Universality of rule 110
towards a formal proof

Nicolas Ollinger
LIF, Univ. de Provence

Gaétan Richard
ENS Lyon student

DMCS, Turku – July 10, 2004

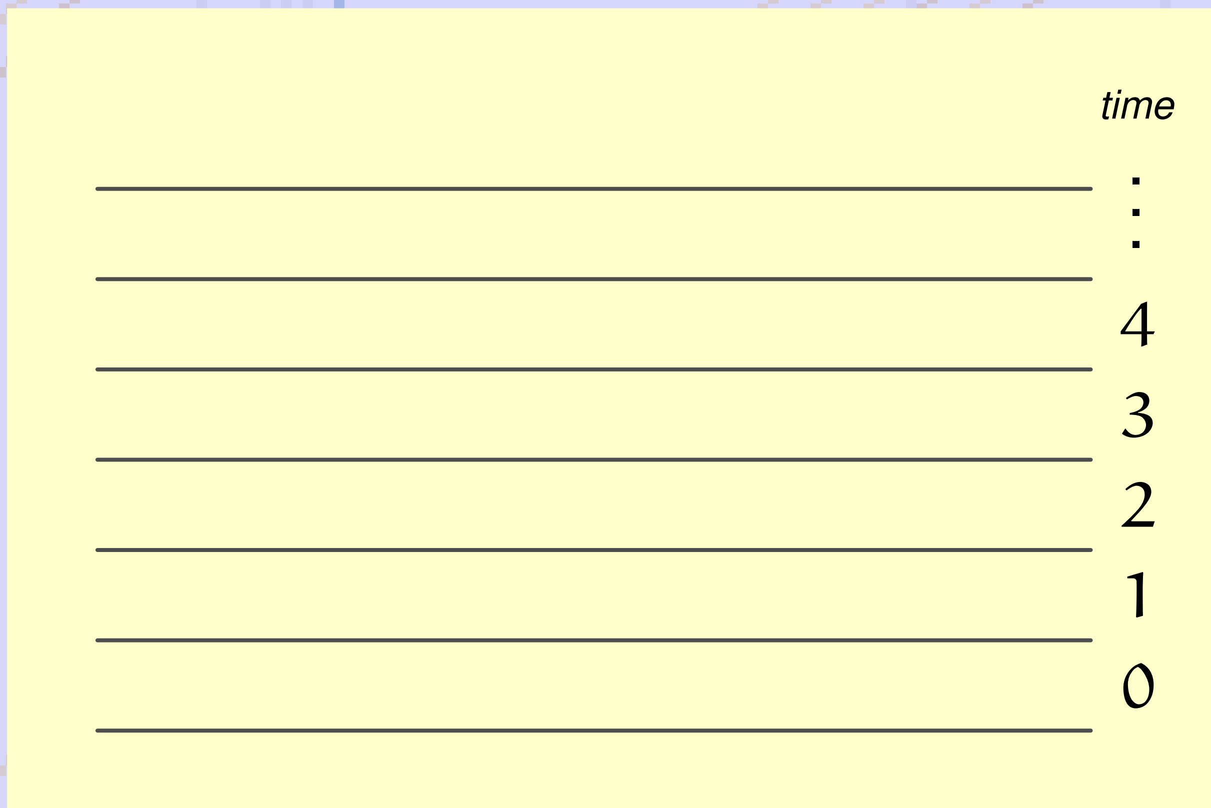
Table of Content

Cellular Automata

- A *1D-CA* $\mathcal{A}$ is a tuple $(\mathbb{Z}, S, \mathcal{N}, \delta)$.

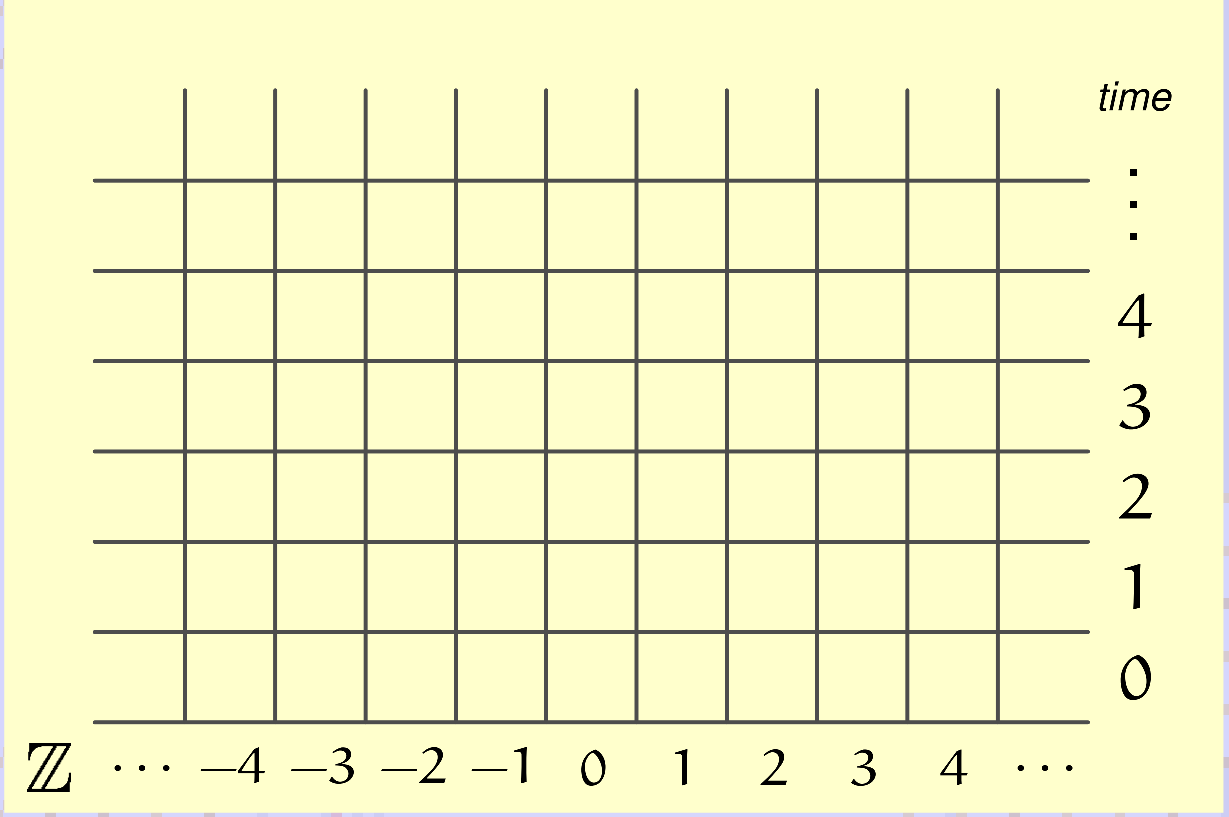
Cellular Automata

- A *1D-CA* $\mathcal{A}$ is a tuple $(\mathbb{Z}, S, \mathcal{N}, \delta)$.



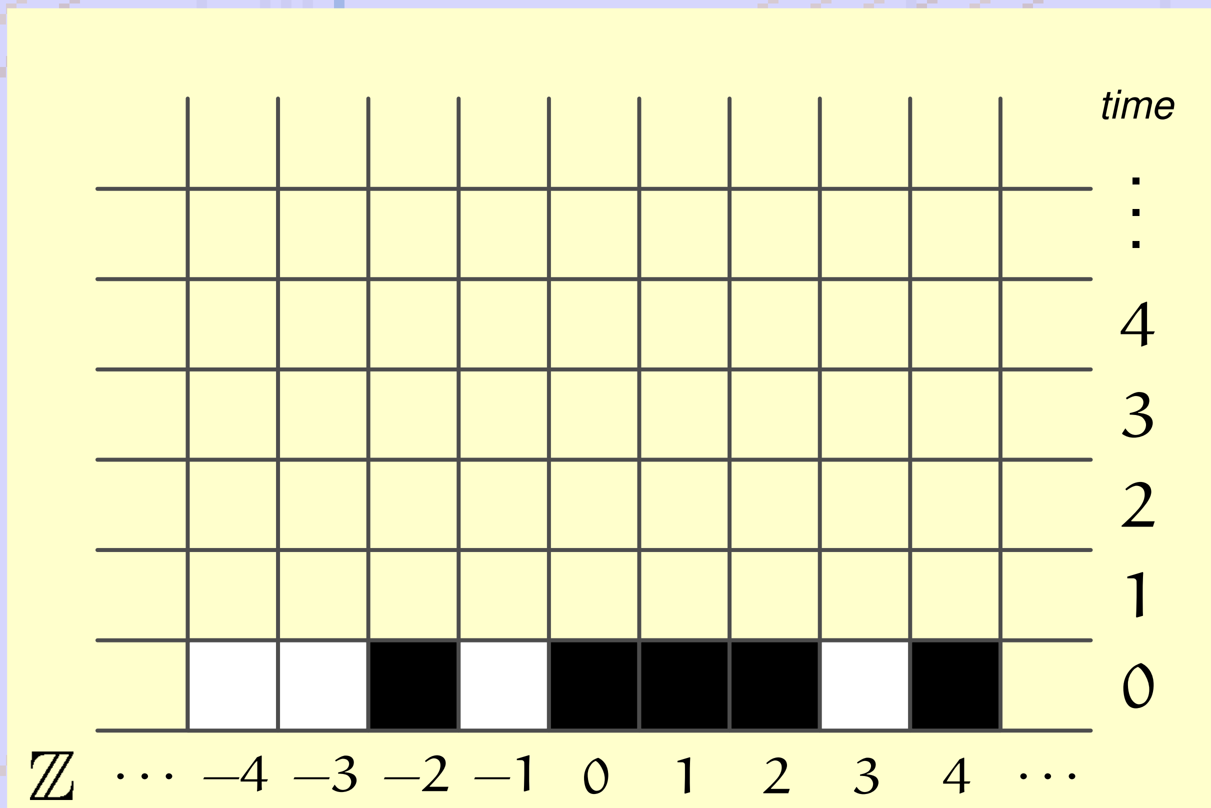
Cellular Automata

- A *1D-CA* $\mathcal{A}$ is a tuple $(\mathbb{Z}, S, \mathcal{N}, \delta)$.



Cellular Automata

- A *1D-CA* $\mathcal{A}$ is a tuple $(\mathbb{Z}, S, \mathcal{N}, \delta)$.

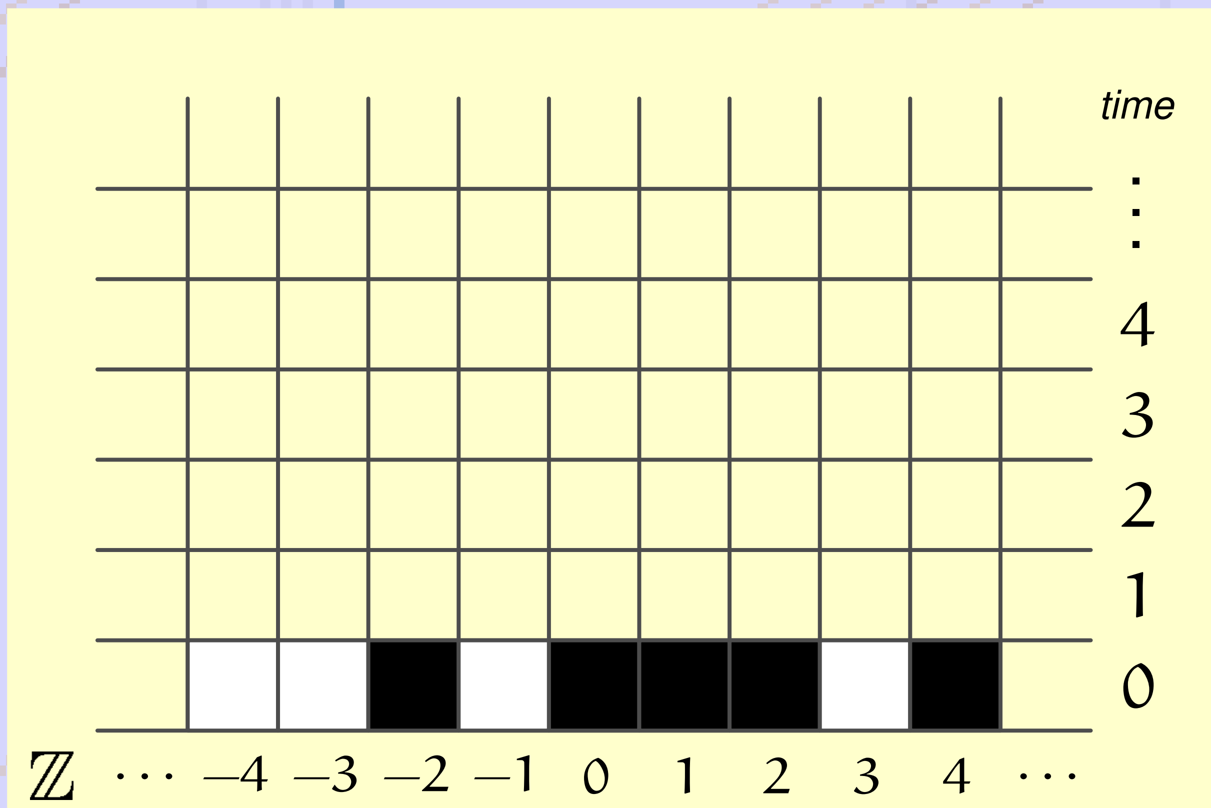


$S = \{\square, \blacksquare\}$

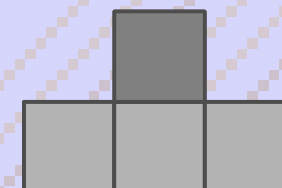
- A configuration C is a mapping from $\mathbb{Z}$ to S.

Cellular Automata

- A *1D-CA* $\mathcal{A}$ is a tuple $(\mathbb{Z}, S, \mathcal{N}, \delta)$.



$S = \{\square, \blacksquare\}$

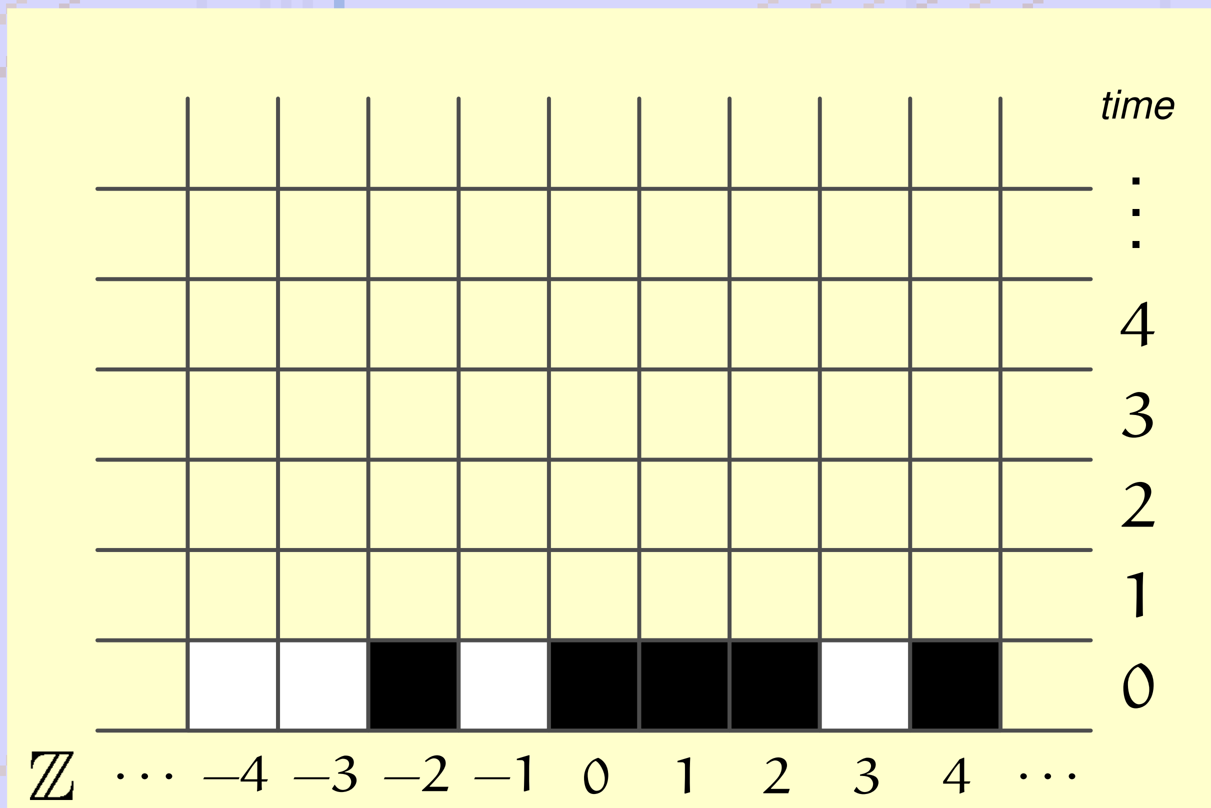


$\mathcal{N} \subseteq_{\text{finite}} \mathbb{Z}$

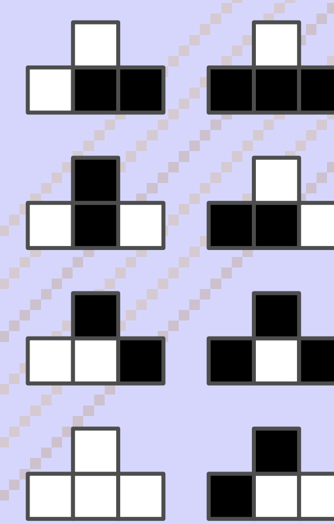
- A configuration C is a mapping from $\mathbb{Z}$ to S.

Cellular Automata

- A *1D-CA* $\mathcal{A}$ is a tuple $(\mathbb{Z}, S, \mathcal{N}, \delta)$.



$S = \{\square, \blacksquare\}$

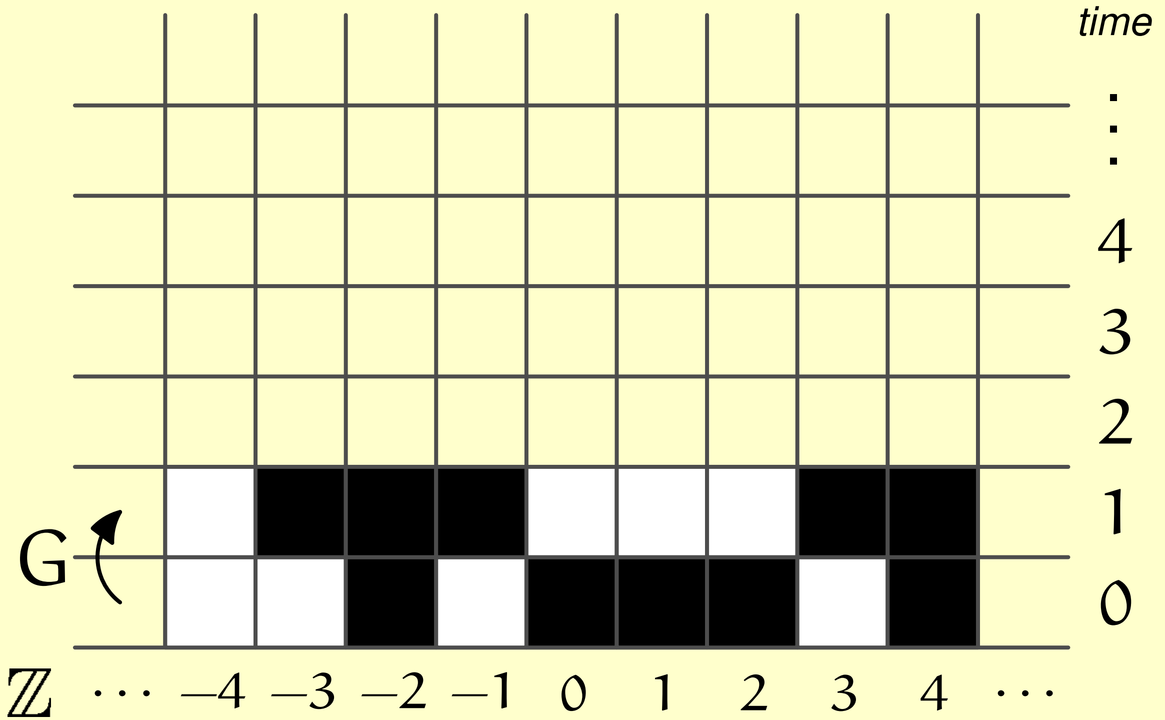


$\delta : S^{|\mathcal{N}|} \to S$

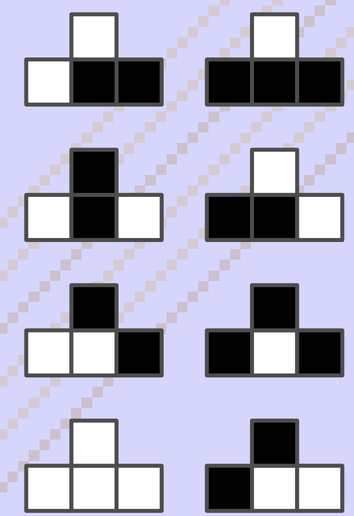
- A configuration C is a mapping from $\mathbb{Z}$ to S.

Cellular Automata

- A *1D-CA* $\mathcal{A}$ is a tuple $(\mathbb{Z}, S, \mathcal{N}, \delta)$.



$S = \{\square, \blacksquare\}$

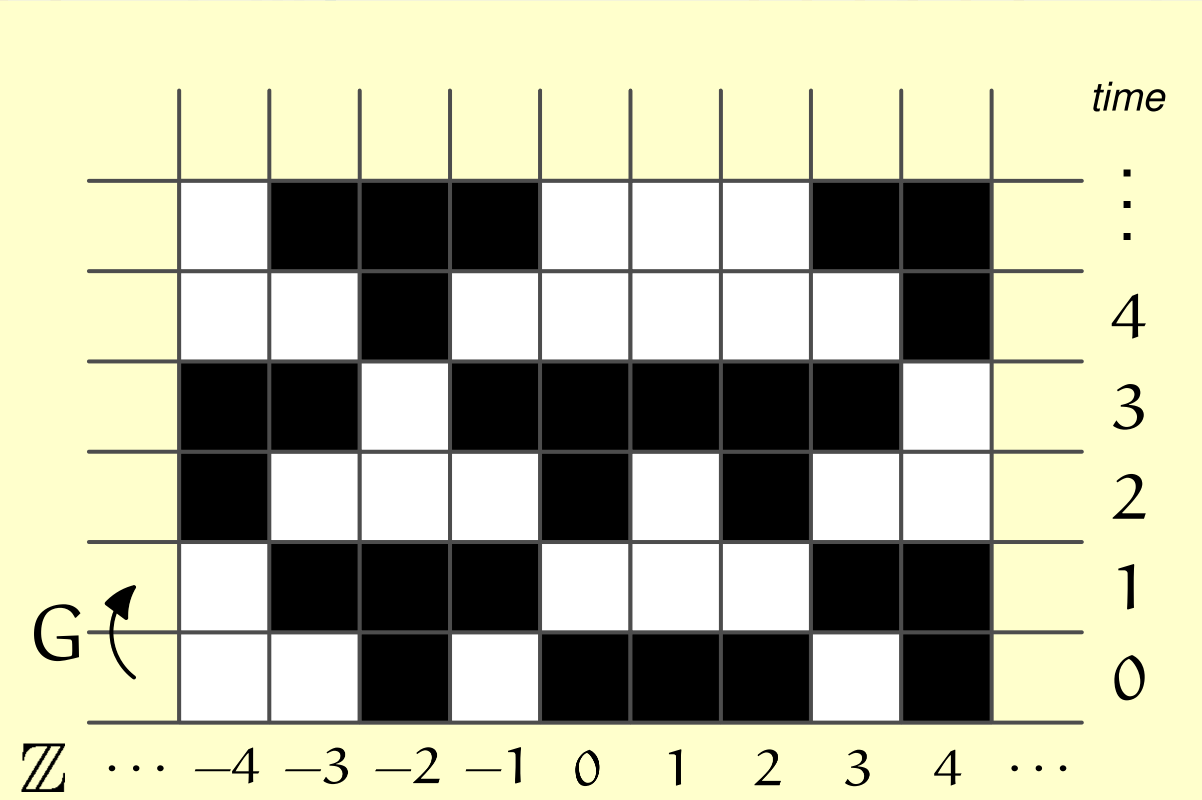


$\delta : S^{|\mathcal{N}|} \to S$

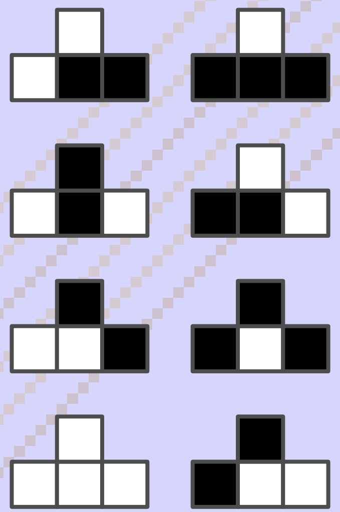
- A configuration C is a mapping from $\mathbb{Z}$ to S.

Cellular Automata

- A *1D-CA* $\mathcal{A}$ is a tuple $(\mathbb{Z}, S, \mathcal{N}, \delta)$.



$S = \{\square, \blacksquare\}$



$\delta : S^{|\mathcal{N}|} \to S$

- A configuration C is a mapping from $\mathbb{Z}$ to S.

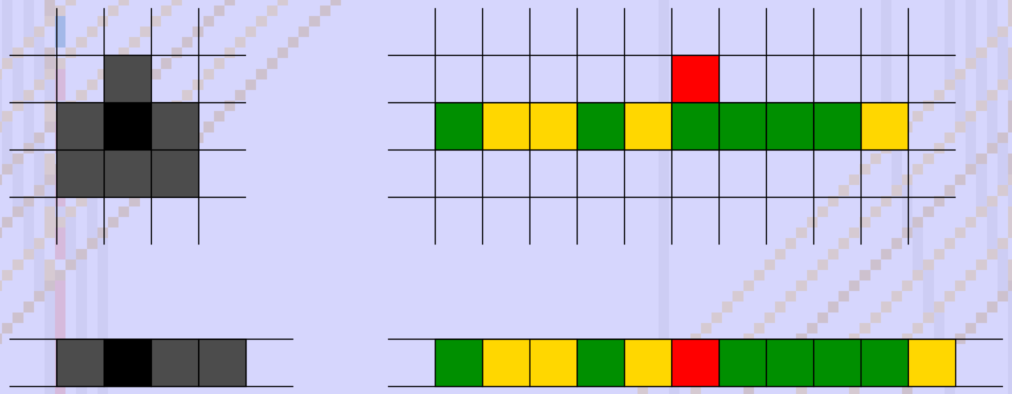
Computation Universality

Idea. A CA is *computation universal* if it can **compute** any partial recursive function.

Computation Universality

Idea. A CA is *computation universal* if it can **compute** any partial recursive function.

- In practice : step-by-step Turing machine simulation.



A. R. Smith III. Simple Computation-Universal Cellular Spaces. 1971

Universalities

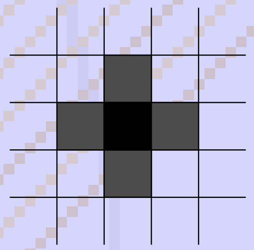
B. Durand and Z. Róka, The game of life: universality revisited, *Cellular automata (Saissac, 1996)* (Kluwer Acad. Publ., Dordrecht, 1999), (pp. 51–74).

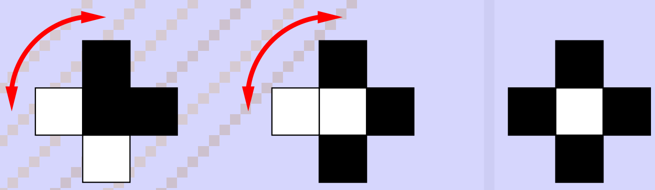
- Several different notions of universality :
  - Turing (computation universality) ;
  - Intrinsic (CA simulating all CA) ;
  - Circuits (CA simulating boolean circuits).
- Problems in the proof of universality of GOL.
- Discusses the difficulty of formalization.

Simple Universal CA

year	author	d	$\lvert N \rvert$	states	universality
1966	von Neumann	2	5	29	intrinsic
1968	Codd	2	5	8	intrinsic
1970	Banks	**2**	**5**	**2**	**intrinsic**
		1	3	18	intrinsic
1971	Smith III	2	7	7	computation
		1	3	18	computation
1987	Albert & Culik II	1	3	14	intrinsic
1990	Lindgren & Nordhal	**1**	**3**	**7**	**computation**
2002	NO	**1**	**3**	**6**	**intrinsic**
2002	Cook & Wolfram	**1**	**3**	**2**	**computation**

Banks' Universal 2D-CA

$$\Big(\mathbb{Z}^2, \{\blacksquare, \square\},$$

$$, \delta\Big)$$

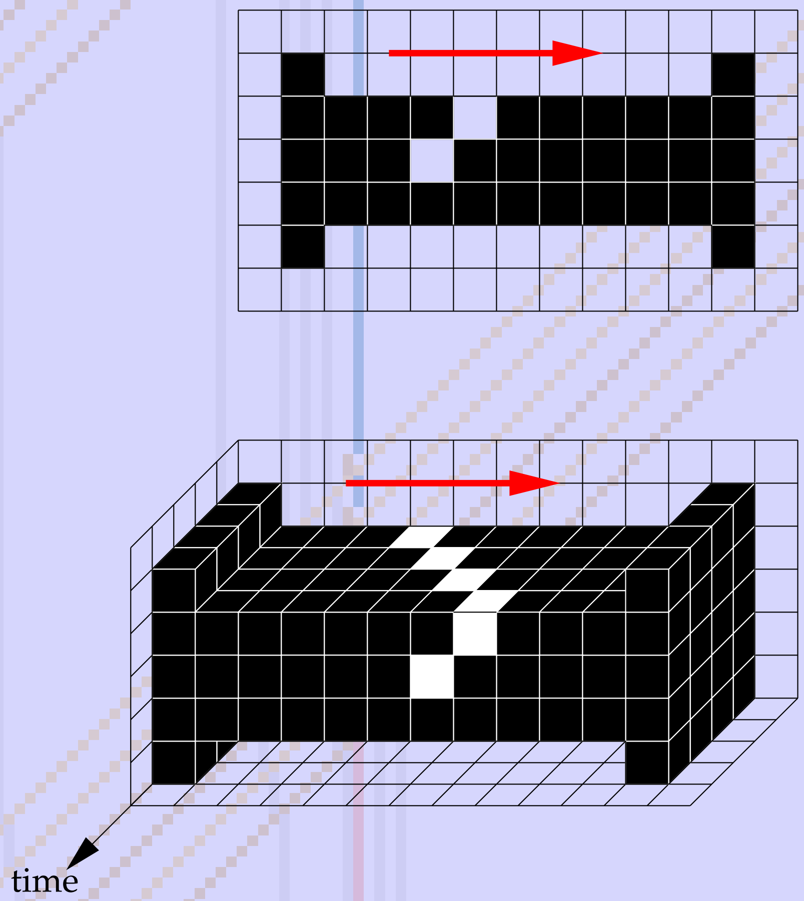


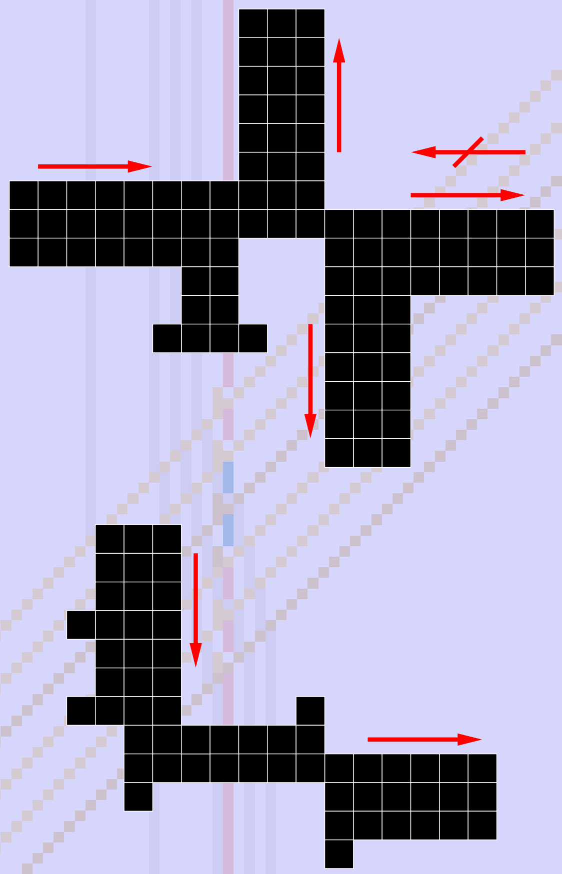
E. R. Banks. Universality in Cellular Automata. 1970

Idea. Emulate logical circuits by building :

- wires transporting binary signals
- logical gates AND, OR and NOT
- wires crossing

Banks' Universal 2D-CA





Banks' Universal 2D-CA

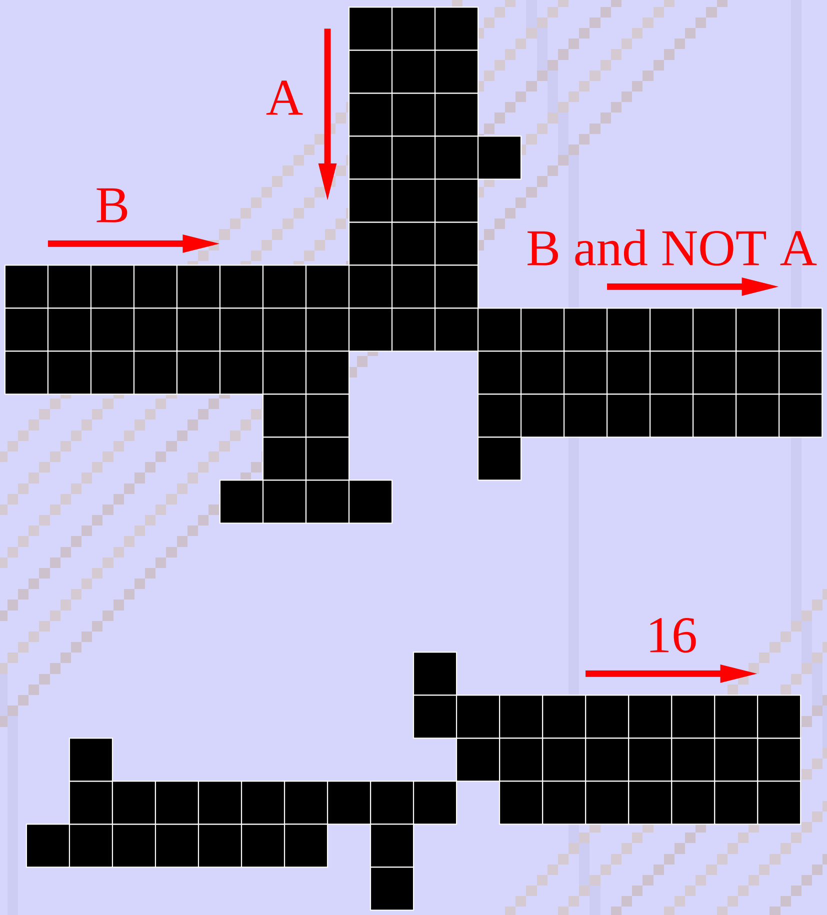
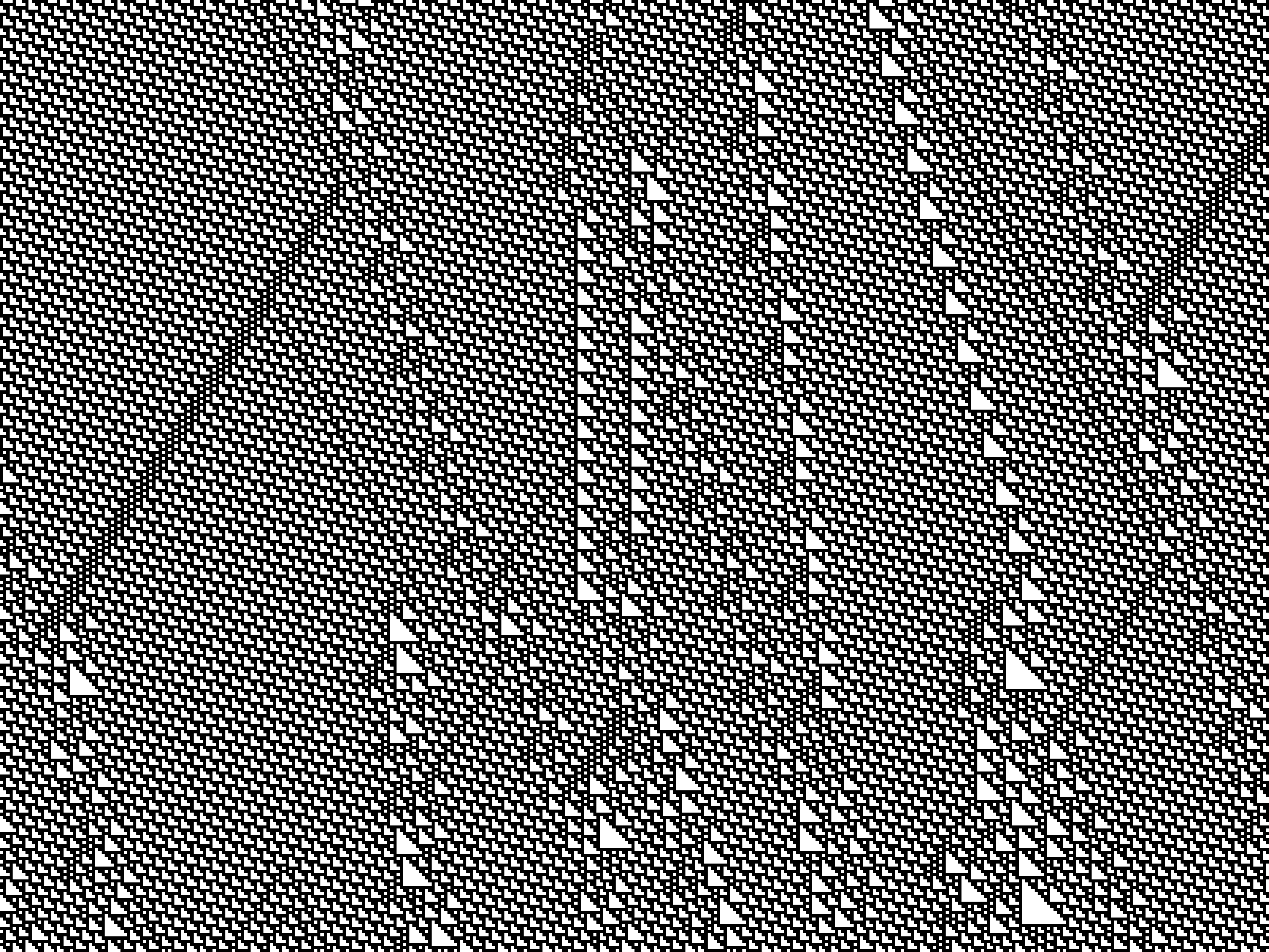


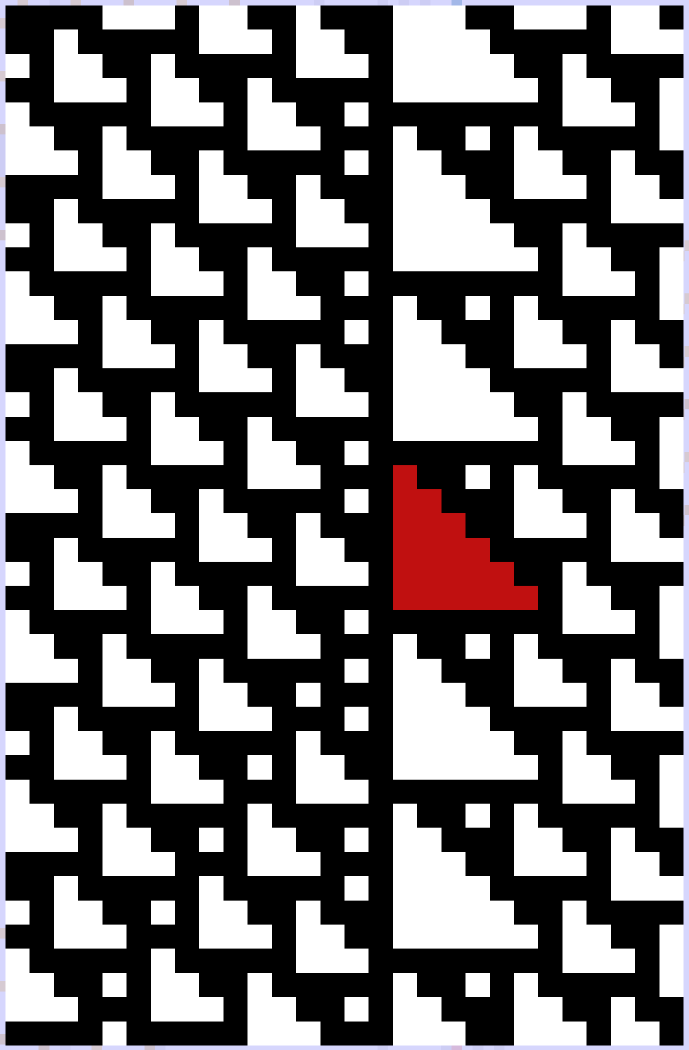
Table of Content



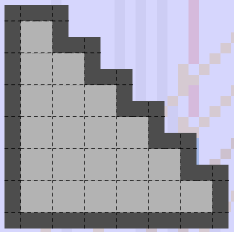
Point of View

- We want to construct huge space-time diagrams.
- We need to prove their existence.
- We cannot simply draw some basis of them because of the size of diagrams involved (squares of millions of cells on a side).

Tiling the plane

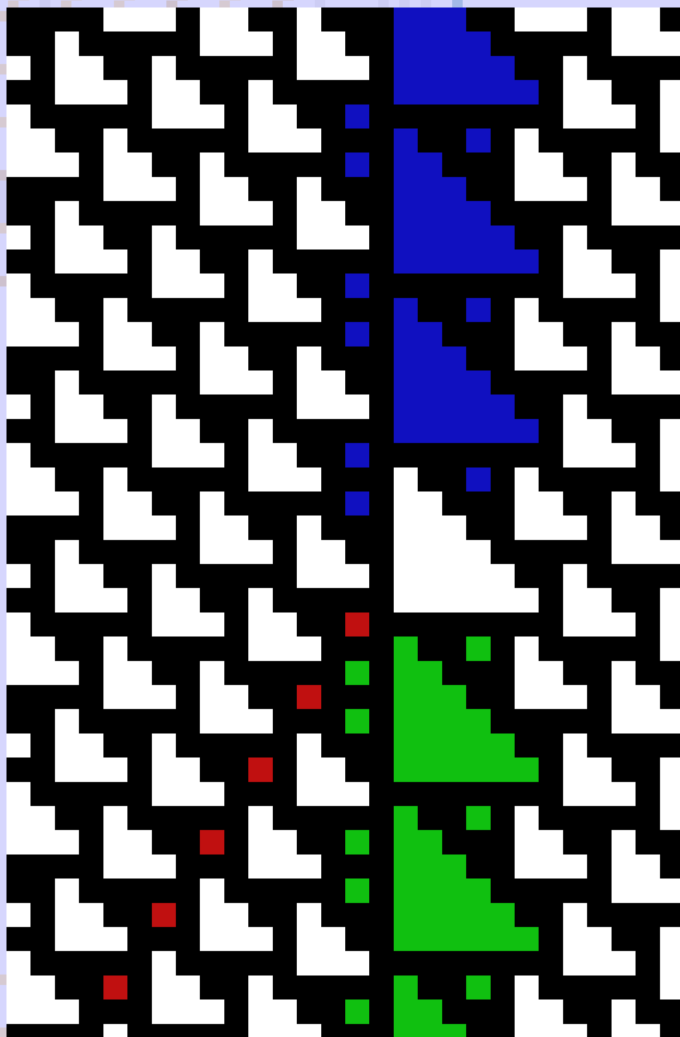


Space-time diagrams as tiling of the plane by triangles

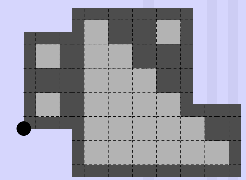


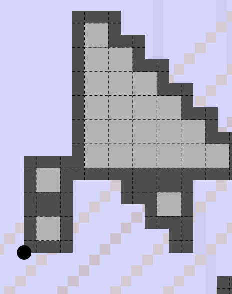
Changing the point of view from 1D to 2D

Particles



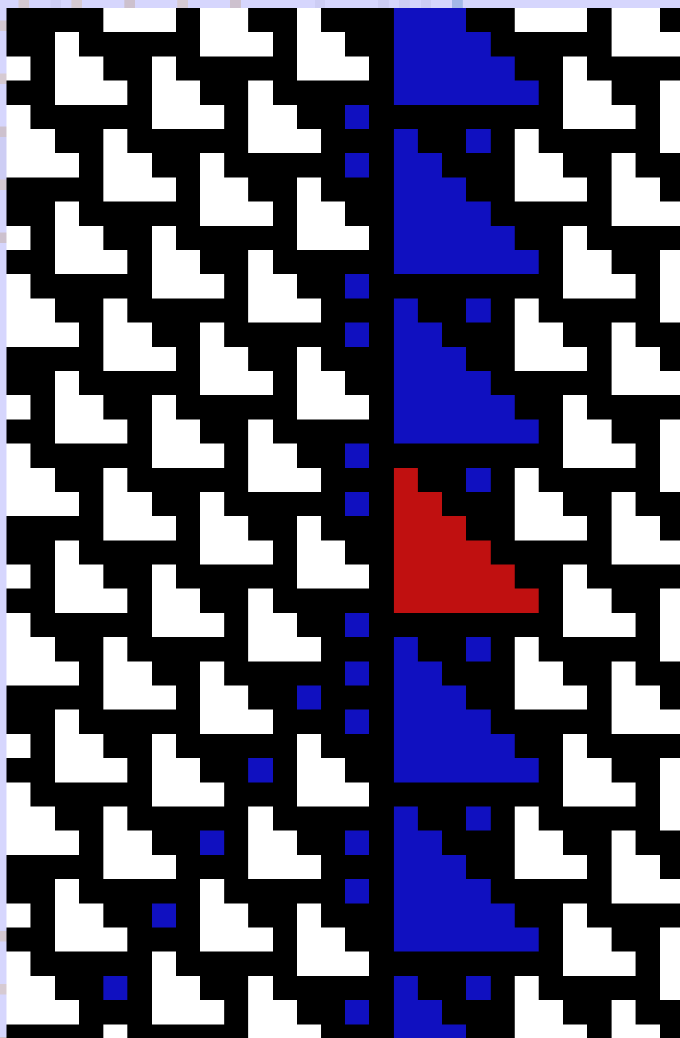
Particles repeats themselves in a uniform background

$A = \left\langle \binom{0}{7}, \right.$  $\left. \right\rangle$

$B = \left\langle \binom{0}{7}, \right.$  $\left. \right\rangle$

$C = \left\langle \binom{2}{3}, \right.$  $\left. \right\rangle$

Collisions



Particles collide when meeting

$$\Gamma : \quad \binom{0}{0}C + \binom{0}{-4}A \vdash \binom{0}{5}B$$

+ some pertubation pattern F

We are given a set of valid elementary particles and elementary collisions

Bindings

- To combine collisions we use one operation : binding.

$$\Gamma' = \left(\begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix} \Gamma_1 + \begin{pmatrix} \alpha_2 \\ \beta_2 \end{pmatrix} \Gamma_2 + \cdots + \begin{pmatrix} \alpha_n \\ \beta_n \end{pmatrix} \Gamma_n \right)_{\text{bind}}$$

Principle Merge incoming and outgoing particles when possible. Some bindings are not valid !

- Binding is easy to construct and validate.

Table of Content

Sketch of the proof

- We prove that rule 110 is Turing-universal.

1. Reduce Turing Machines to Post Tag Systems.

2. Reduce Tag Systems to Cyclic Tag Systems.

3. Encode Cyclic Tag Systems with collisions.

Post Tag Systems

M. Minsky, *Computation : Finite and Infinite Machines* (Prentice Hall, Englewoods Cliffs, 1967).

- A classical model used to prove universality of small Turing Machines.
- Configurations are words on Σ, a system is given by $(k, v_1, \ldots, v_{|\Sigma|})$. A transition from u is done as follows :

$$u_1 \ldots u_k u_{k+1} \ldots u_m \;\vdash\; u_{k+1} \ldots u_m \cdot v_{u_1}$$

- When the rule cannot be applied, the system accepts.

Cyclic Tag systems

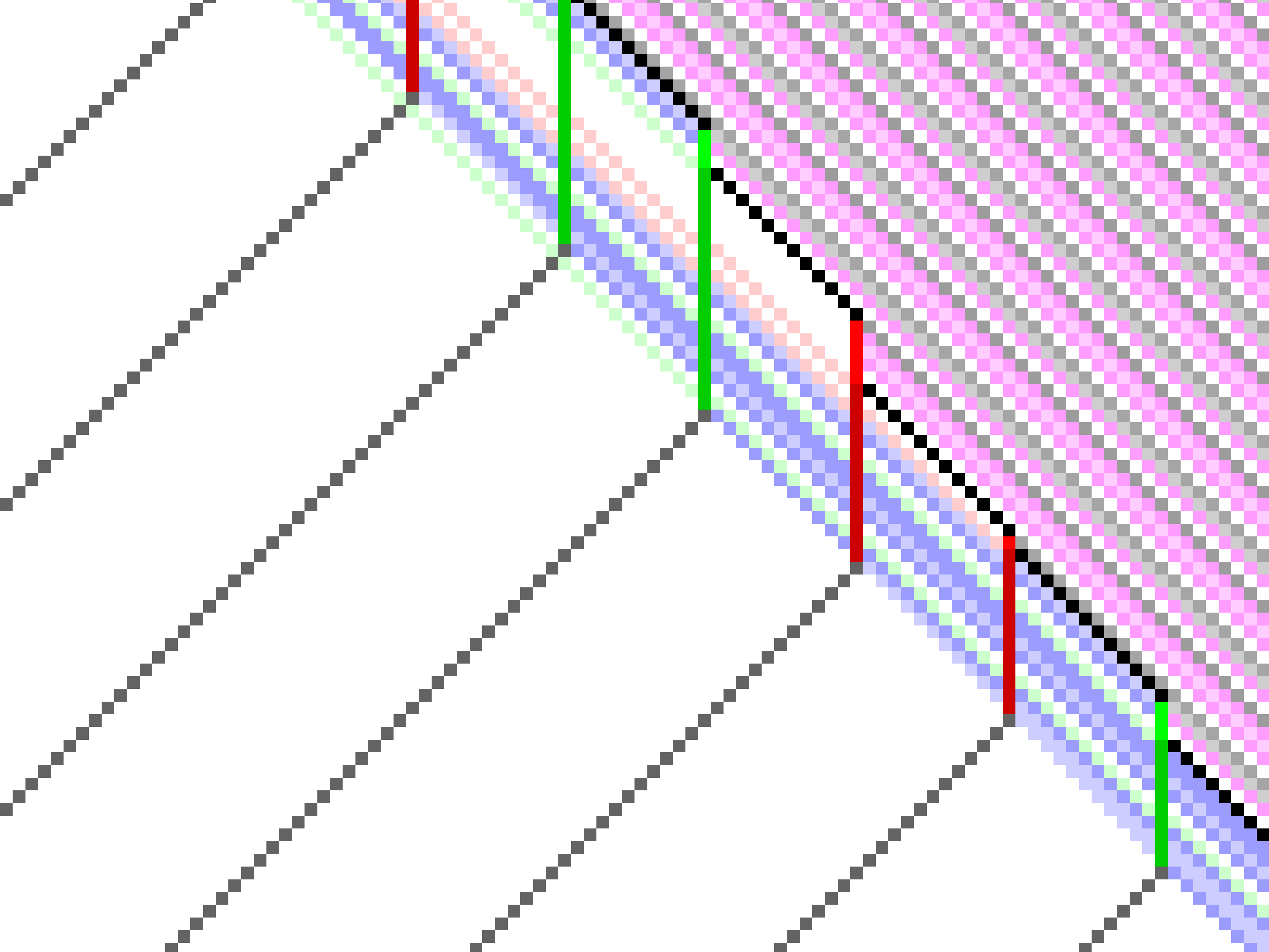
- A cyclic tag system acts only on the binary alphabet.
- A configuration is given by a word u and a set of finite words $(v_1, \ldots, v_n)$.
- A transition is done as follows :
  1. if the first letter of u is 1 then catenate v_1 to u ;
  2. erase the first letter of u ;
  3. rotate the list of words as $(v_2, \ldots, v_n, v_1)$.
- Such systems can simulate any Post Tag System.

A Local Dynamical System

Idea Replace the finite set of words by a periodic one.

Idea Make the first letter cross the word letter by letter.

- A transition is done as follows :
  1. the first letter of u crosses the word to the right ;
  2. when it meets a boundary, it destroys it ;
  3. it begins either to erase of unfreeze letters ;
  4. when it meets the second boundary, it stops.



A Sample CA

- 16 states, a large neighborhood $(-1, 0, 1, 2)$.
- Locally it can simulate the cyclic Tag system.

A Sample CA

- 16 states, a large neighborhood $(-1, 0, 1, 2)$.
- Locally it can simulate the cyclic Tag system.

Claim This CA may not work ! Why ?

A Sample CA

- 16 states, a large neighborhood $(-1, 0, 1, 2)$.
- Locally it can simulate the cyclic Tag system.

Claim This CA may not work ! Why ?

- Synchronization problems may appear. Be careful.

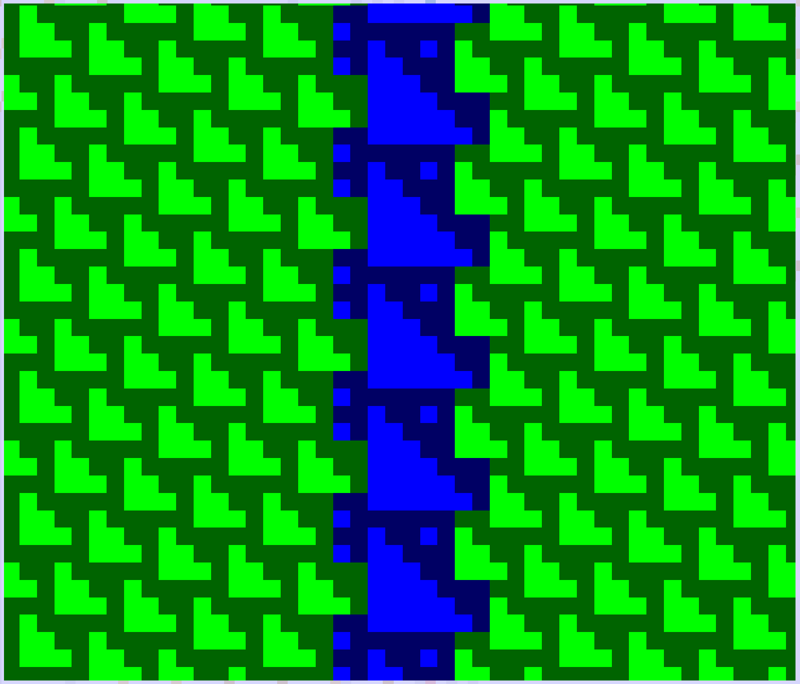
Table of Content

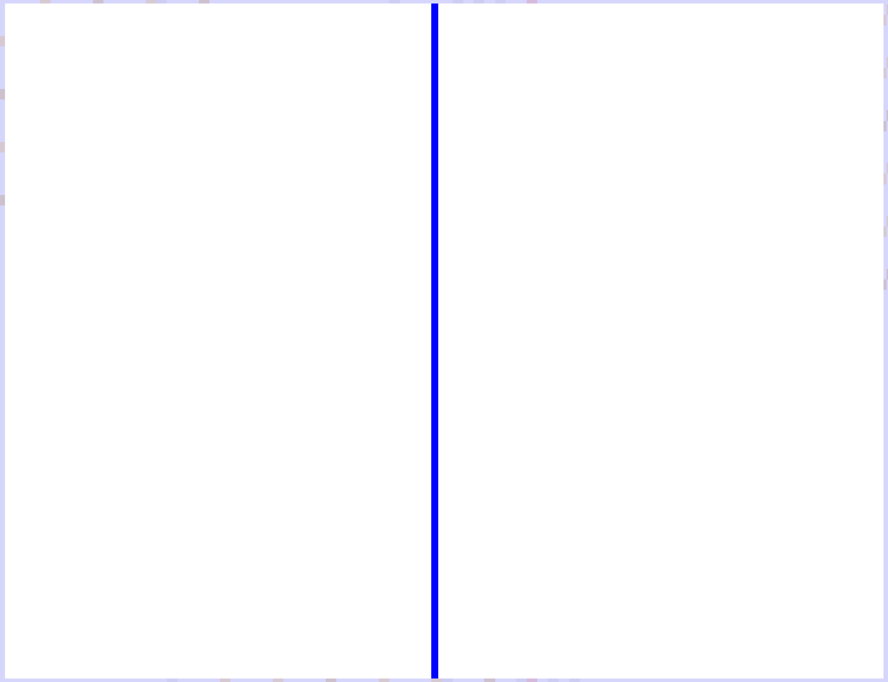
Roadmap

- Now we need to exhibit the gadgets for rule 110.
- This is very technical and requires an Oracle.
- M. Cook and S. Wolfram “tour de force”.

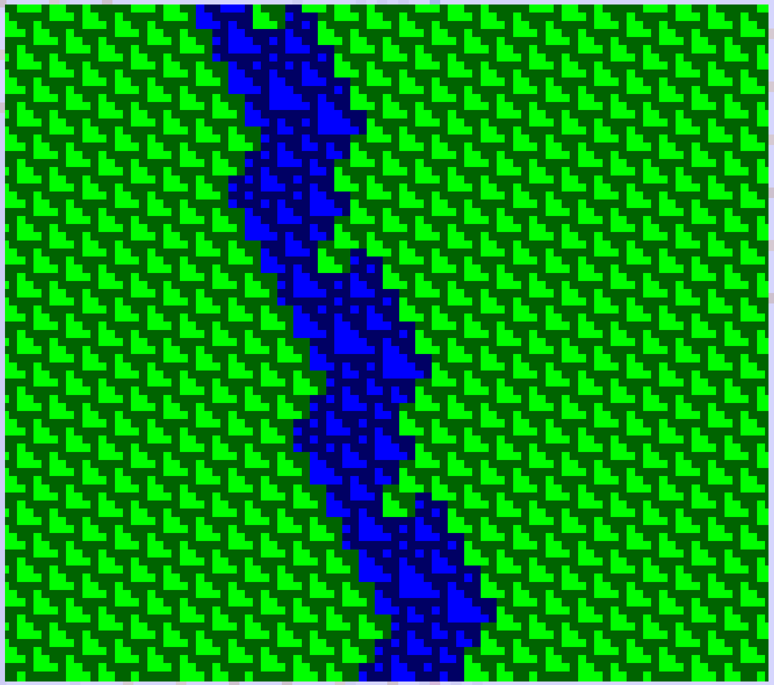
  S. Wolfram, *A New Kind of Science*, 2002

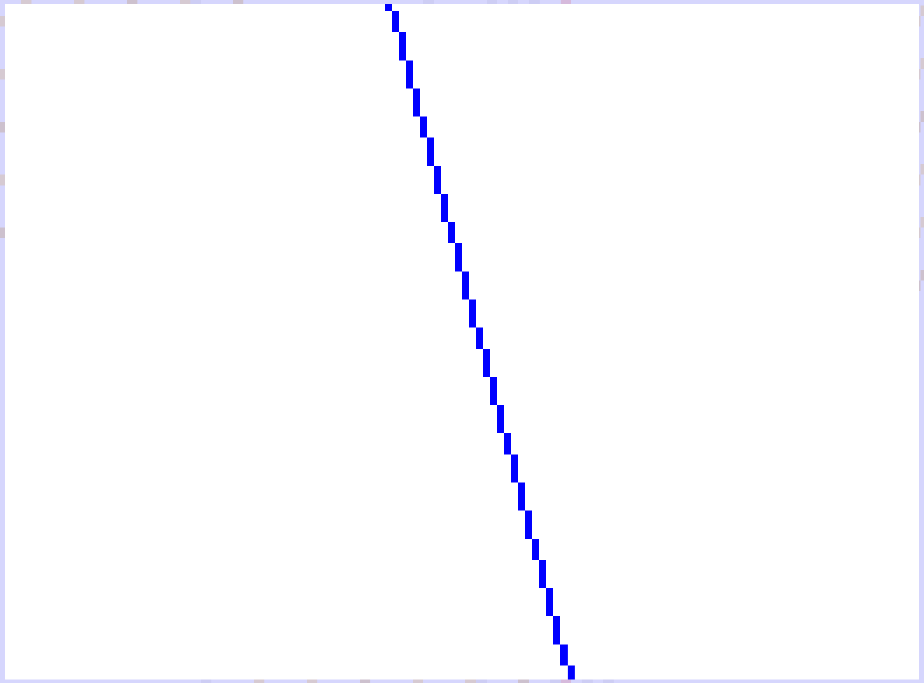
information active



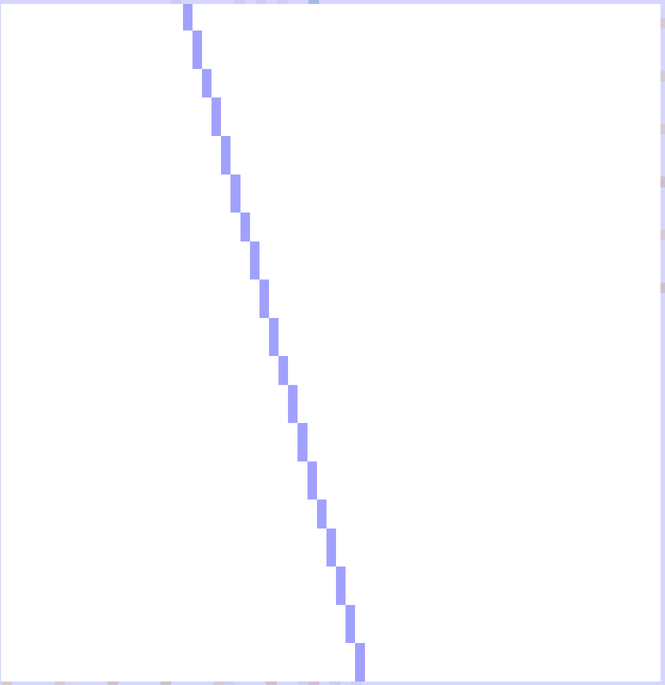


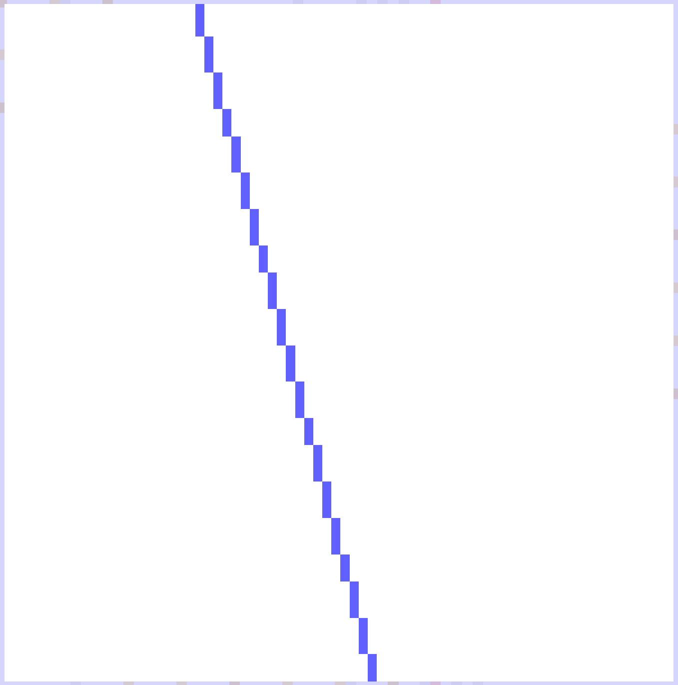
information passive



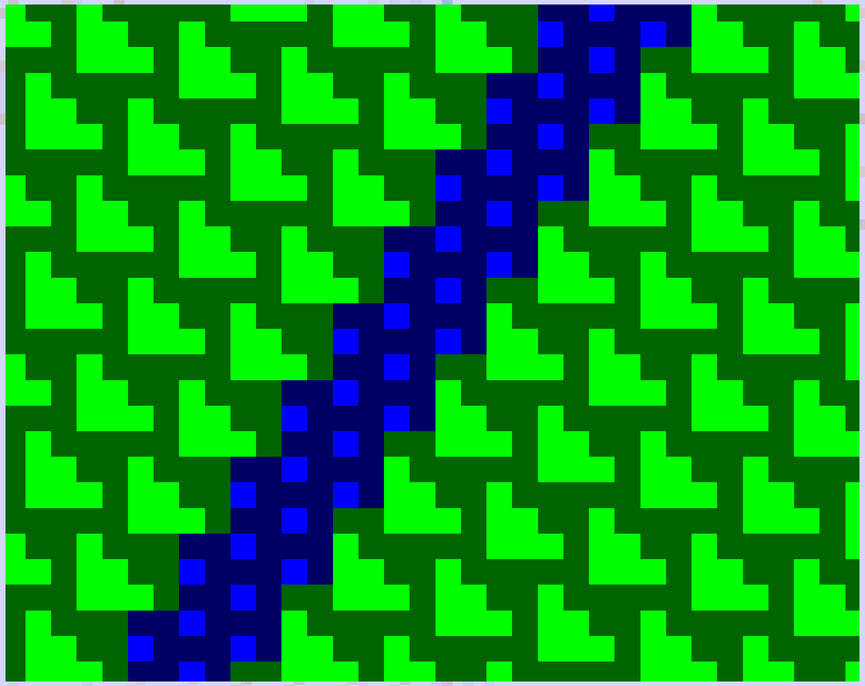


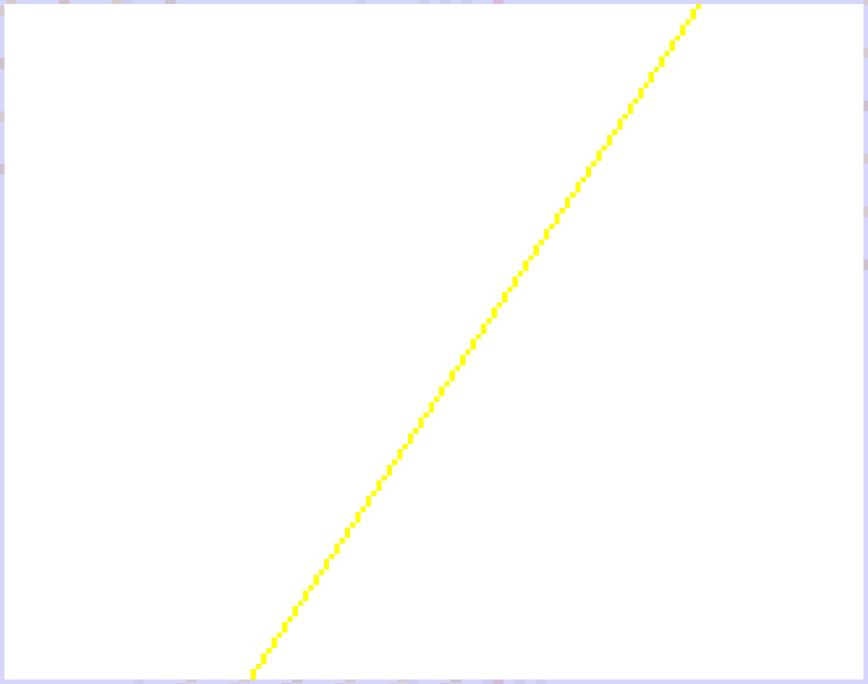
info passive (x2) (x3)



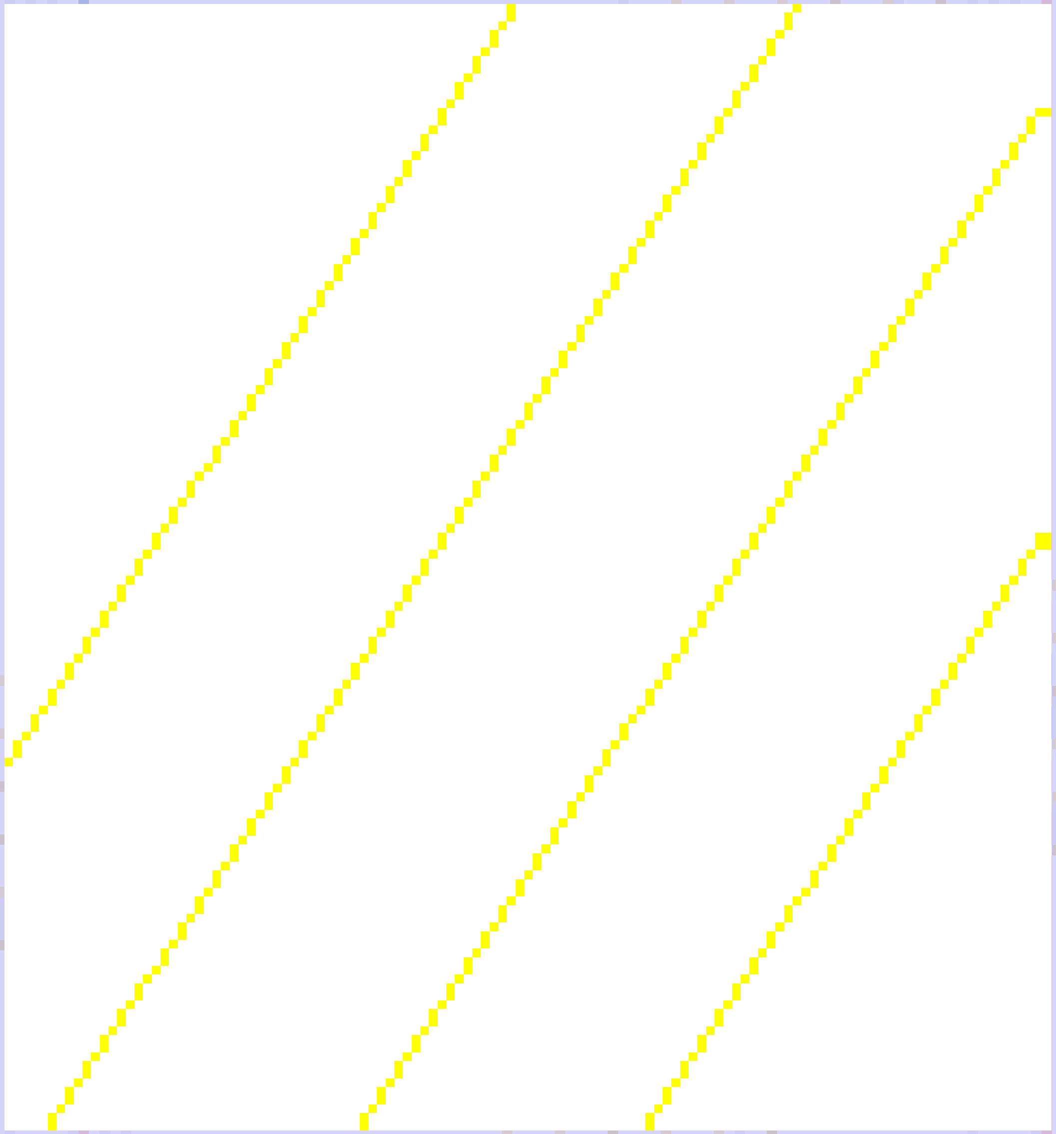


synchronisation

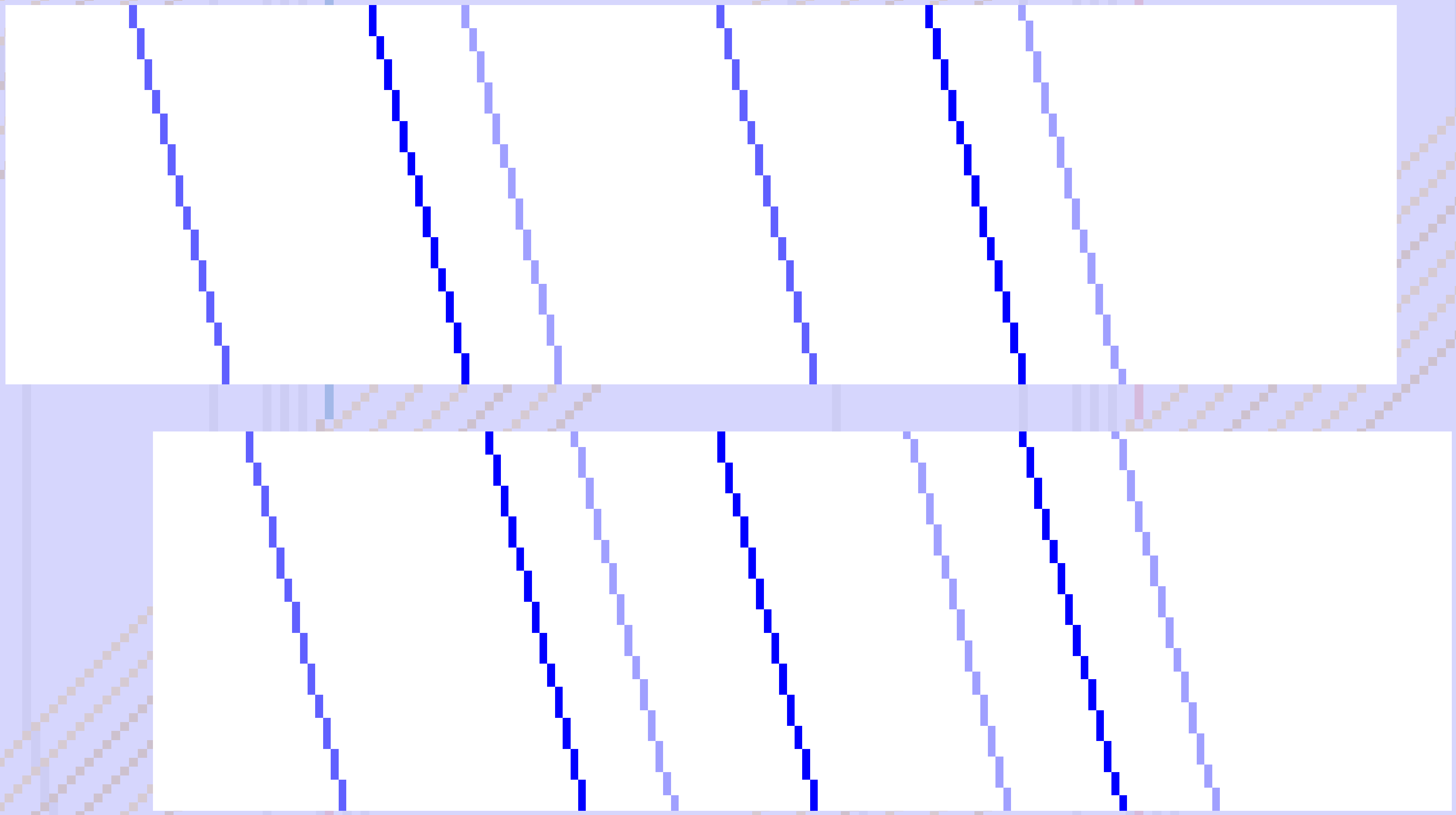




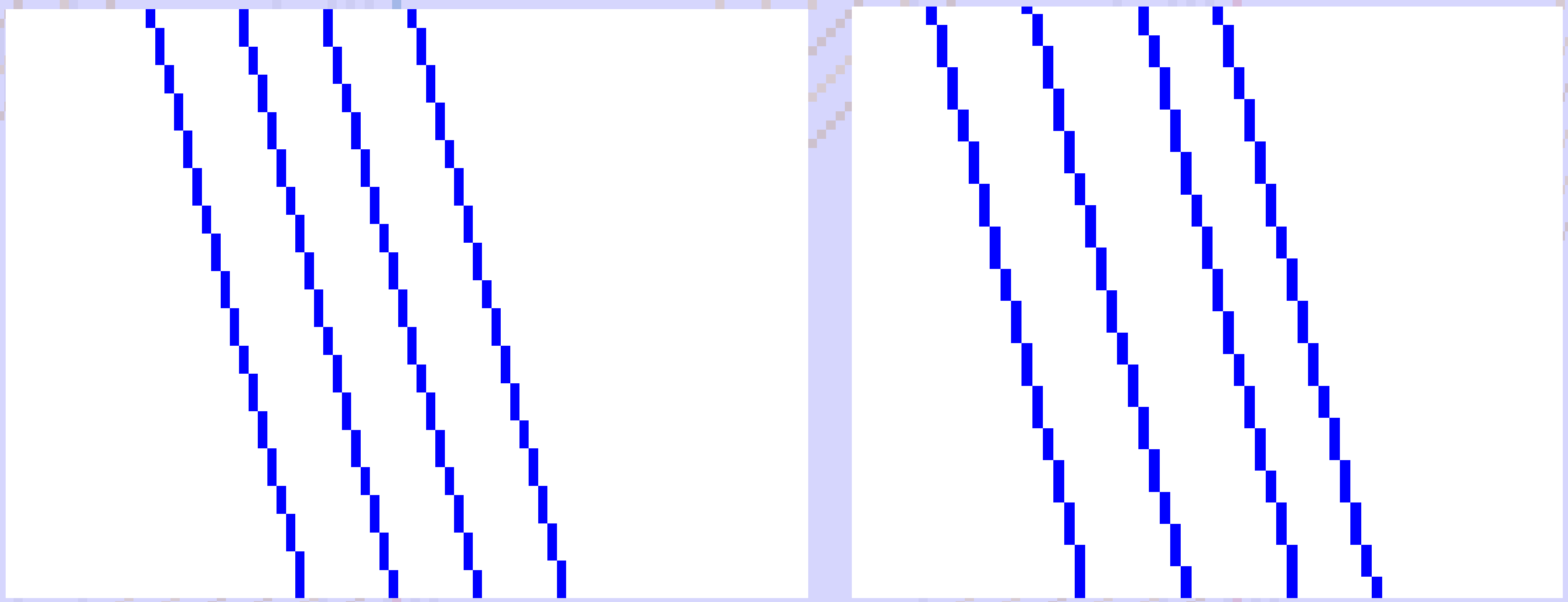
(E) synchronisation



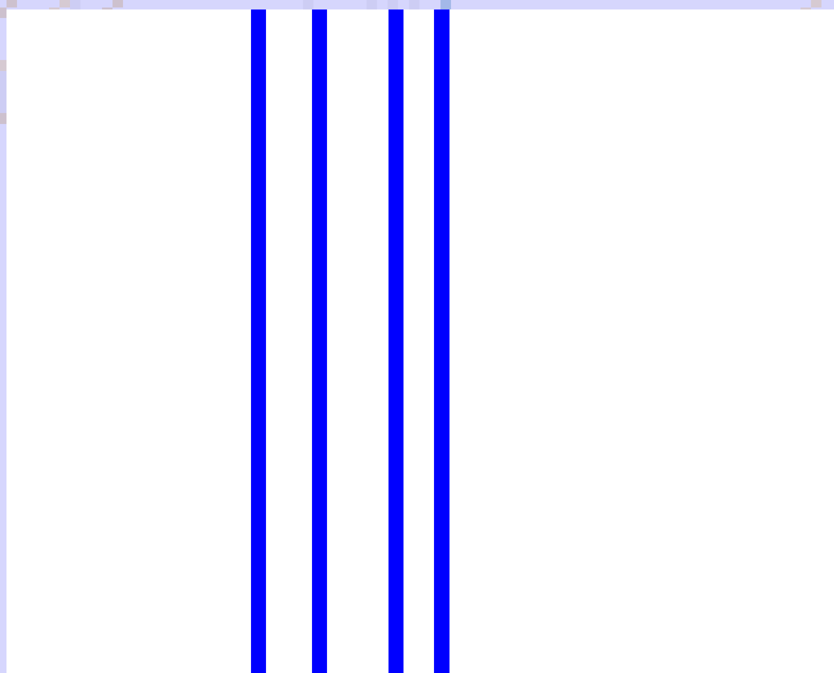
(E) pré-bits N B

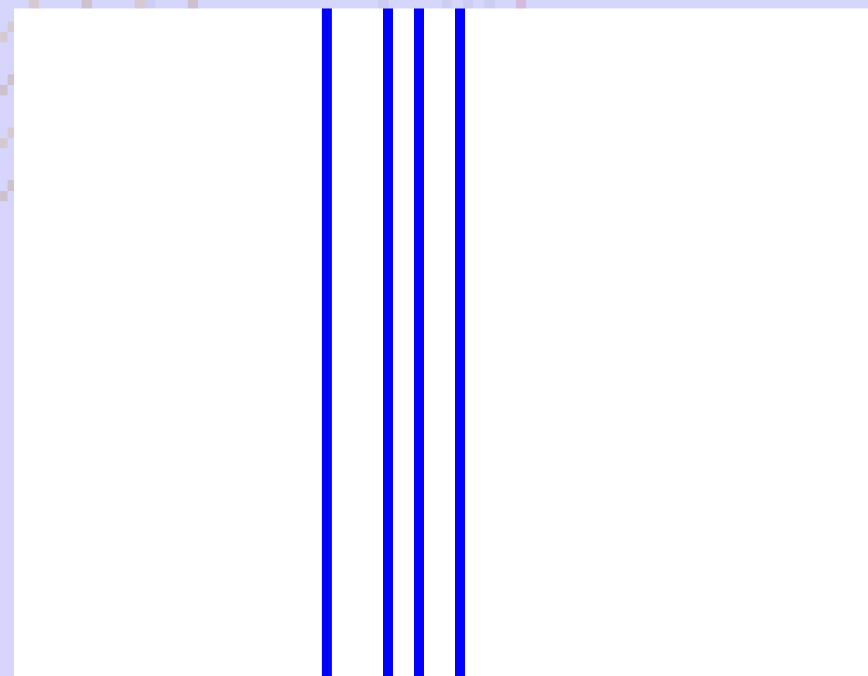


(E) bits passifs N B

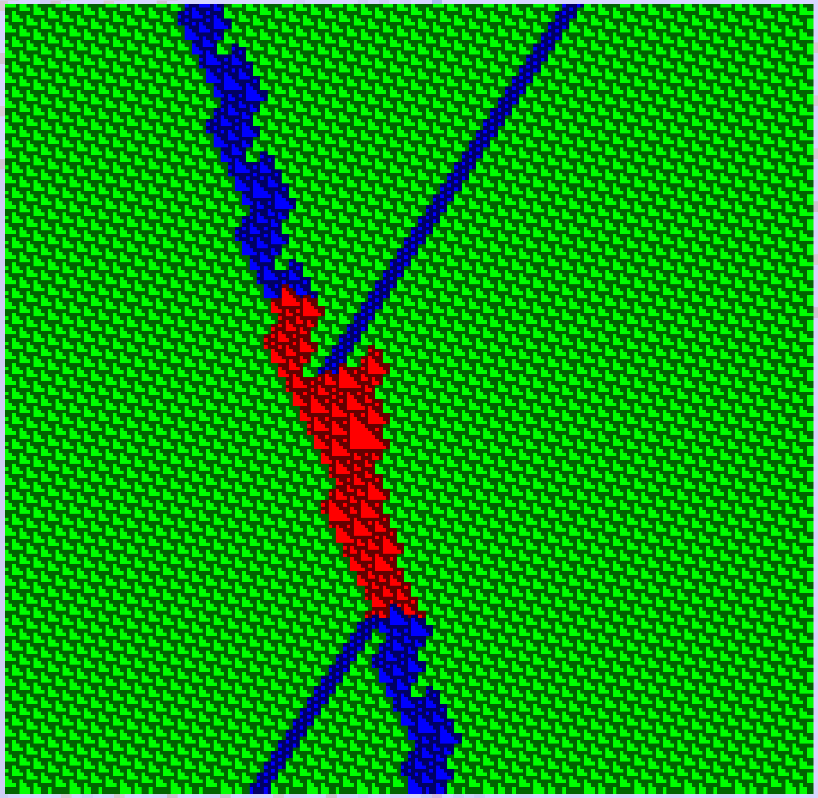


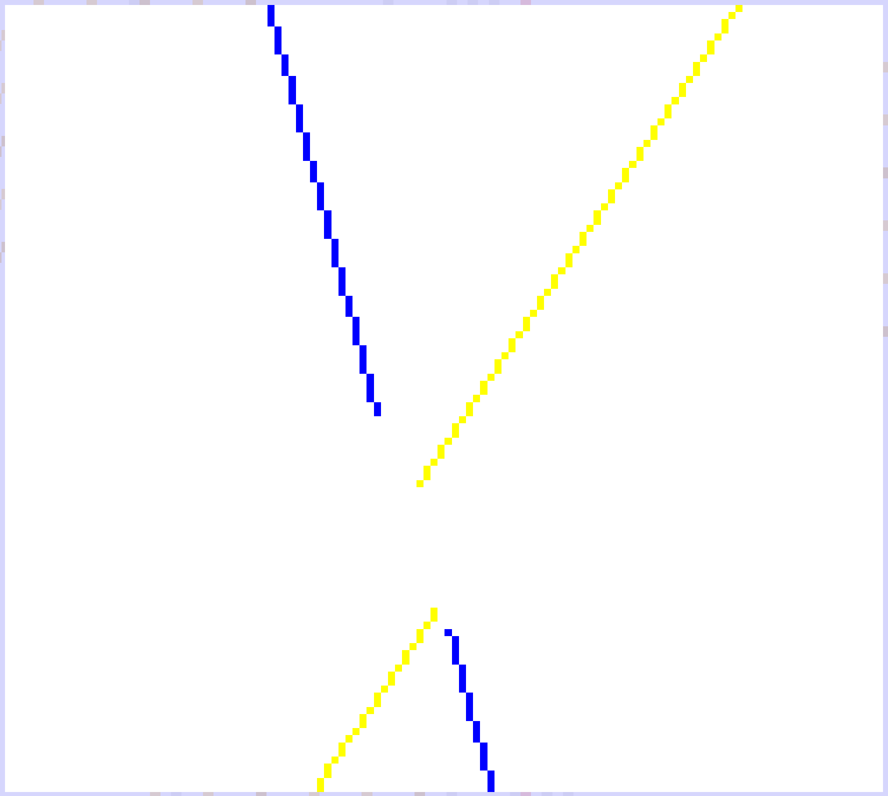
(E) bits actifs N B



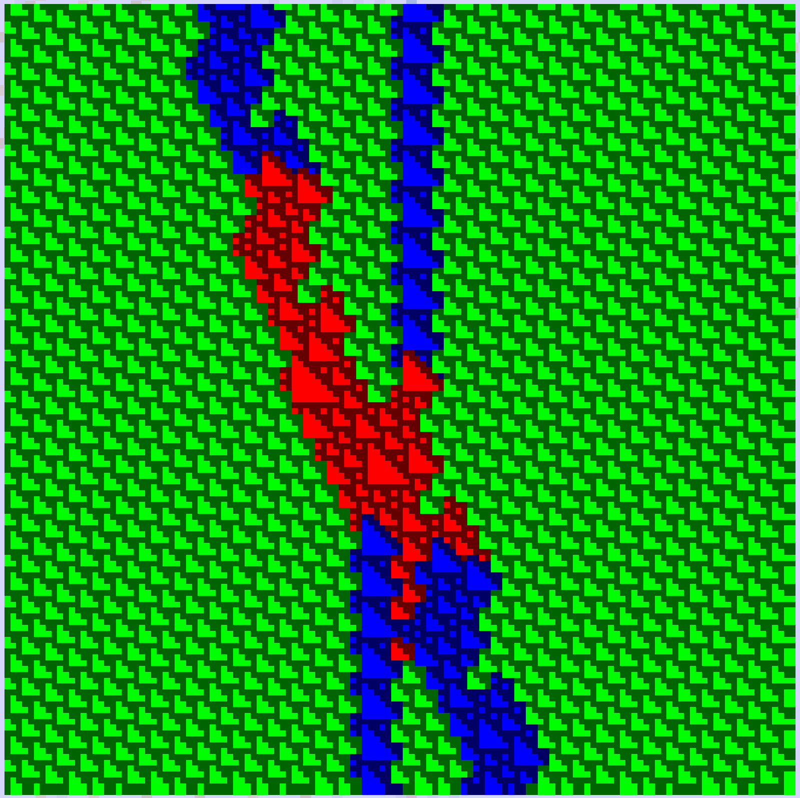


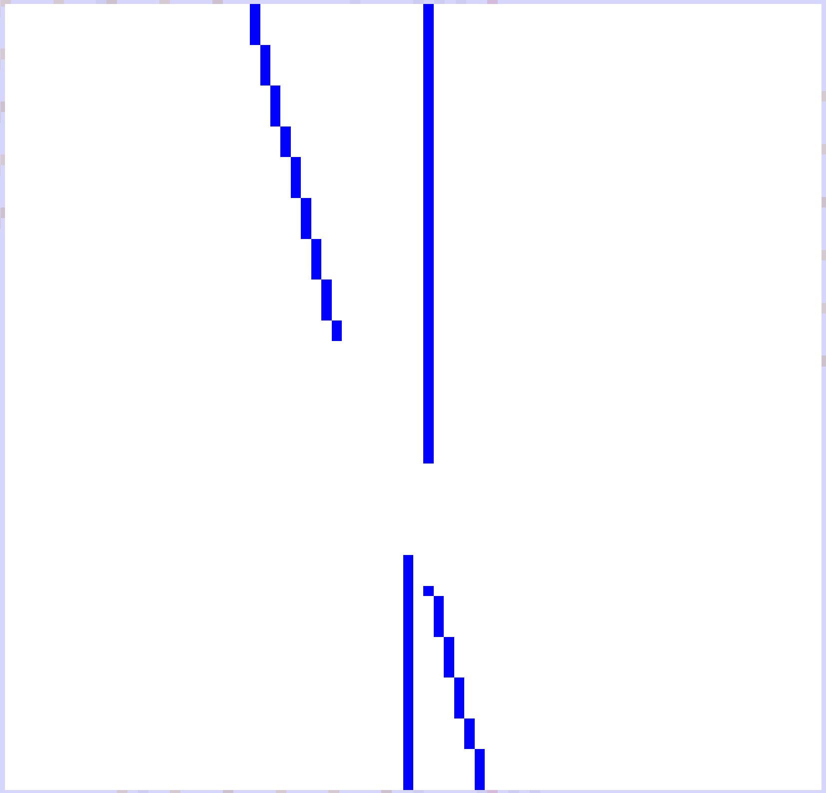
croisement1



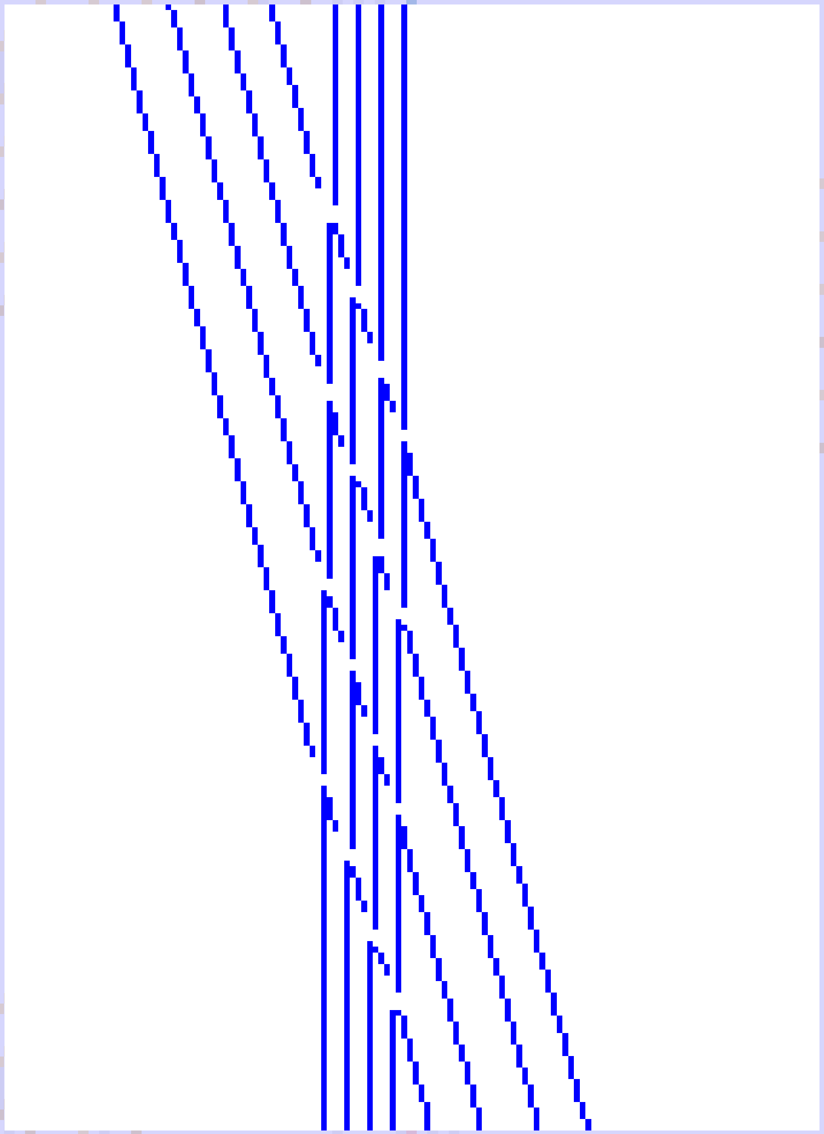


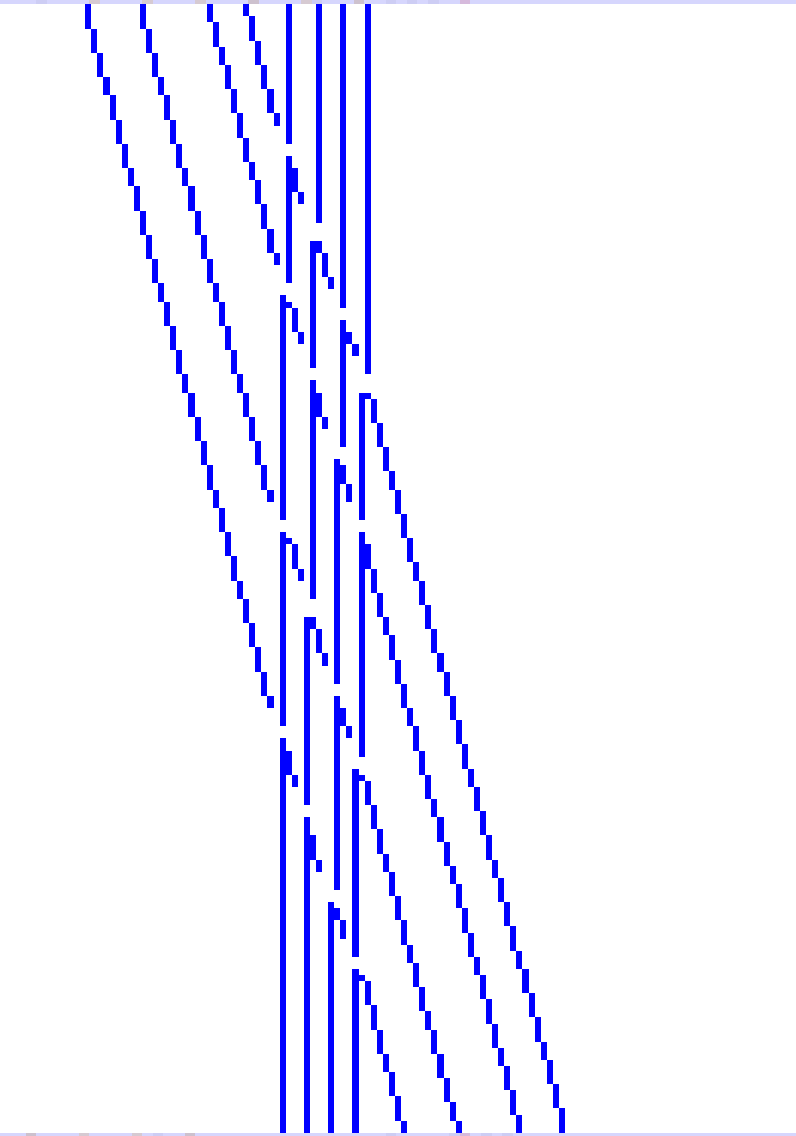
croisement2



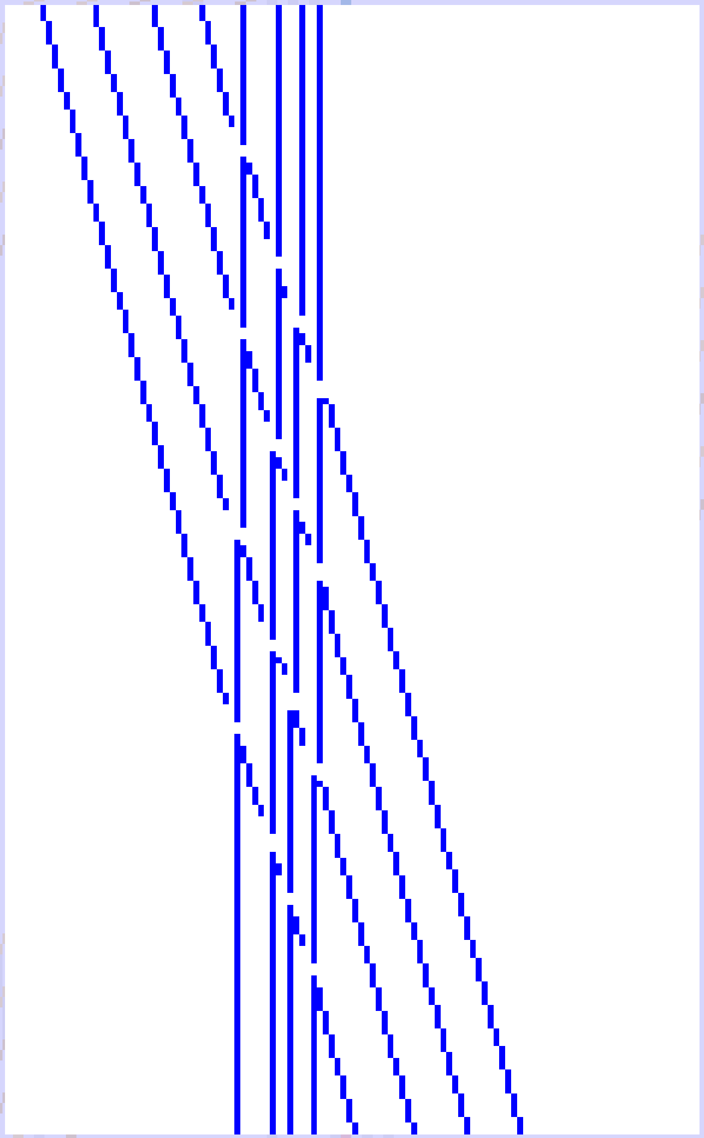


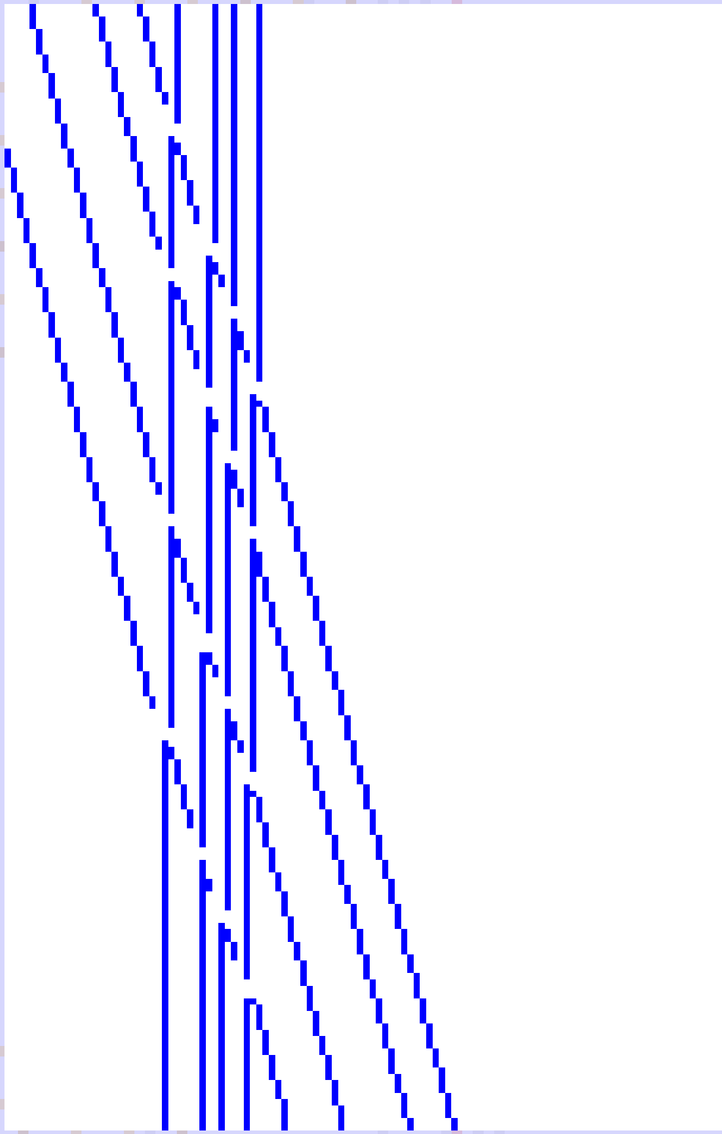
(E) croisement NN NB



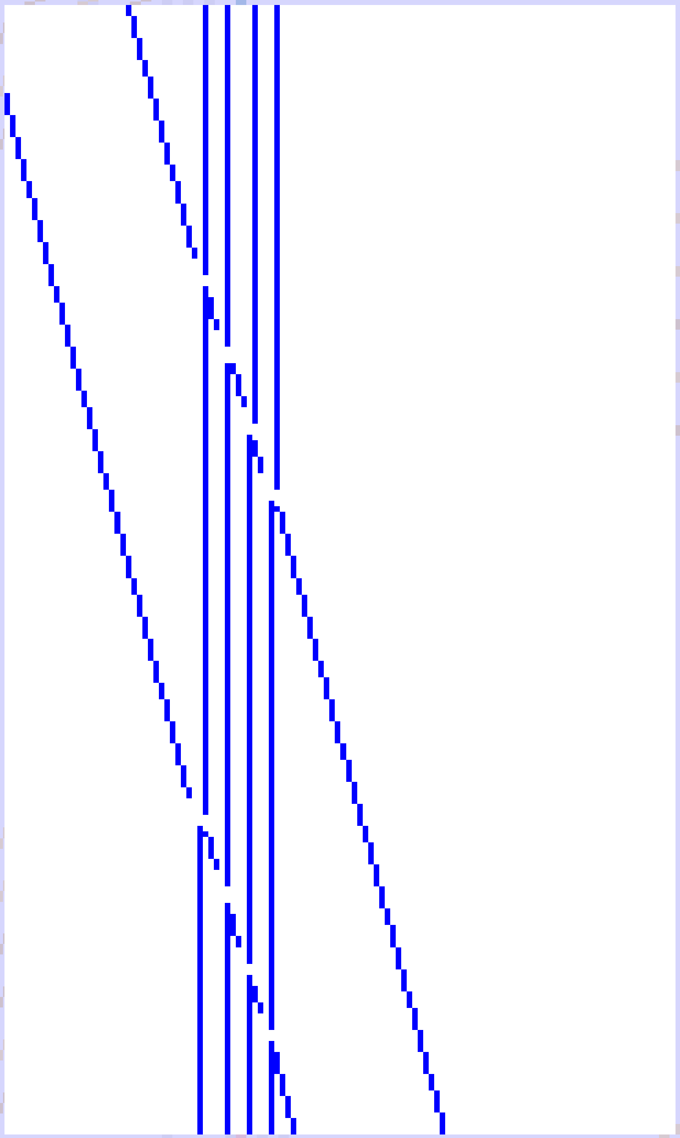


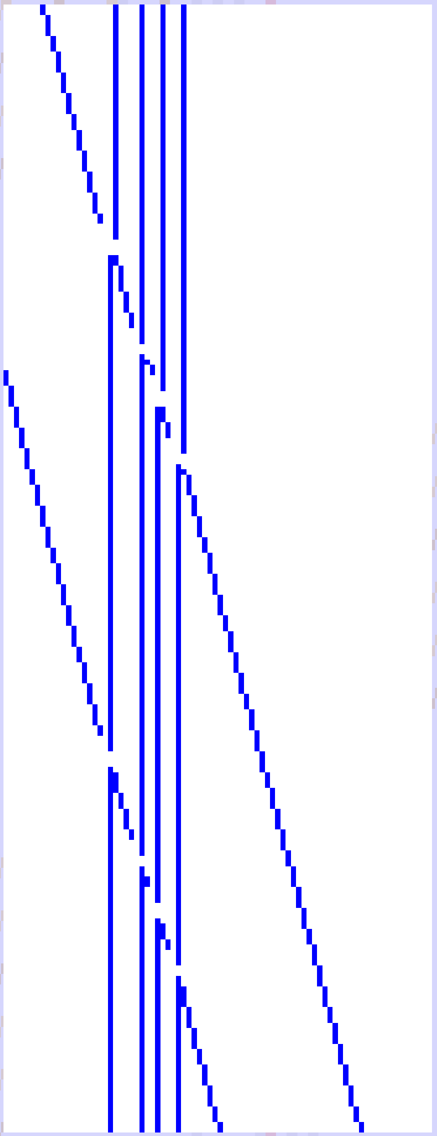
(E) croisement BN BB



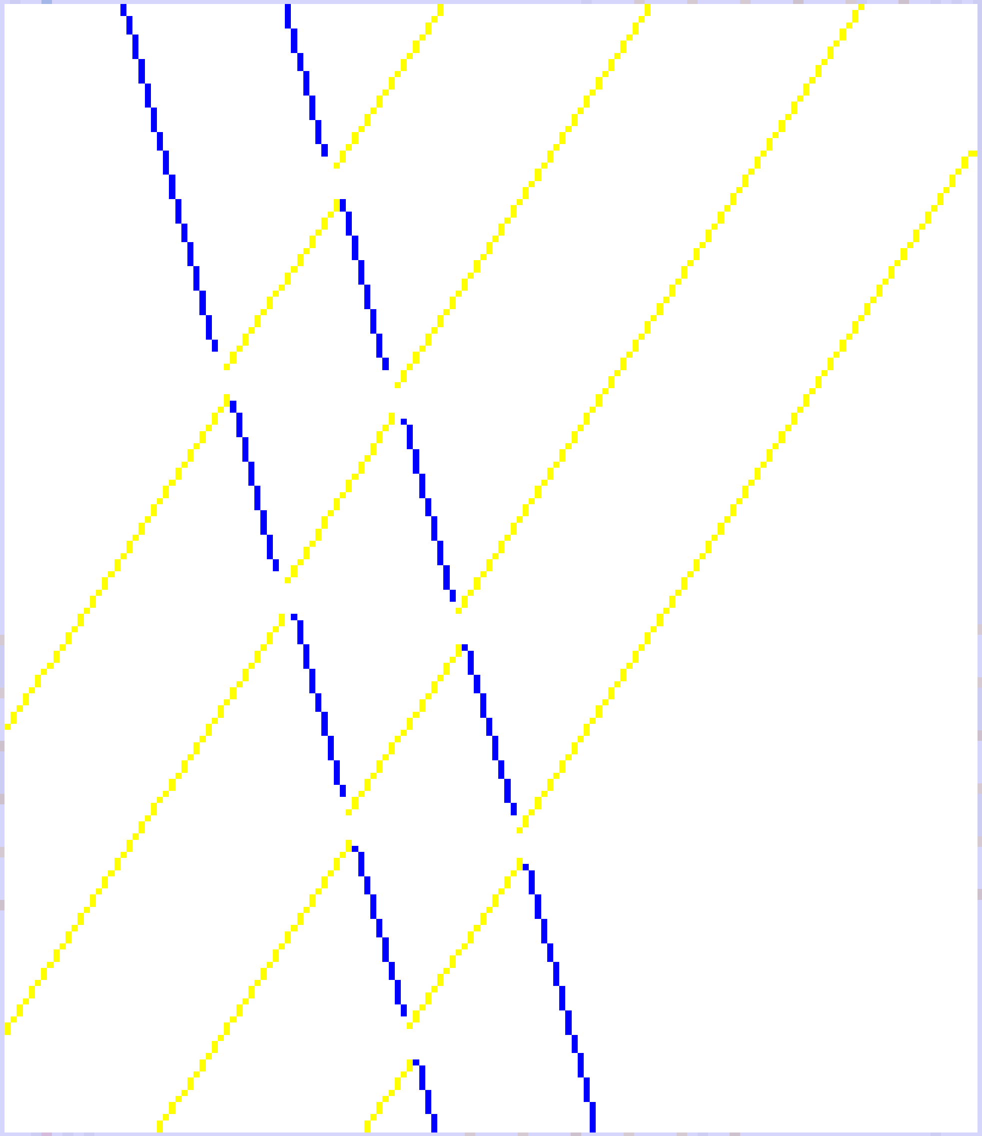


(E) poubelle N B

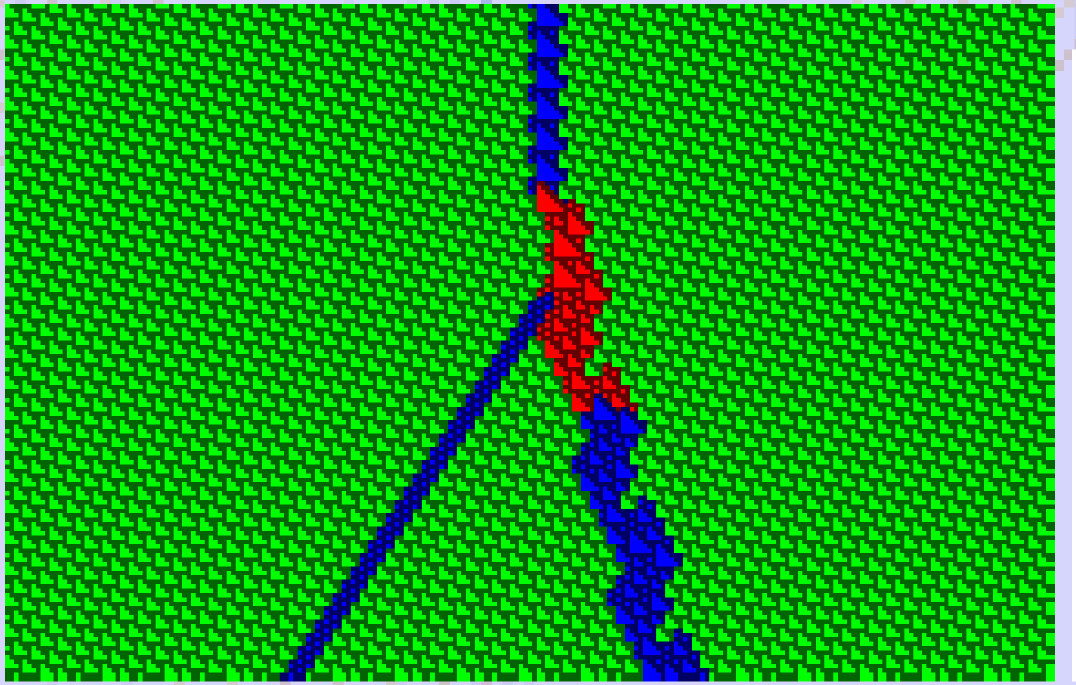


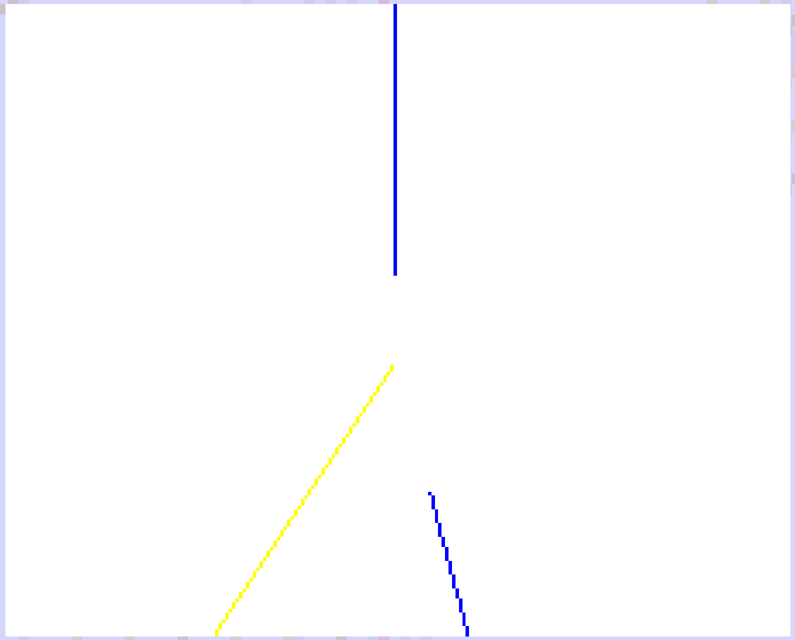


(E) poubelle sync

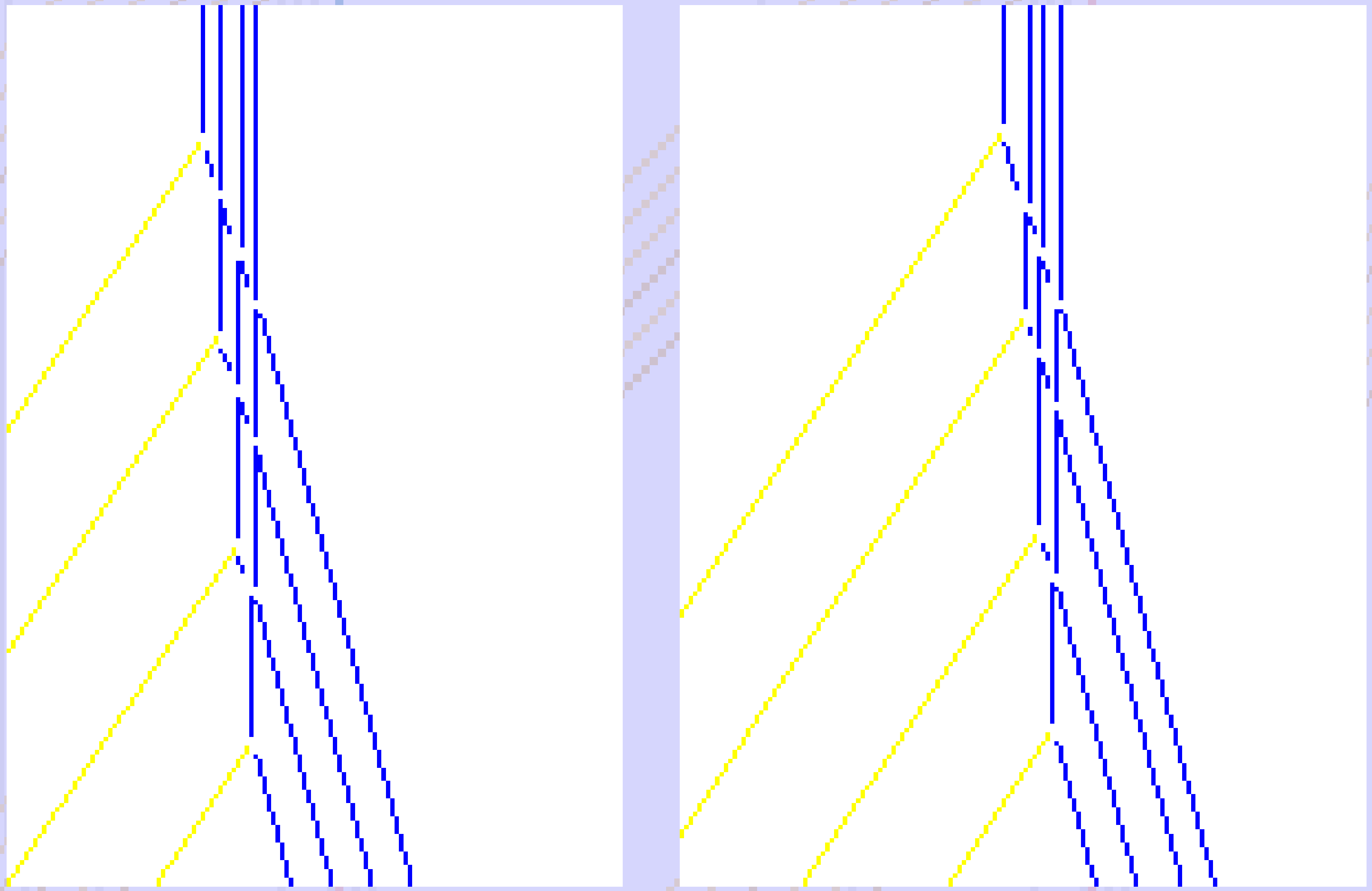


redressement

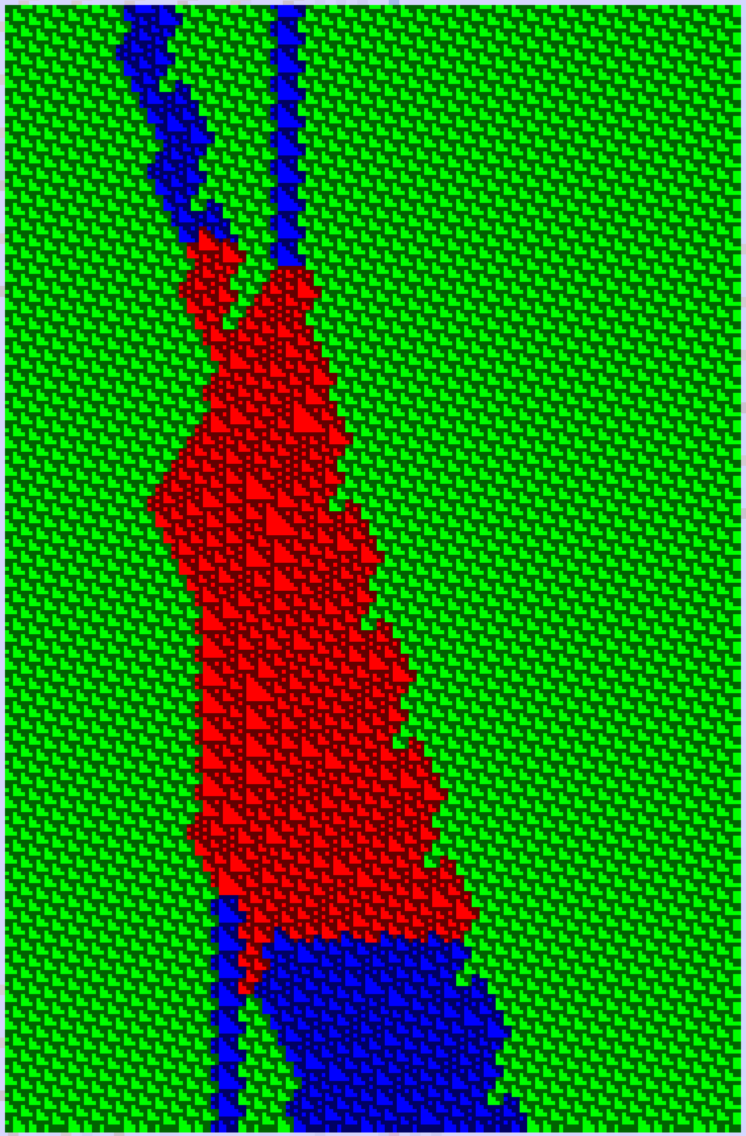


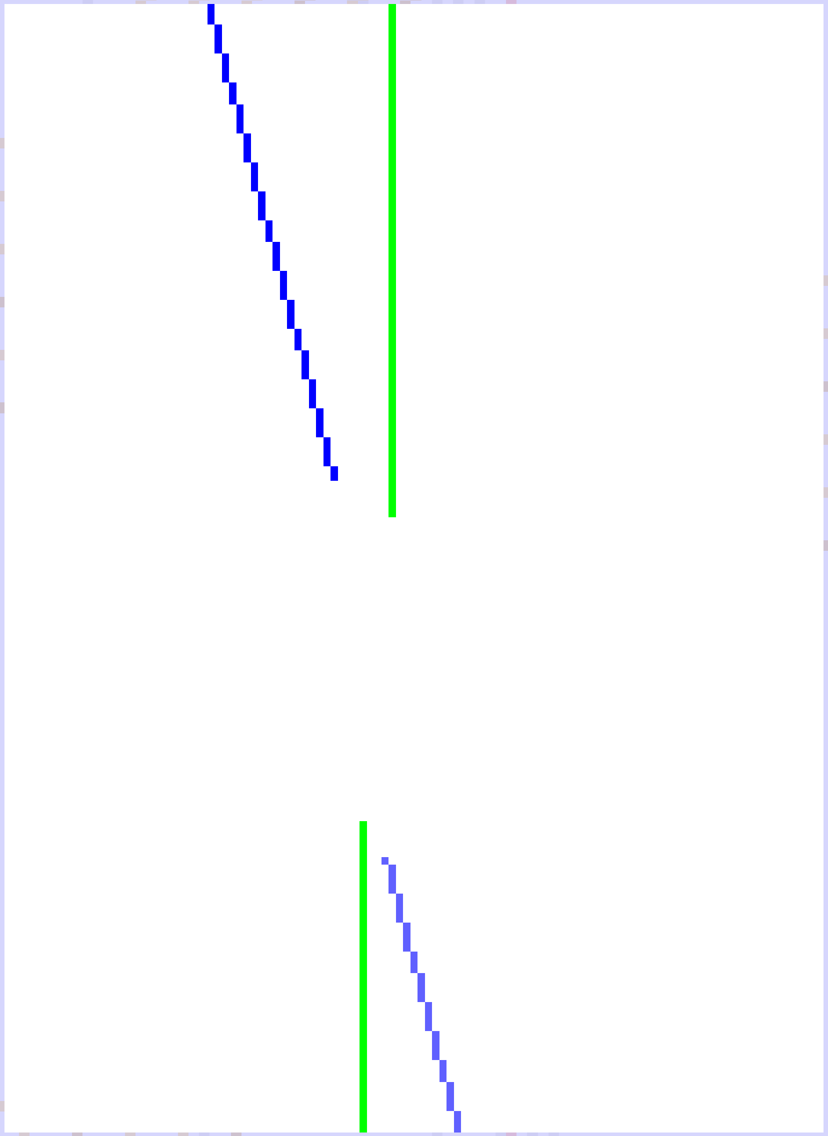


(E) redressement N B

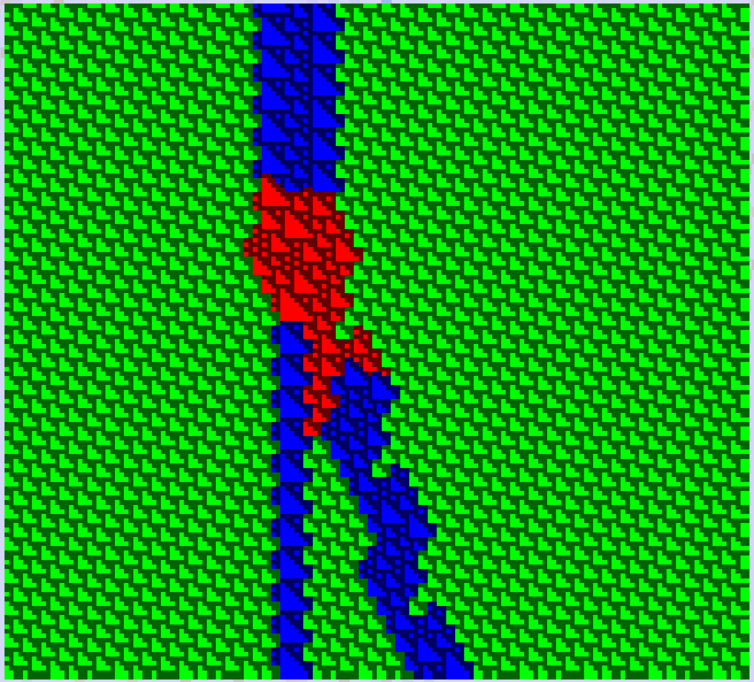


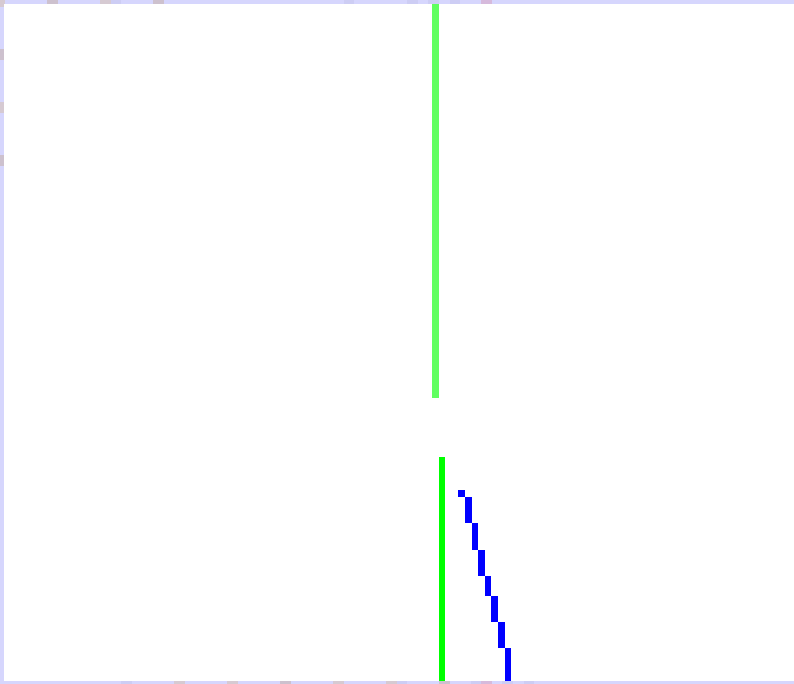
passage1



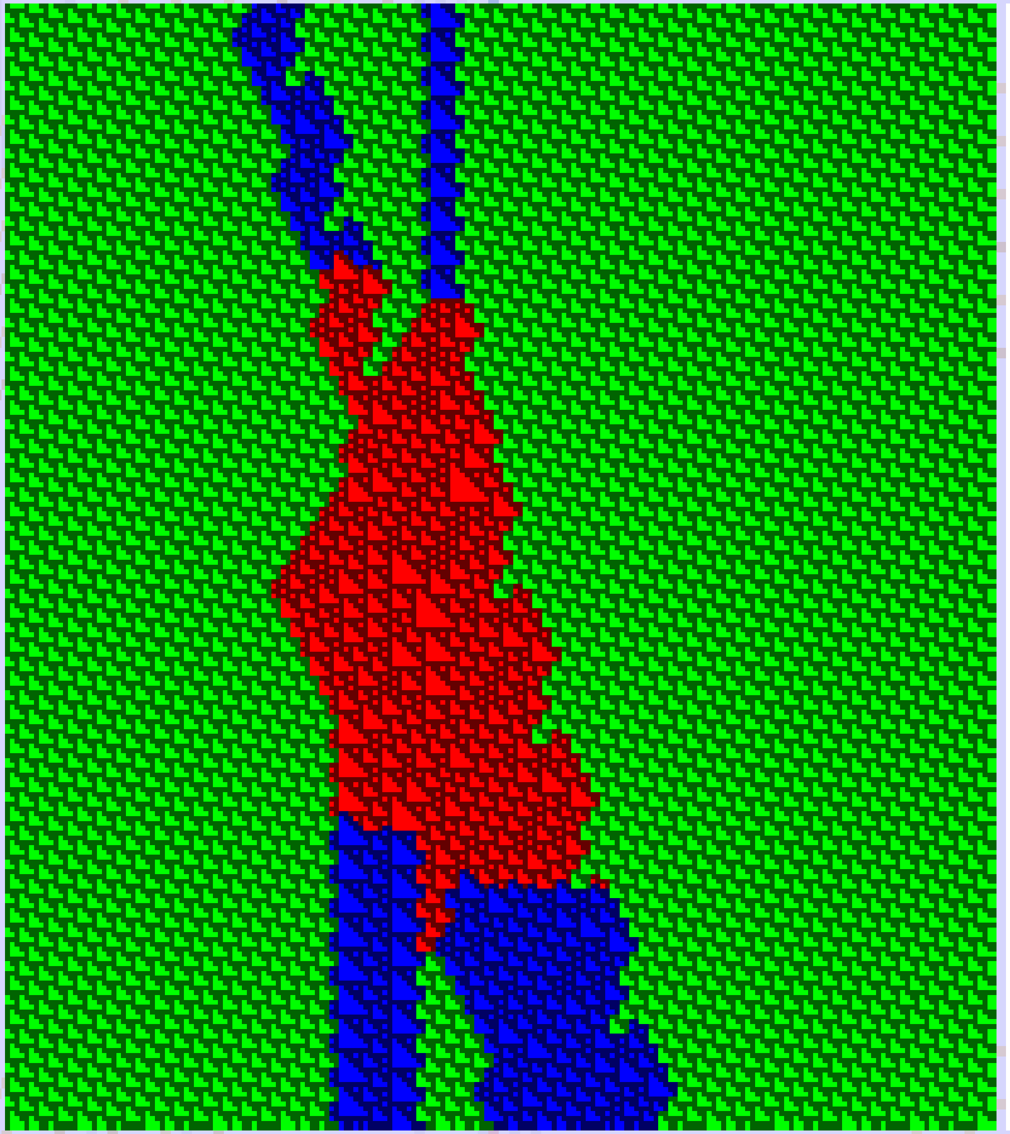


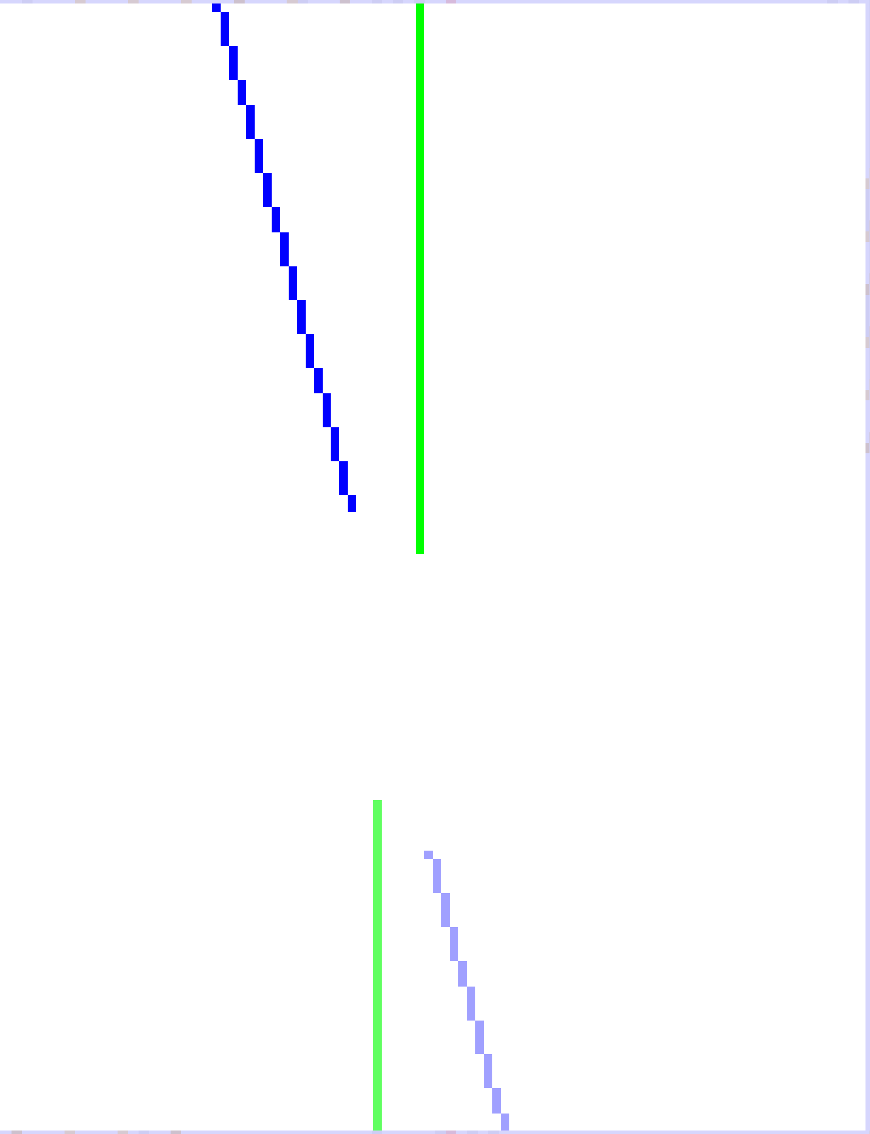
passage2a



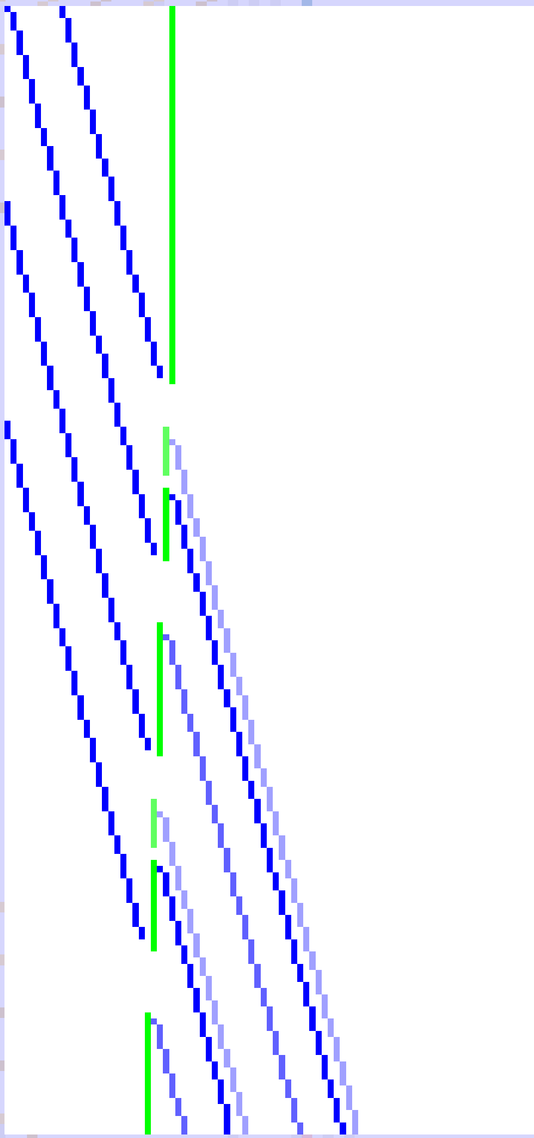


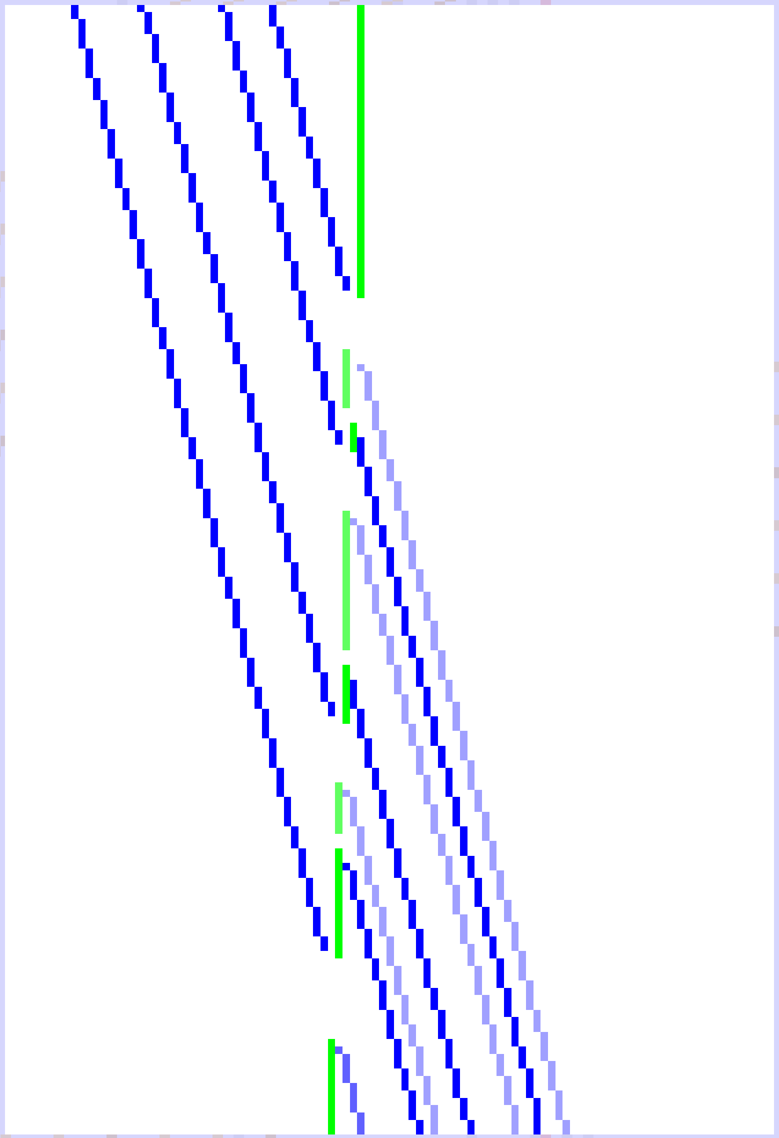
passage2b



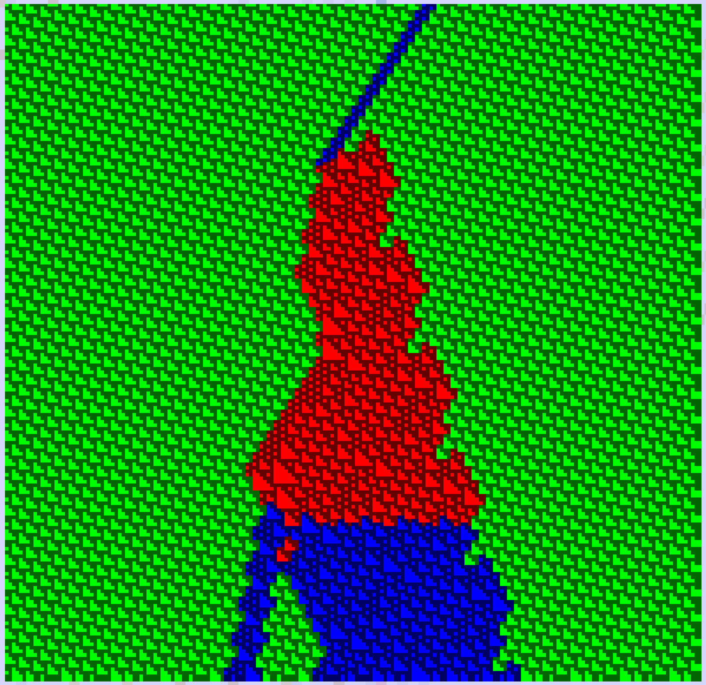


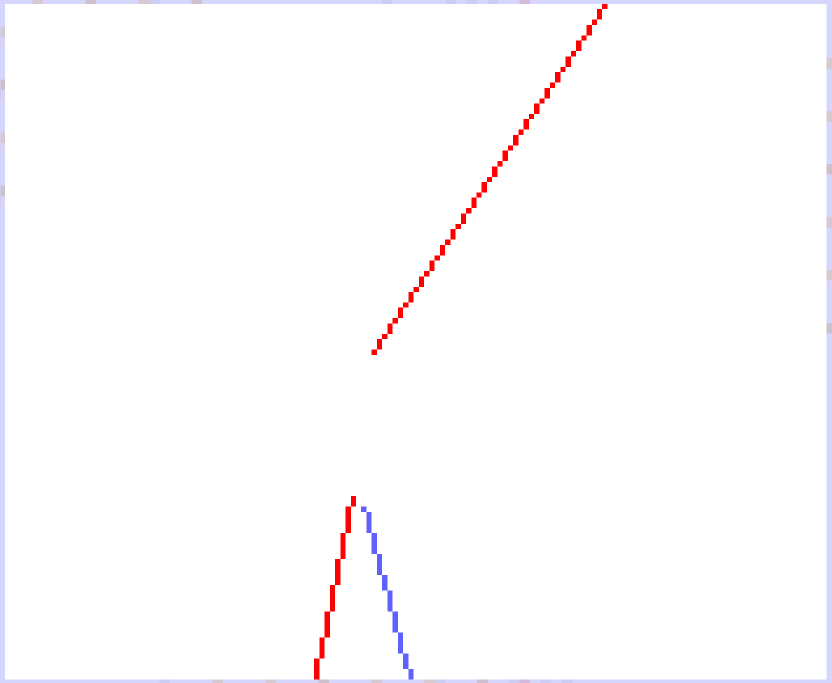
(E) passage N B



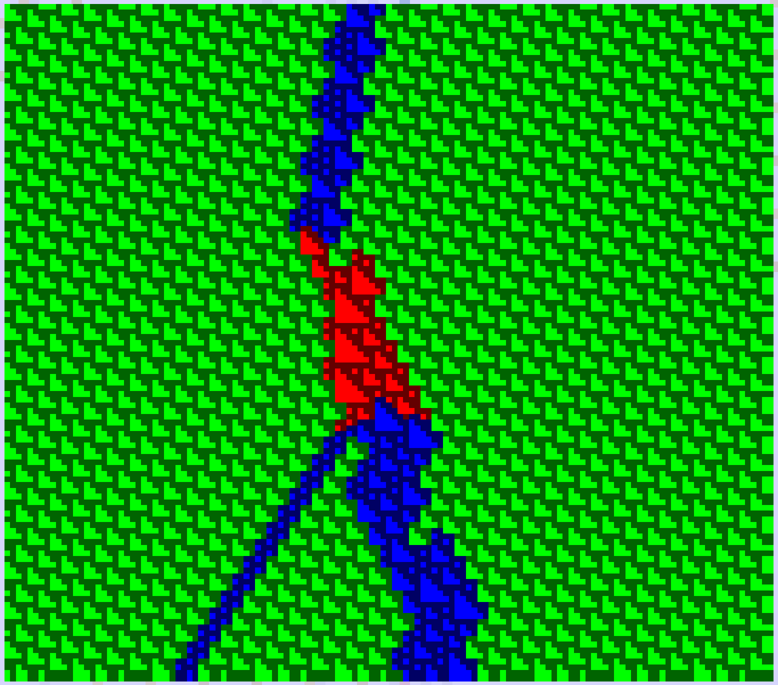


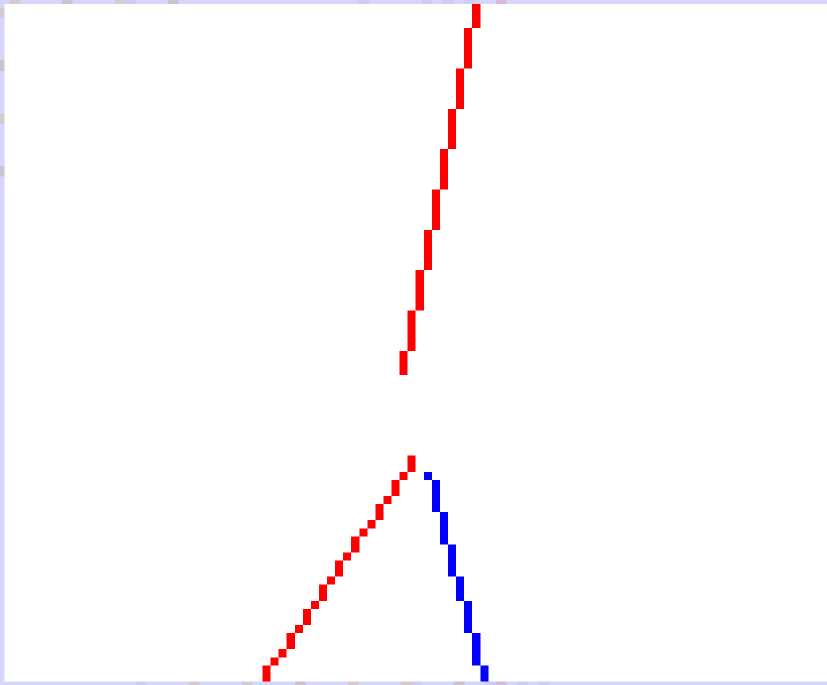
blocage1



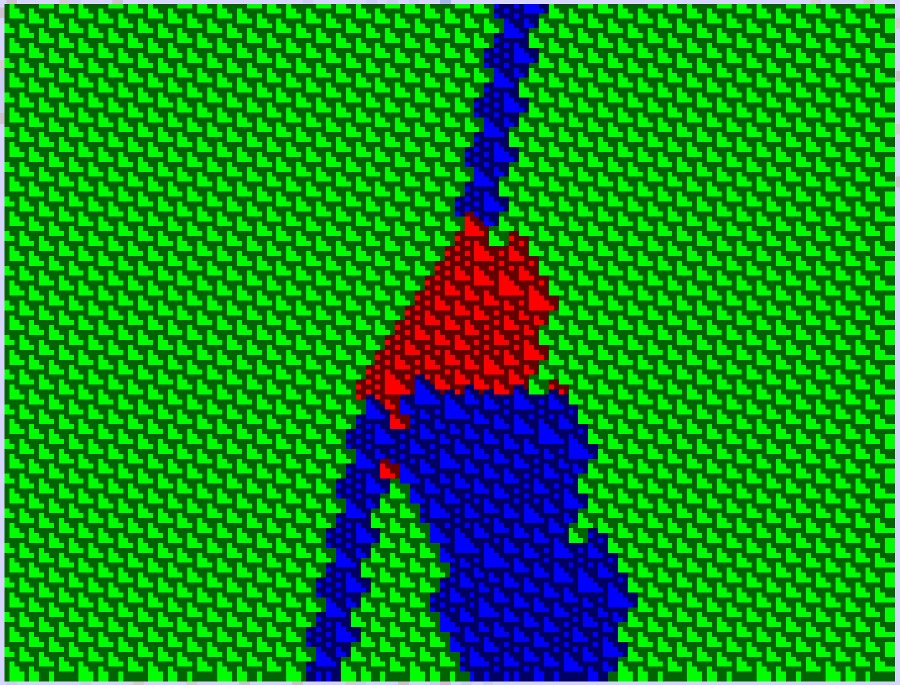


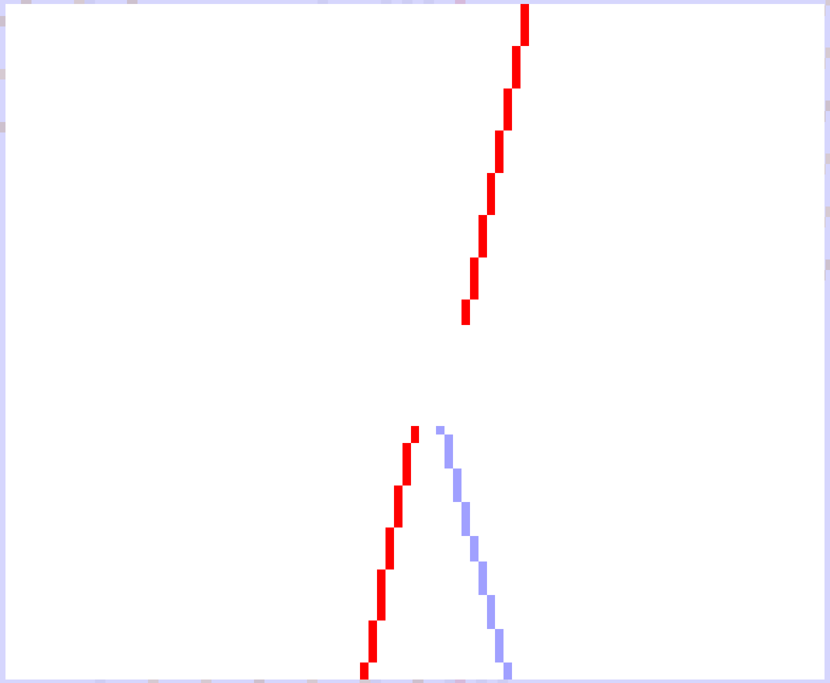
blocage2a



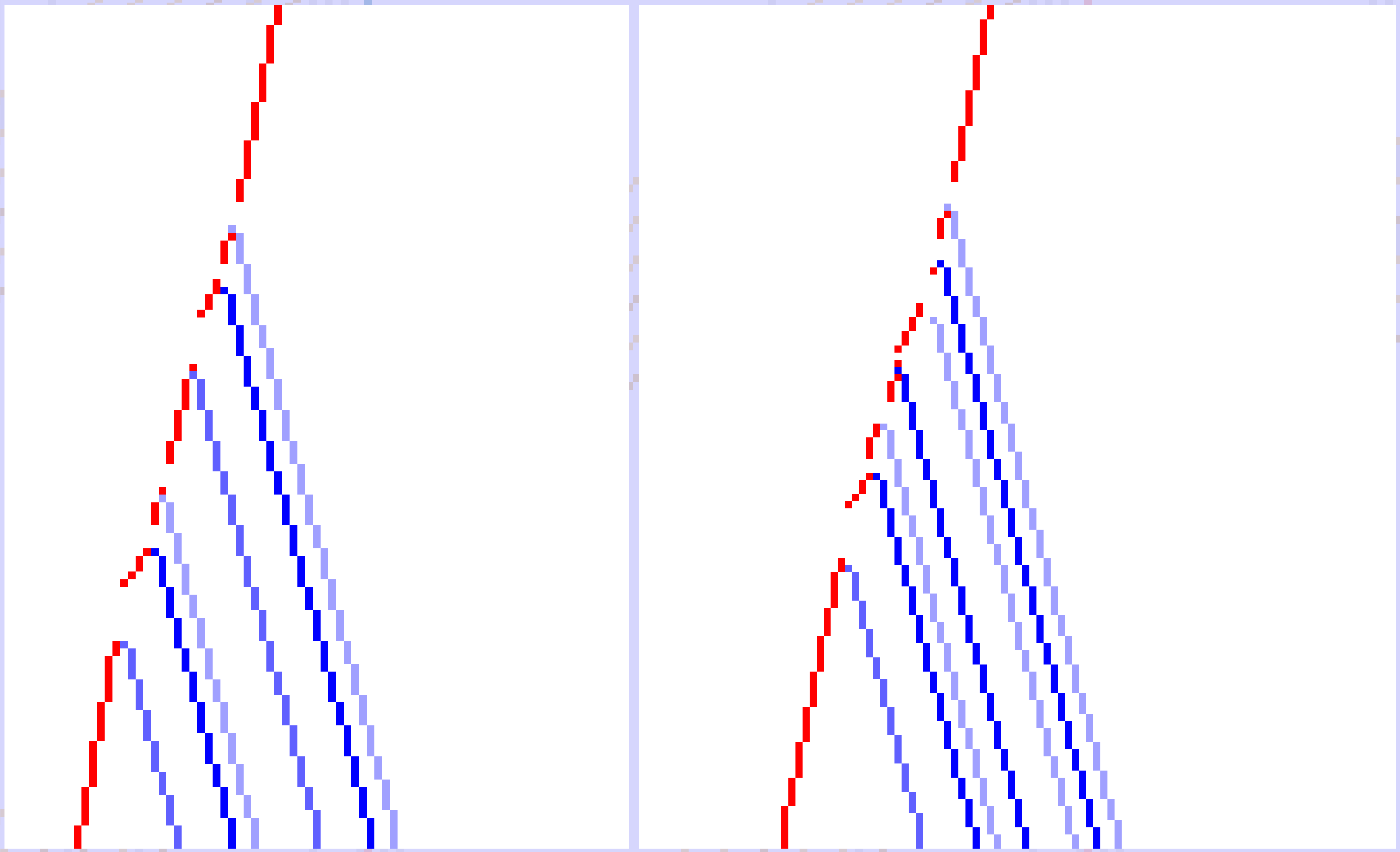


blocage2b

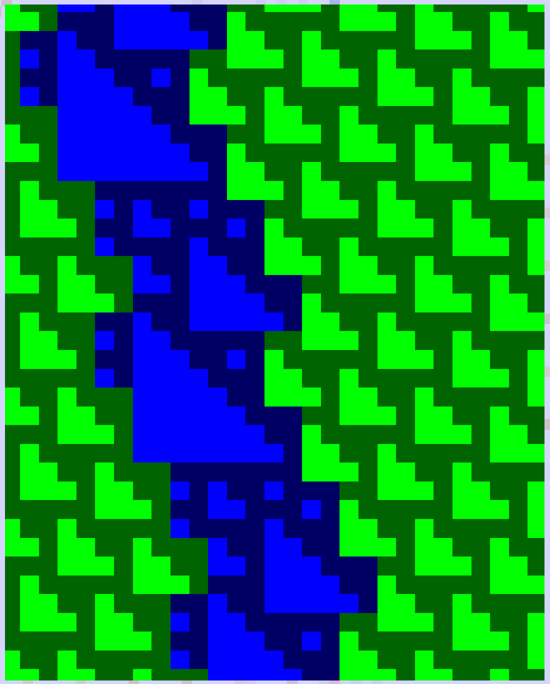


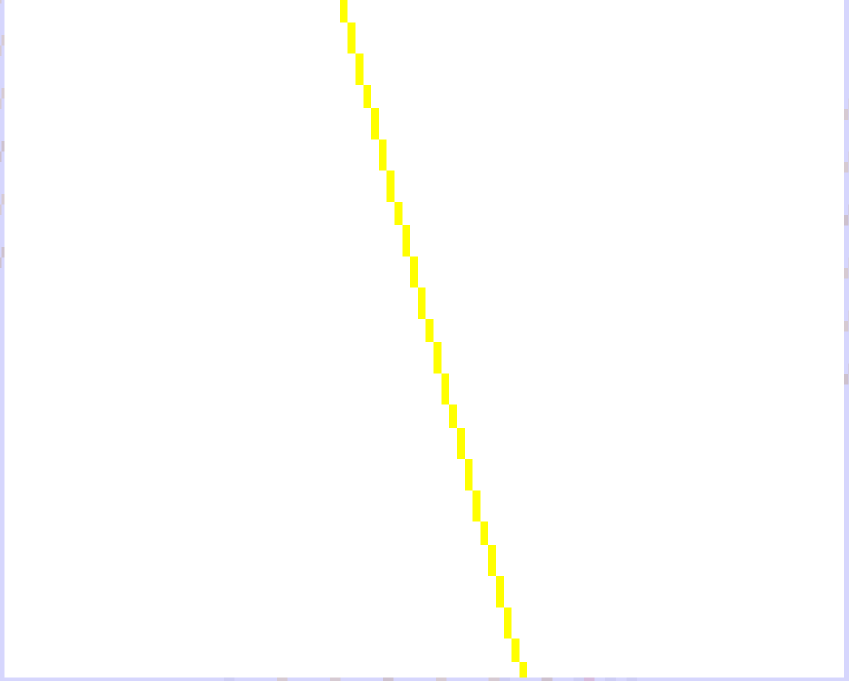


(E) blocage N B

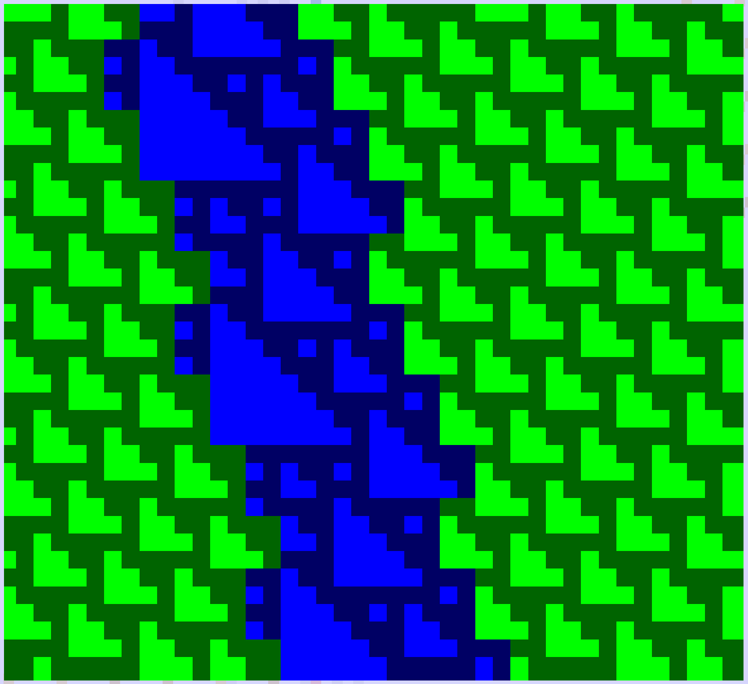


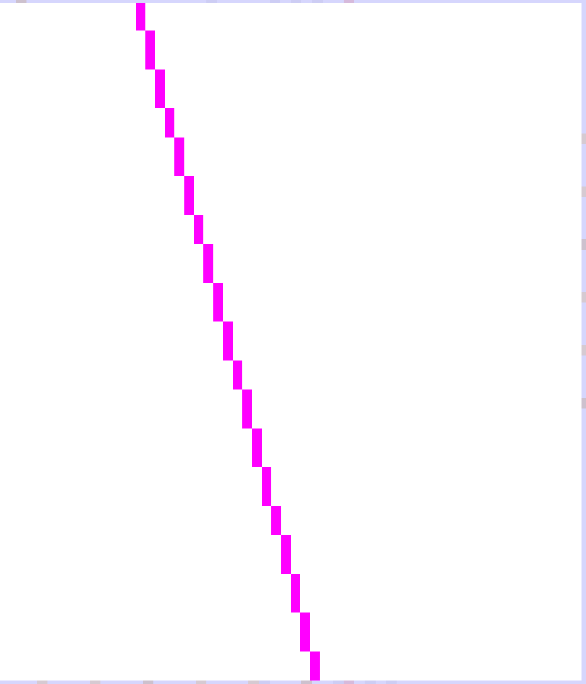
délimiteur 1



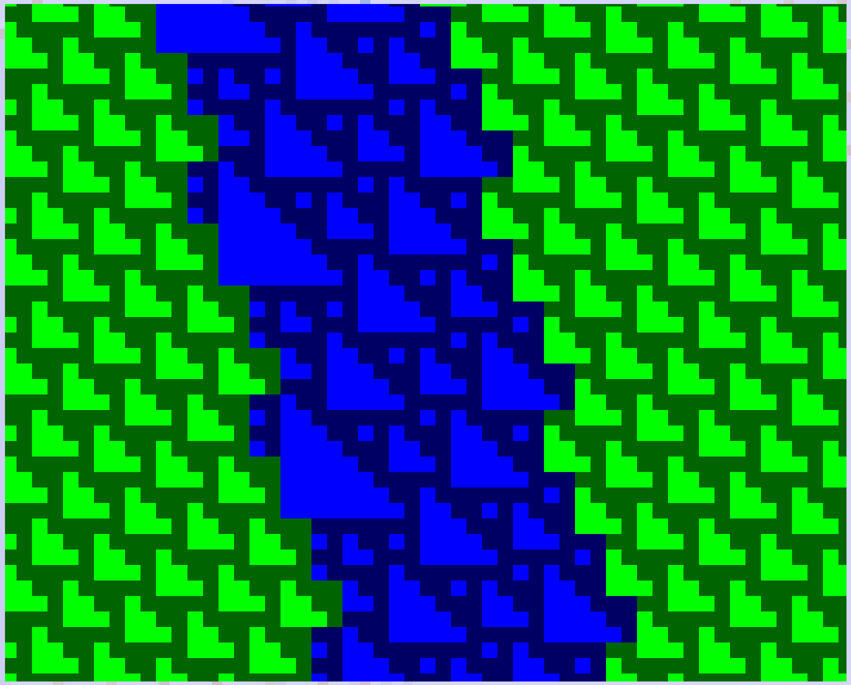


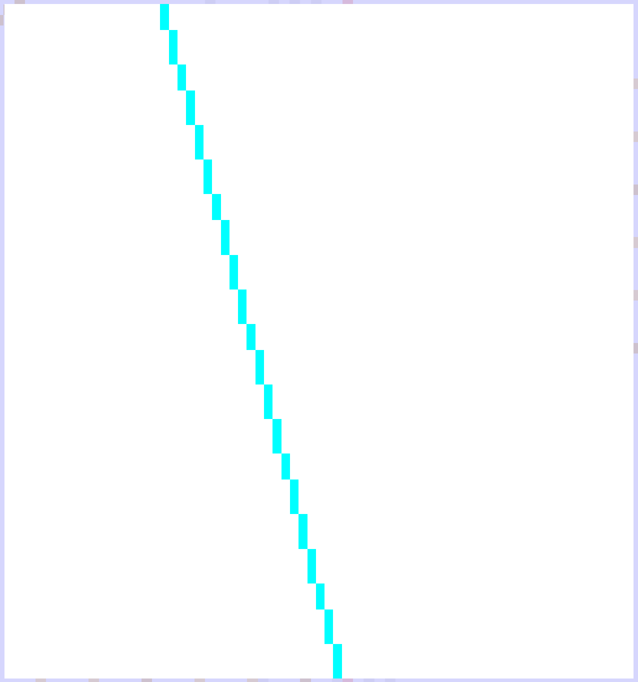
délimiteur 2



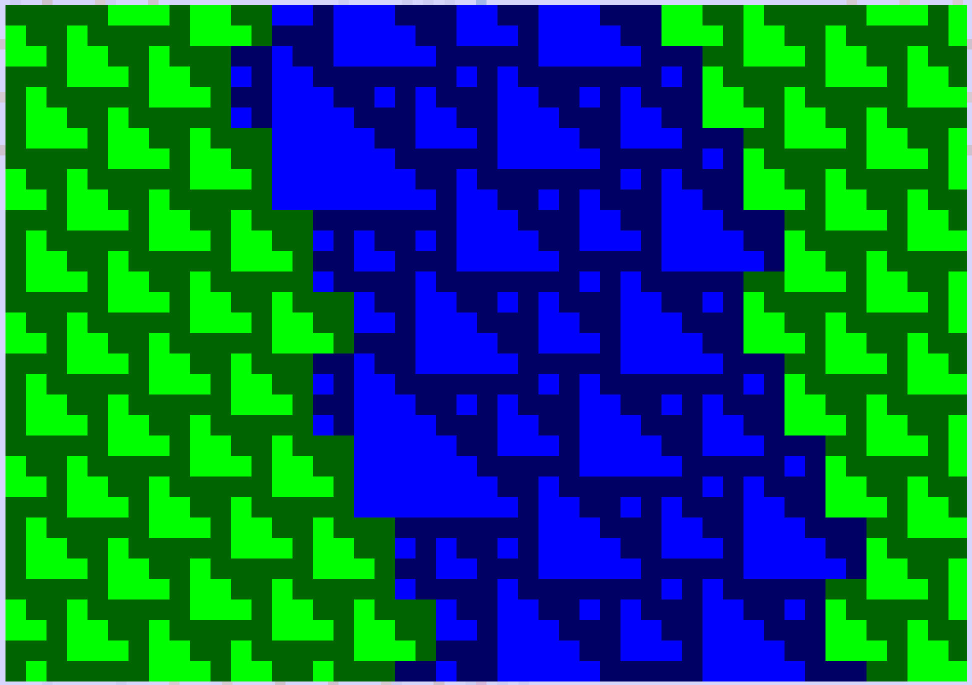


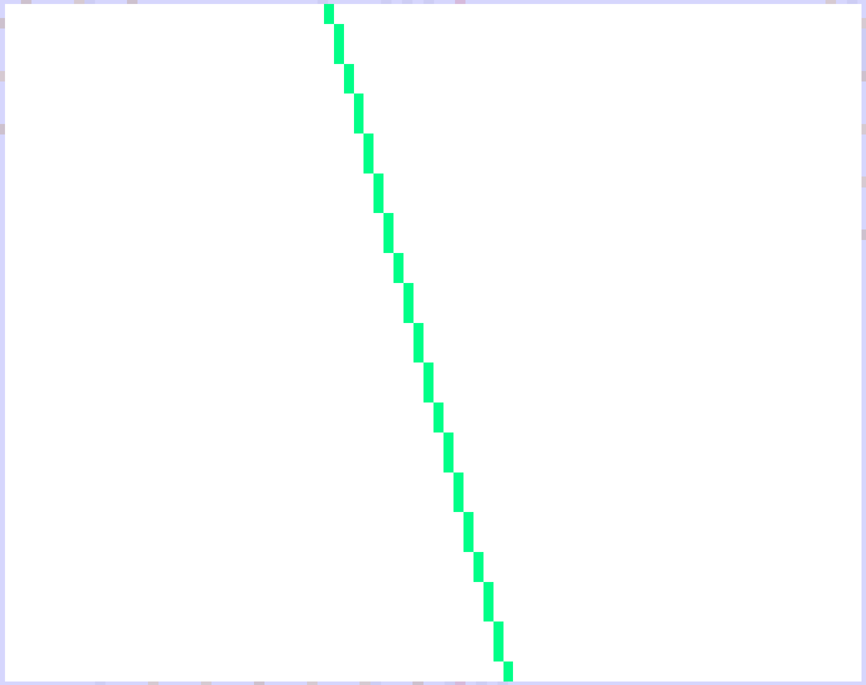
délimiteur 3



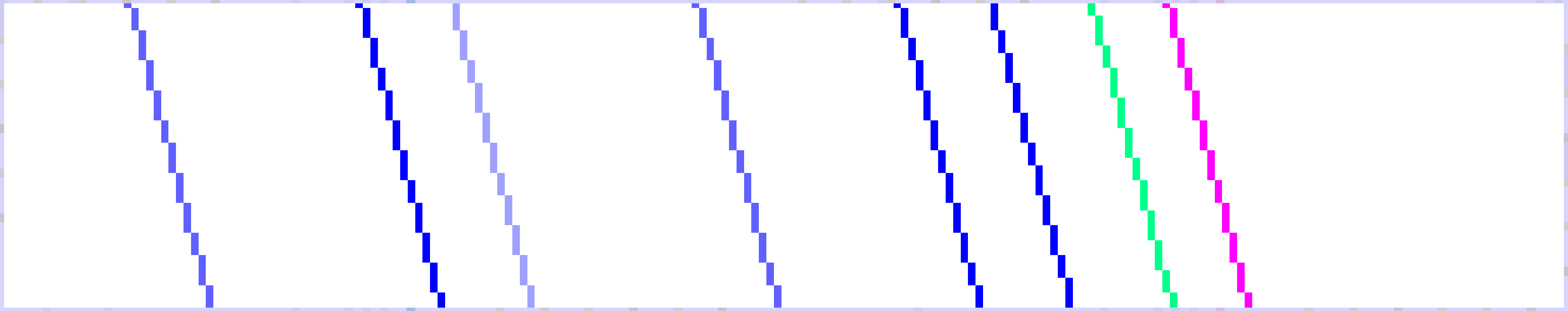


délimiteur 4

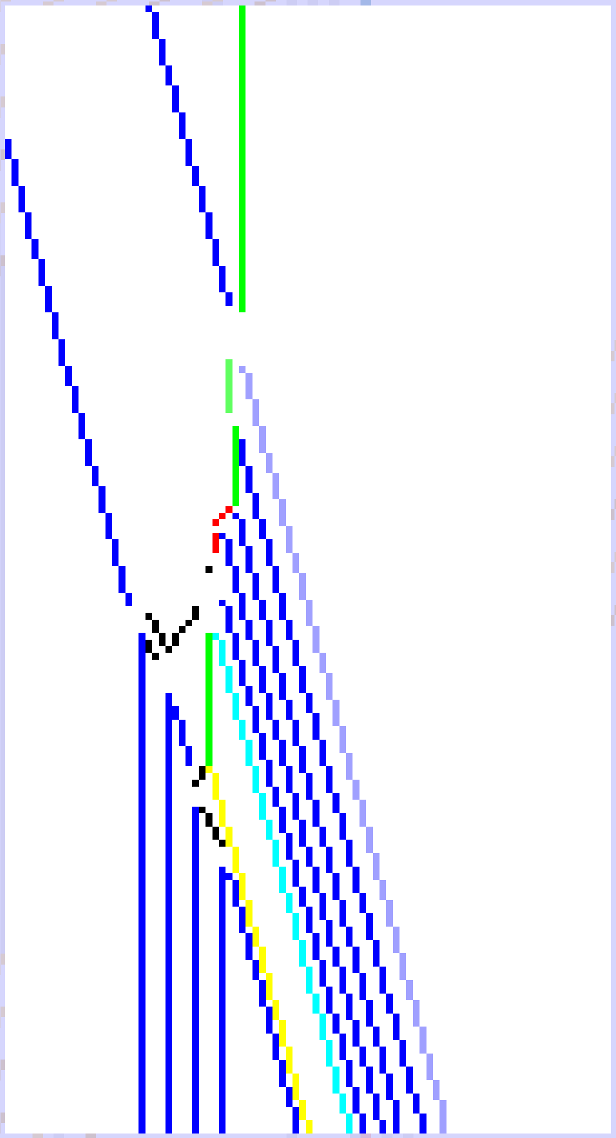


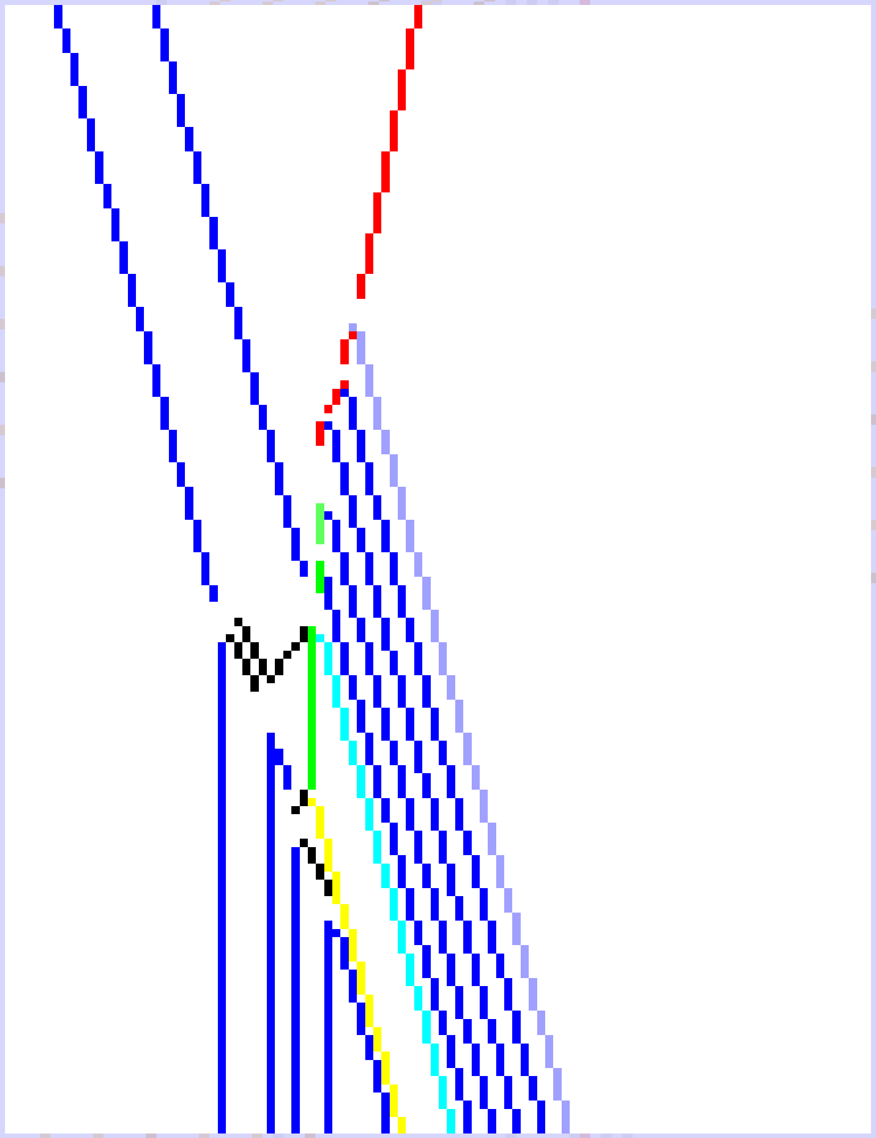


(E) pré-bits N B fin

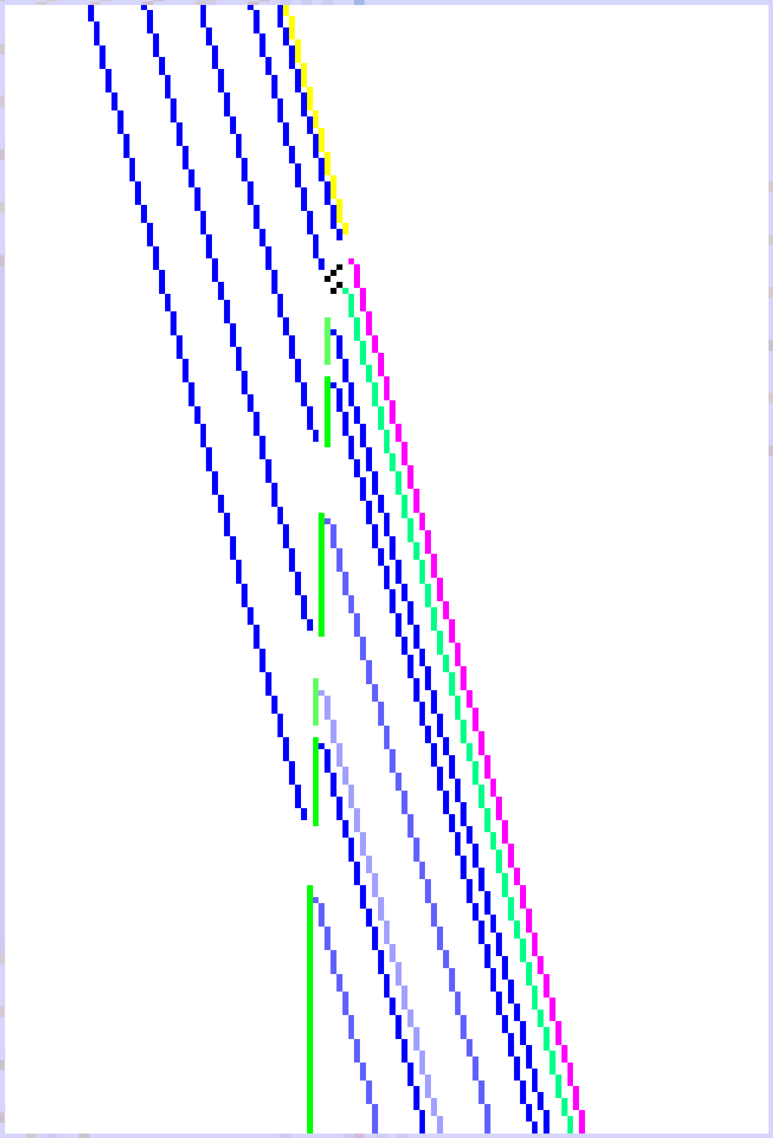


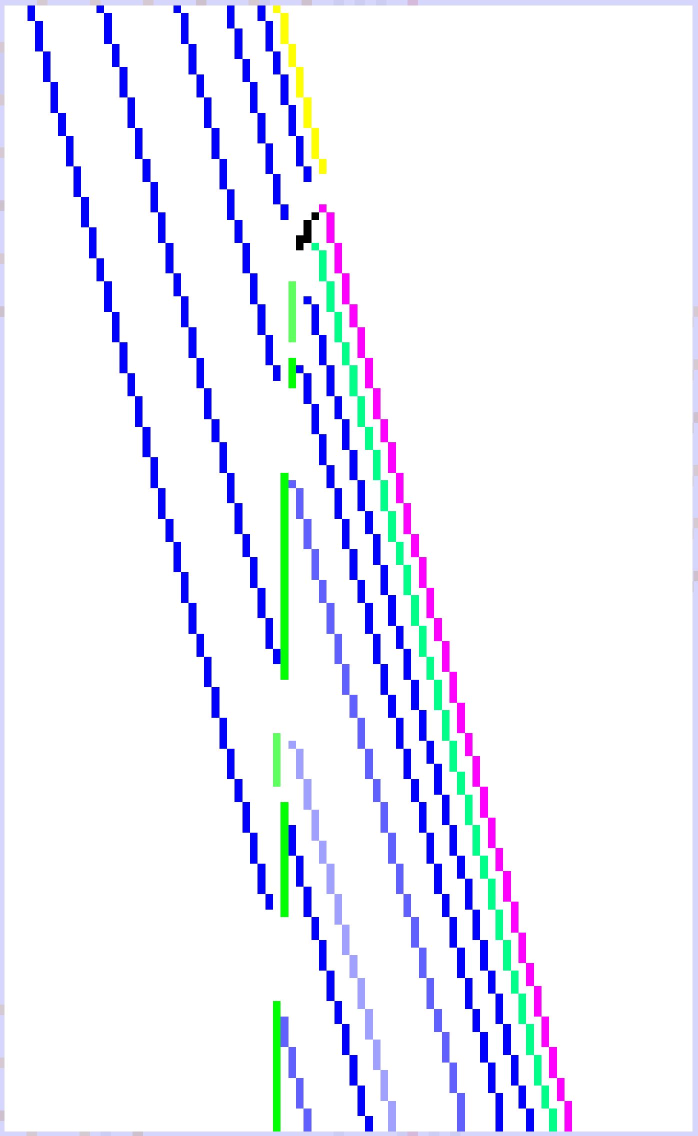
(E) analyse N B



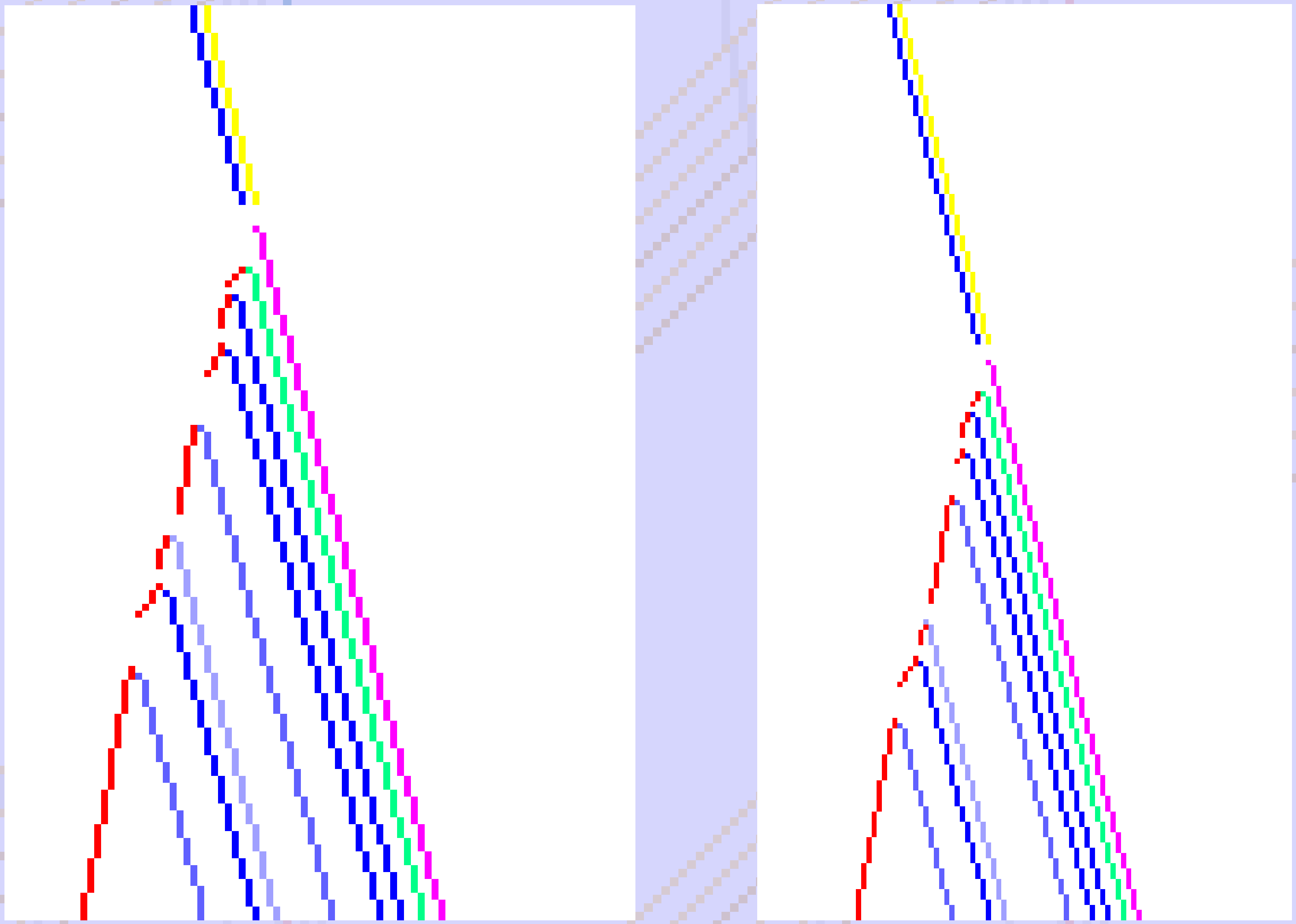


(E) passage final N B





(E) blocage final N B



Combining the gadgets

Claim Only synchronization invariants are missing.

Idea 1 Combine groups of particles.

Idea 2 Express synchronization as a big system of linear equalities and solve it.

Test Page (+ pdfTEX & Acrobat issue)

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
012345789

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
012345789

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
012345789

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
012345789