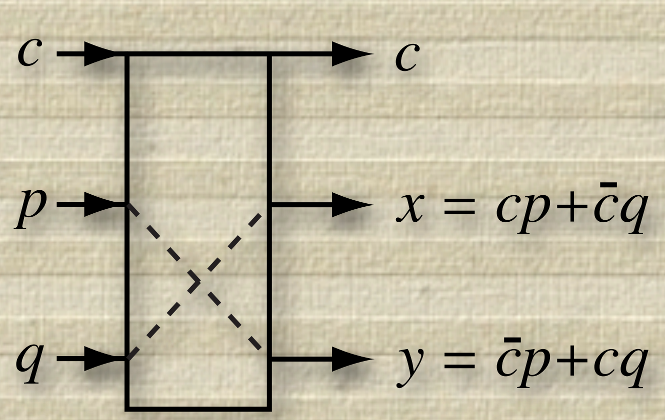
Realization of a Switch Gate

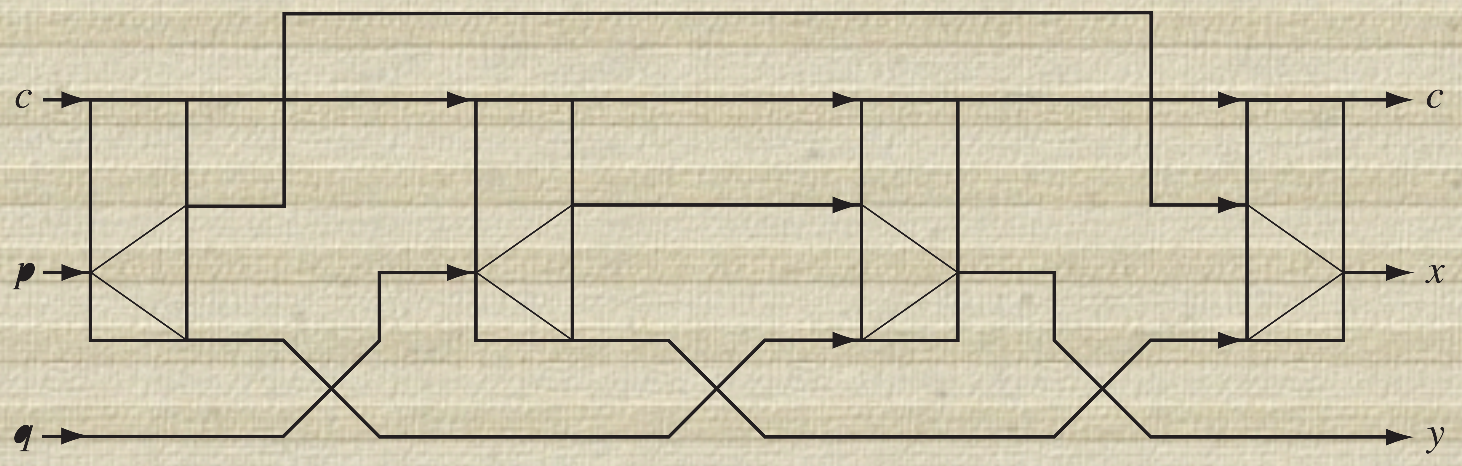






Realization of a Fredkin Gate





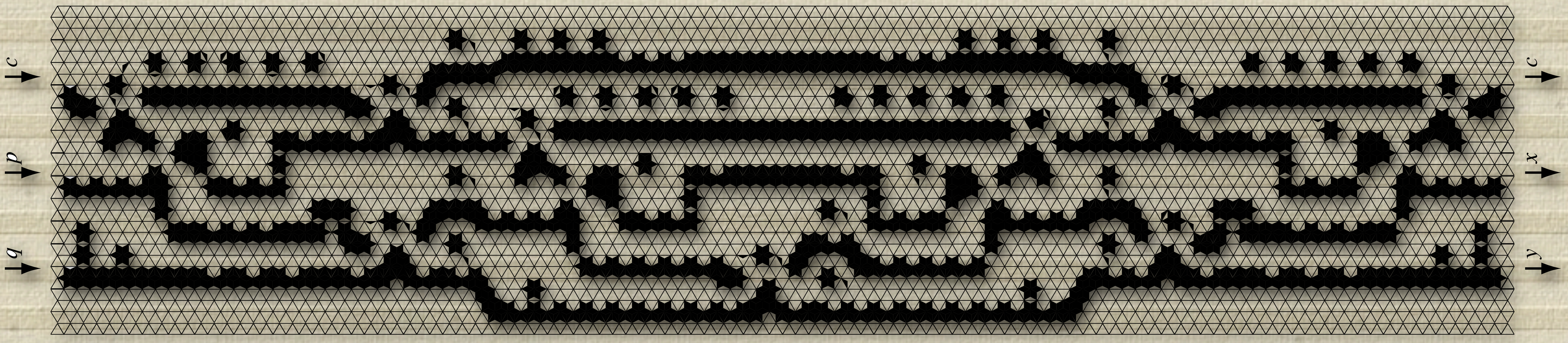


Figure 19: A configuration of F-gate

How about reversible self-reproduction?

Self-reproducing CA were intensively studied by many researchers of Artificial Life.

Reversible and **non-dissipative** self-reproducing organism has a flavor of paradox and the setting seemed to be fun for us.

Low power self-reproducing organism!
Is it possible to find any application to mesoscopic systems?

Langton's self-reproducing loop

An 8-state von Neumann neighbor cellular automaton

"genetic information is handled **interpreted** and **uninterpreted**

Langton 1984

The condition of computation and construction universality is too excess for the model of biological self-reproduction.

Langton's self-reproducing loop

An 8-state von Neumann neighbor cellular automaton

"genetic information is handled **interpreted** and **uninterpreted**

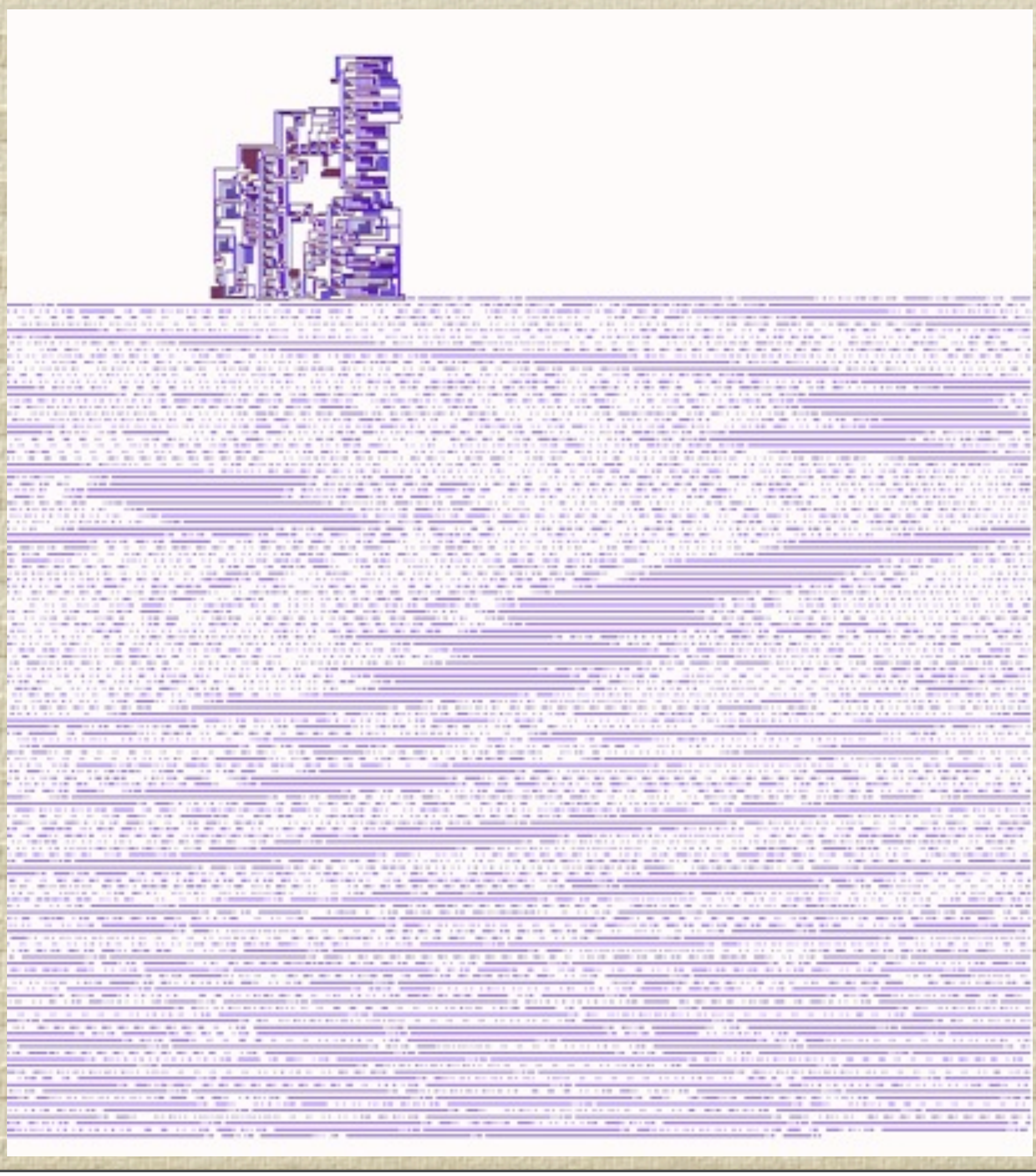
Langton 1984

The condition of computation and construction universality is too excess for the model of biological self-reproduction.

Self-inspection: another way of self-reproduction

1979 Laing

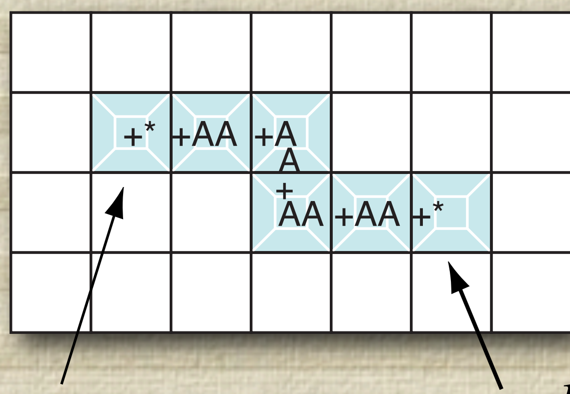
- Without genetic description
- Self-inspection is a very simple framework.
- On the fly composition of the blueprint of itself.



A Reversible Self-reproducing Cellular Automaton

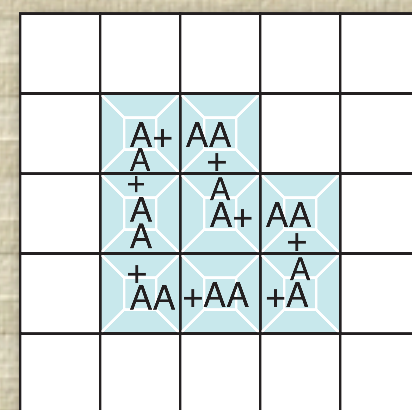
"Worms" and "Loops" can be self-reproduce in RCA.

(Morita, Imai 1996)



tail
(encodes its shape)

head
(interprets commands to advance an arm)

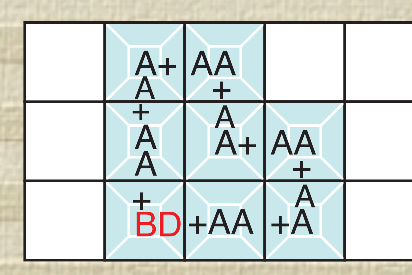


Advance & Branch commands

Command		Operation
First signal	Second signal	
A	A	Advance the head forward
A	B	Advance the head leftward
A	C	Advance the head rightward
B	A	Branch the wire in three ways
B	B	Branch the wire in two ways (leftward)
B	C	Branch the wire in two ways (rightward)

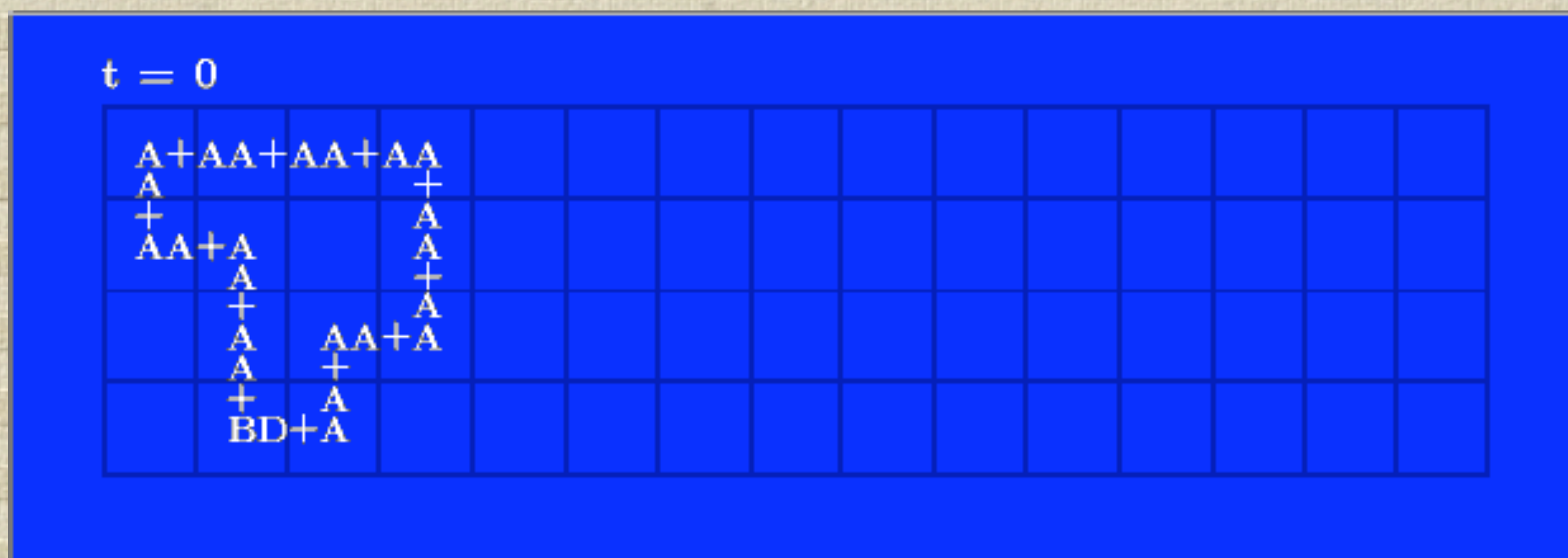
Self-reproducing commands DB,DC

Command		Operation
First signal	Second signal	
D	B	Create an arm
B	C	Encode the shape of the Loop



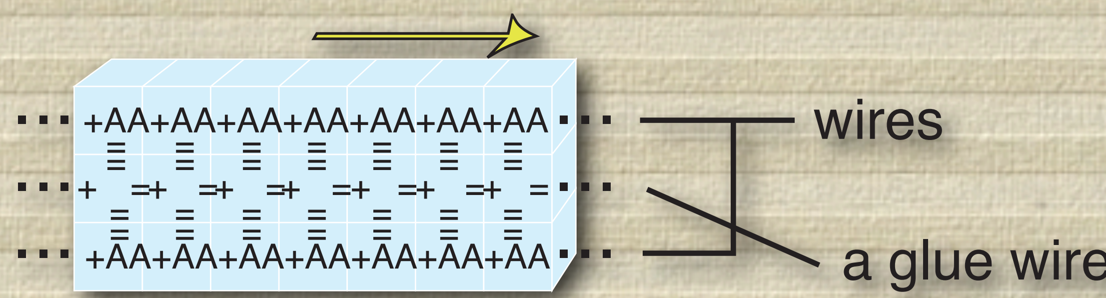
The DB command advances an arm

An Example:

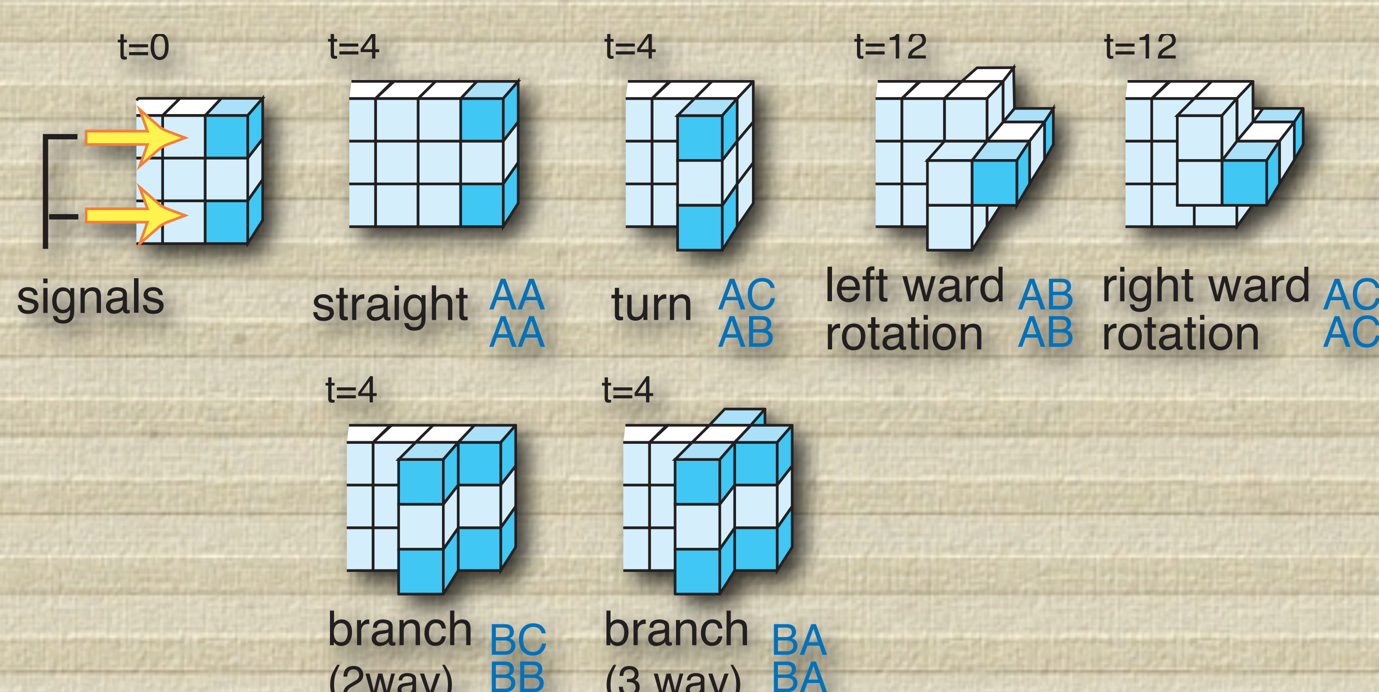


A Three Dimensional Reversible Self-reproducing Model

Three-ribbon worm

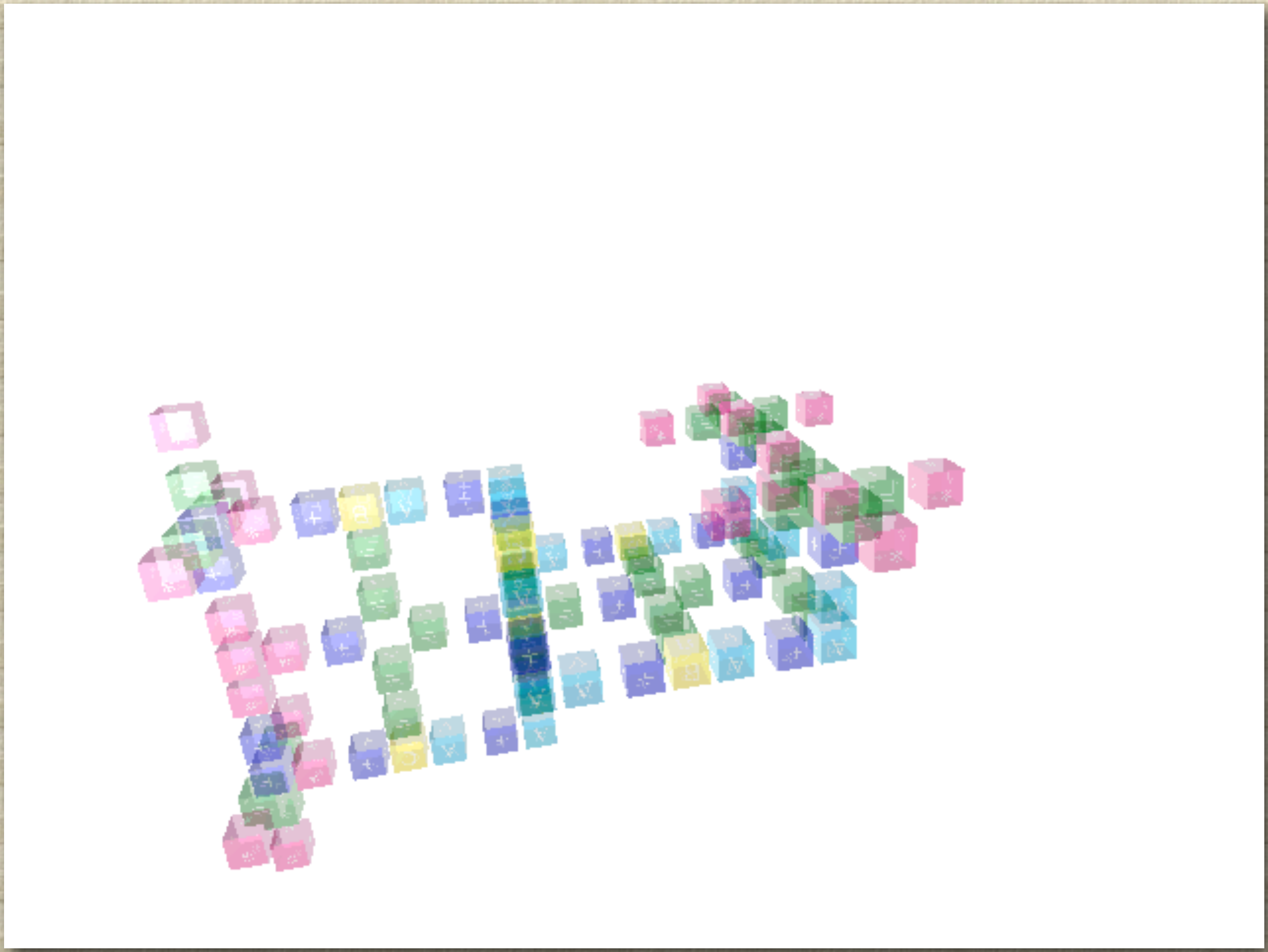


Commands



Imai, Hori, Morita 2002

Exapmle: a simple worm



Positioning commands and shape commands

The shape of a daughter Loop can be changed.

Original Positioning commands: $\begin{matrix}\mathrm{AA}^{24}\\ \mathrm{AA}^{24}\end{matrix}$

Positioning commands of this example:

$$\begin{matrix}\mathrm{AA}^5\,\mathrm{ABAA}^6\,\mathrm{AA}^3\,\mathrm{ABAAABAA}^6\\ \mathrm{AA}^5\,\mathrm{ACAA}^6\,\mathrm{AA}^3\,\mathrm{ACAAACAA}^6\end{matrix}$$

The complete description of its commands:

$$\underbrace{\begin{matrix}(\mathrm{AA}^5\,\mathrm{ABAA}^6\,\mathrm{AA}^3\,\mathrm{ABAAABAA}^6)\\ (\mathrm{AA}^5\,\mathrm{ACAA}^6\,\mathrm{AA}^3\,\mathrm{ACAAACAA}^4)\end{matrix}}_{\text{positioning signals}}\underbrace{\begin{matrix}\mathrm{AA}^5\,\mathrm{ABAA}^5\,\mathrm{ABAA}^5\,\mathrm{ABAA}^6\\ \mathrm{AA}^5\,\mathrm{ACAA}^5\,\mathrm{ACAA}^5\,\mathrm{ACAA}^6\end{matrix}}_{\text{shape signals}}$$

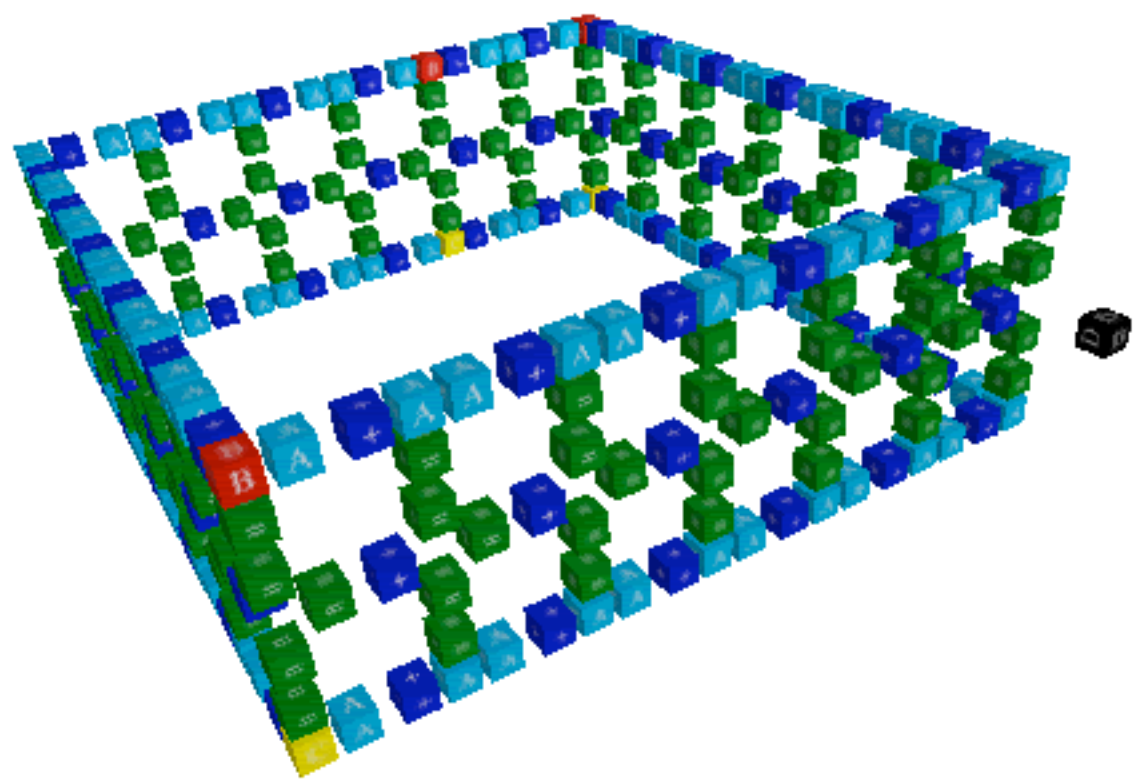
Positioning commands and shape commands

The shape of a daughter Loop can be changed.

Original Positioning commands: $\begin{matrix}\mathrm{AA}^{24}\\ \mathrm{AA}^{24}\end{matrix}$

Positioning commands of this example:

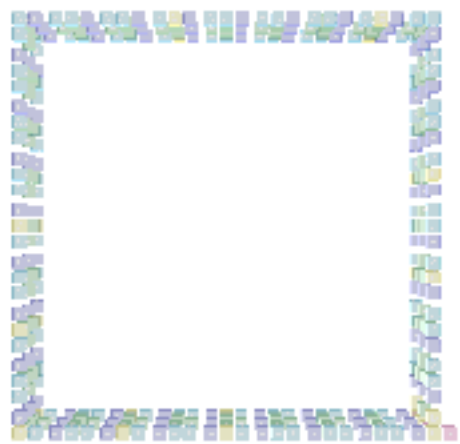
$$\begin{matrix}\mathrm{AA}^{5}\,\mathrm{ABAA}^{6}\,\mathrm{AA}^{3}\,\mathrm{ABAAABAA}^{6}\\ \mathrm{AA}^{5}\,\mathrm{ACAA}^{6}\,\mathrm{AA}^{3}\,\mathrm{ACAAACAA}^{6}\end{matrix}$$



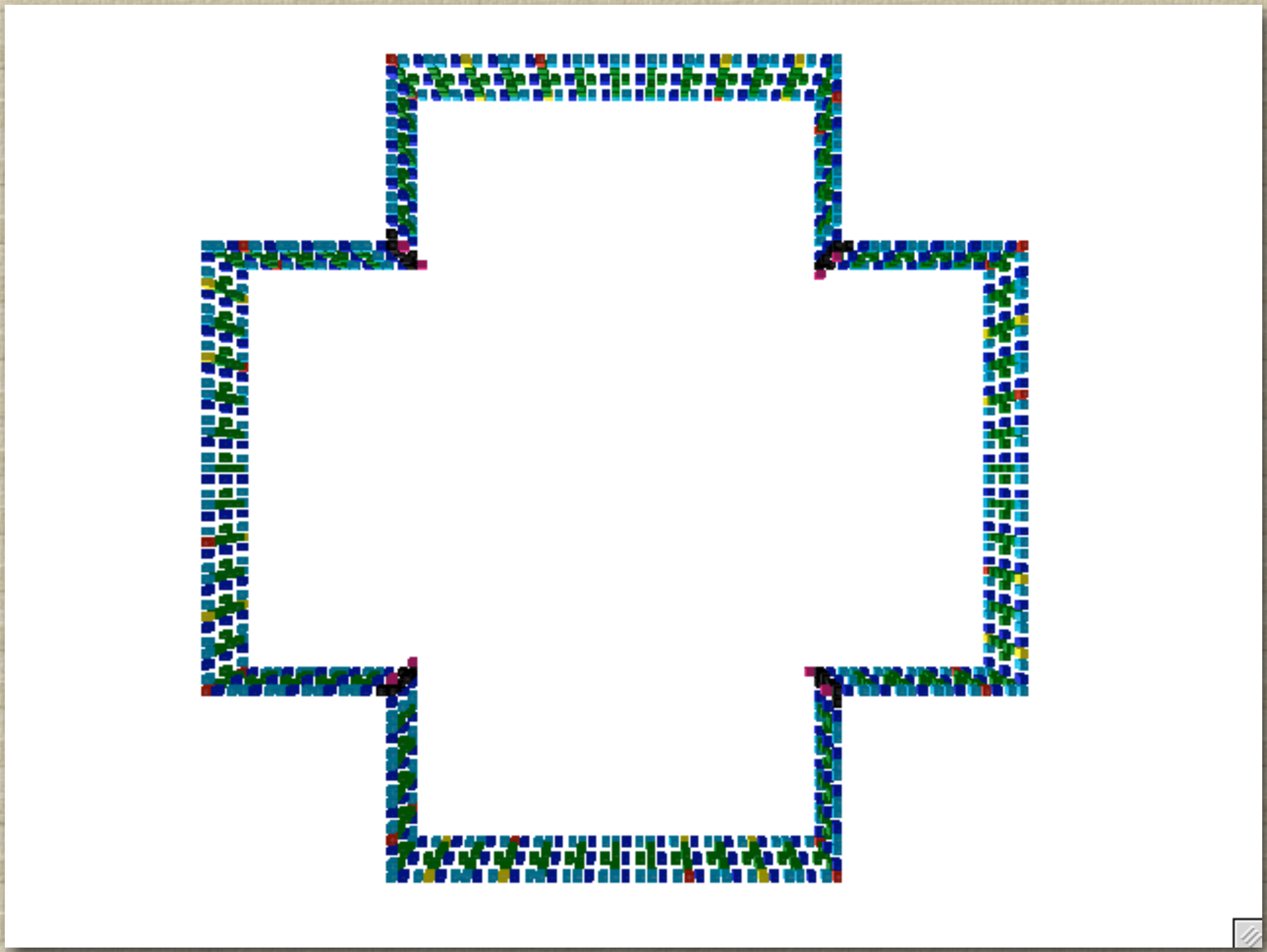
The complete description of its commands:

$$\underbrace{\begin{matrix}(\mathrm{AA}^{5}\,\mathrm{ABAA}^{6}\,\mathrm{AA}^{3}\,\mathrm{ABAAABAA}^{6})\\ (\mathrm{AA}^{5}\,\mathrm{ACAA}^{6}\,\mathrm{AA}^{3}\,\mathrm{ACAAACAA}^{4})\end{matrix}}_{\text{positioning signals}}\;\underbrace{\begin{matrix}\mathrm{AA}^{5}\,\mathrm{ABAA}^{5}\,\mathrm{ABAA}^{5}\,\mathrm{ABAA}^{6}\\ \mathrm{AA}^{5}\,\mathrm{ACAA}^{5}\,\mathrm{ACAA}^{5}\,\mathrm{ACAA}^{6}\end{matrix}}_{\text{shape signals}}$$

Example: a chain



Exapmle: speeding up



To self-reproduce, eat something anyway!

A Number-Conserving Loop

based on the Langton's loop

$$A = (\mathbf{Z}, N_{395}, f, 235)$$

(Imai, Fujita, Iwamoto, and Morita 2001)

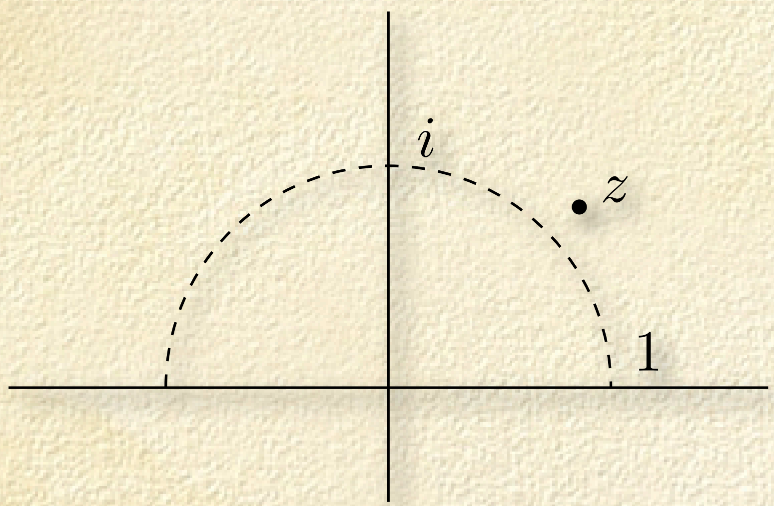
180	195	195	195	195	195	195	195	210	180					
210	275	355	315	275	355	315	285	345	195					
195	315	195	195	195	195	195	195	315	195					
195	355	195					195	315	195					
195	275	195					195	315	195					
195	315	195					195	315	195					
195	355	195					195	315	195					
195	275	195	195	195	195	195	195	315	195	195	195	195	195	203
195	315	355	275	315	355	275	315	315	315	315	315	315	315	184
180	210	195	195	195	195	195	195	195	195	195	195	195	195	198

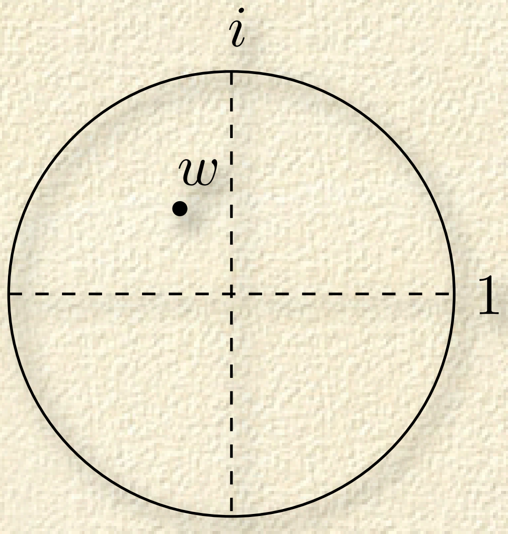
(the value of blank cells is 235)

The Hyperbolic Plane

Poincaré Model for the Hyperbolic Plane

half-plane model versus disk model





$$H = \{z = x + iy \in \mathbf{C}; y > 0\}$$

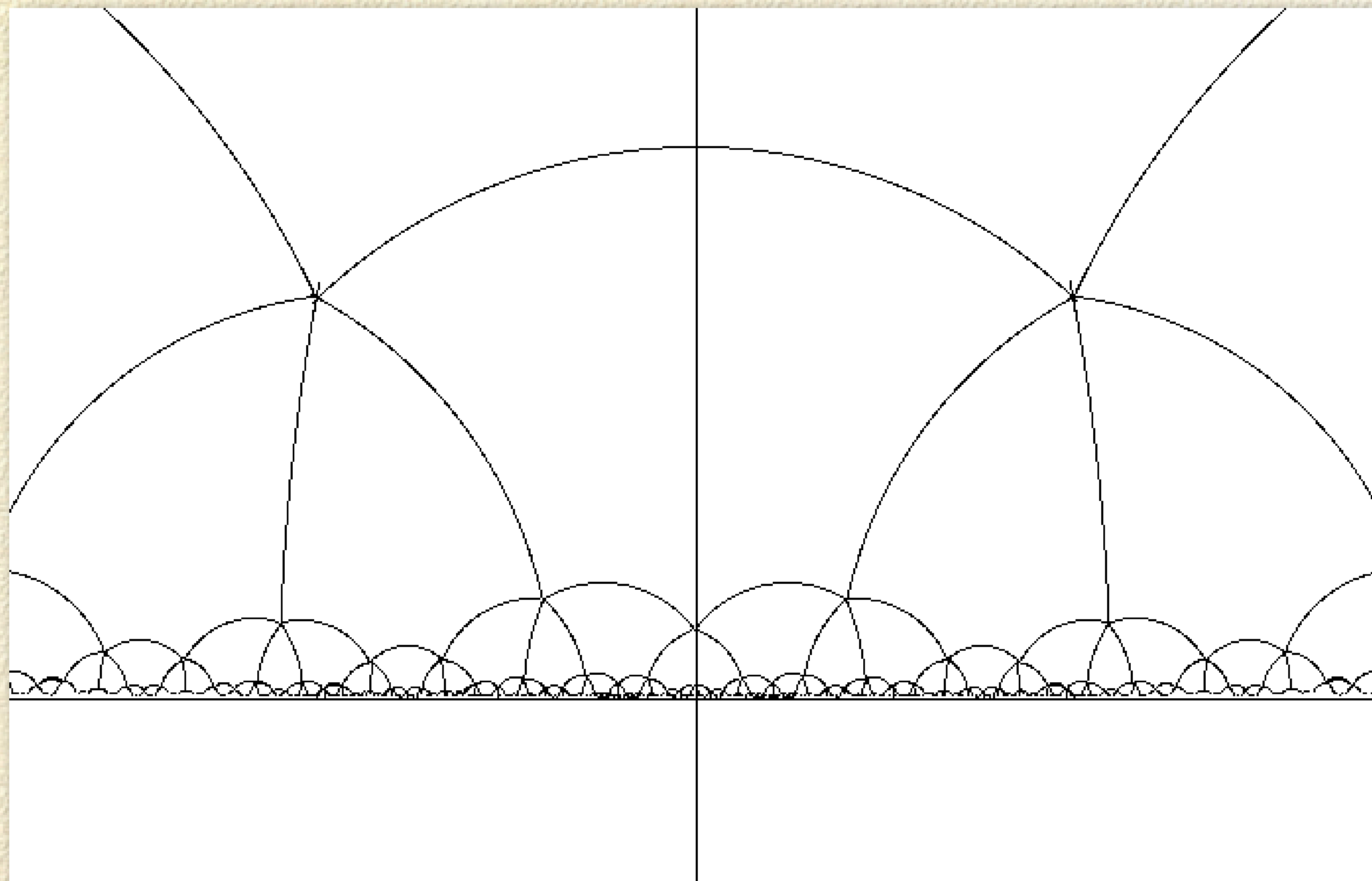
$$D = \{w = u + iv \in \mathbf{C}; |w|^2 = u^2 + v^2 < 1\}$$

$$w = \frac{i - z}{i + z}, \quad z = \frac{i(1 - w)}{1 + w}$$

Hyperbolic Cellular Automata

Margenstern, Morita 2001

Pentagrid
Hexagrid
Heptagrid
...



Pentagrids are bit confusing for me!

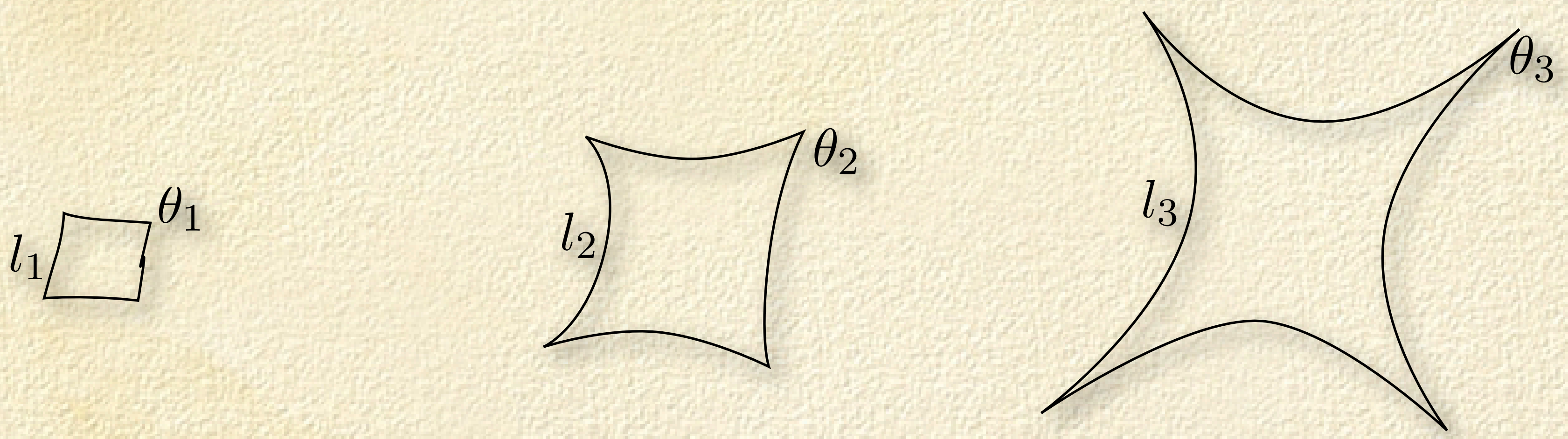
How about von Neuman neighbor hyperbolic CA?

unfortunately no square tiling...

Quadrangles on the Hyperbolic Plane

There exists infinite different kinds of quadrangles such that all edges are the same length.

(not rectangle)



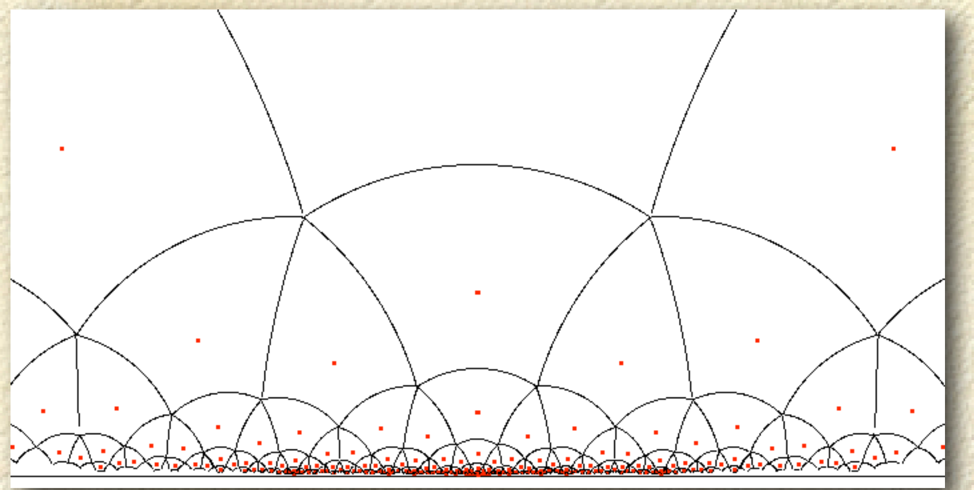
$l_1 < l_2 < l_3, \frac{\pi}{2} > \theta_1 > \theta_2 > \theta_3$

$\theta = \frac{2\pi}{n}, n = 5, 6, \cdots$

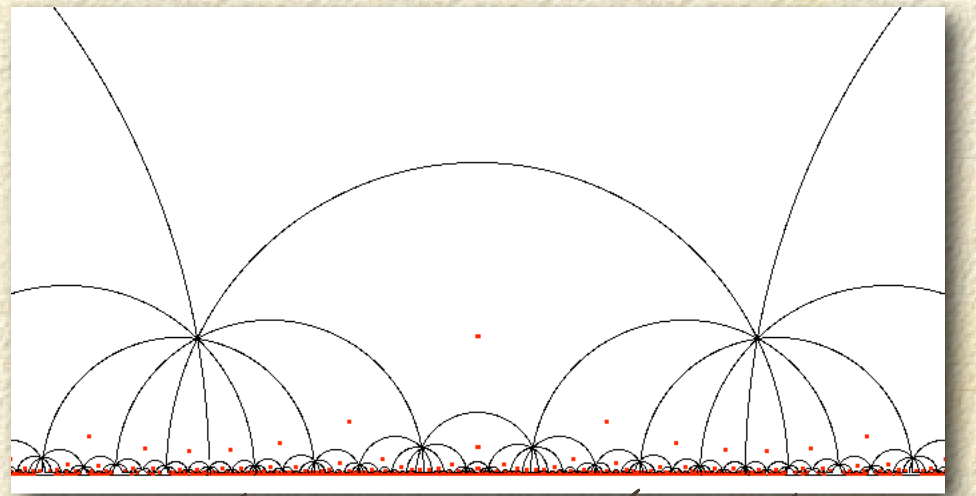
There are infinite kinds of tiling with such quadrangles.

Tilings by Quadrangles

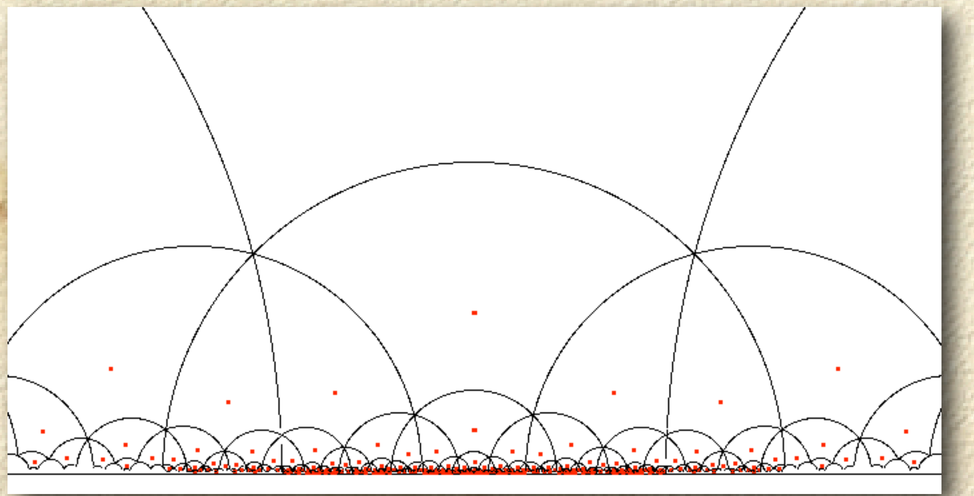
Quadrangles with angle = 2π/n (n >4) can tile the hyperbolic plane.



degree 5 (angle = 2π/5)



degree 10 (angle = 2π/10)

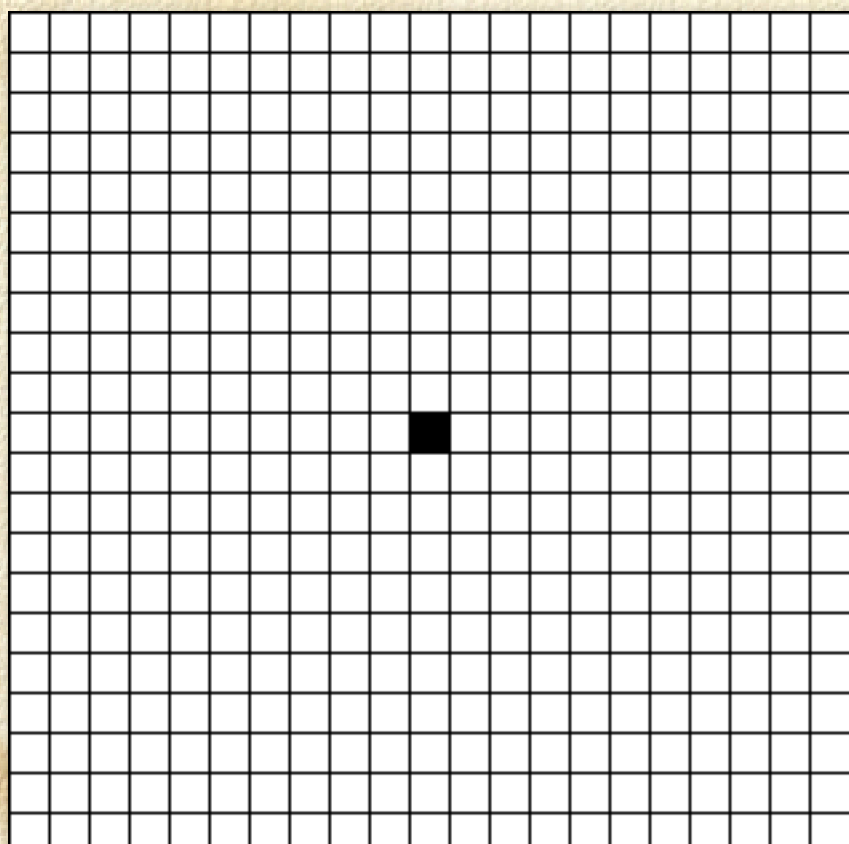


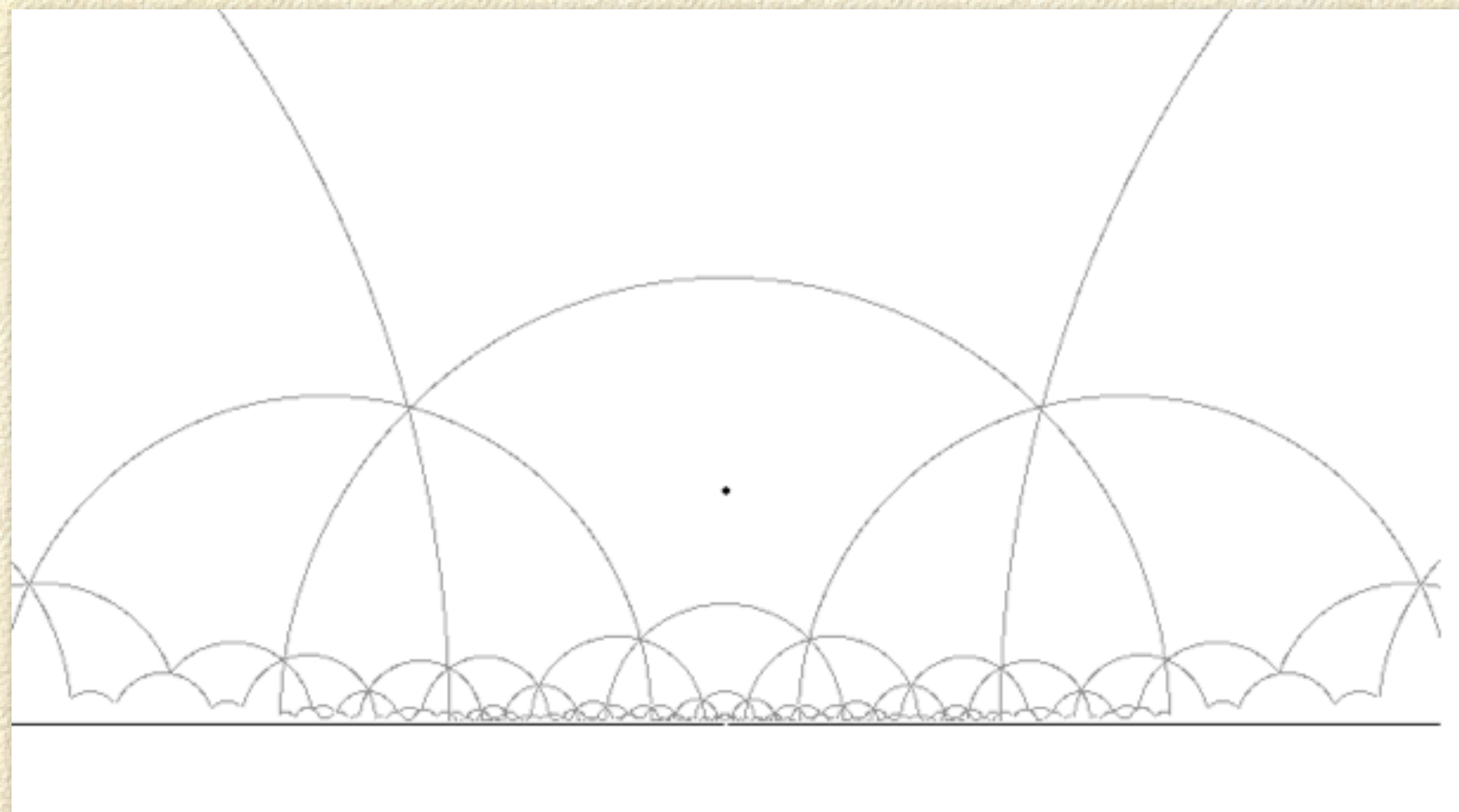
degree 6 (angle = 2π/6)

An example: Fredkin (Parity) CA

Euclidian

Hyperbolic





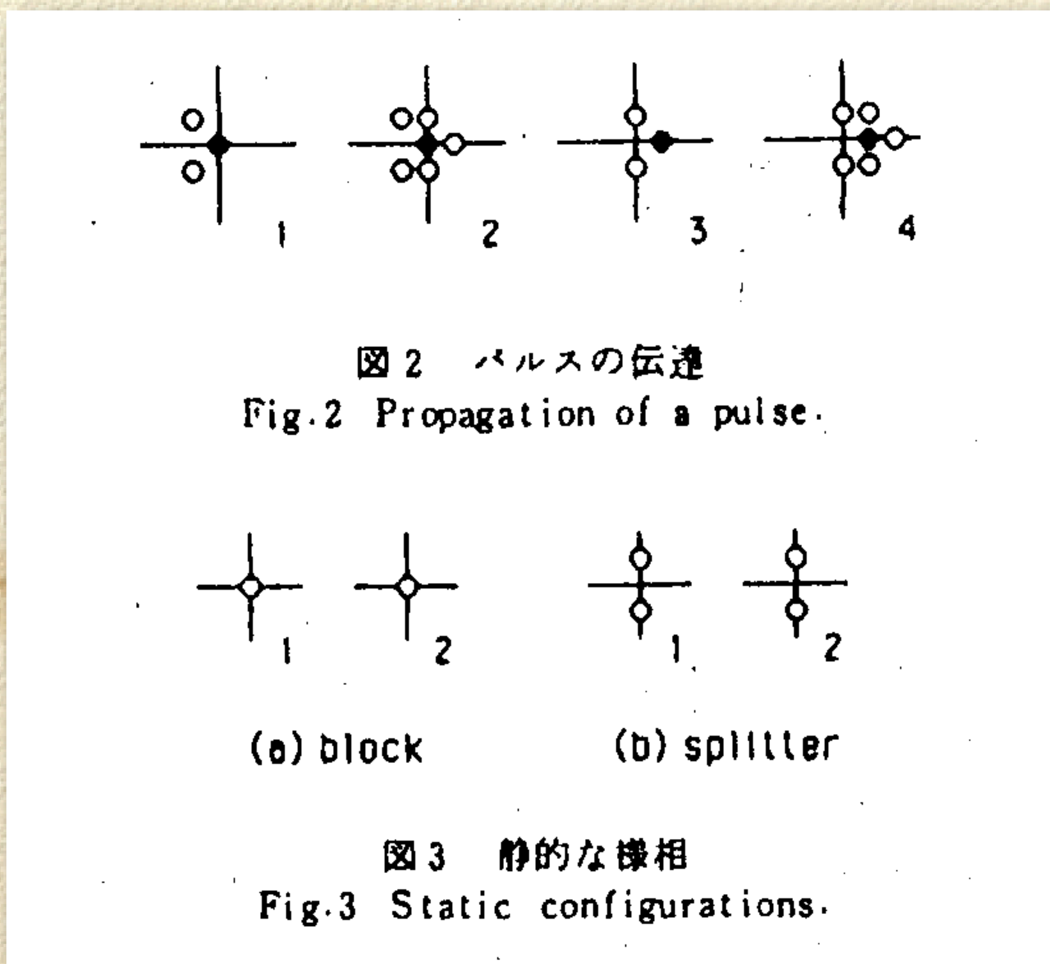
Both rules are the same. (von Neumann neighborhood)

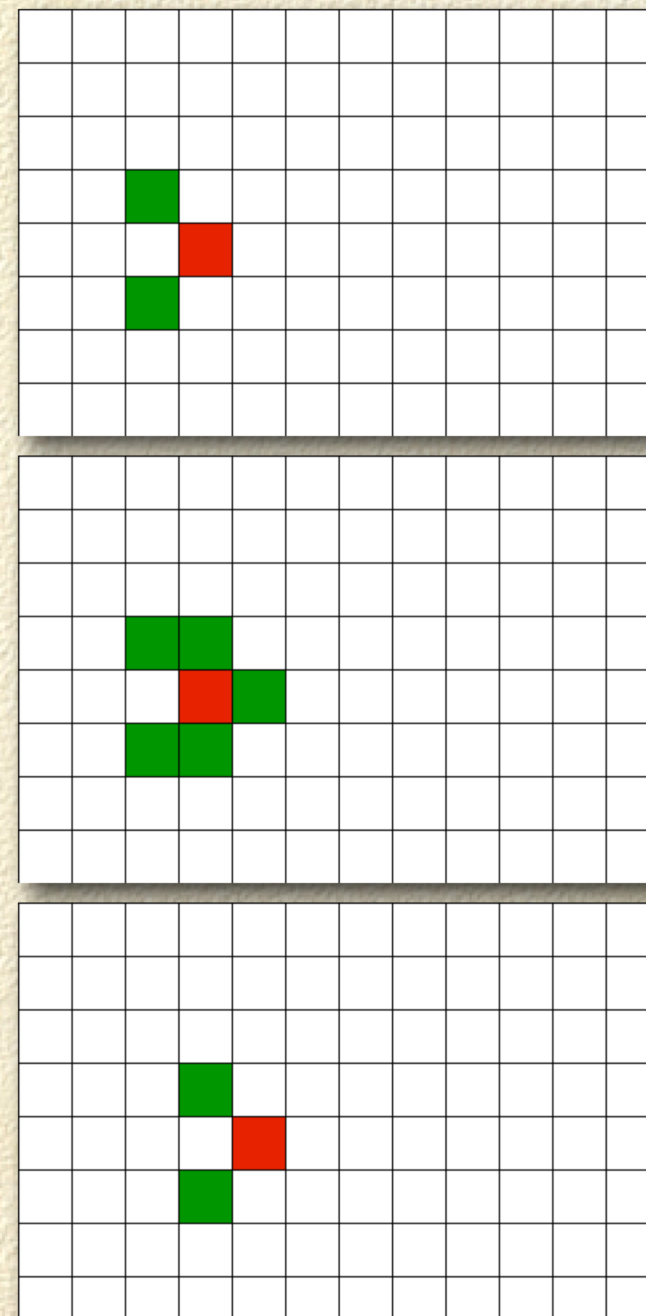
Let's build a glider on Hyperbolic CA...

Serizawa's 3-state Universal Euclidean CA

3-state, von Neumann neighborhood (Serizawa 1986)

A signal is represented by a “glider”.





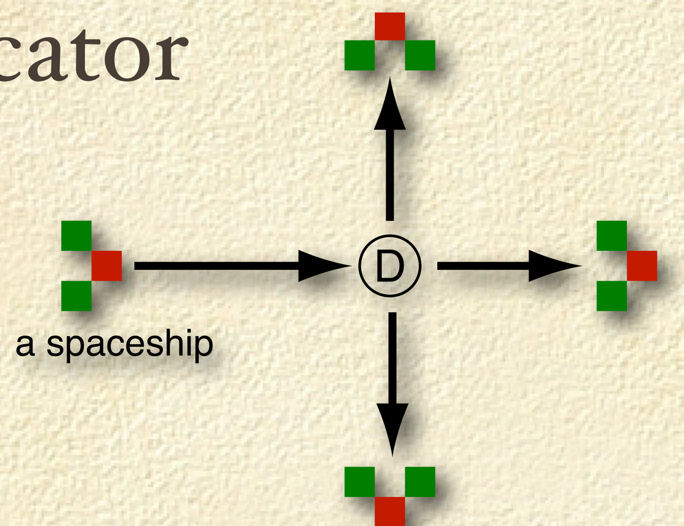
$t = 0$

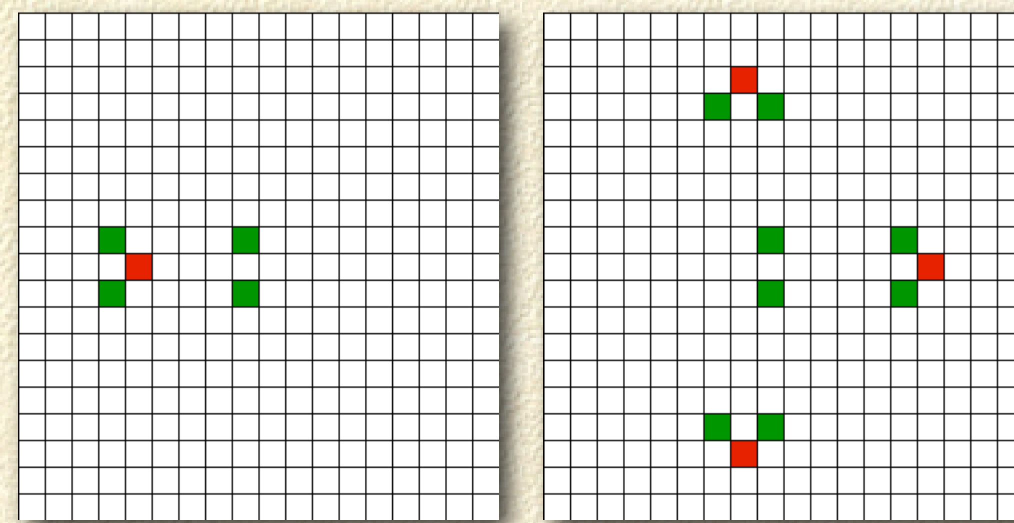
$t = 1$

$t = 2$

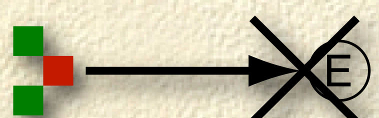
Basic Actions in Serizawa's Universal Euclidean CA

1. duplicator

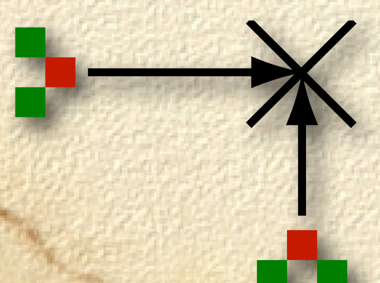




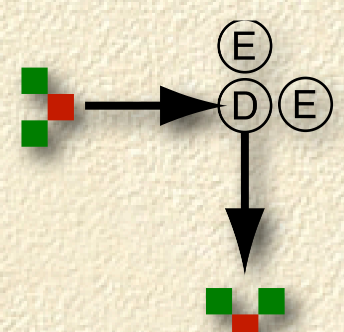
2. eater



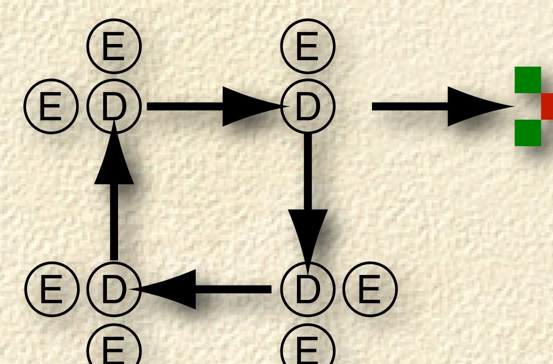
3. collision of two gliders



change direction

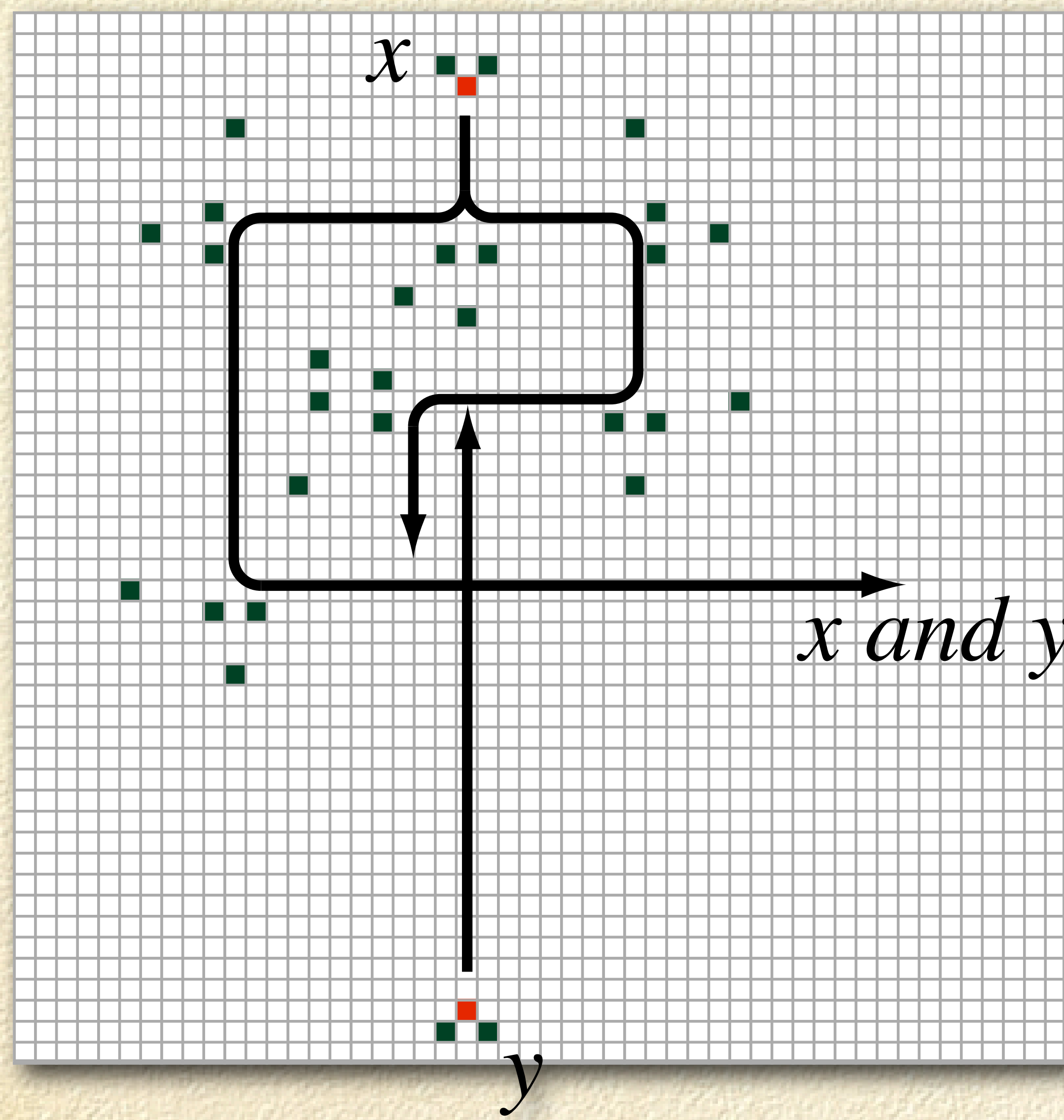


glider generator



Logical universality of Serizawa's Universal Euclidean CA

For example: AND gate can be constructed by duplicators, eaters, and collisions.

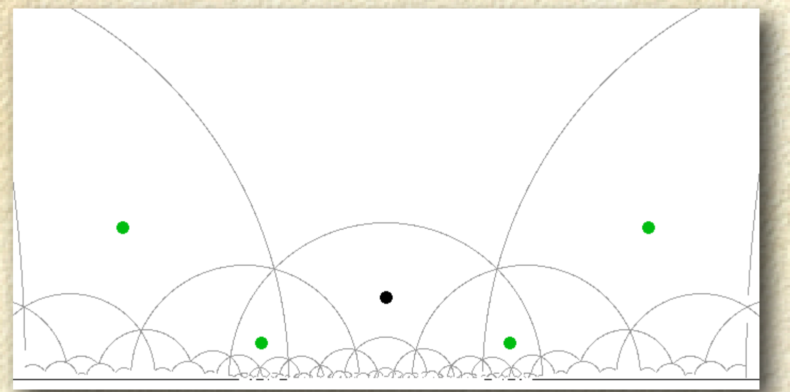


Let's embed such a model in a hyperbolic CA!

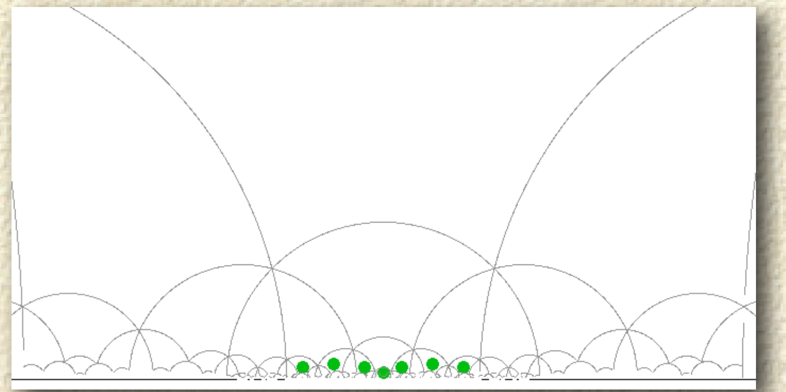
A glider on 4-state Hyperbolic CA

(Imai, Ogawa 2000)

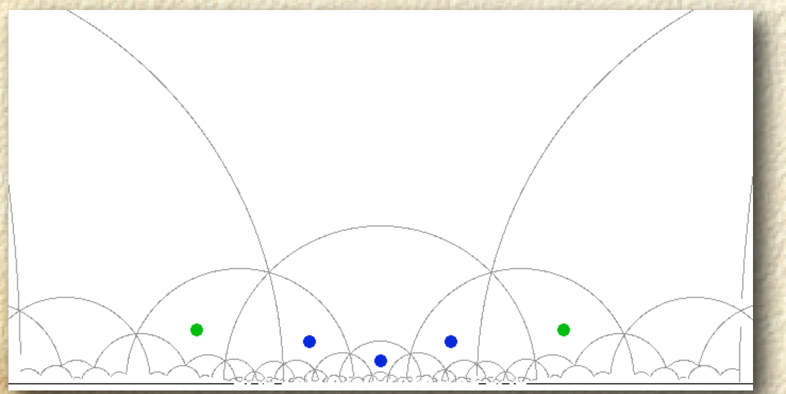
$t = 0$



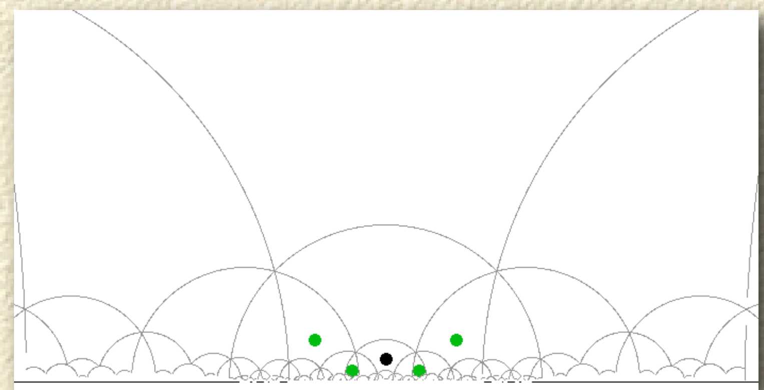
$t = 2$



$t = 1$



$t = 3$

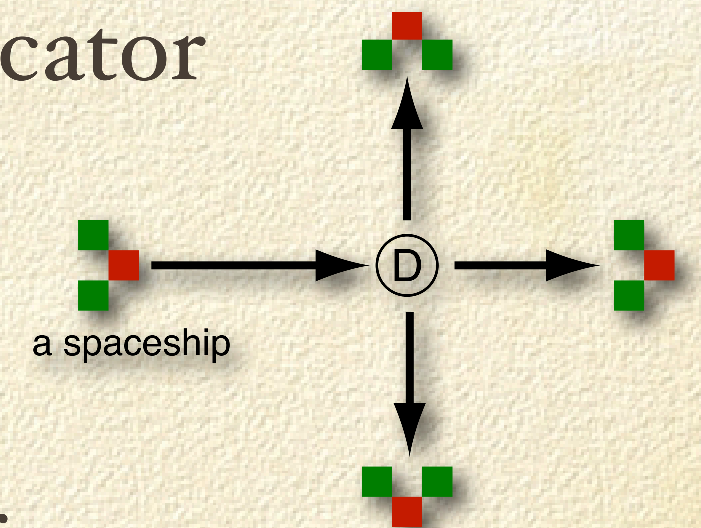


H1: 5-state Quadrangular Hyperbolic CA (degree 6)

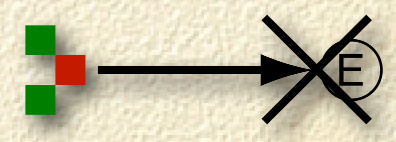
(Imai, Ogawa 2000)

- degree 6 quadrangle
  - 5-state, 5-neighbor
  - H1 can perform three basic actions.

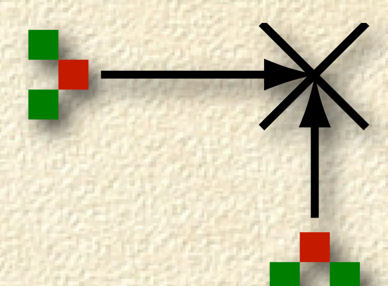
1. duplicator



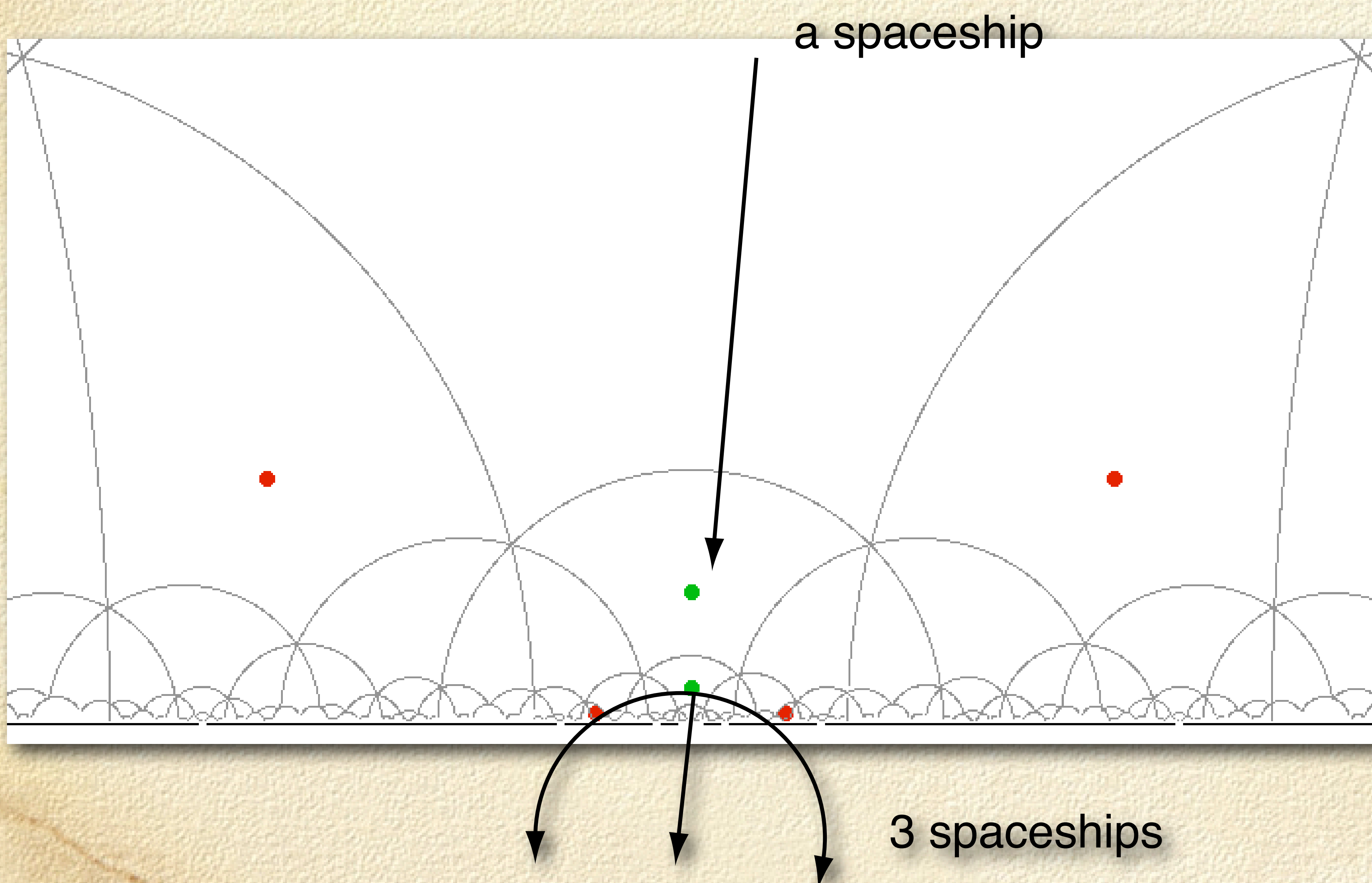
2. eater



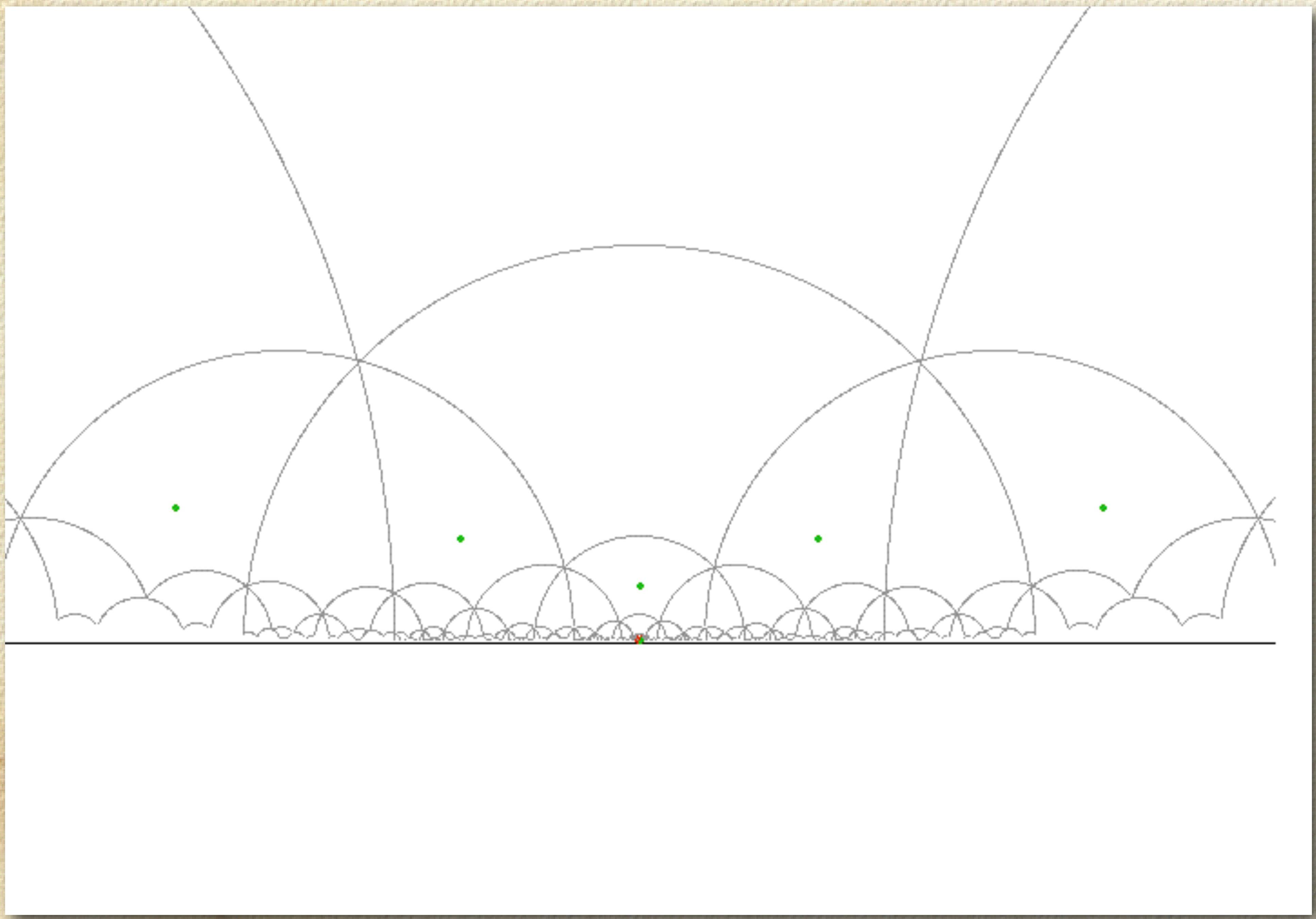
3. collision of two glider



A Duplicator on H1

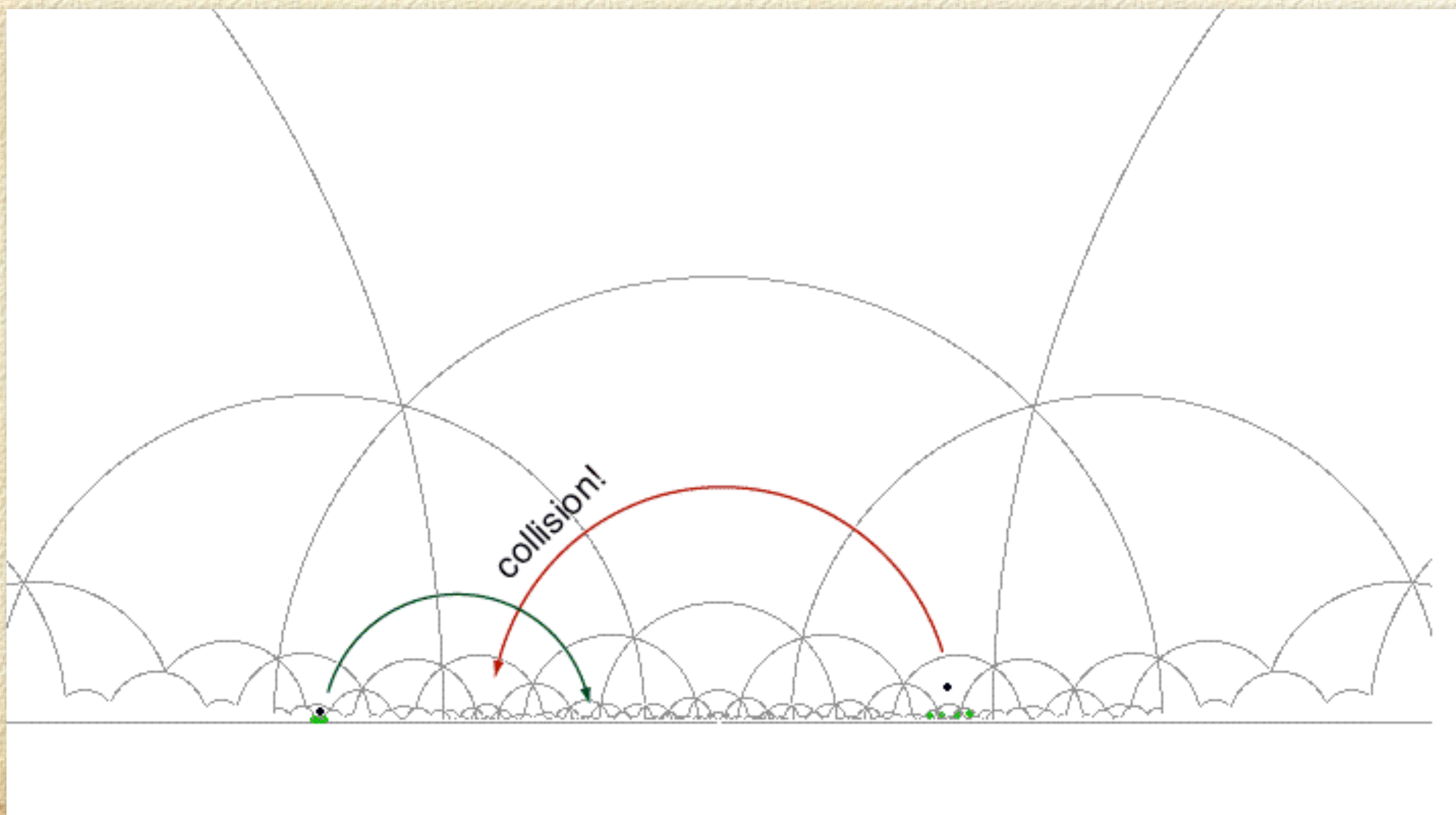


A Duplicator on H1



Collision of two spacehips

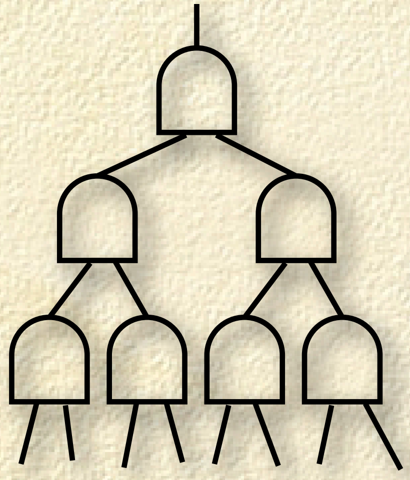
Collision of two spacehips



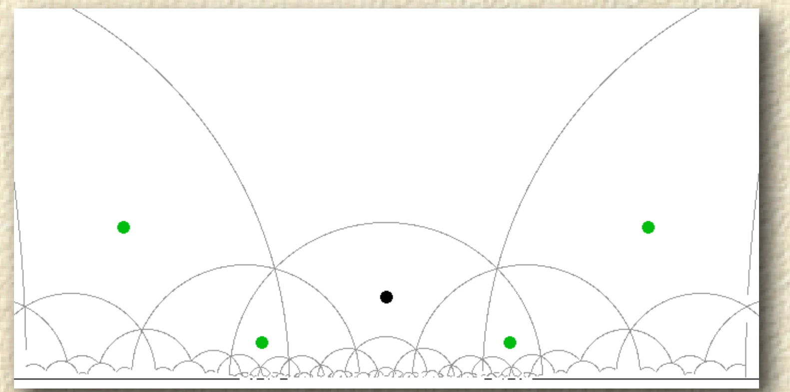
Can H1 Perform Universal Computation?

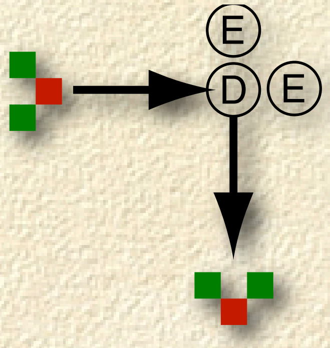
Maybe no.

It is impossible to realize feedback signals.



changing direction





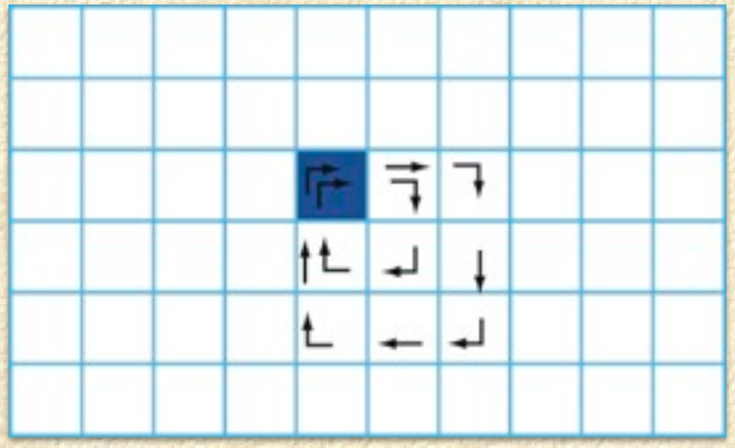
'Sharp Turns' are Inevitable in the Hyperbolic Plane

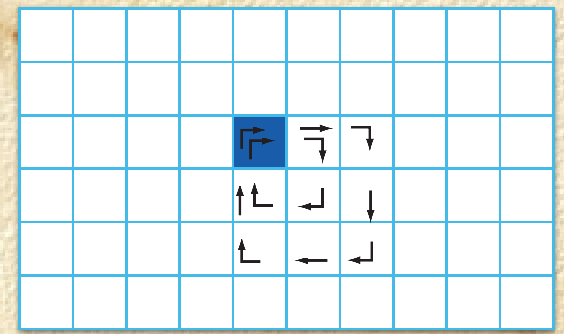
Euclidean:

RRRR

RSRSRSRS

(both turn to the same cell)



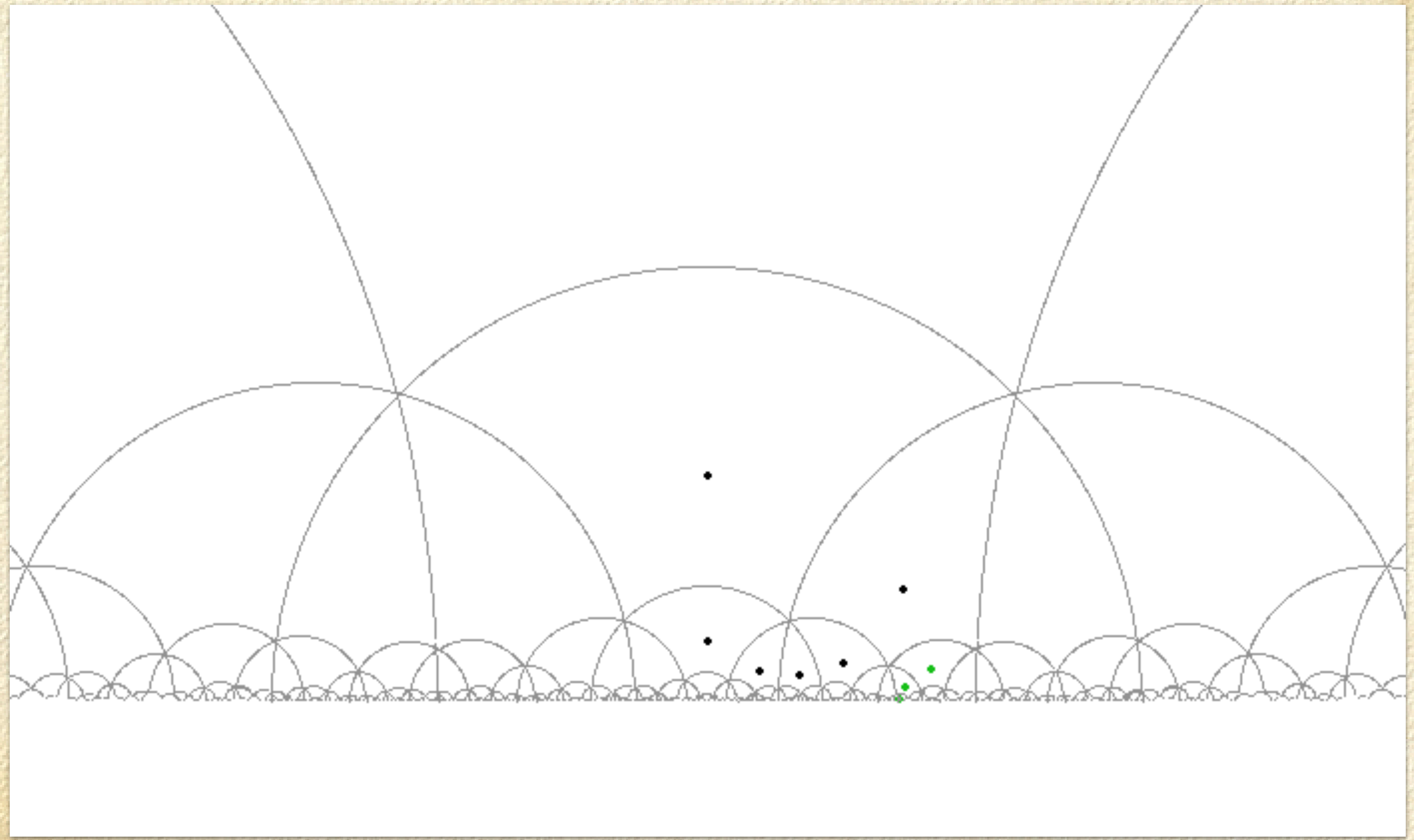


Hyperbolic:

RRRRRR

RSRSRSRSRSRS

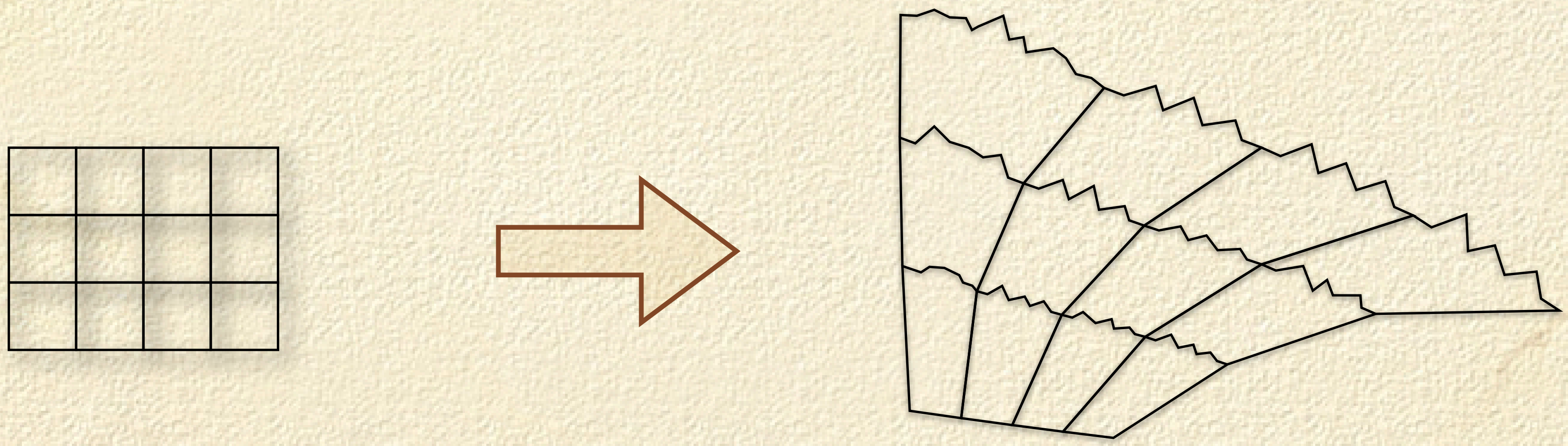
(impossible to turn back)



How to Embed a Logical Circuit?

Laying out a “mesh” on the hyperbolic space.

Euclidean Hyperbolic



Edges (length 1) ⇨ Paths (should be zigzag paths)

(actually zigzig? paths...)

Length of Each Path is Difficult to Control

Exponentially long sharply turning zigzag path....

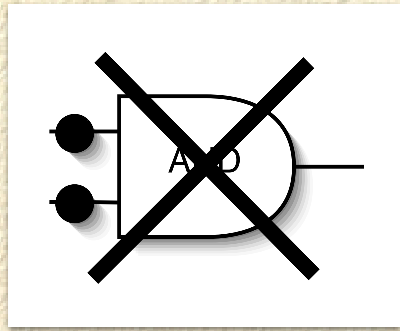
It is difficult to embed synchronous circuits.

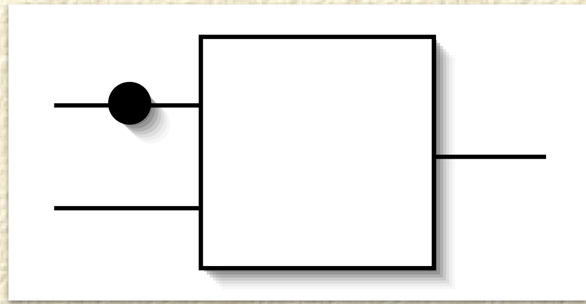
How about asynchronous circuits?

Delay insensitive (DI) circuits (Keller 1974)

Serial Modules

Def. A module is called *serial* if its specification prohibits concurrent inputs or outputs, i.e., each output event has exactly one unique input event that causes it.





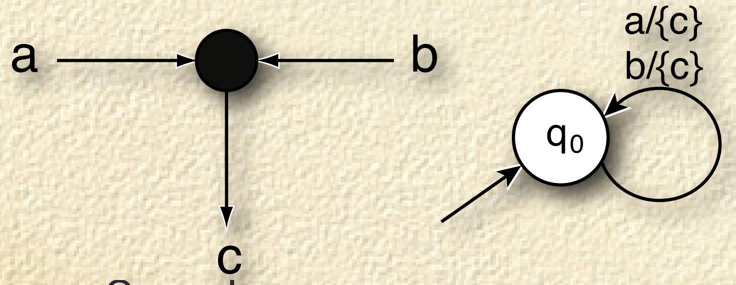
Serial Universality

(Keller 1974)

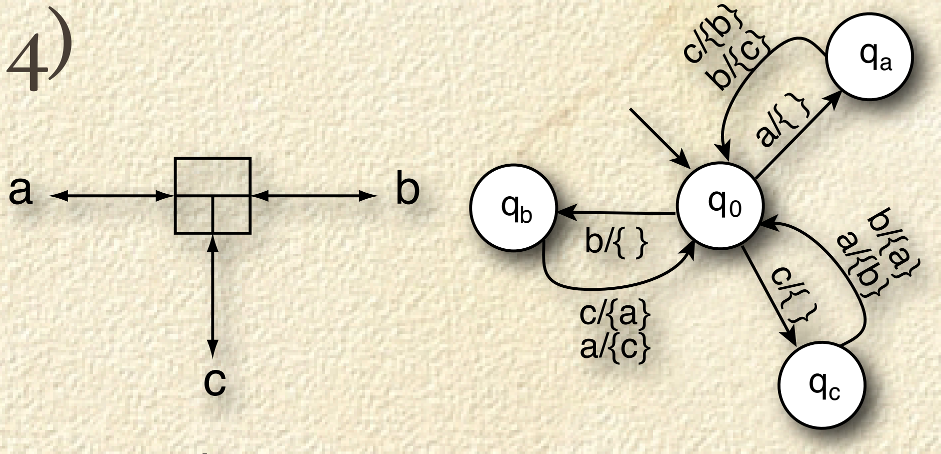
Def. A set of primitive modules is *serial universal*, if any arbitrary serial module is realizable by a network of modules in the set.

A serial universal element set (Lee et, al. 2004)

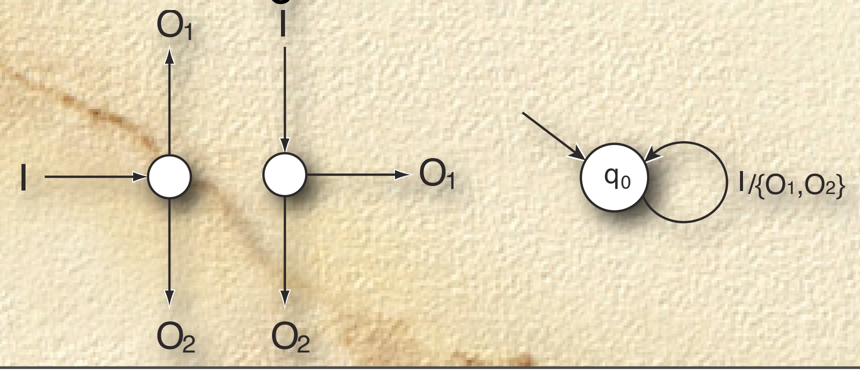
MERGE



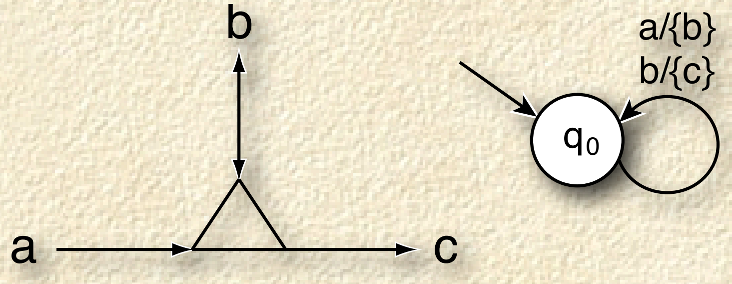
S-JOIN



FORK



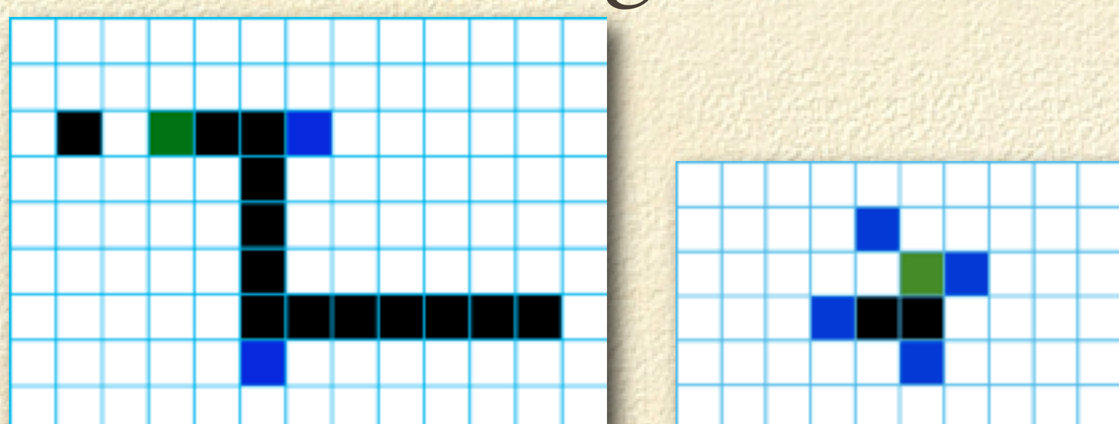
IOM



E1: A 5-state Serial Universal Euclidean CA

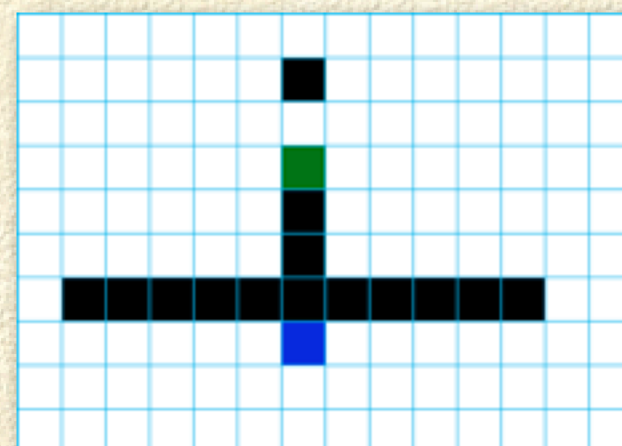
It simulates FORK, MERGE, S-JOIN, IOM, and crossing modules.

A wire and a signal

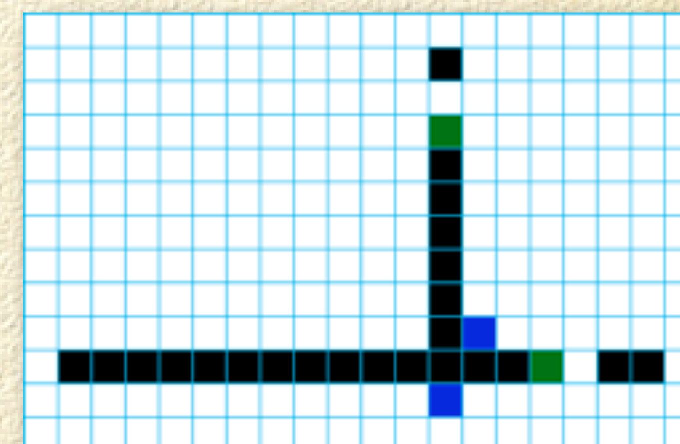


A crossing element

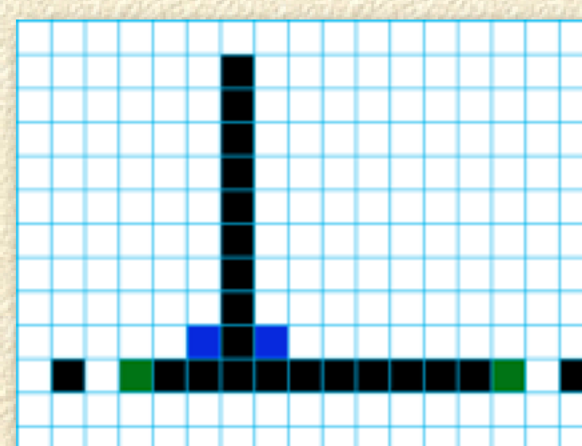
FORK



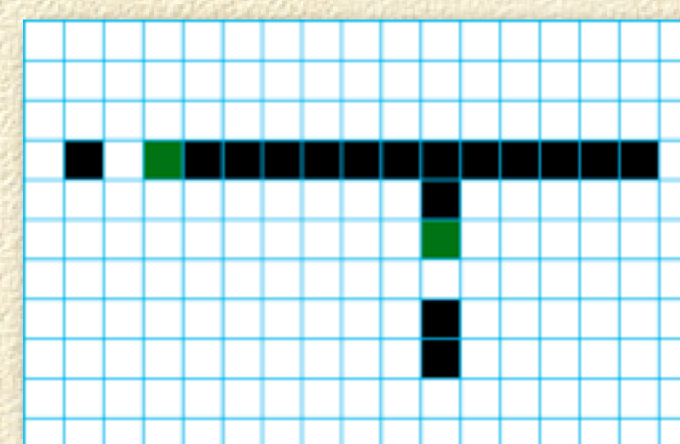
IOM



MERGE



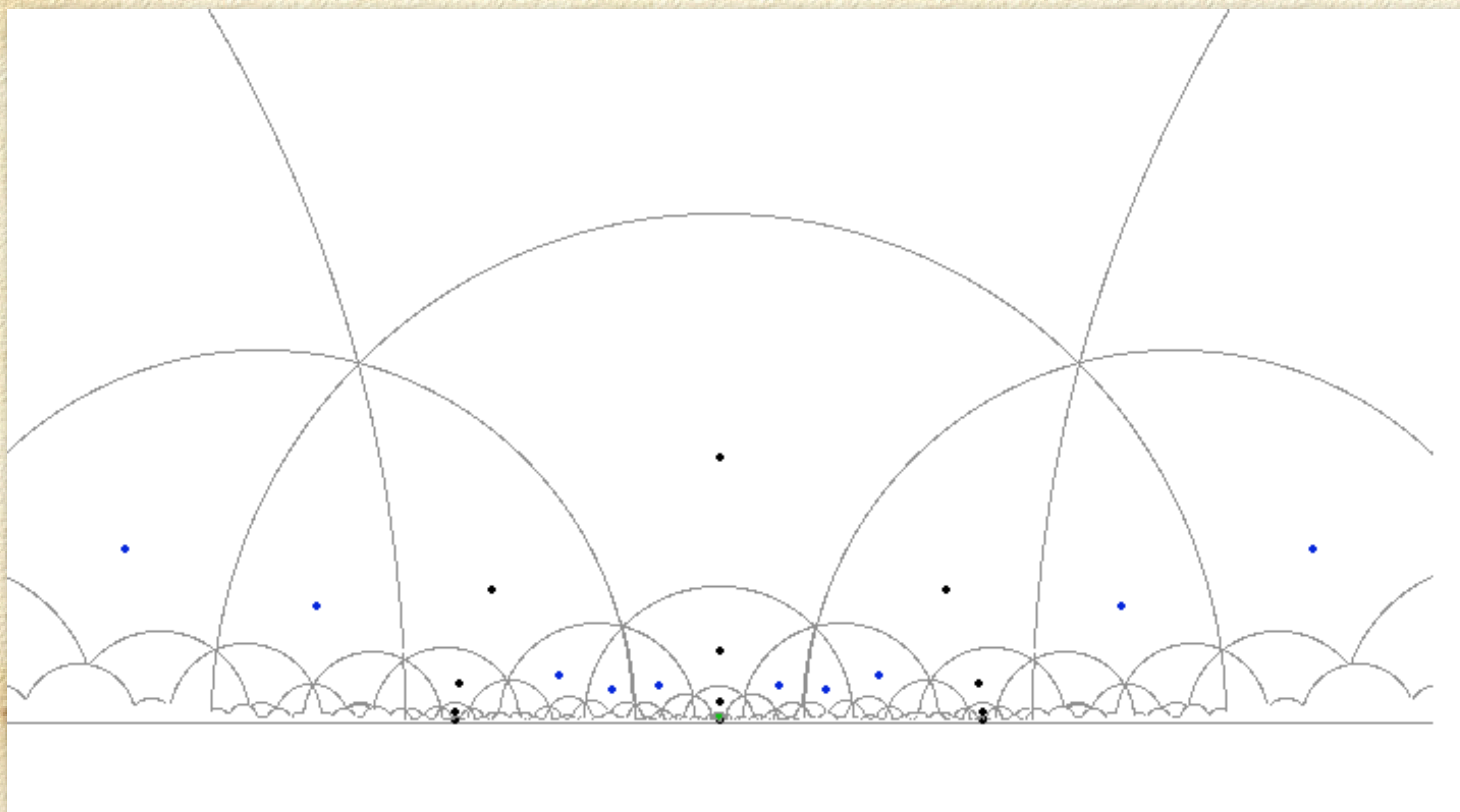
S-JOIN

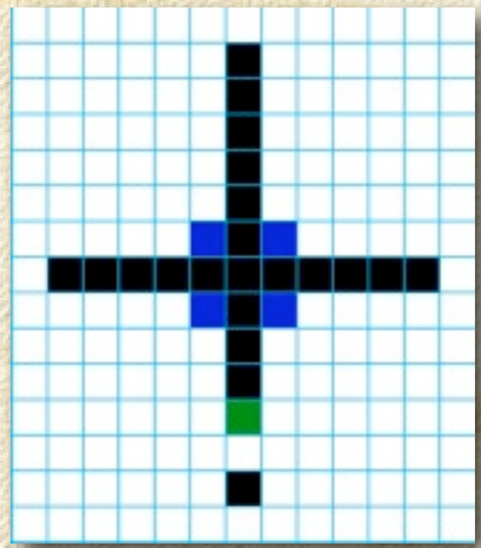


H2: 5-state Universal Hyperbolic CA

- Based on the CA E1 and added several hyperbolic specific rules
- Hyperbolic von Neumann-neighborhood
- Degree-independent of quadrangles
  - Euclidean case (degree 4) is included

A Crossing Module



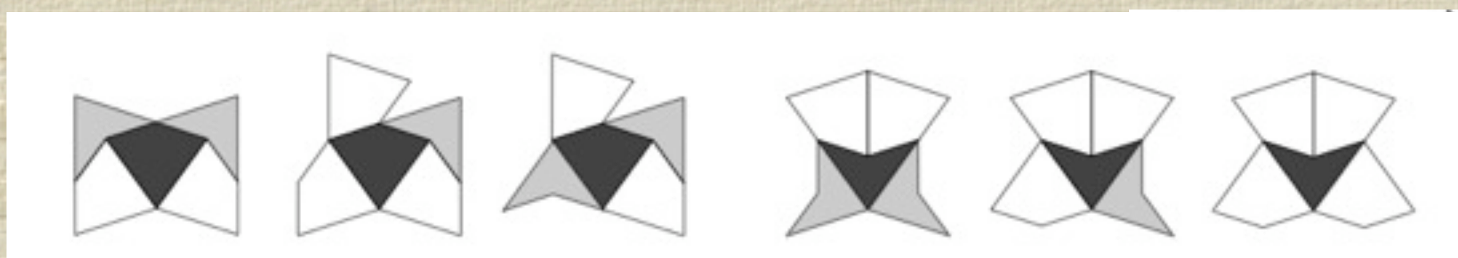


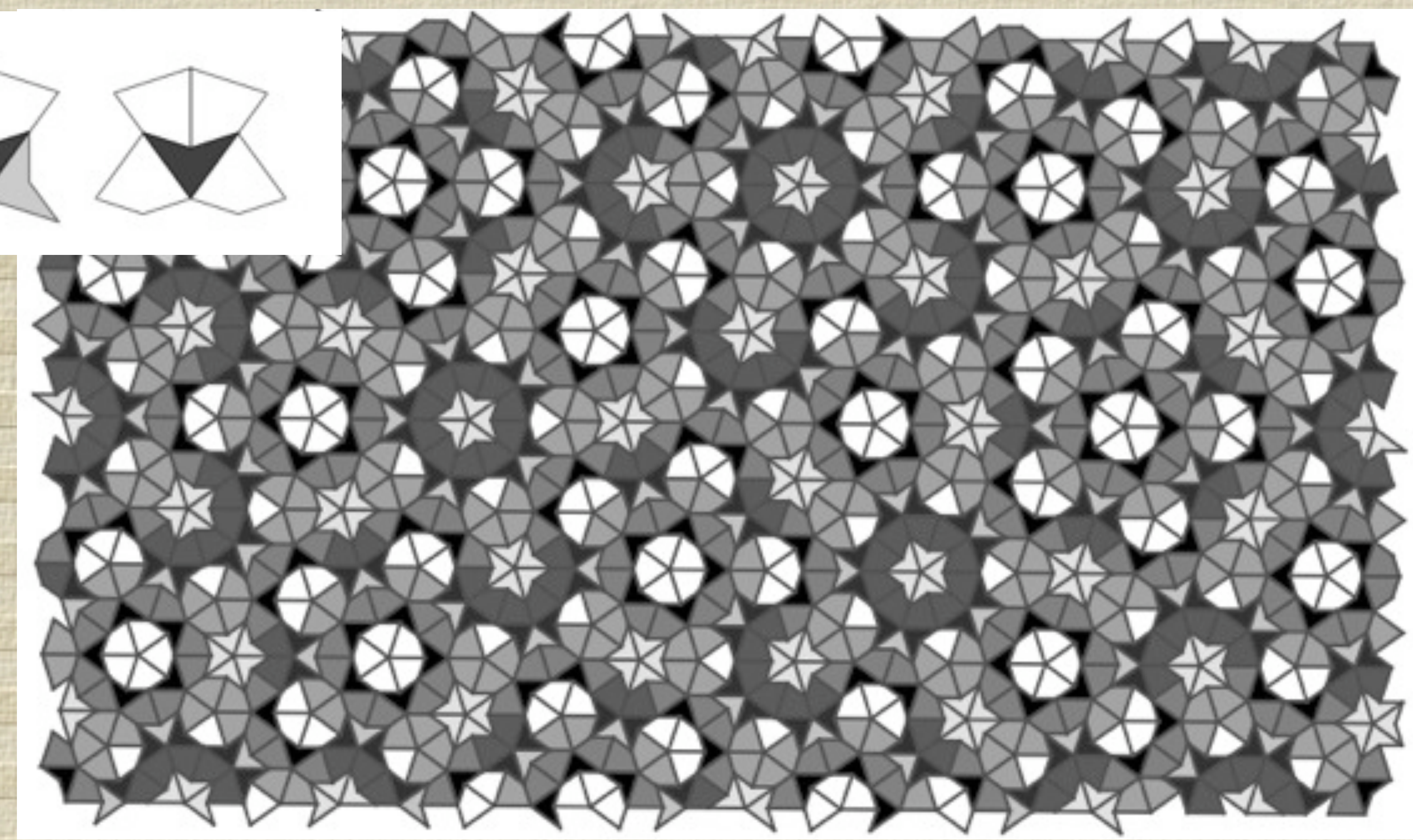
What's next?

- Cellular Automata on Penrose Tilings
  - Hill, Stepney, Wan 2005, Owens, Stepney 2008
  - Investigation of their behaviors in the case of random initial configurations.

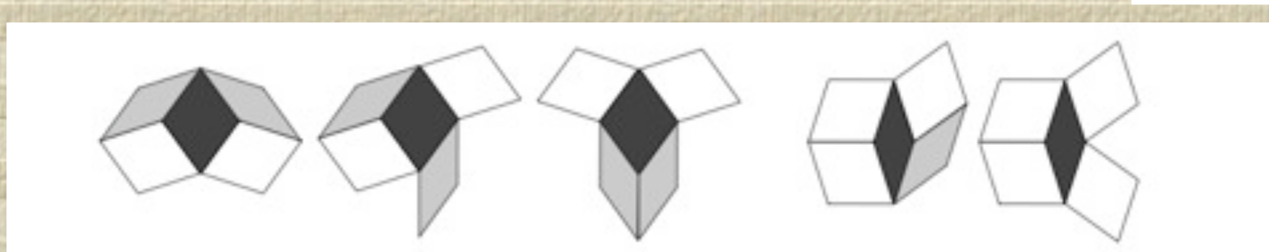
Generalized von Neumann neighborhoods

kite and dart





rhomb

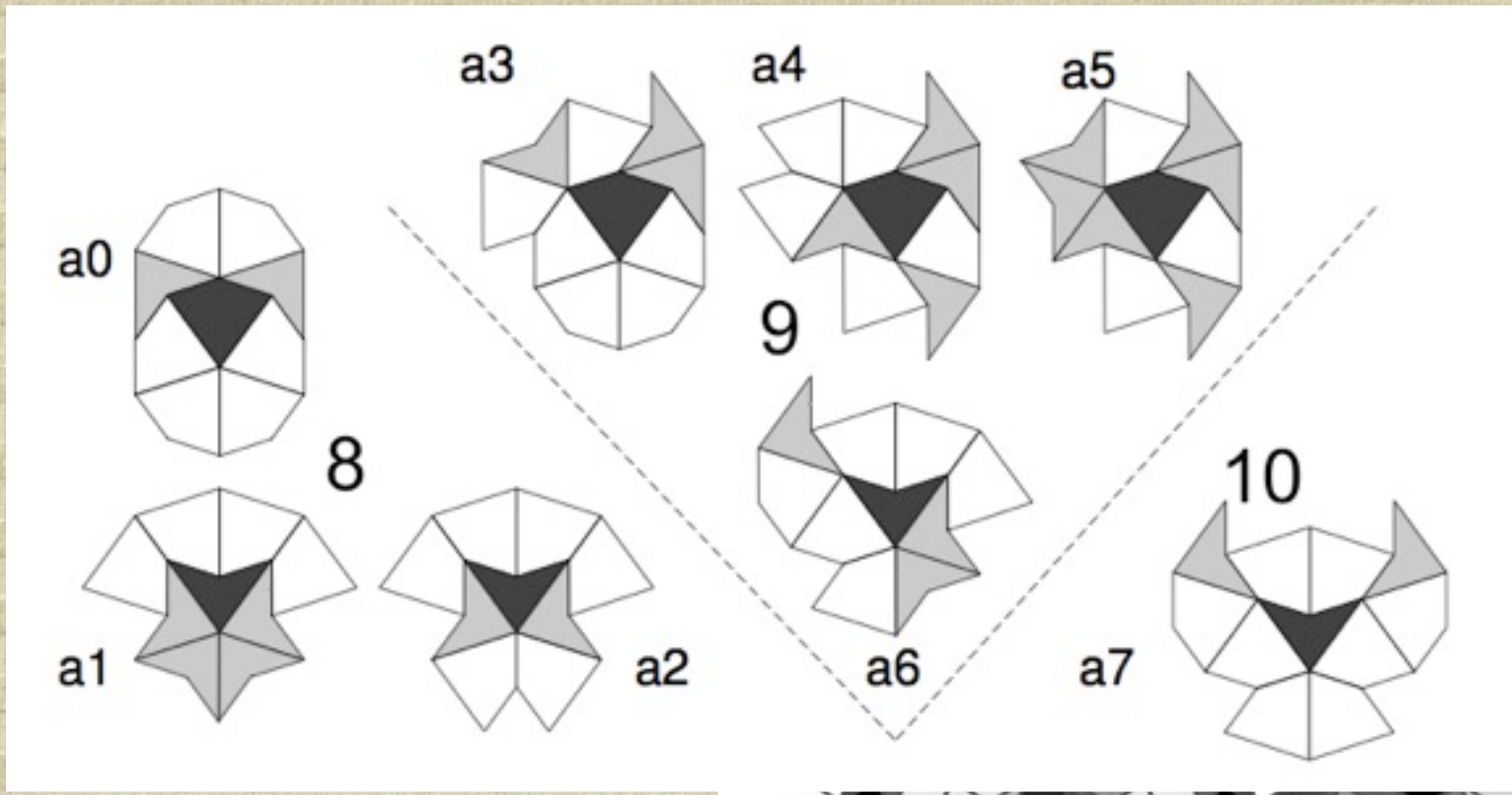


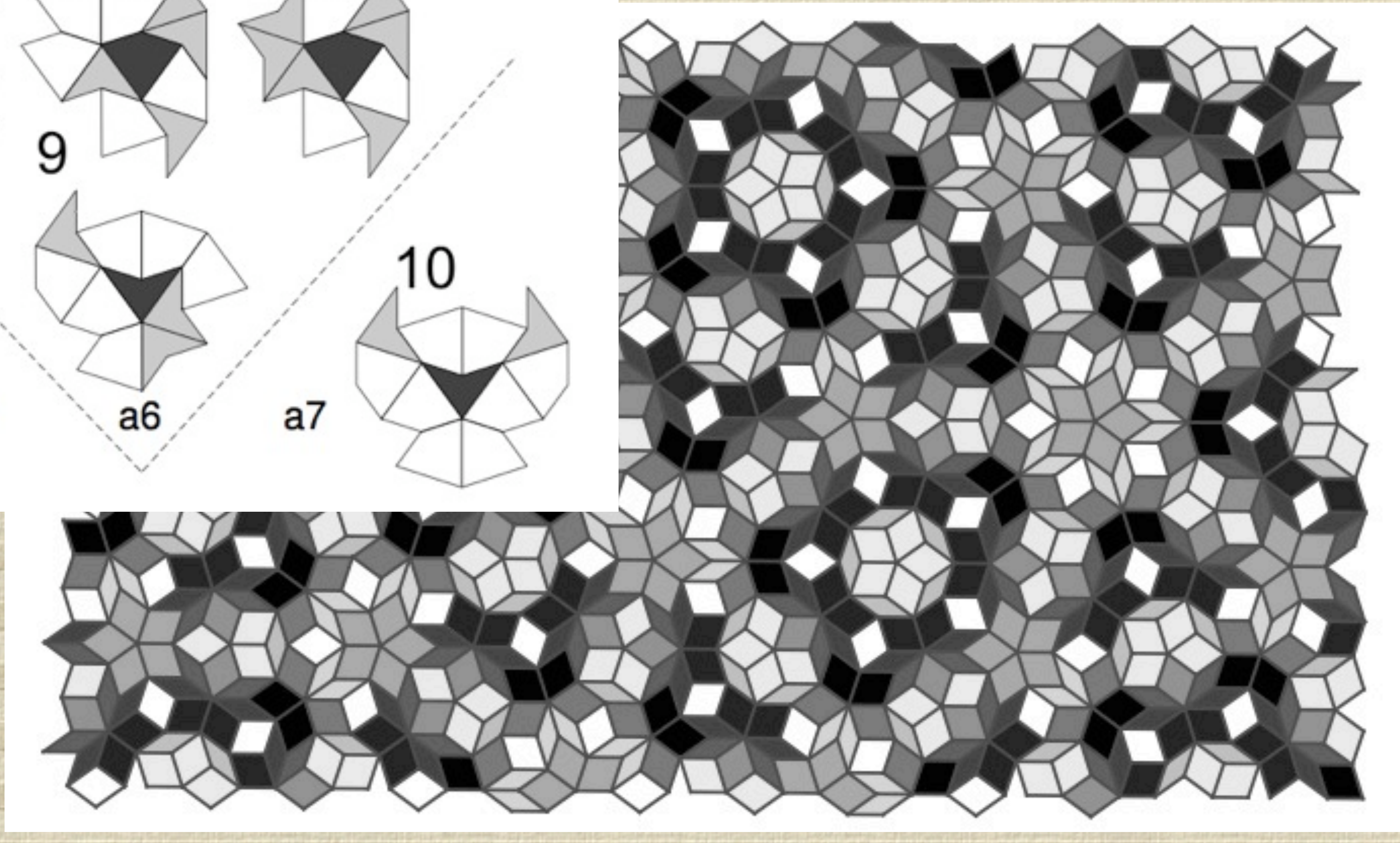
Owens, Stepney 2008

Generalized Moore neighborhoods

rhomb

Owens, Stepney 2008





Thank you