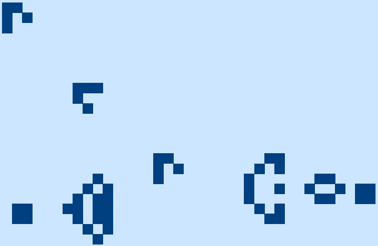
# Tutorial on Cellular Automata

Nicolas Ollinger

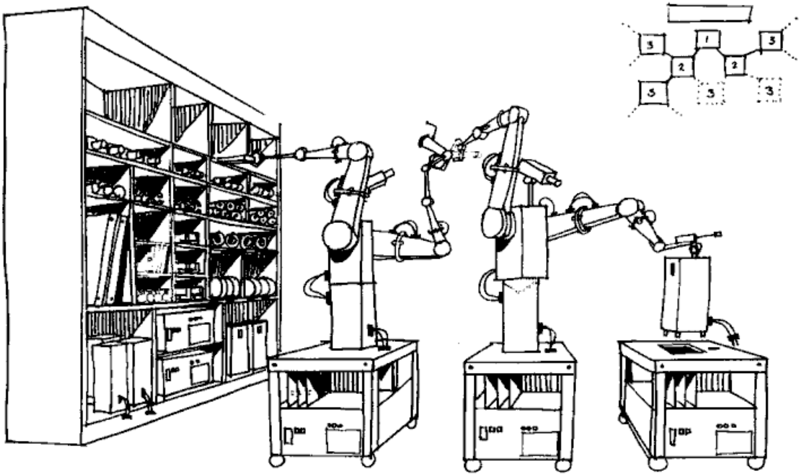
LIFO, Université d'Orléans

IRIF, CNRS et Univ. Paris Cité (sep. 2023 → aug. 2024)

**DMCS, CIRM, Marseille**
**January 2024**



# Theory of self-reproducing automata



**Cellular automata** emerged in the late 40s from the work of Ulam and von Neumann.

# Cellular Automata

A **cellular automaton** (**CA**) is a discrete dynamical model.

**Space** is discrete and consists of an infinite regular grid of cells. Each cell is described by a **state** among a common **finite set**.

**Time** is discrete. At each clock tick cells change their state **deterministically**, **synchronously** and **uniformly** according to a common **local update rule**.

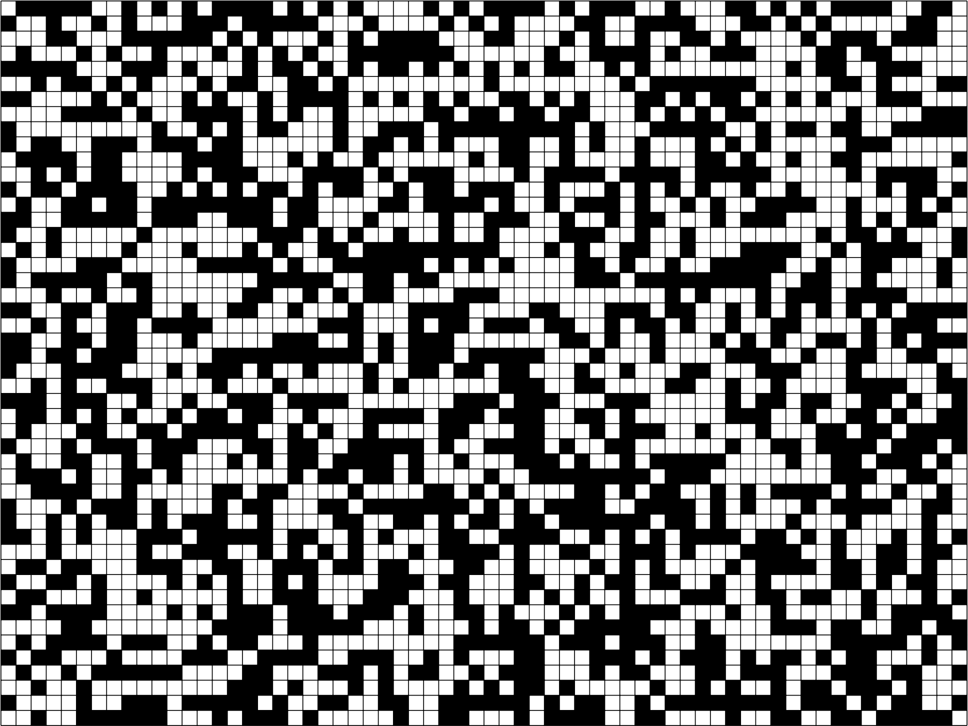
# Conway's famous Game of Life

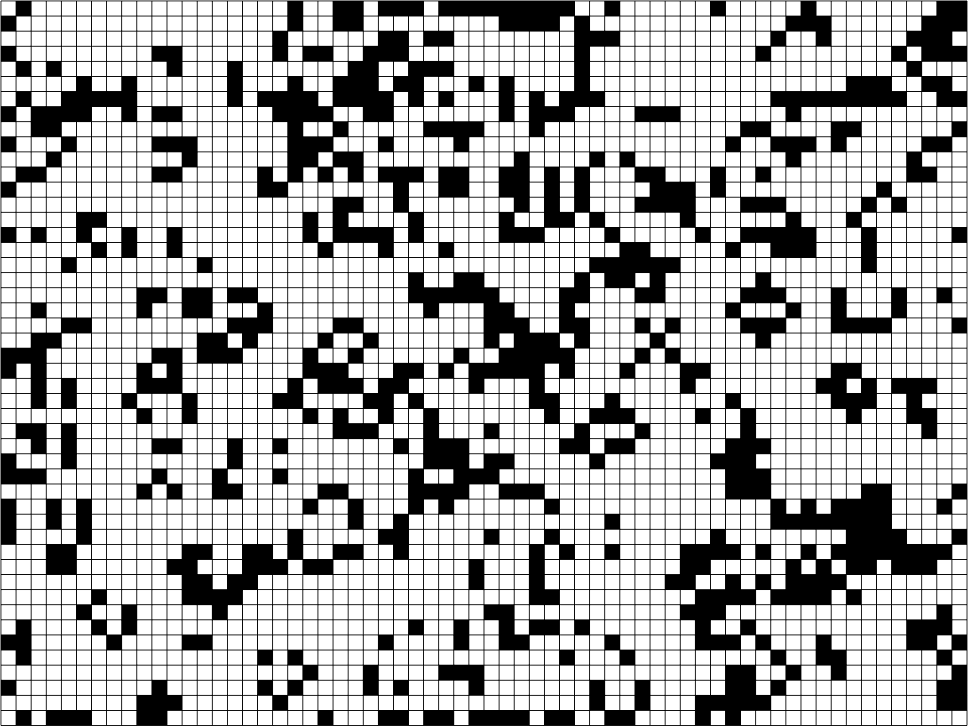
The **Game of Life** is a 2D CA invented by Conway in 1970.

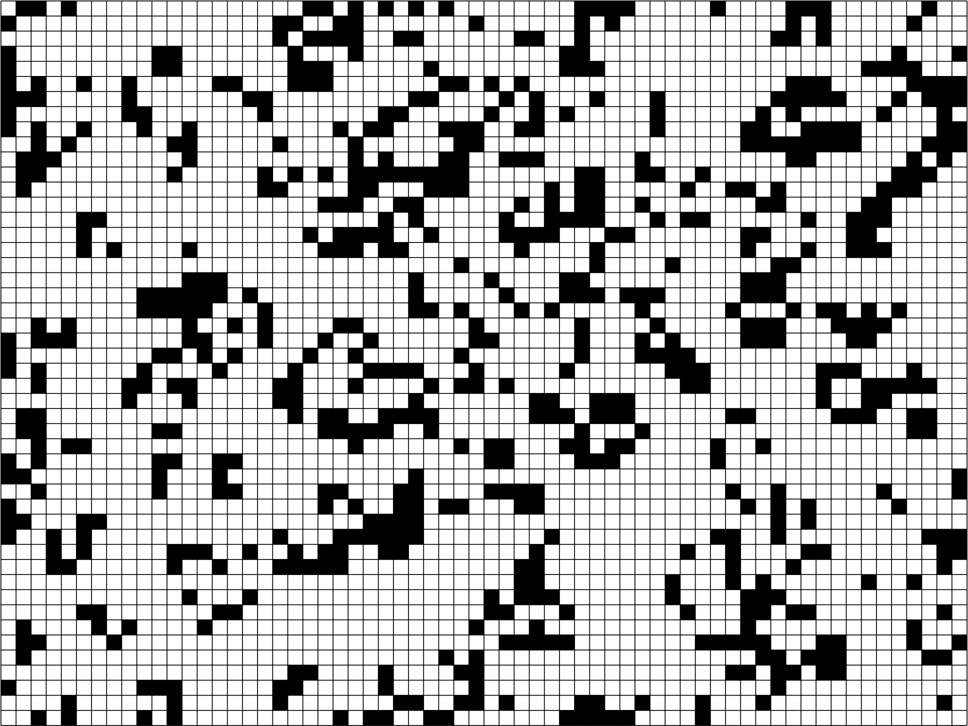
**Space** is an infinite chessboard of **alive** or **dead** cells.

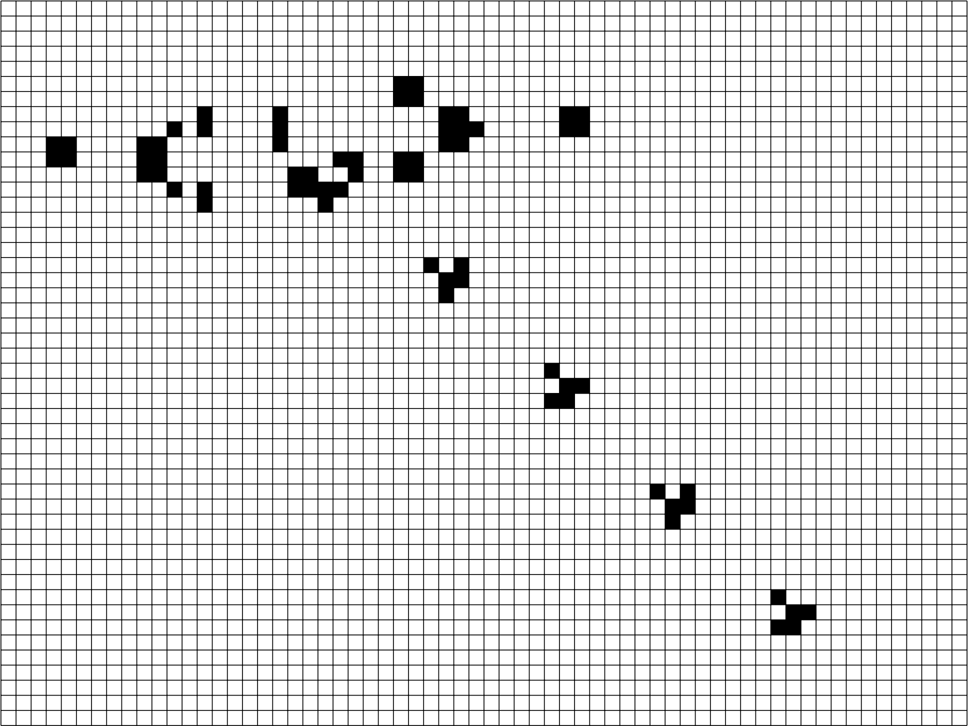
The **local update rule** counts the number of alive cells among the eight surrounding cells :

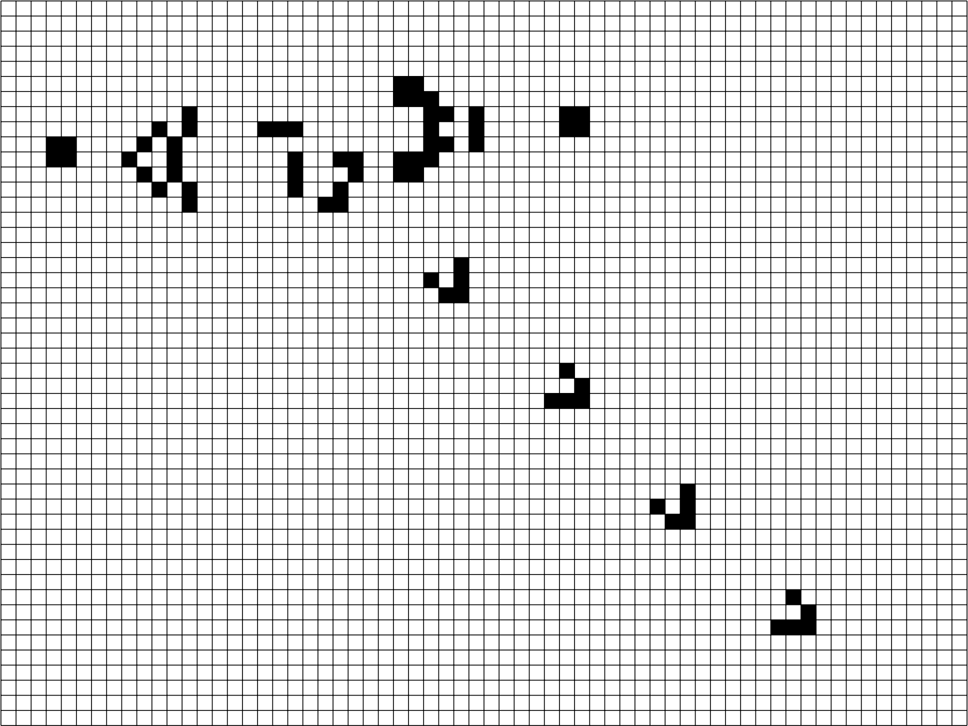
- **exactly three** alive cells give **birth** to dead cells ;
- **less than three** alive cells kill by **loneliness** ;
- **more than four** alive cells kill by **overcrowding** ;
- otherwise the cell remains in the same state.

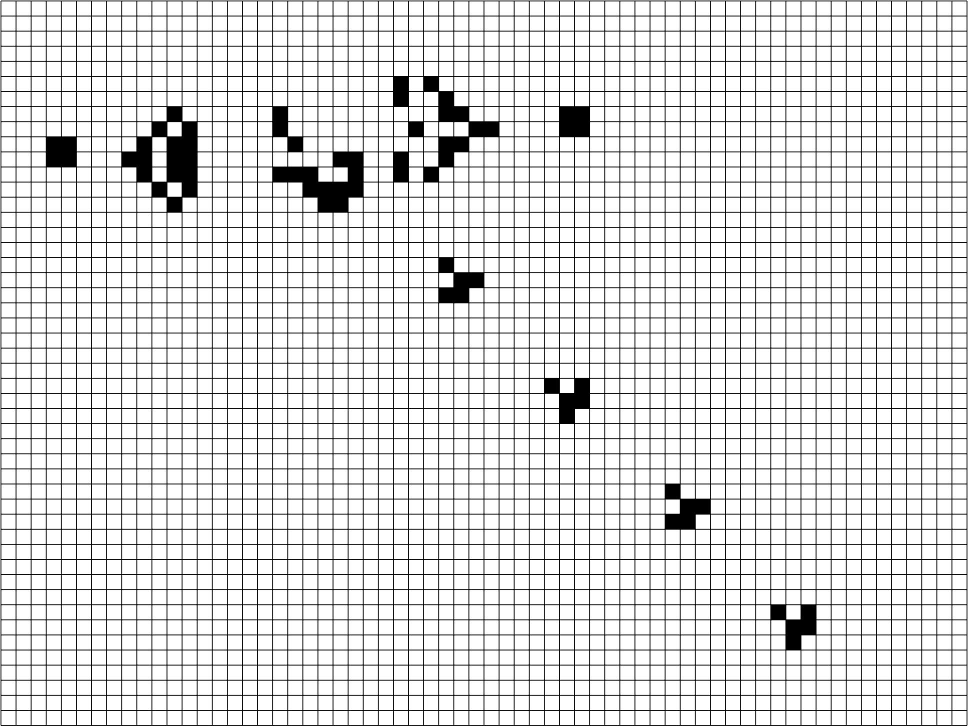


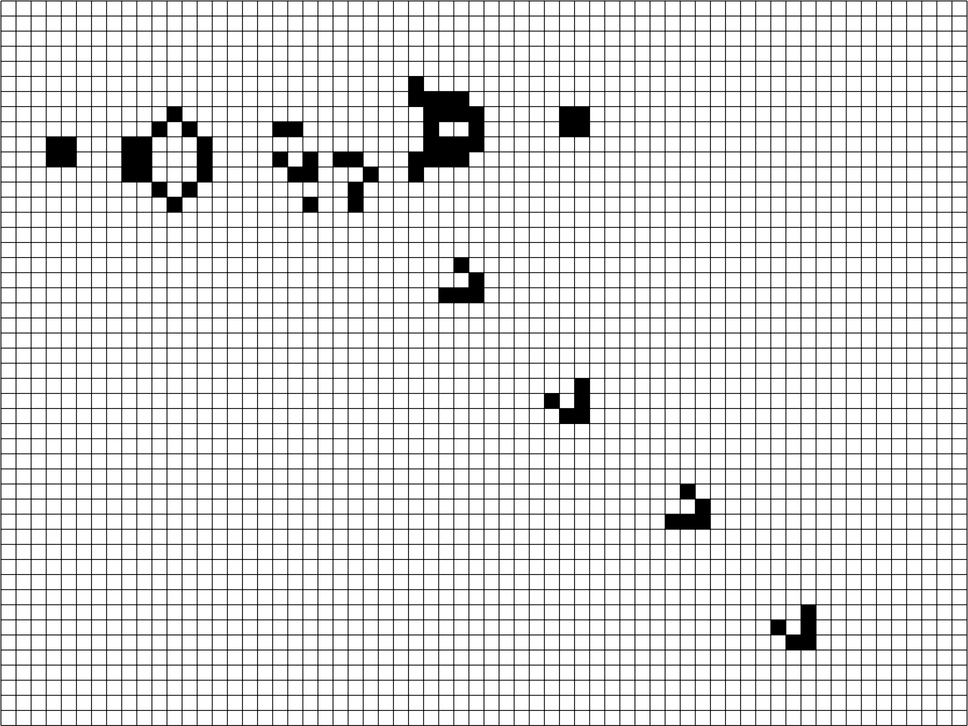


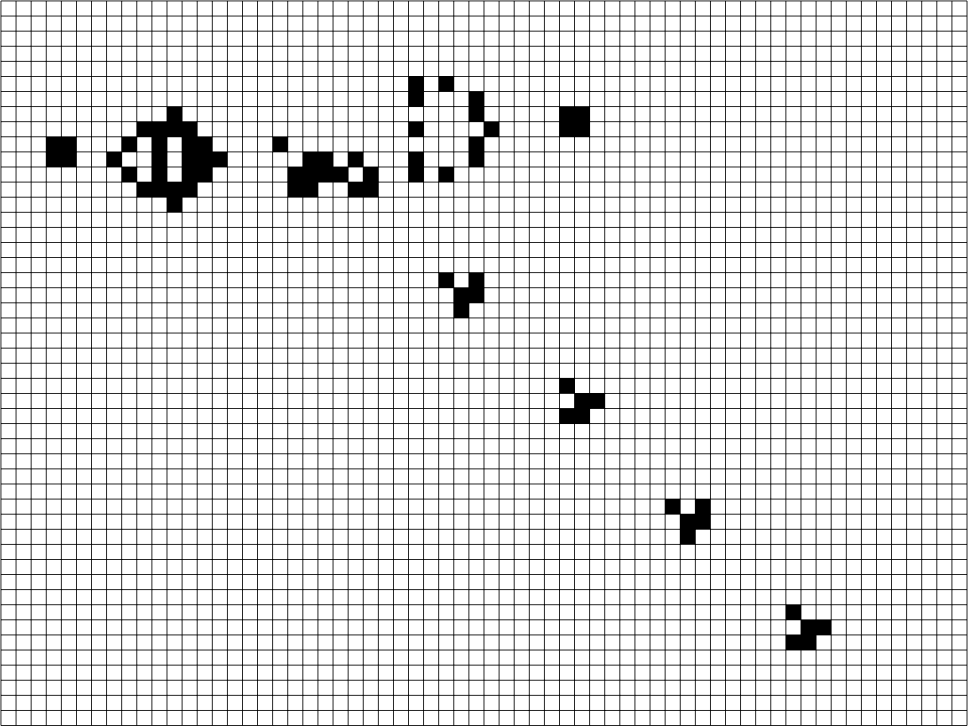


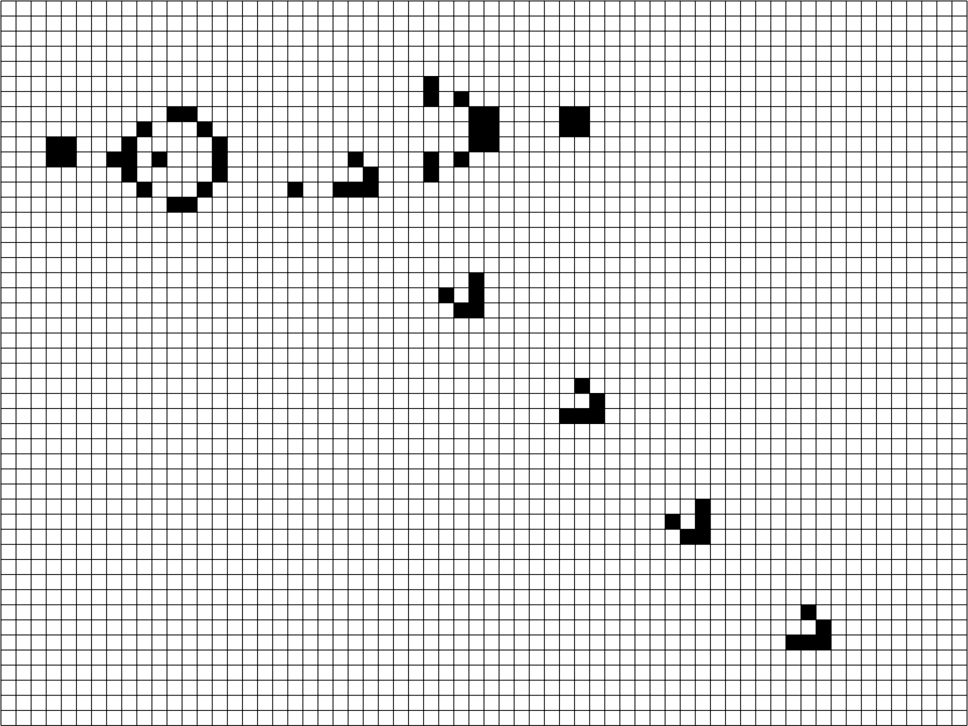


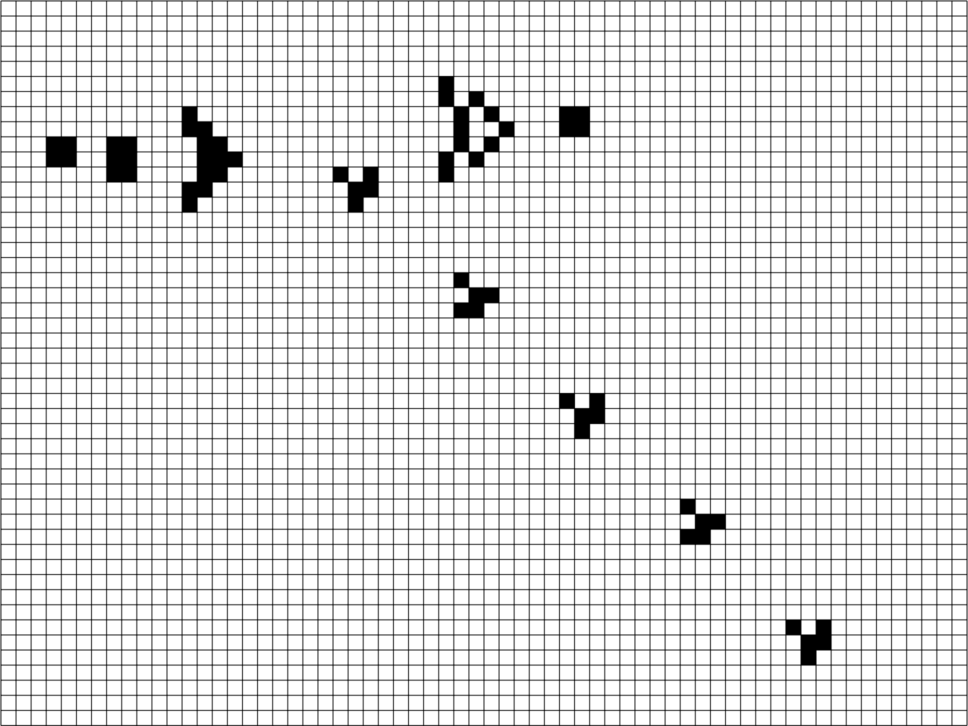


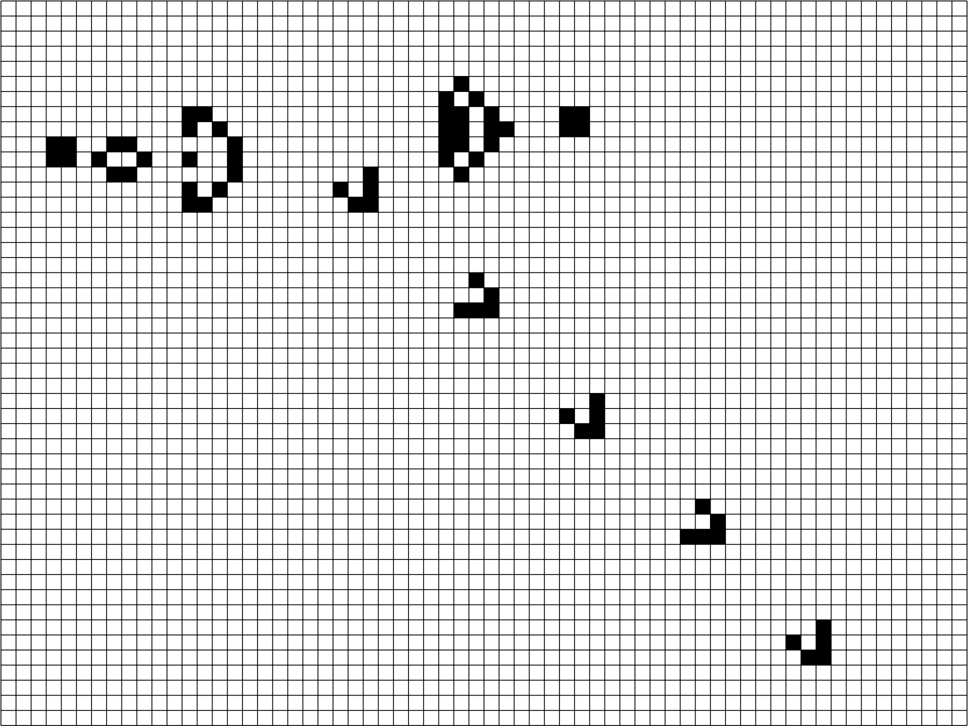








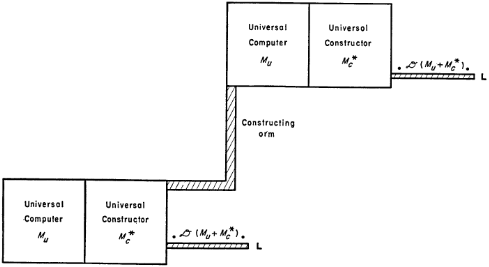




# von Neumann self-reproducing CA

| Class | Name | Symbol | | Number | Summary of transition rule |
|---|---|---|---|---|---|
| | Unexcitable | $\mathbf{U}$ | | 1 | Direct process changes $\mathbf{U}$ into sensitized states and then into $T_{u\alpha 0}$ or $C_{00}$. Reverse process kills $\mathbf{T}_{u\alpha\epsilon}$ or $\mathbf{C}_{\epsilon\epsilon'}$ into $\mathbf{U}$. |
| Ordinary | Confluent | $\mathbf{C}_{\epsilon\epsilon'}$ | $\begin{cases}\epsilon = 0 \text{ (quiescent)} \\ \epsilon = 1 \text{ (excited)}\end{cases}$ $\begin{cases}\epsilon' = 0 \text{ (next quiescent)} \\ \epsilon' = 1 \text{ (next excited)}\end{cases}$ | 4 | Receives conjunctively from $\mathbf{T}_{0\alpha\epsilon}$ directed toward it; emits with double delay to all $\mathbf{T}_{u\alpha\epsilon}$ not directed toward it. Killed to $\mathbf{U}$ by $\mathbf{T}_{1\alpha 1}$ directed toward it; killing dominates reception. |
| Ordinary | Transmission ($\mathbf{T}_{u\alpha\epsilon}$) | $\mathbf{T}_{0\alpha\epsilon}$: $\overset{0}{\rightarrow}$, $\uparrow 0$, $\overset{0}{\leftarrow}$, $\downarrow 0$ | *Class:* $u = 0$ (ordinary), $u = 1$ (special)<br>*Excitation:* $\epsilon = 0$ (quiescent), $\epsilon = 1$ (excited) | 8 | Receives disjunctively from $\mathbf{T}_{0\alpha\epsilon}$ directed toward it and from $\mathbf{C}_{\epsilon\epsilon'}$; emits in output direction with single delay (a) to $\mathbf{T}_{0\alpha\epsilon}$ not directed toward it and to $\mathbf{C}_{\epsilon\epsilon'}$ (b) to $\mathbf{U}$ or sensitized states by direct process (c) to kill $\mathbf{T}_{1\alpha\epsilon}$ by reverse process. Killed to $\mathbf{U}$ by $\mathbf{T}_{1\alpha 1}$ directed toward it; killing dominates reception. |
| Special | Transmission ($\mathbf{T}_{u\alpha\epsilon}$) | $\mathbf{T}_{1\alpha\epsilon}$: $\overset{1}{\rightarrow}$, $\uparrow 1$, $\overset{1}{\leftarrow}$, $\downarrow 1$ | *Output direction:* $\alpha = 0$ (right), $\alpha = 1$ (up), $\alpha = 2$ (left), $\alpha = 3$ (down) | 8 | Receives disjunctively from $\mathbf{T}_{1\alpha\epsilon}$ directed toward it and from $\mathbf{C}_{\epsilon\epsilon'}$; emits in output direction with single delay (a) to $\mathbf{T}_{1\alpha\epsilon}$ not directed toward it (b) to $\mathbf{U}$ or sensitized states by direct process (c) to kill $\mathbf{T}_{0\alpha\epsilon}$ or $\mathbf{C}_{\epsilon\epsilon'}$ by by reverse process. Killed to $\mathbf{U}$ by $\mathbf{T}_{0\alpha 1}$ directed toward it; killing dominates reception. |
| Special | Sensitized | $\mathbf{S}_{\emptyset}$, $\mathbf{S}_{0}$, $\mathbf{S}_{1}$, $\mathbf{S}_{00}$, $\mathbf{S}_{01}$, $\mathbf{S}_{10}$, $\mathbf{S}_{11}$, $\mathbf{S}_{000}$ | | 8 | These are intermediary states in the direct process. $\mathbf{T}_{u\alpha 1}$ directed toward $\mathbf{U}$ converts it to $\mathbf{S}_{\emptyset}$. Thereafter, $\mathbf{S}_{\Sigma}$ is followed by (a) $\mathbf{S}_{\Sigma 1}$ if some $\mathbf{T}_{u\alpha 1}$ is directed toward the cell (b) $\mathbf{S}_{\Sigma 0}$ otherwise, until the direct process terminates in a $\mathbf{T}_{u\alpha 0}$ or $\mathbf{C}_{00}$. See Figure 10. |

A **29 states** CA with **von Neumann** neighborhood with wires and construction/destruction abilities.

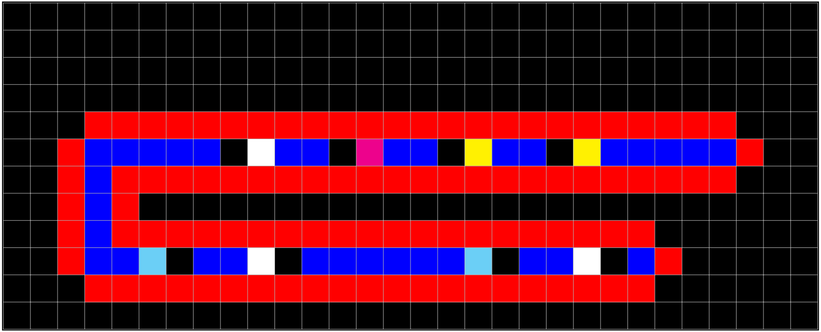



**Self-reproduction** using **Universal Computer** + **Universal Constructor**.

*(Theory of Self-Reproducing Automata, edited by Burks, 1966)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

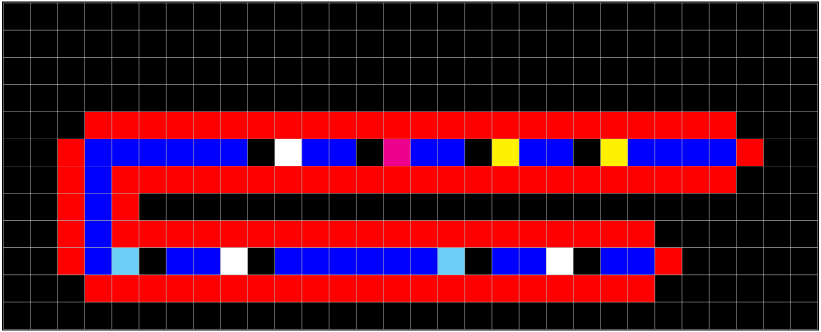


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

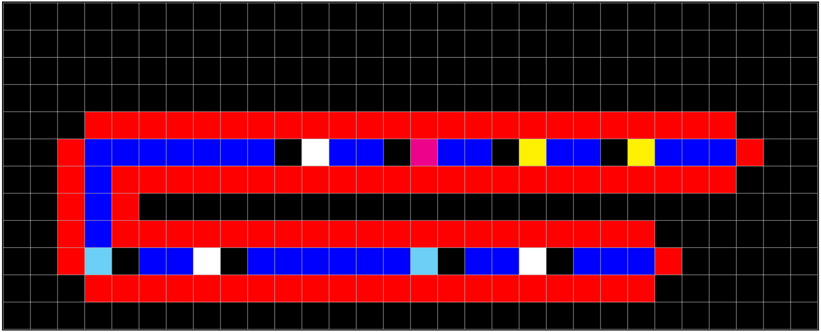


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

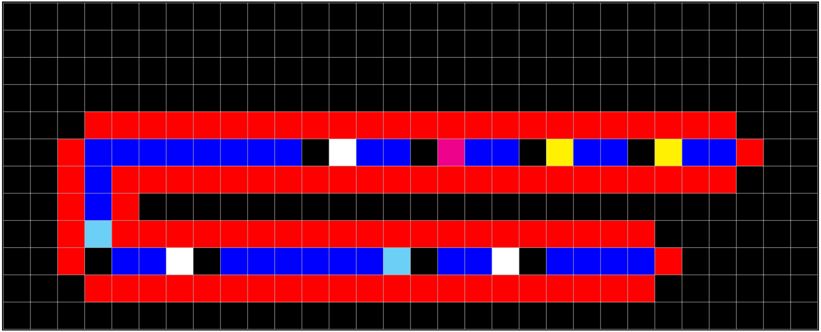


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

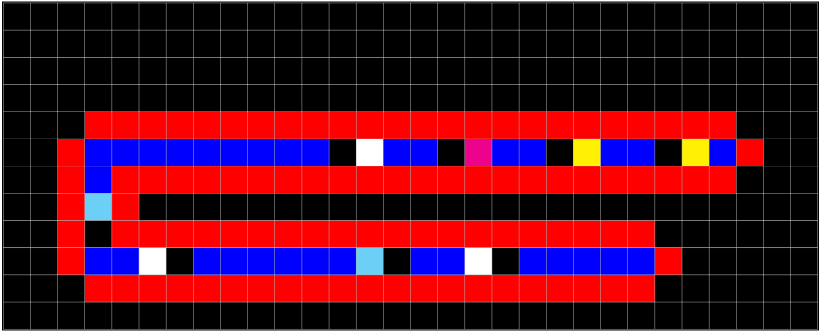


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

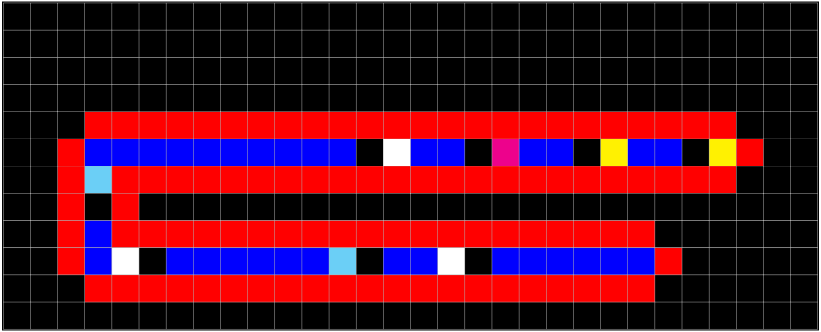


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

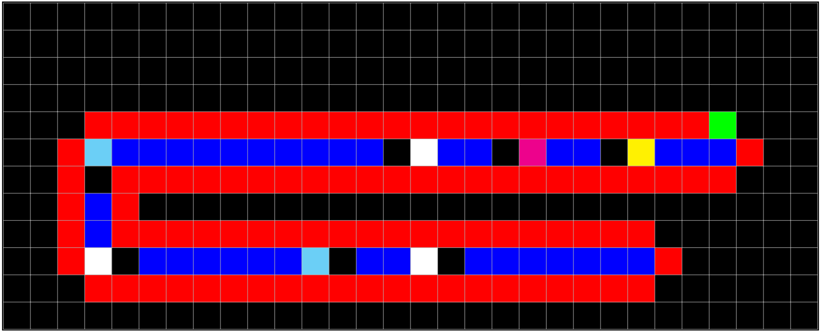


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

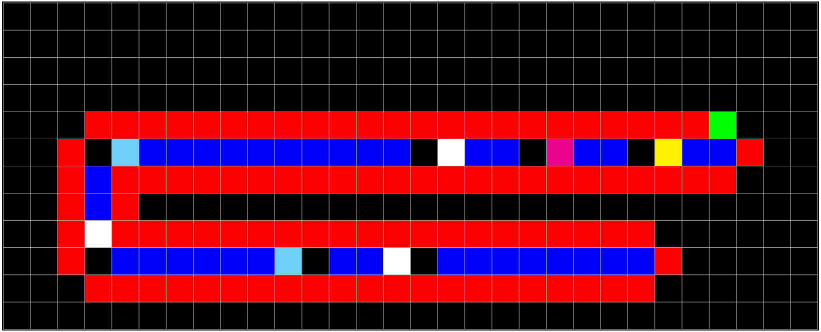


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

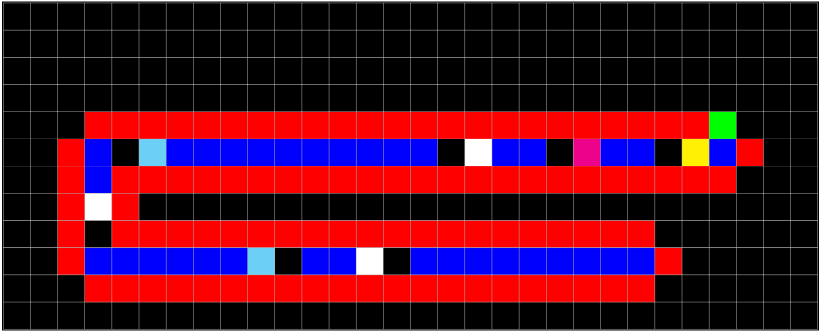


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

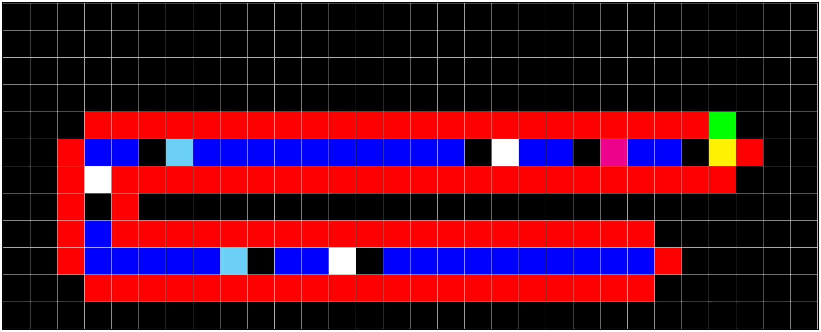


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

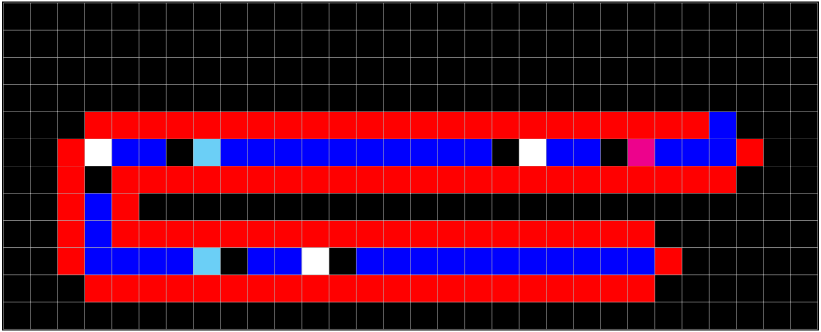


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

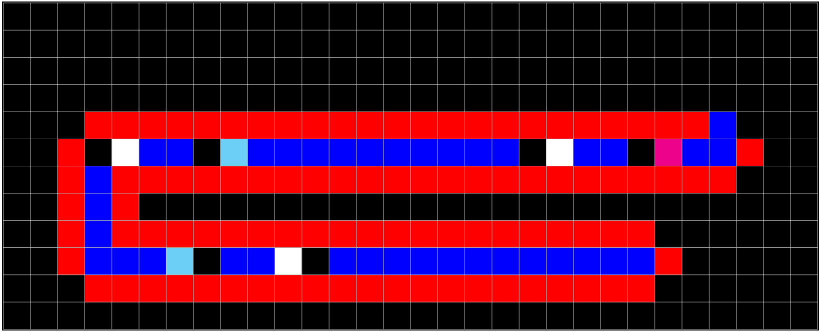


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

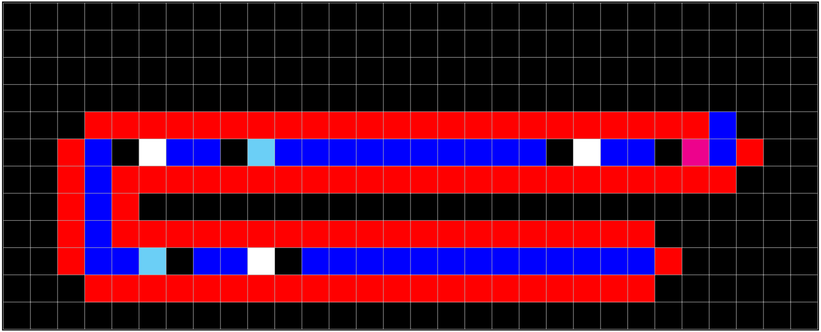


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

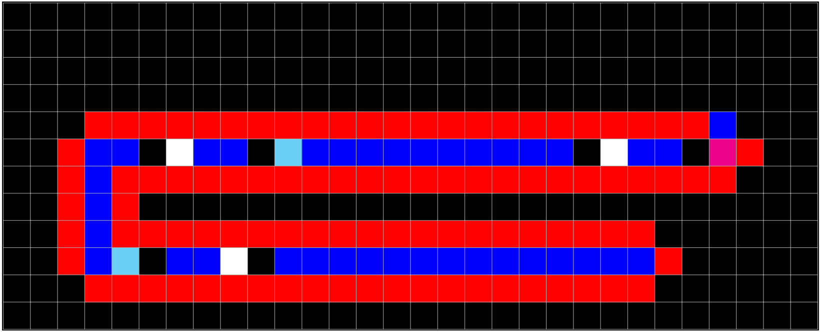


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

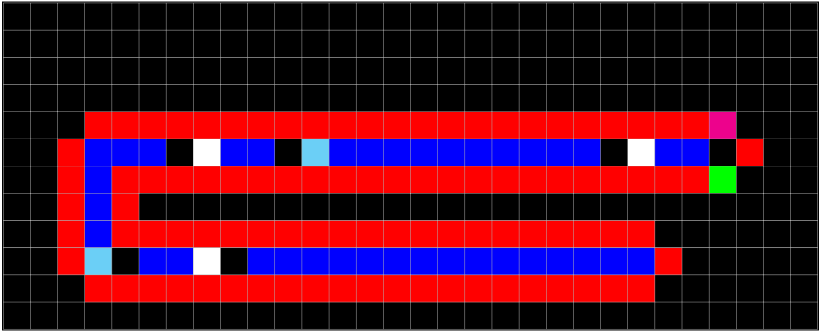


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

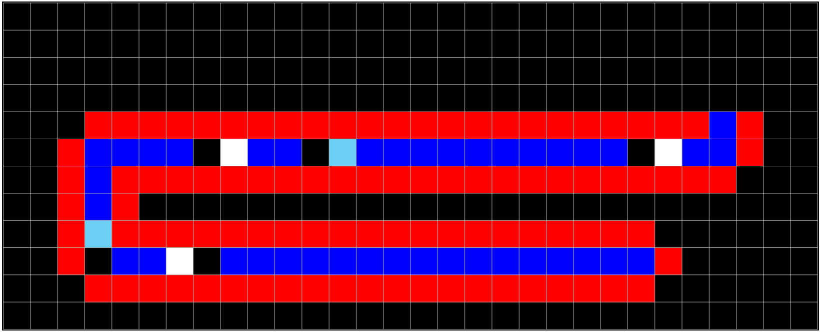


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

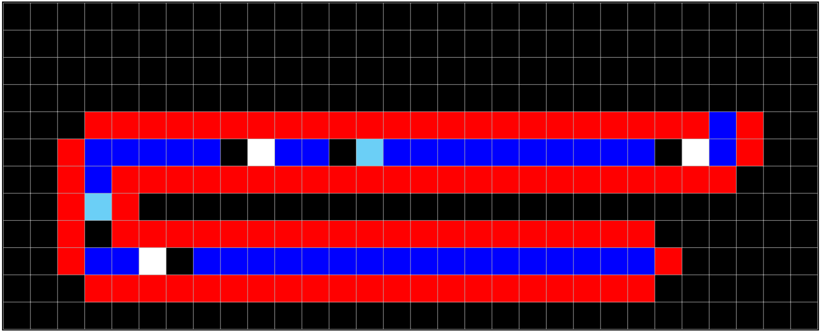


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

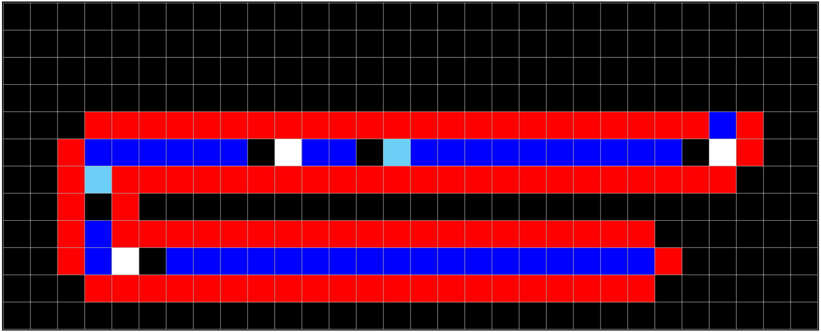


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

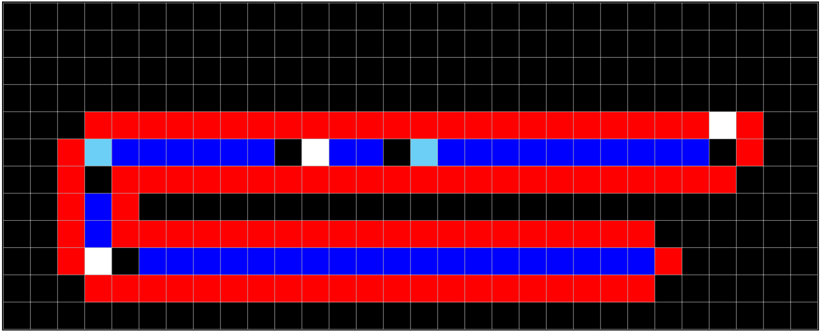


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

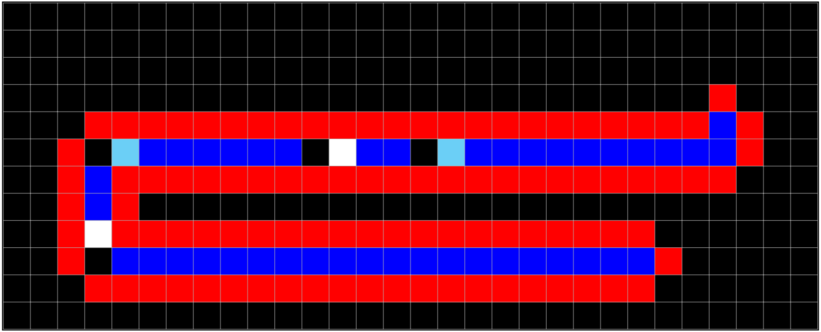


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

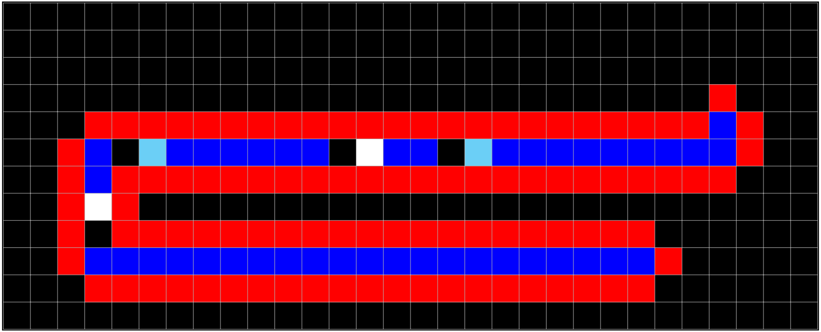


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

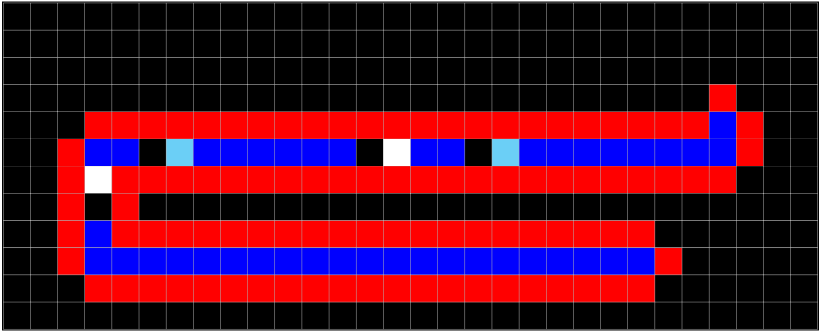


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

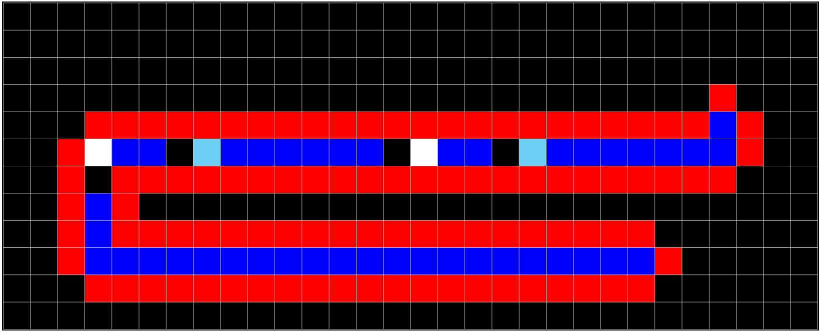


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

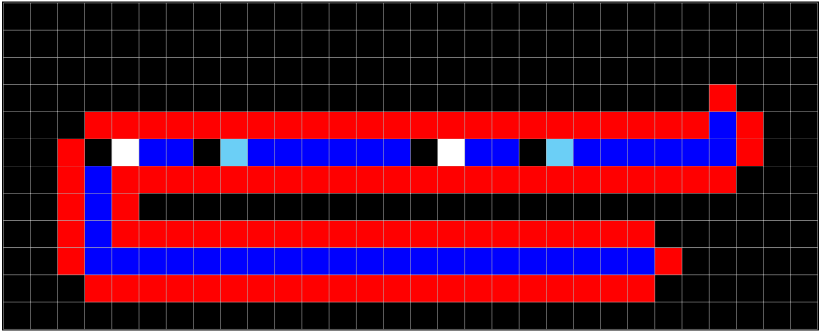


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

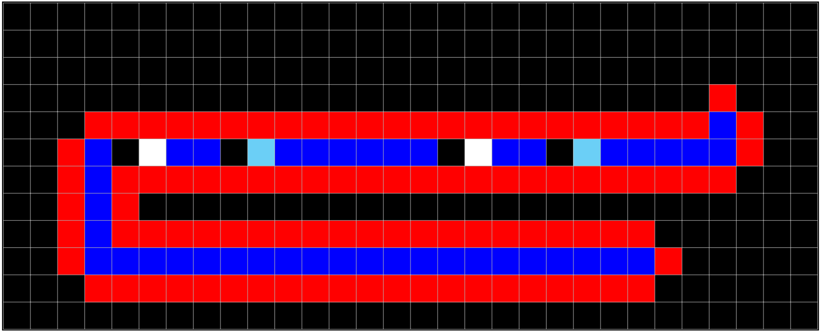


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

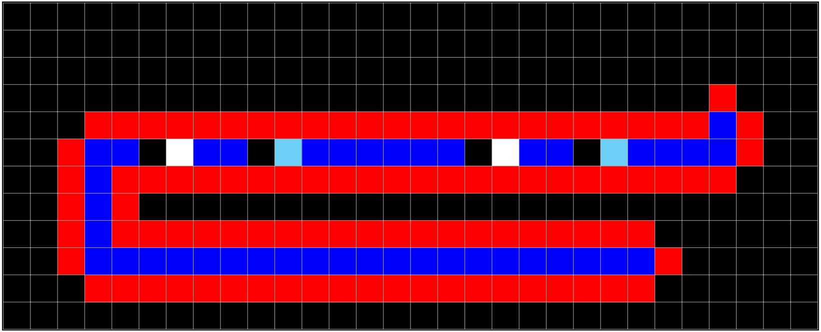


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

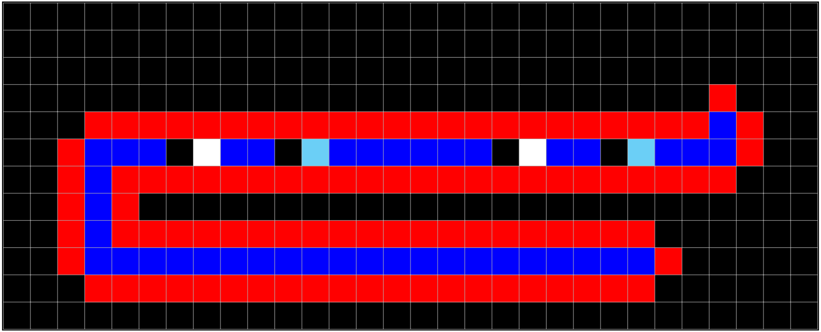


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

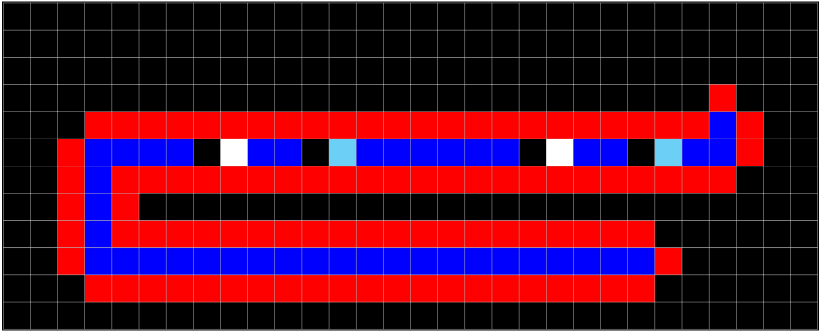


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

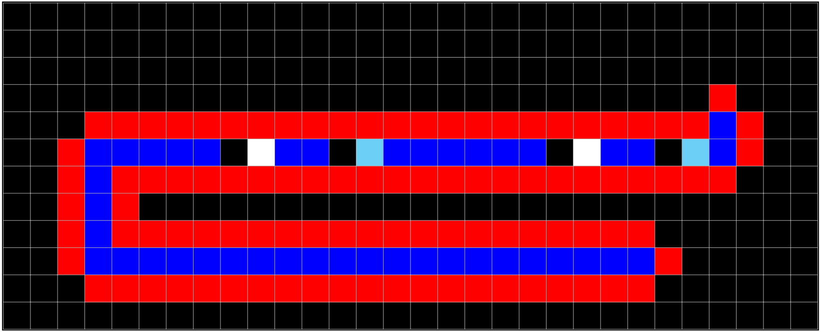


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

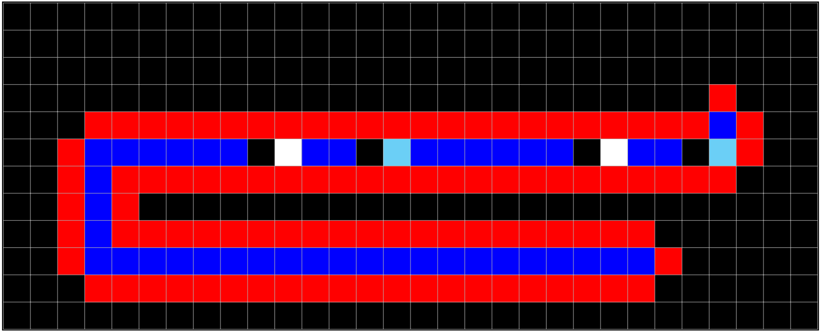


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

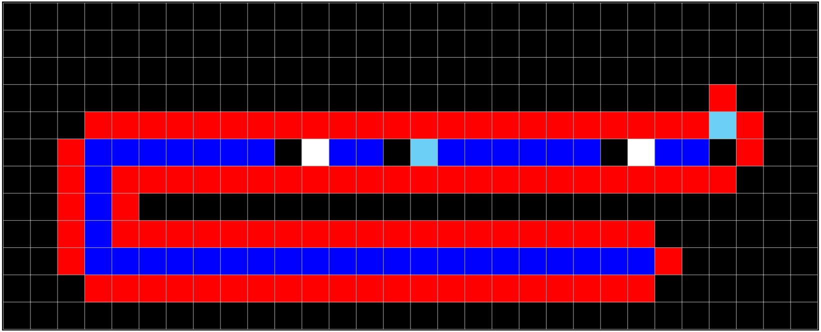


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

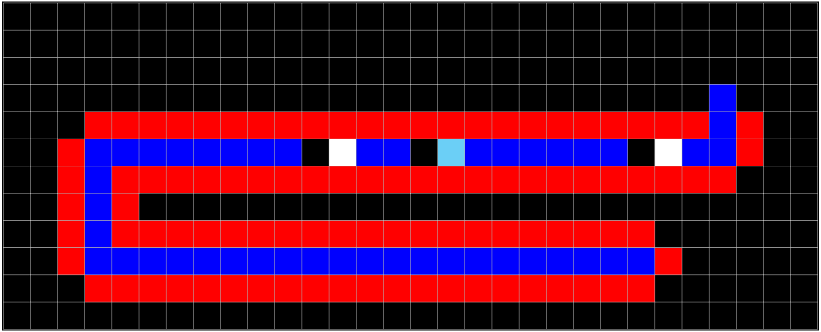


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

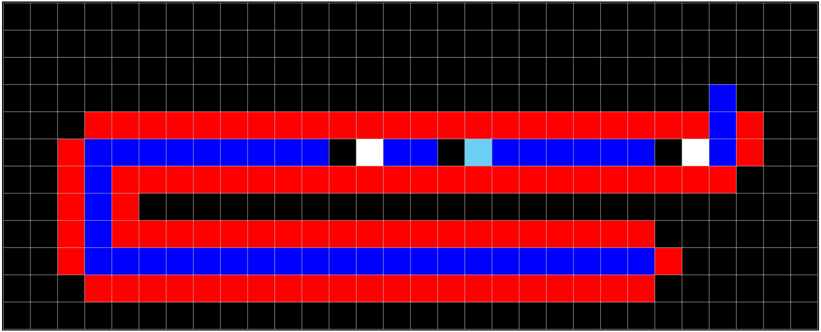


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

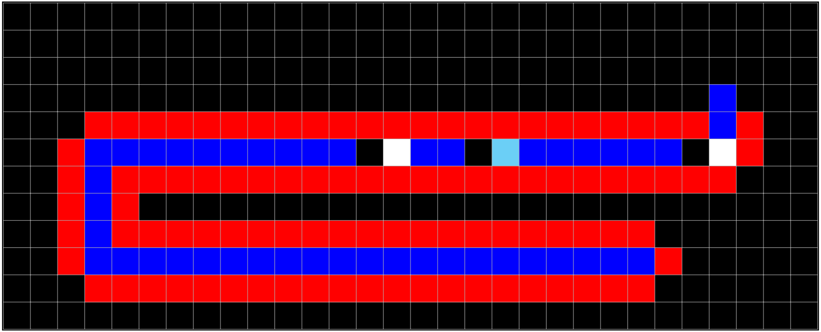


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

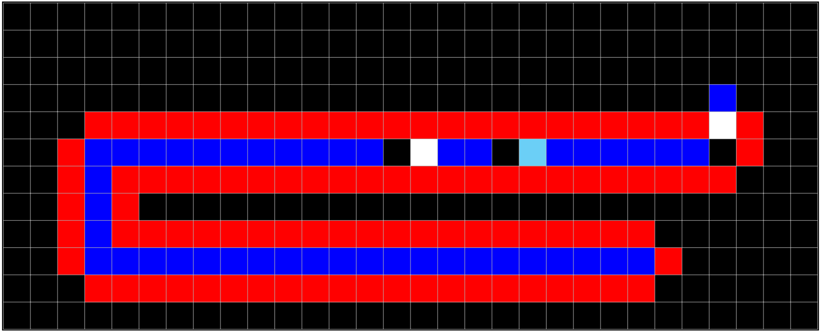


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

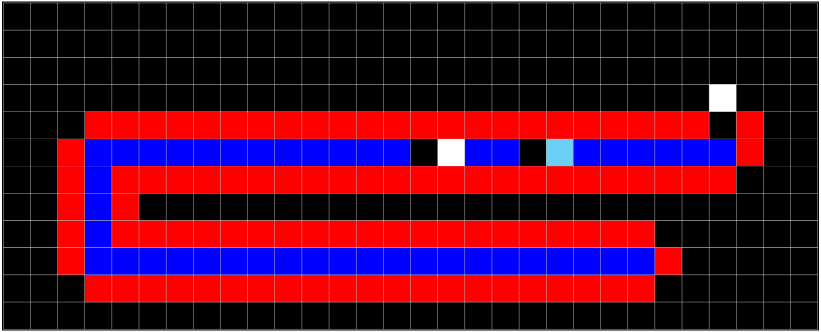


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

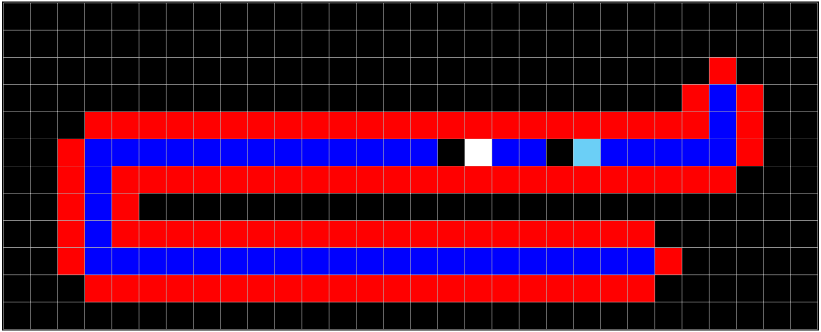


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

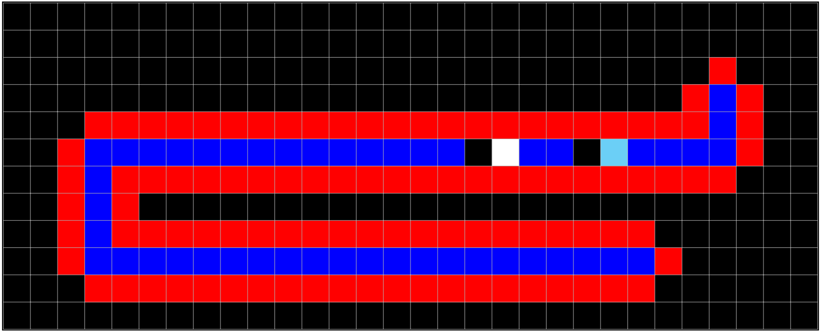


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

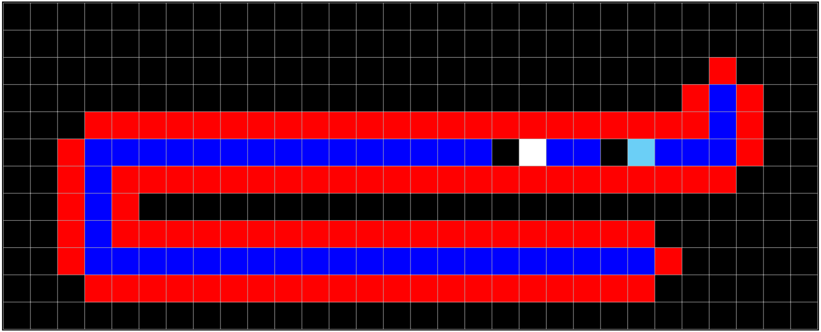


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

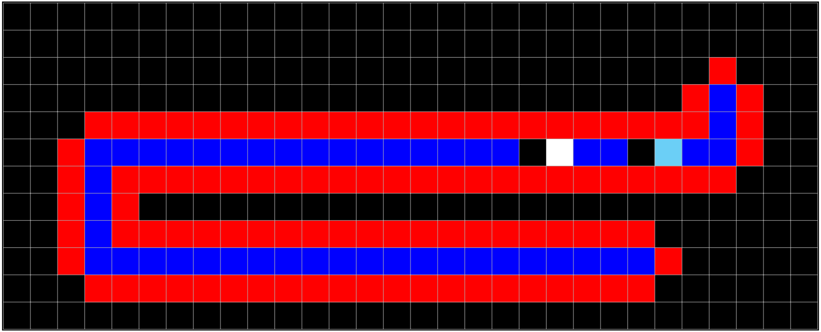


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

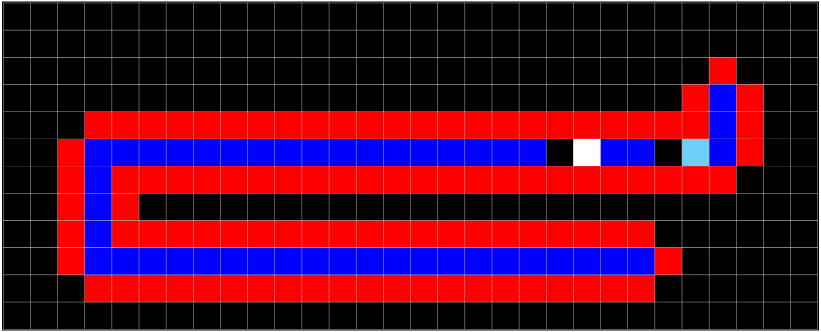


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

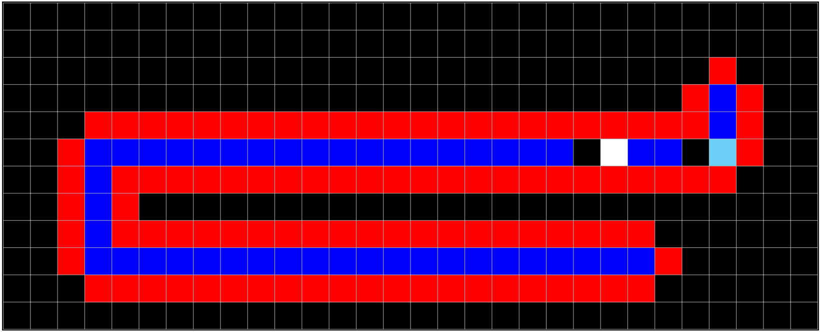


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

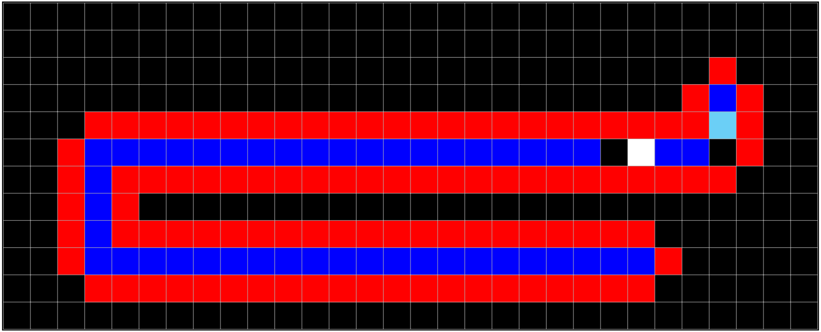


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

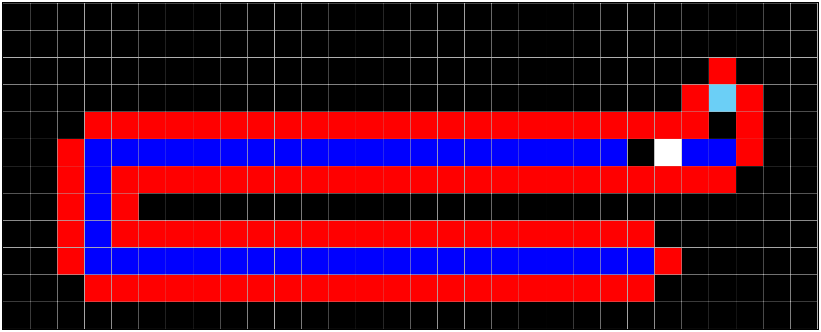


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

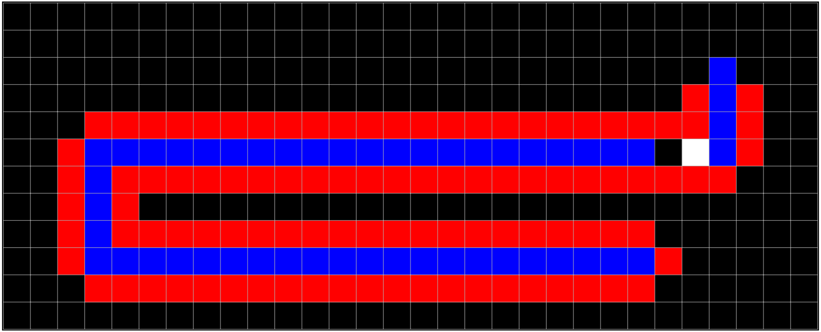


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

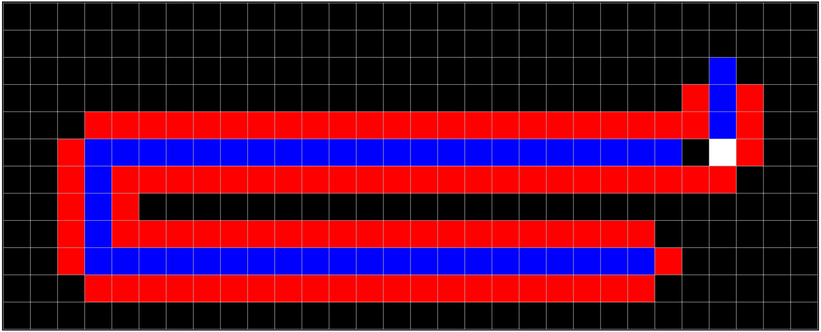


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

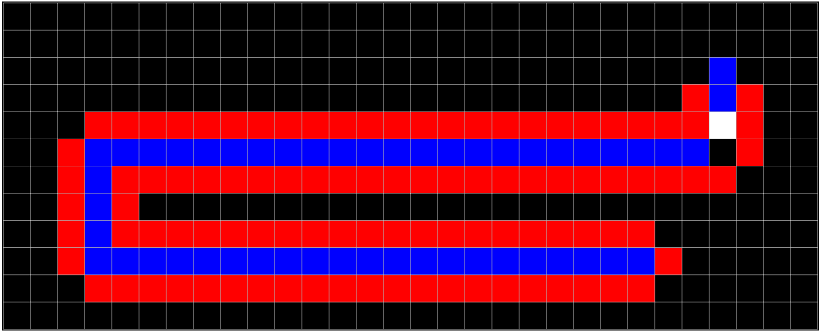


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

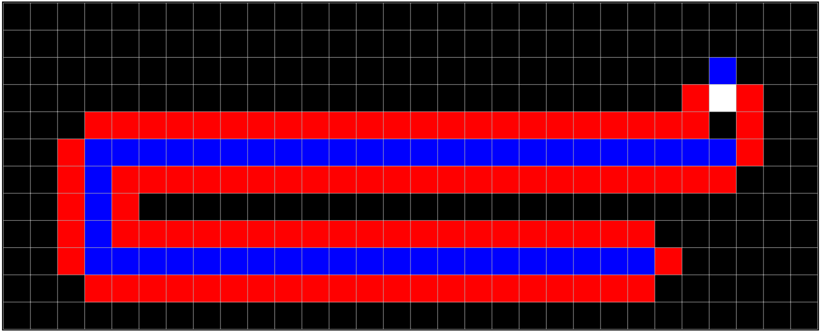


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

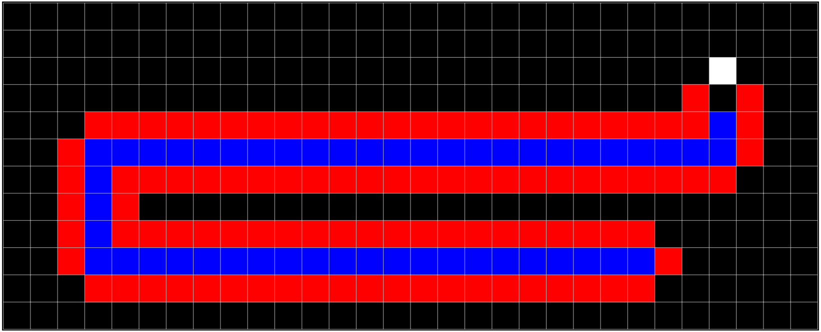


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.

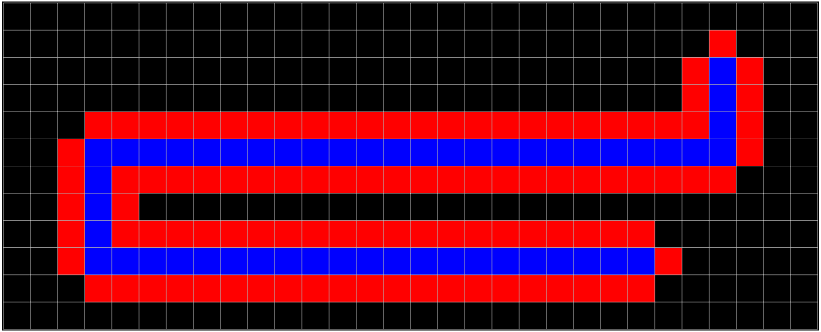


Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

# Codd self-reproducing CA

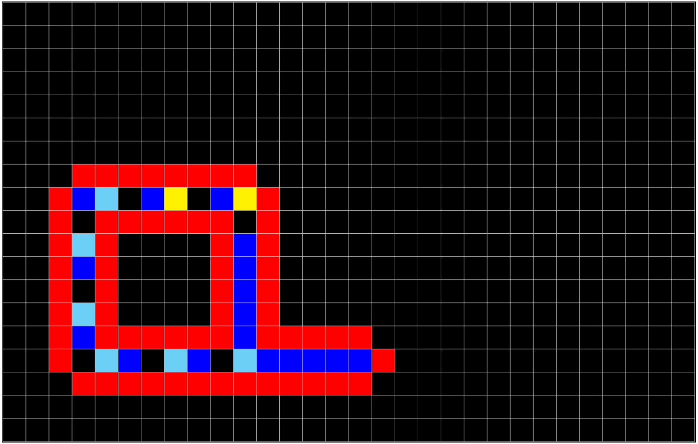
A **8 states self-reproducing** CA with von Neumann neighborhood using sheathed wires.



Implemented by Hutton in 2009, several millions cells, self-reproducing in $1.7 \times 10^{18}$ steps (estimated).

*(Cellular Automata, Codd, 1968)*

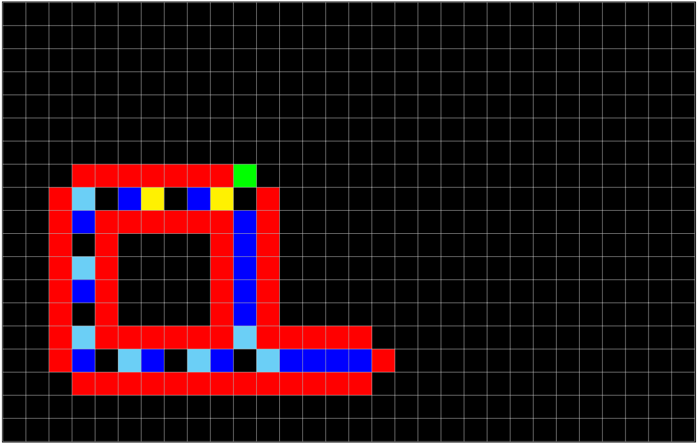
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

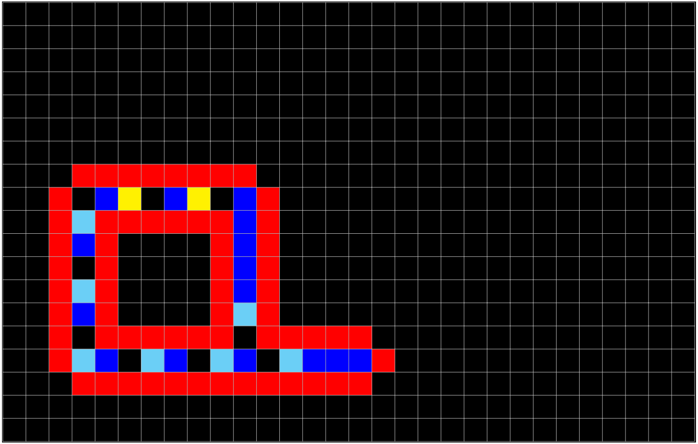
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

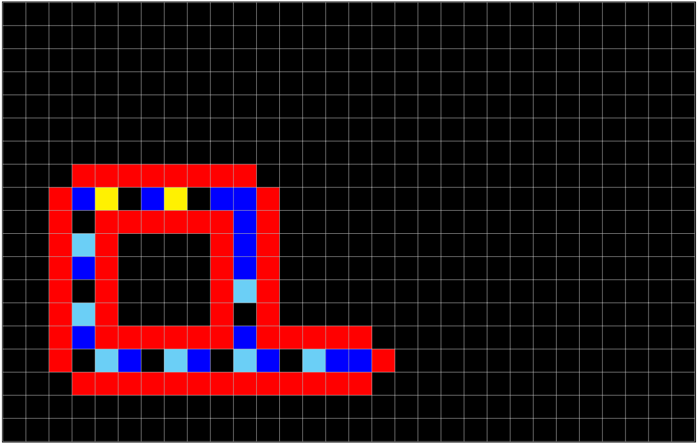
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

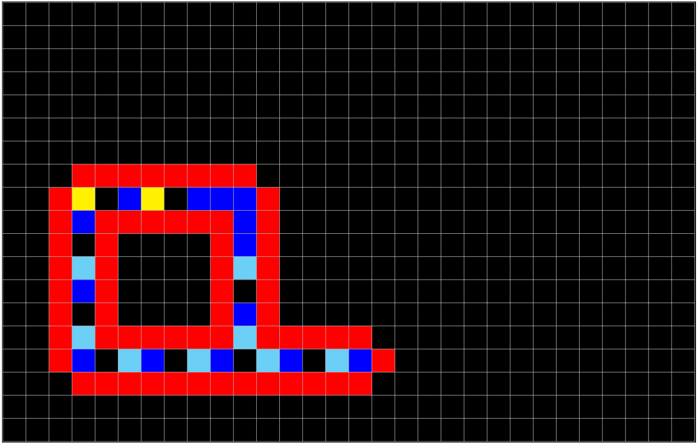
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

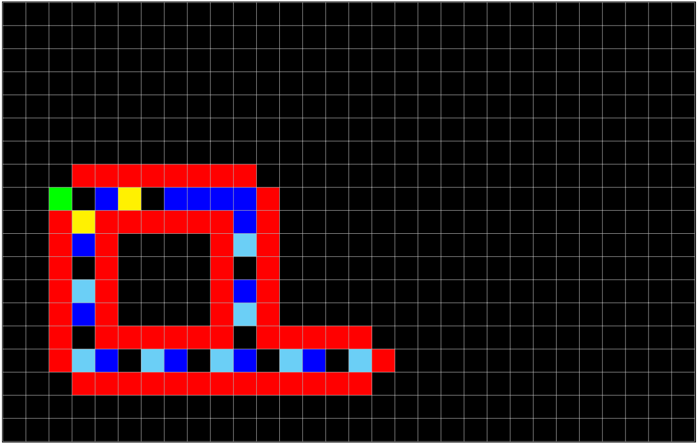
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

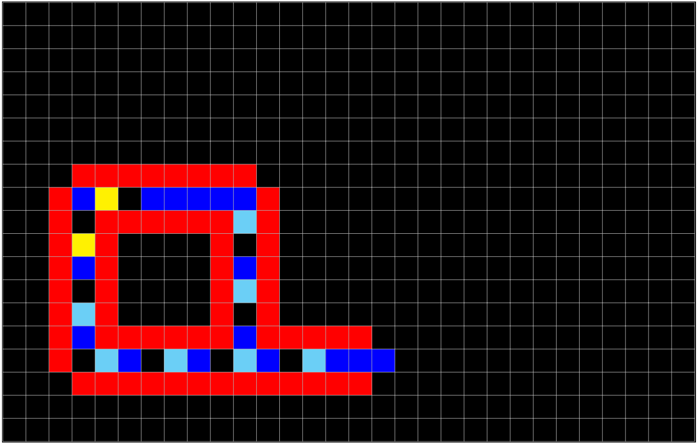
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

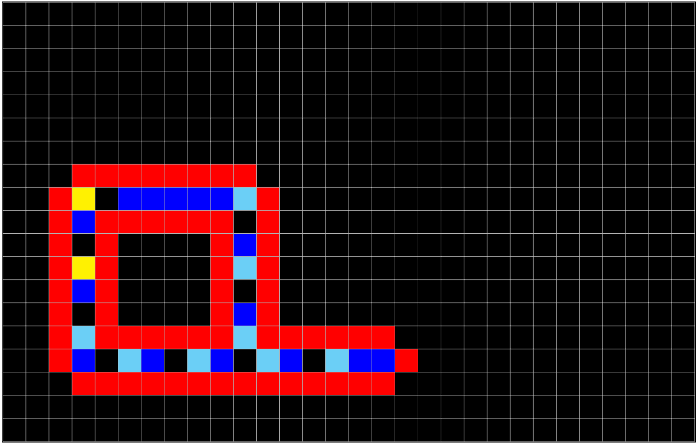
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

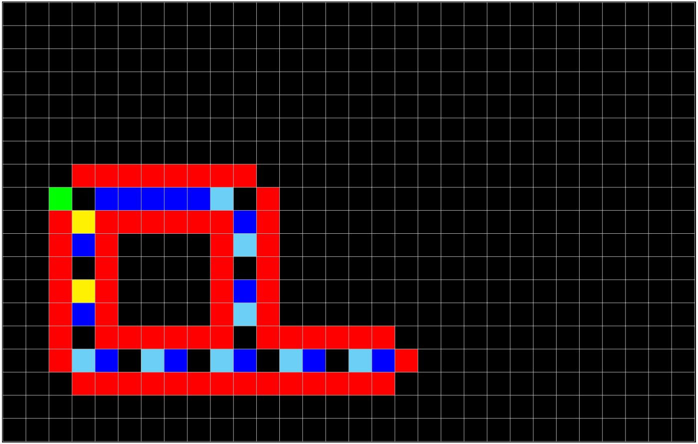
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

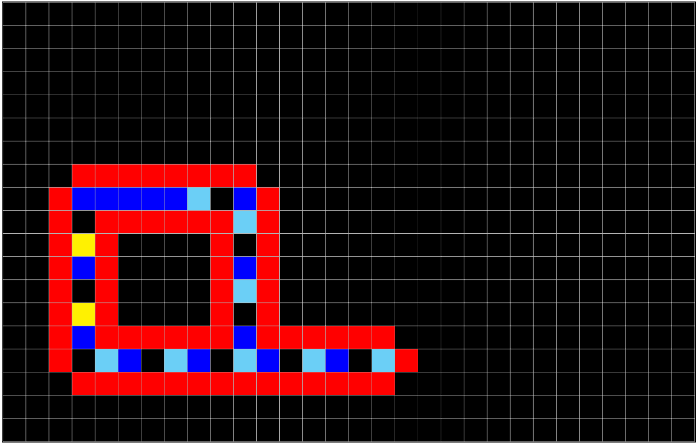
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

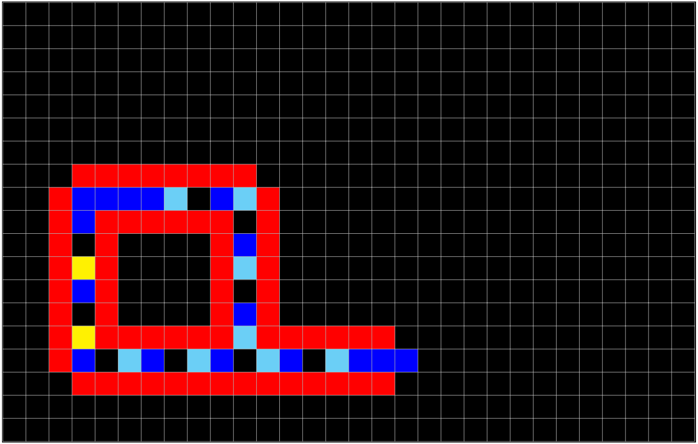
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

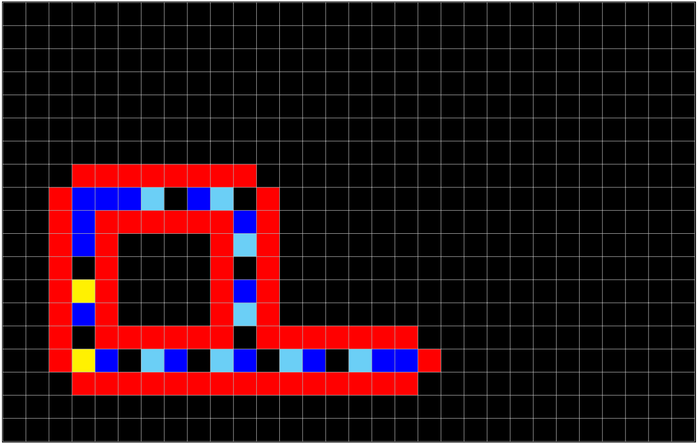
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

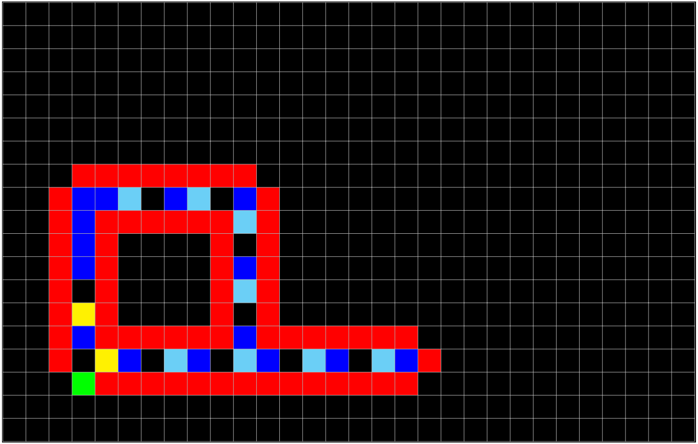
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

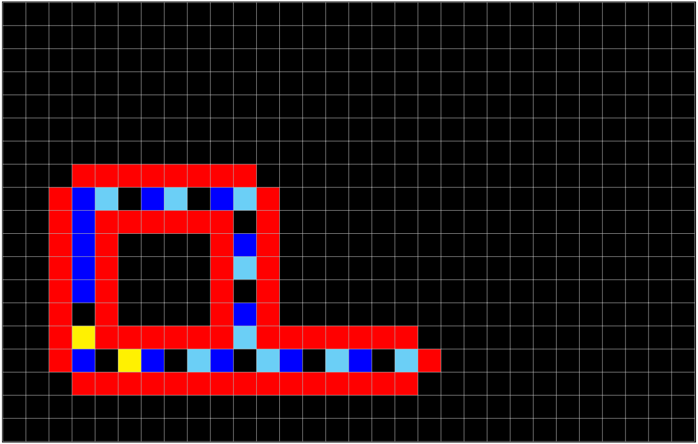
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

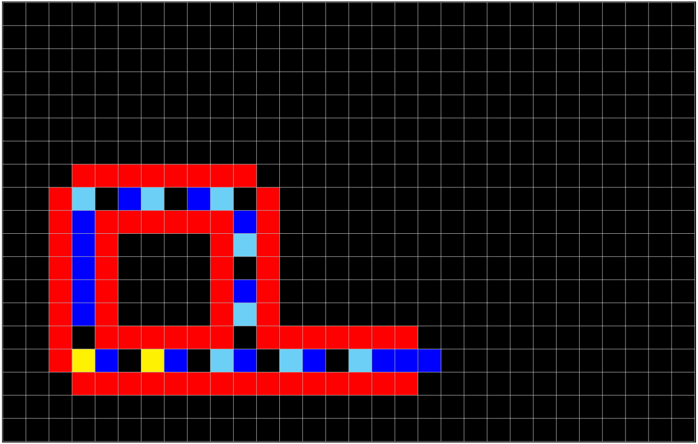
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

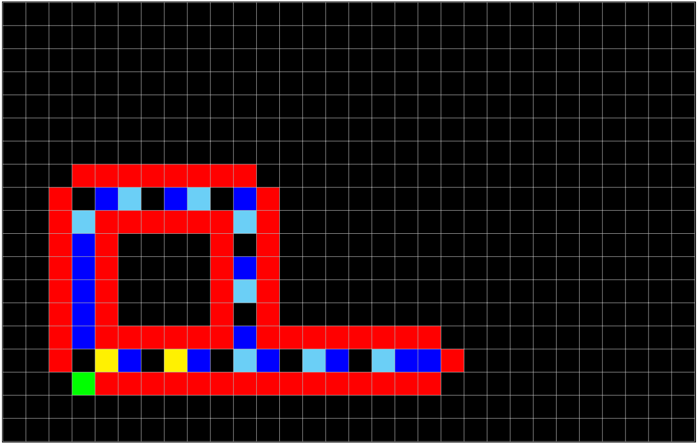
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

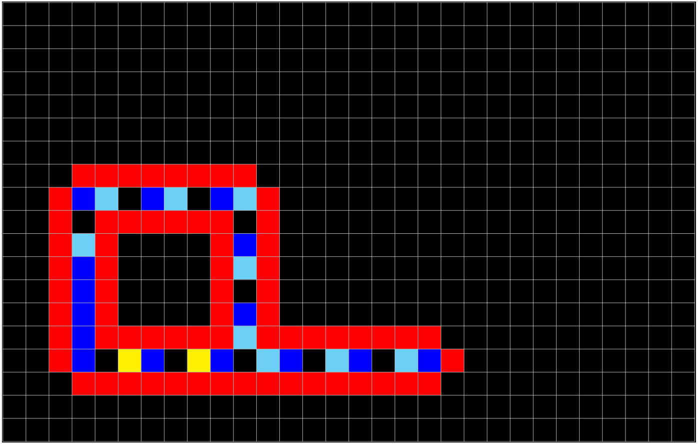
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

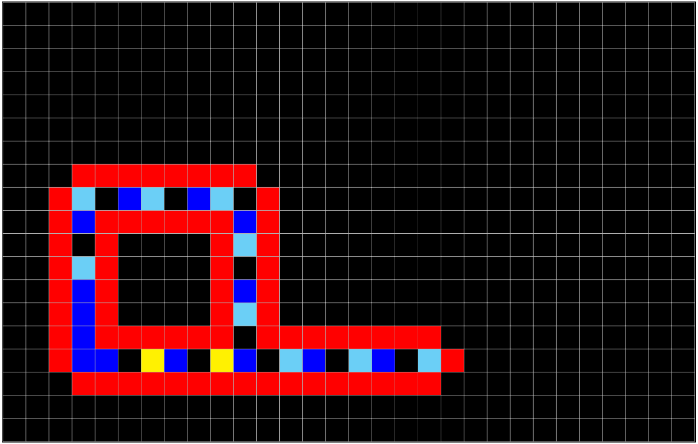
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

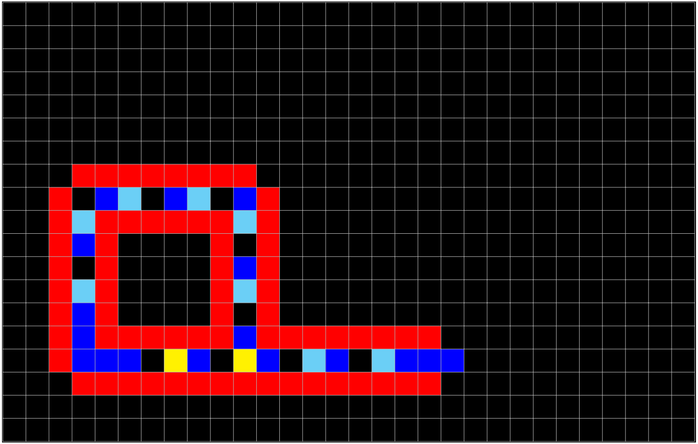
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

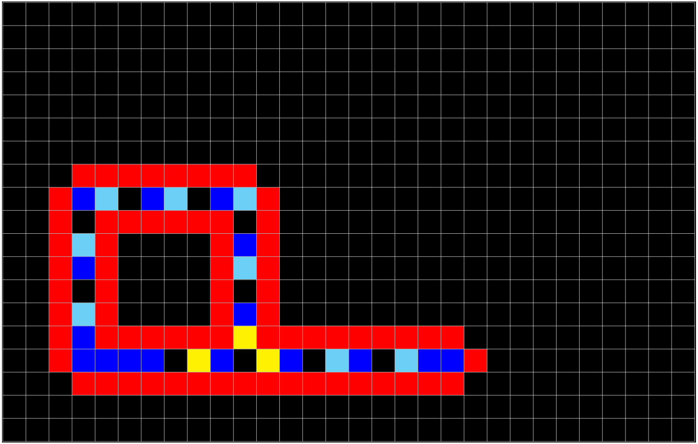
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

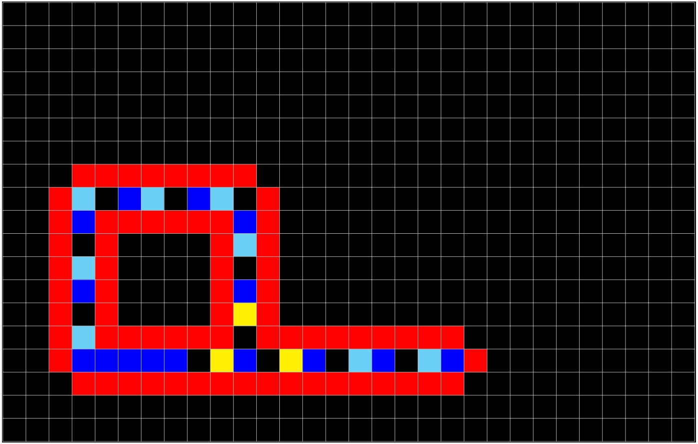
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

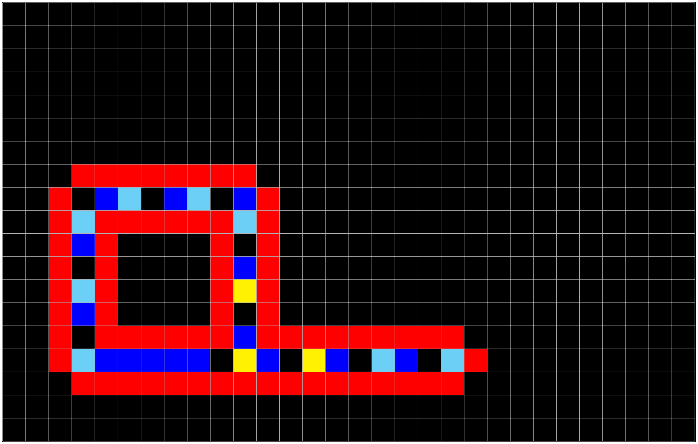
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

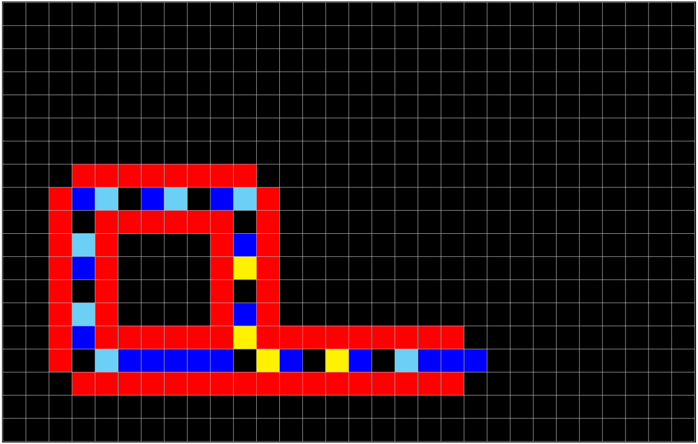
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

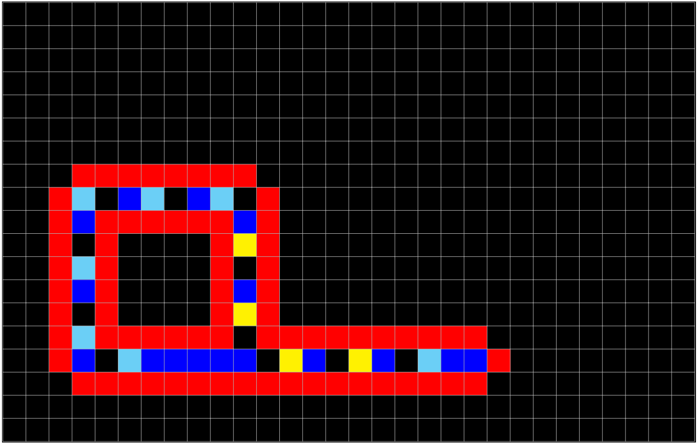
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

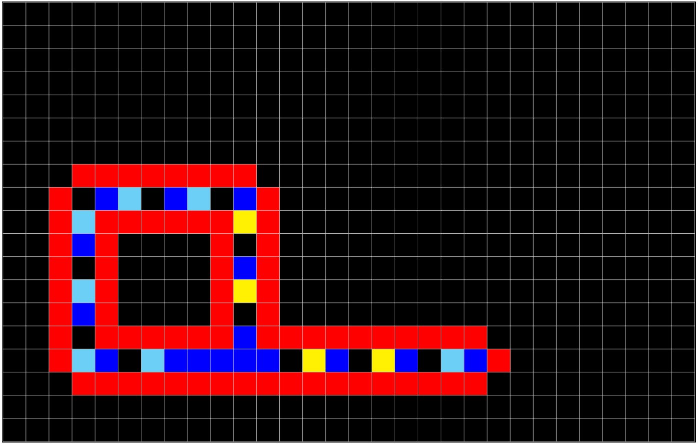
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

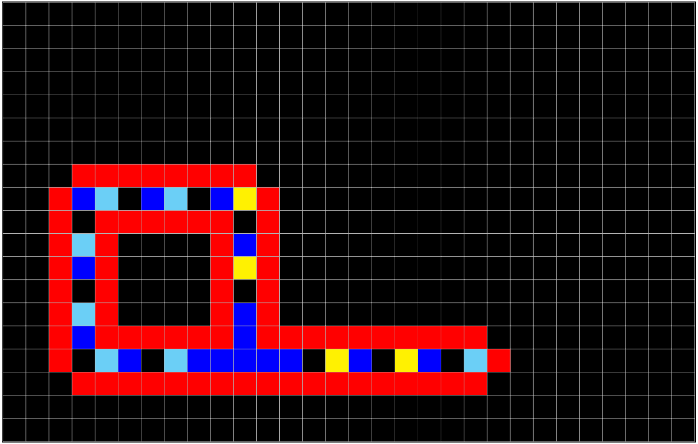
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

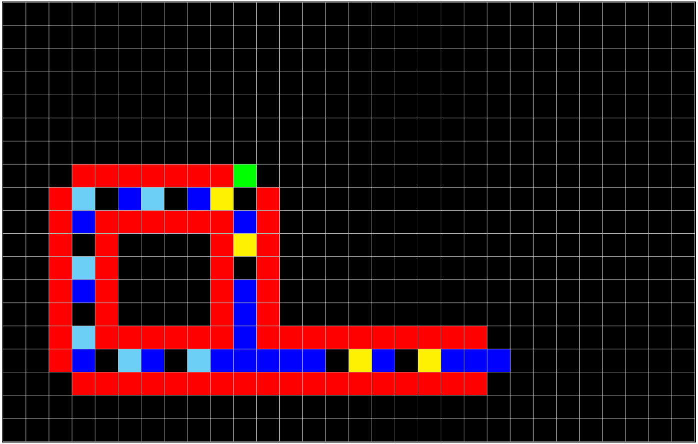
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

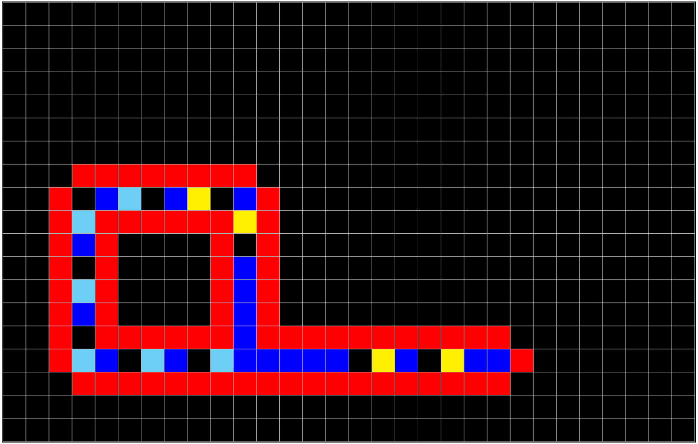
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

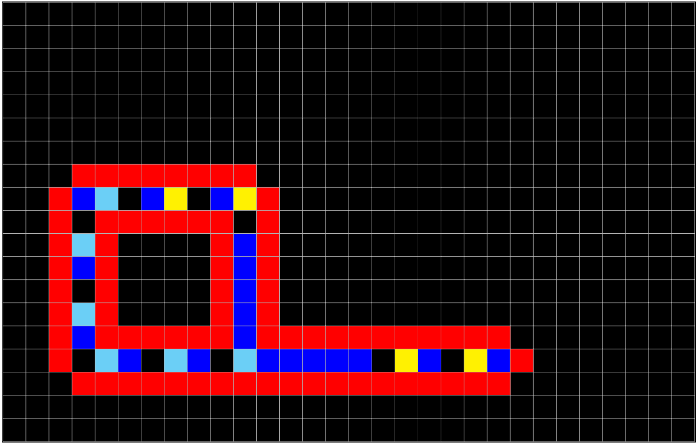
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

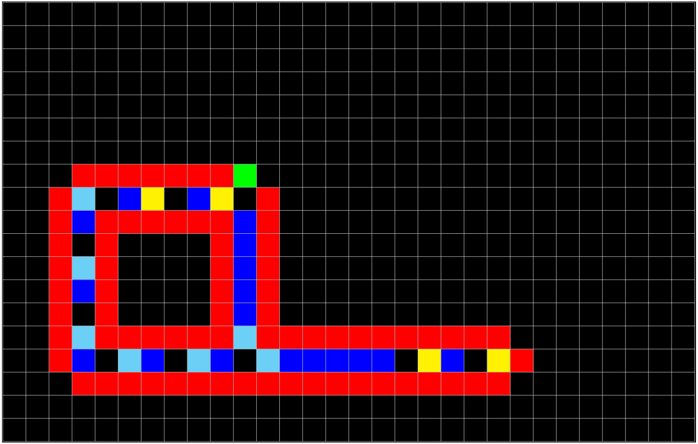
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

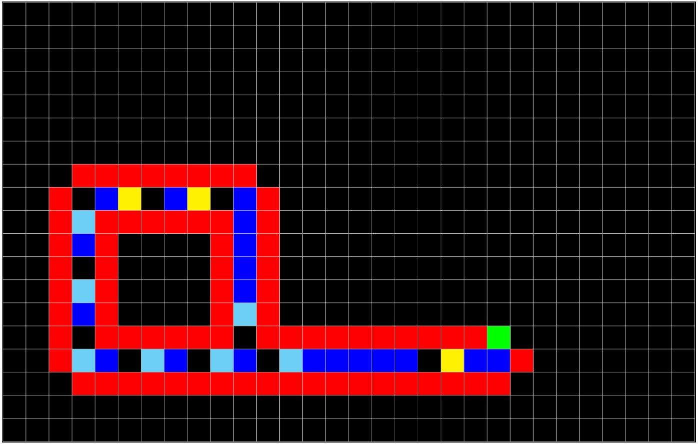
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

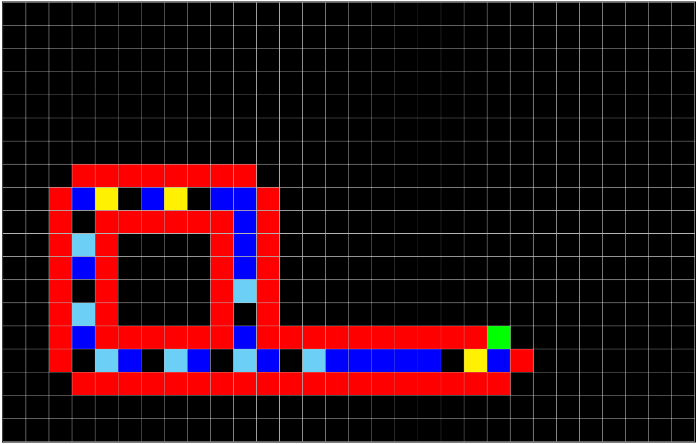
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

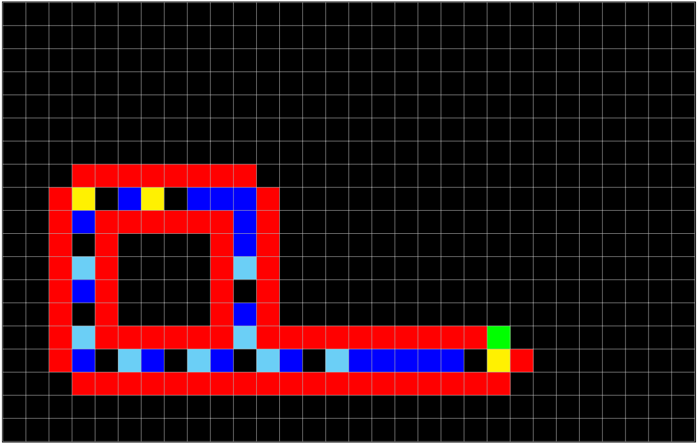
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

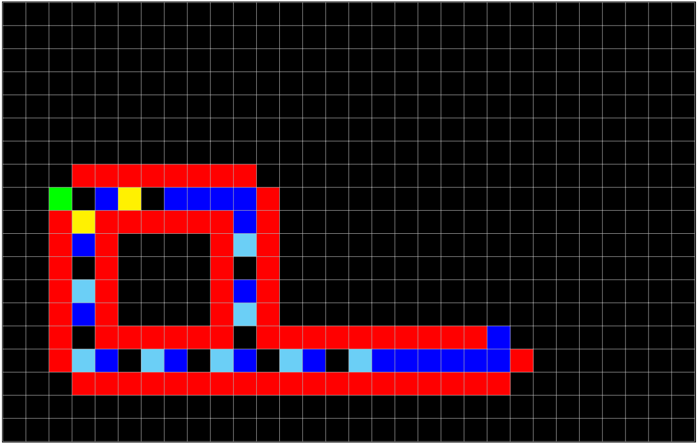
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

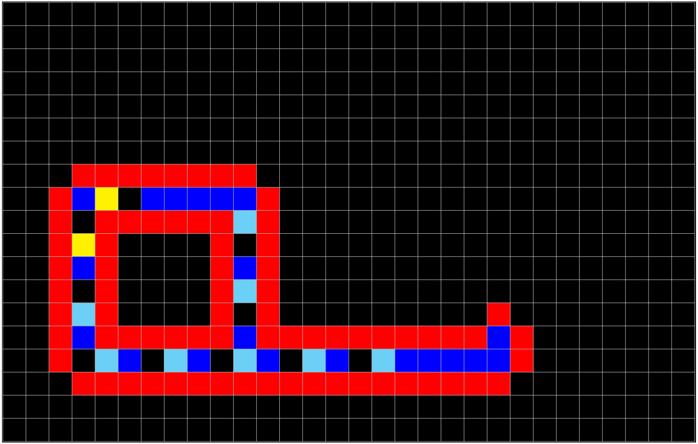
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

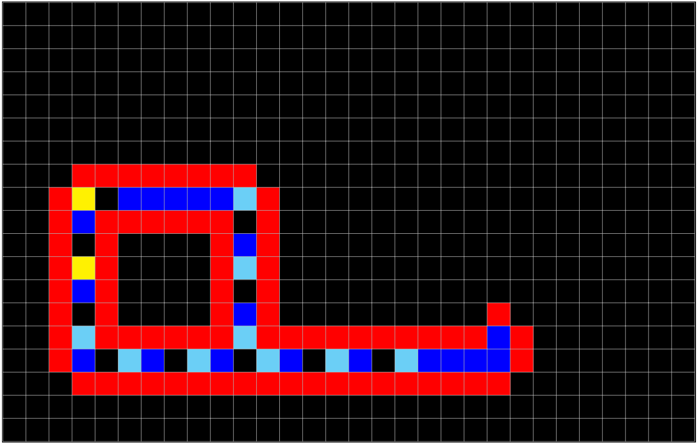
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

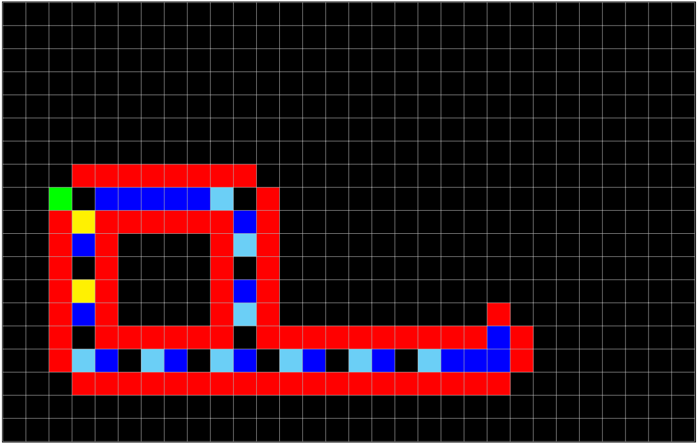
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

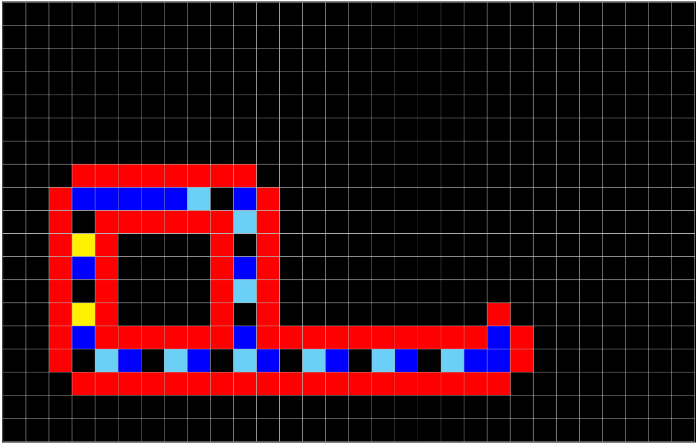
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

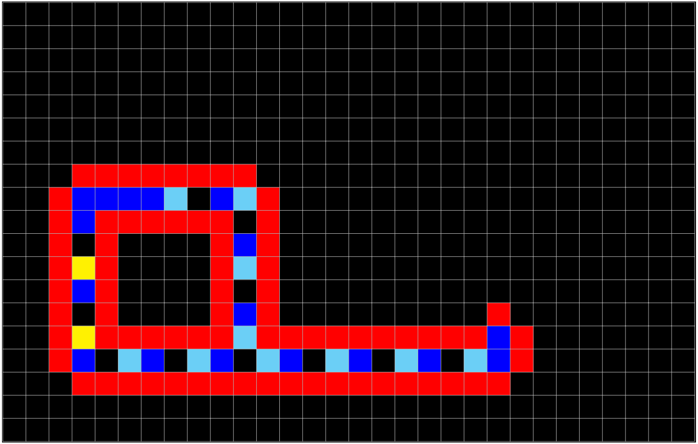
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

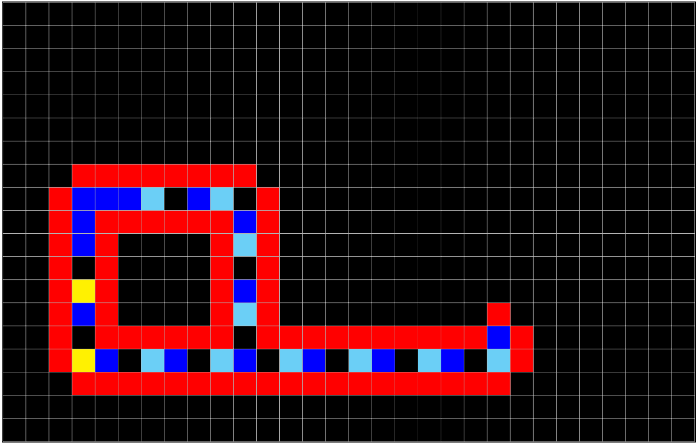
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

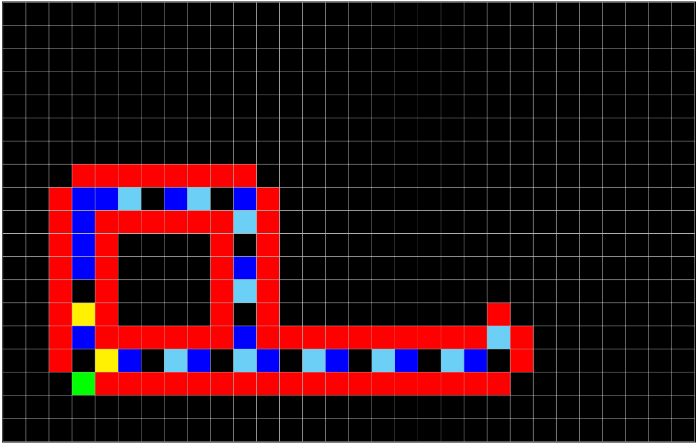
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

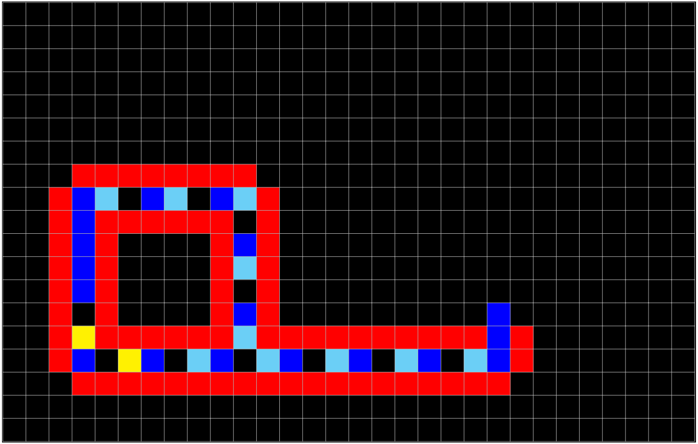
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

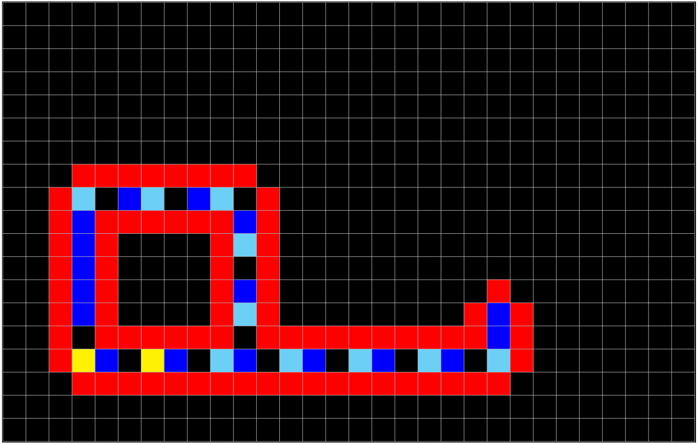
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

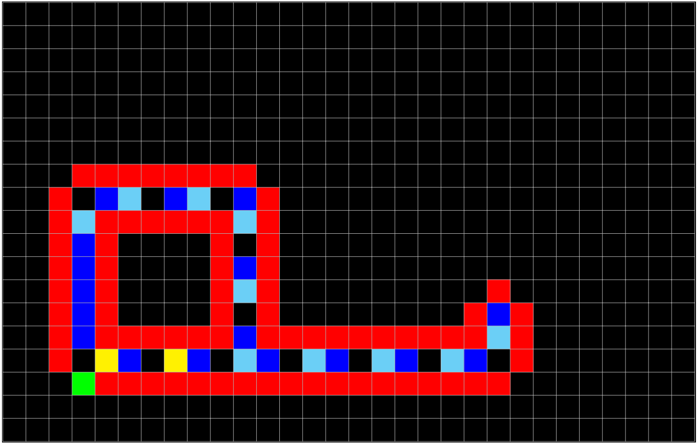
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

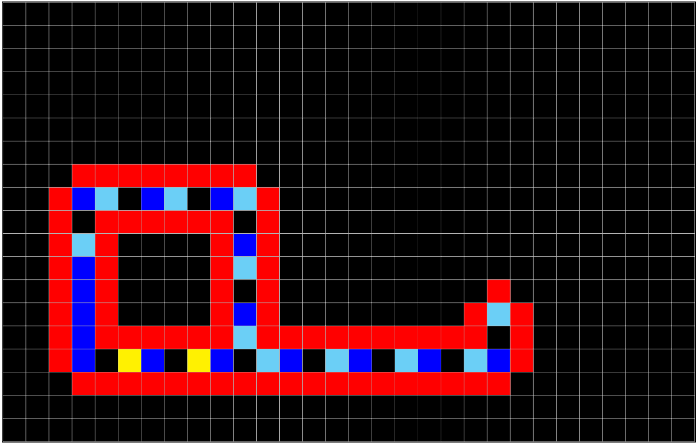
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

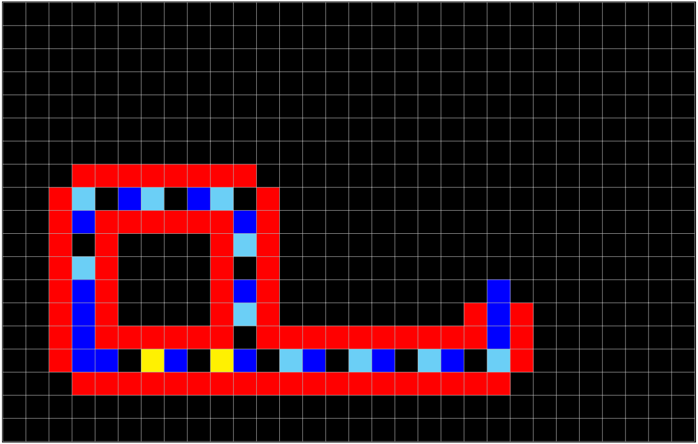
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

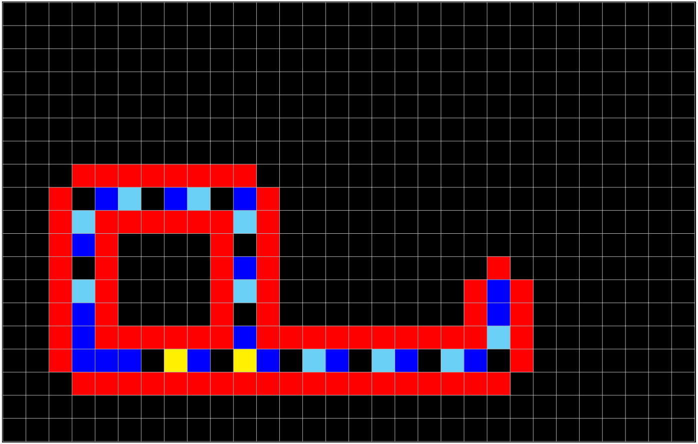
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

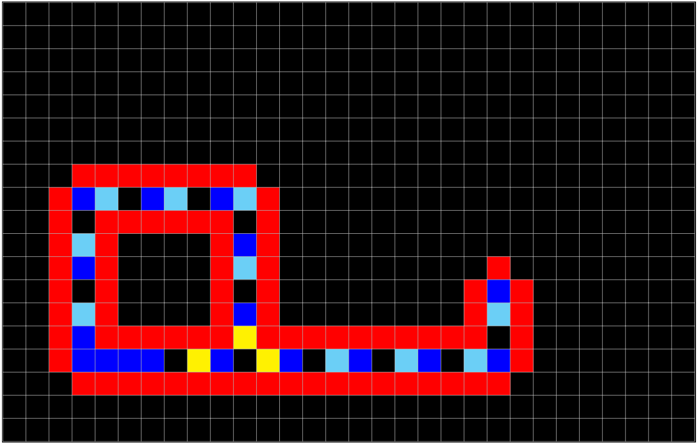
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

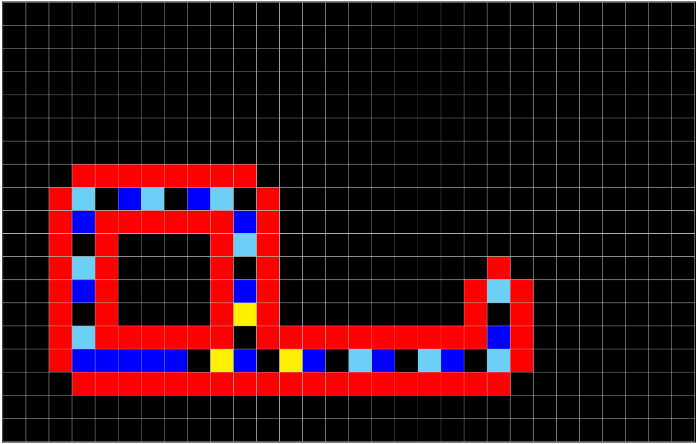
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

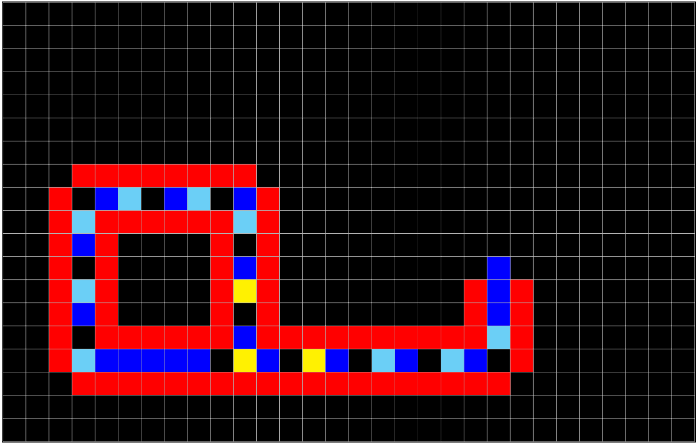
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

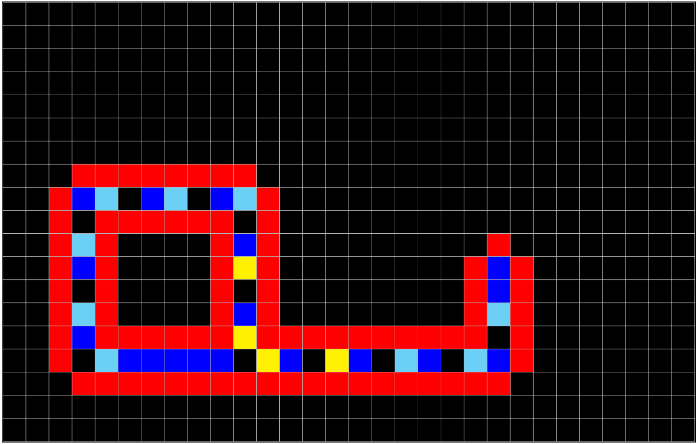
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

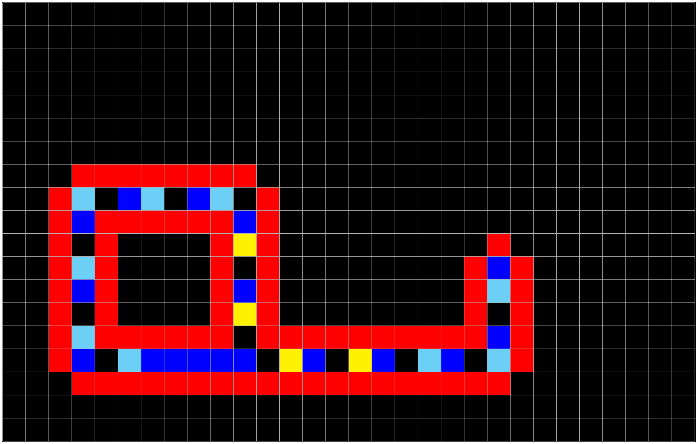
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

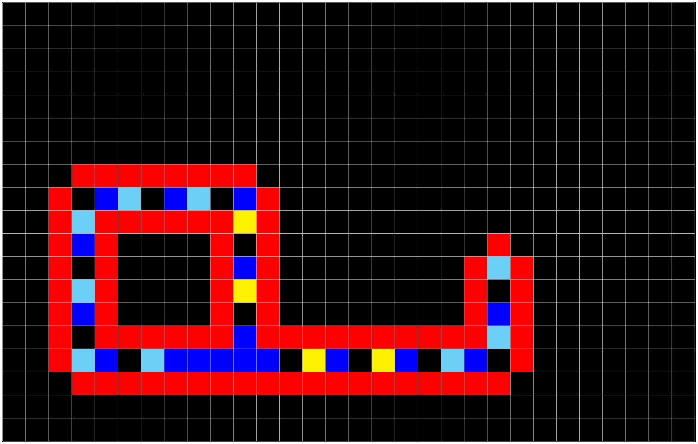
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

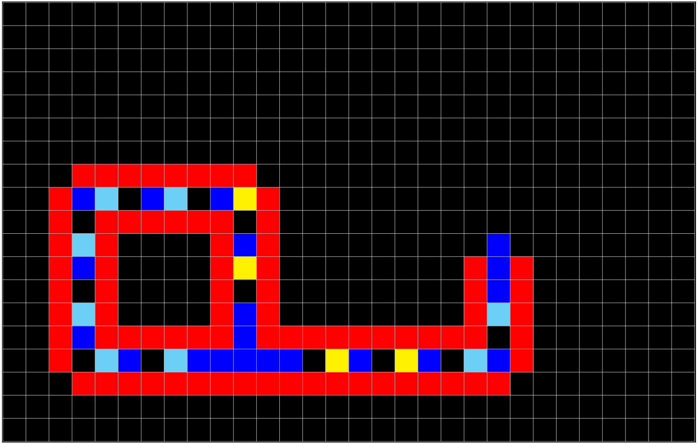
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

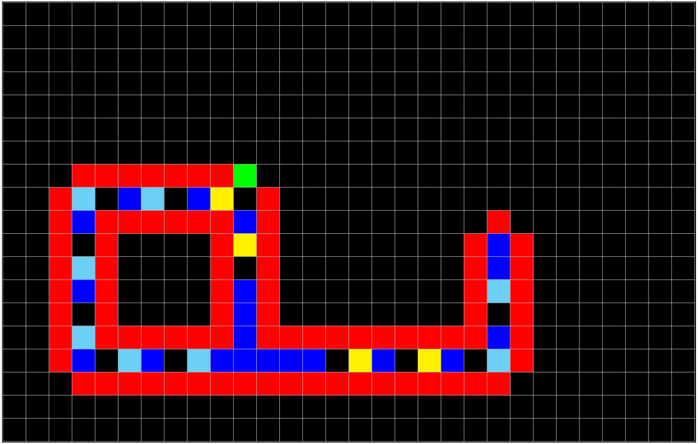
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

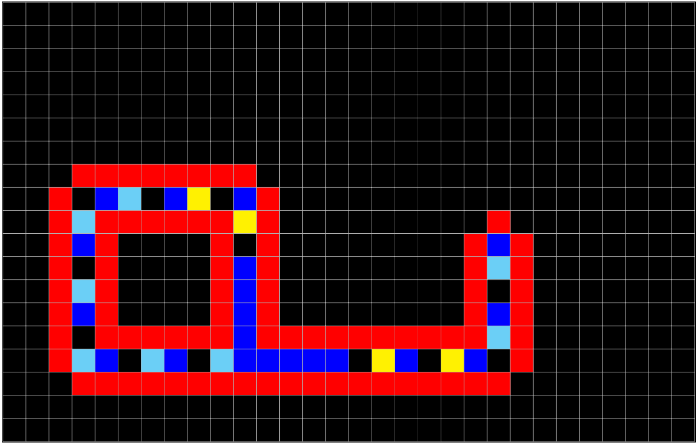
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

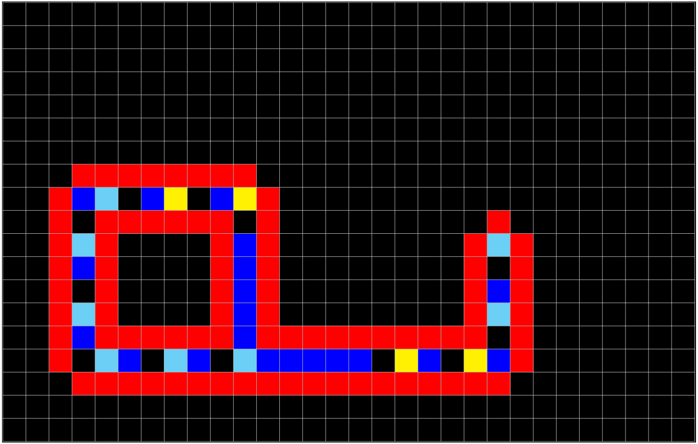
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

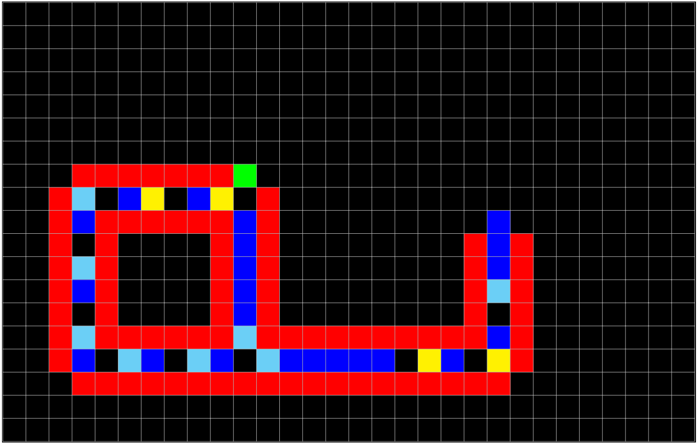
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

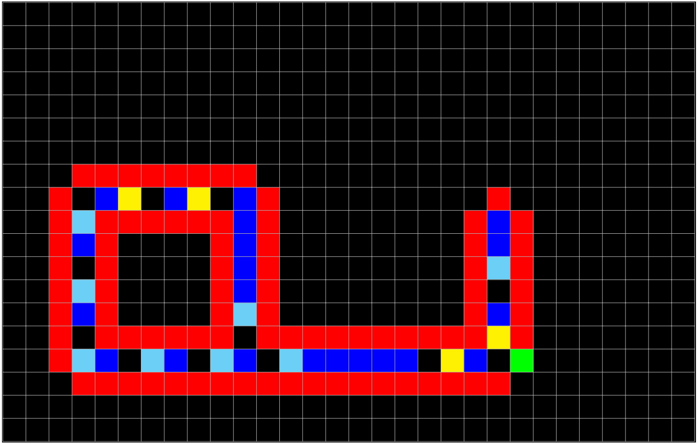
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

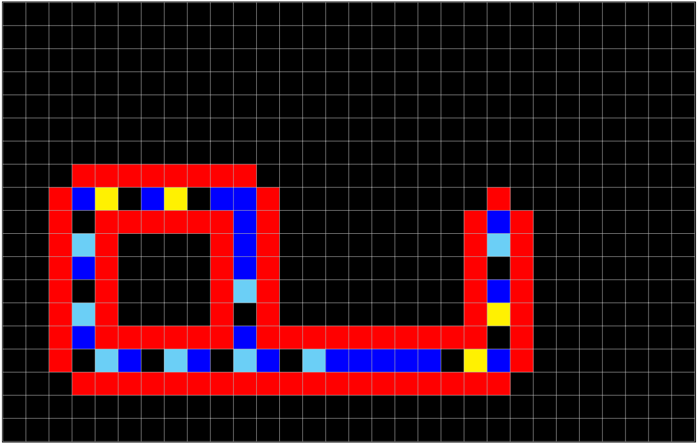
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

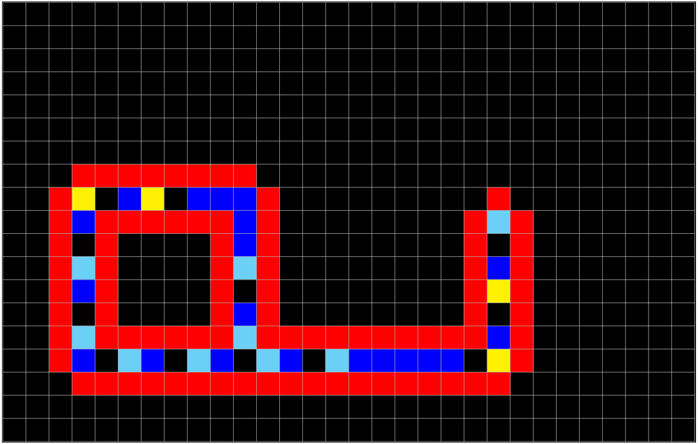
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

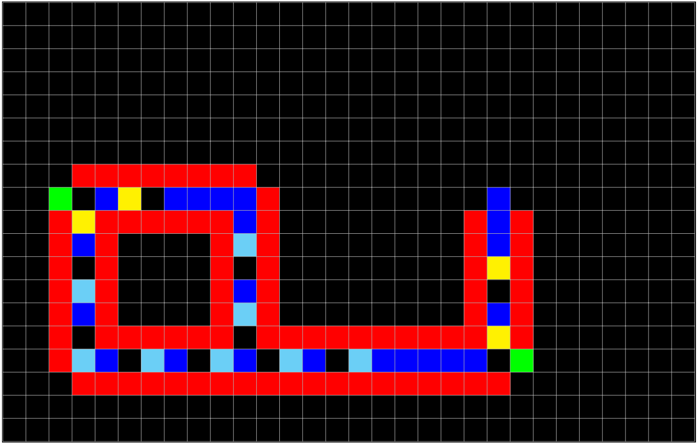
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

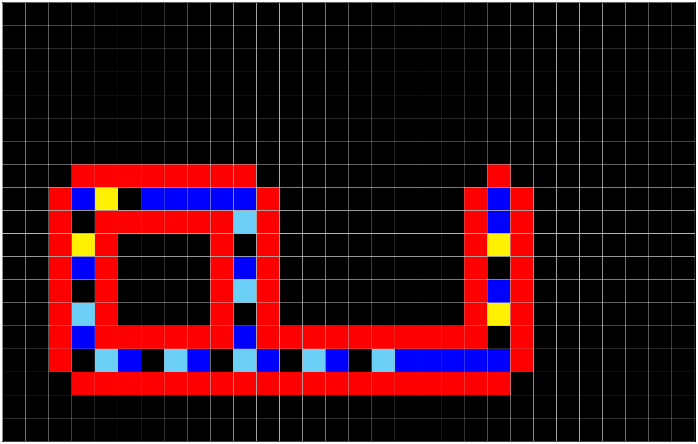
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

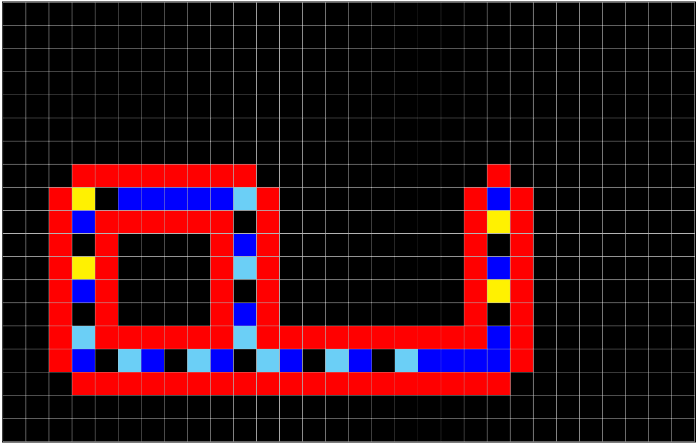
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

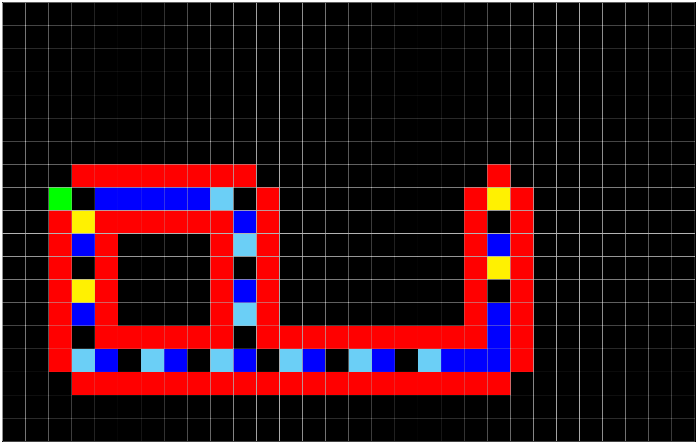
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

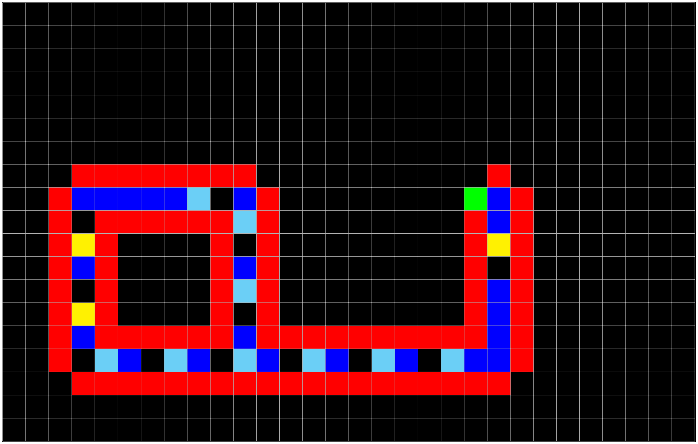
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

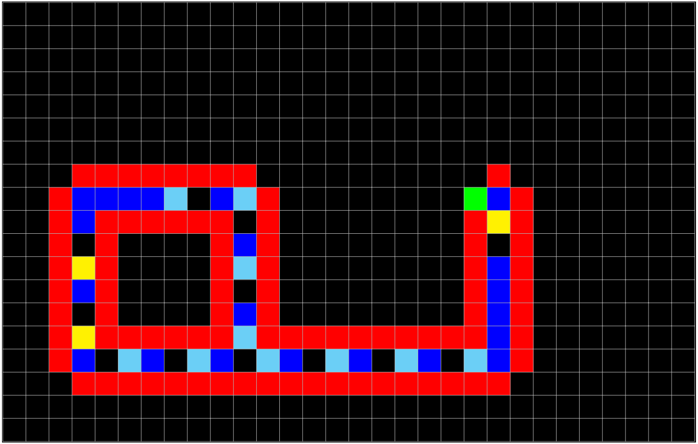
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

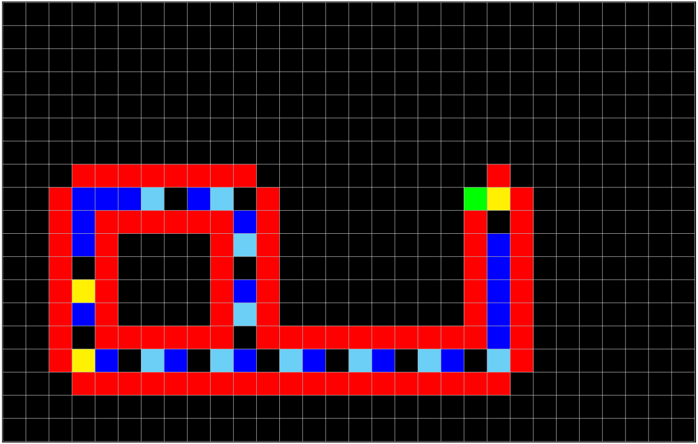
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

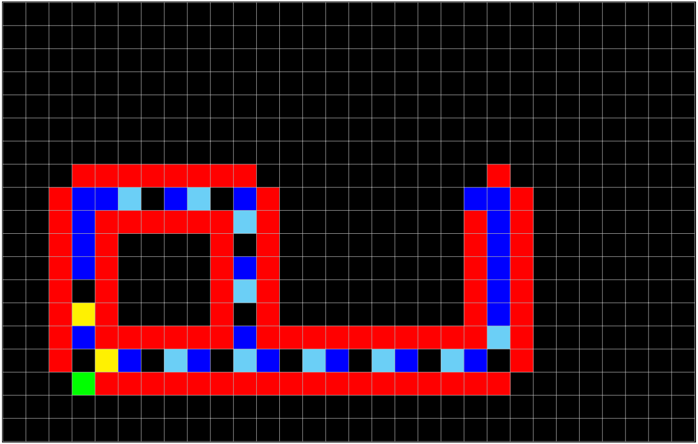
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

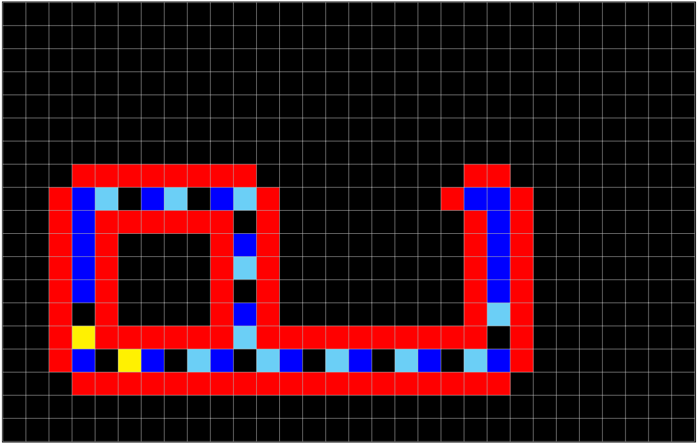
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

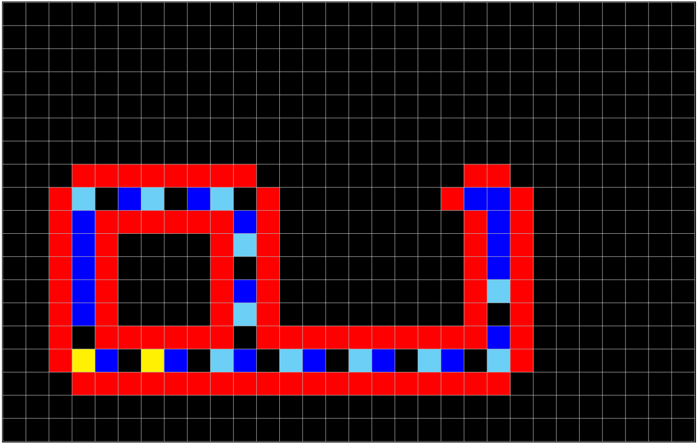
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

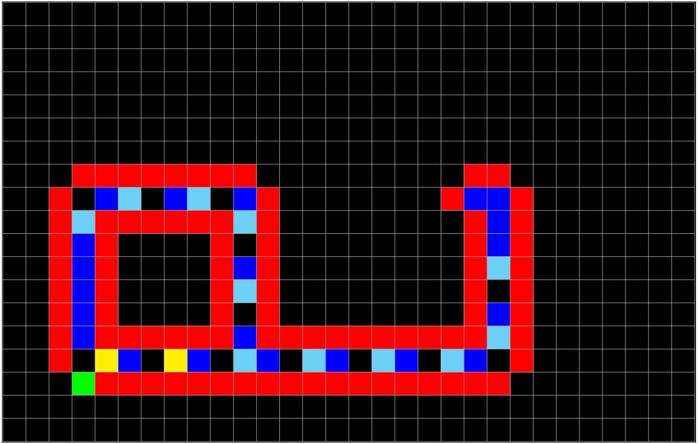
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

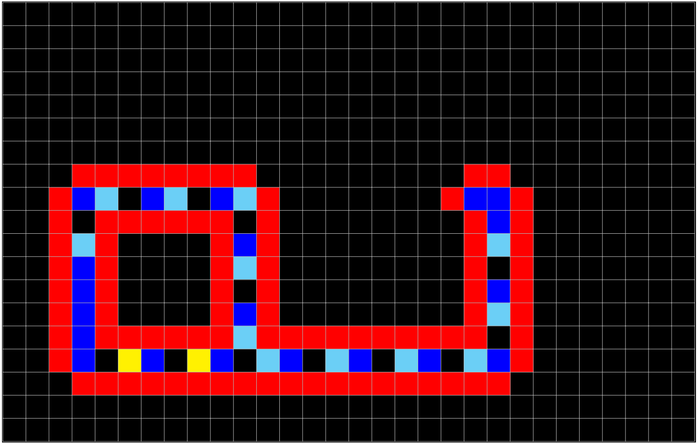
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

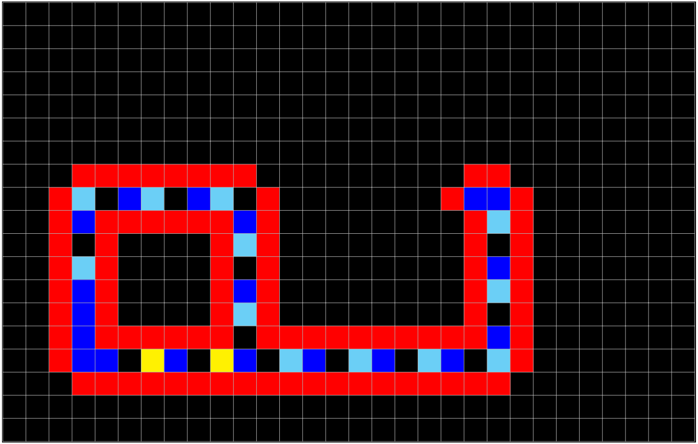
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

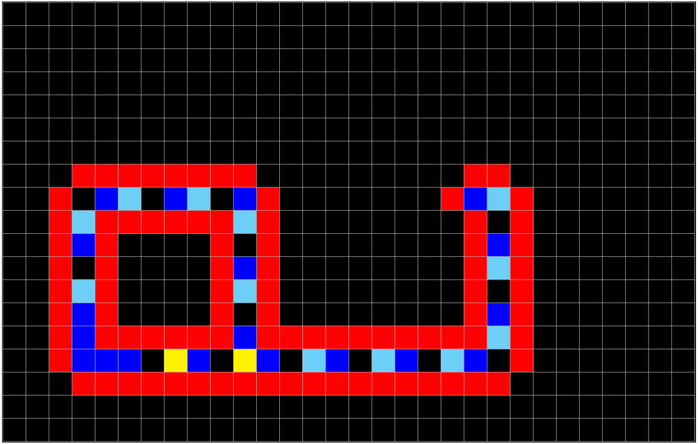
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

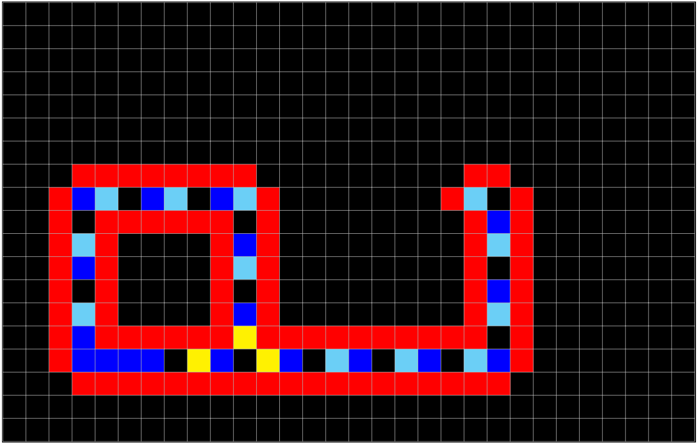
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

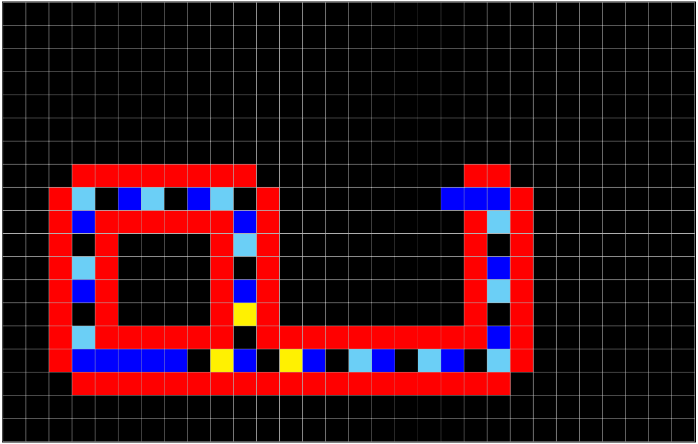
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

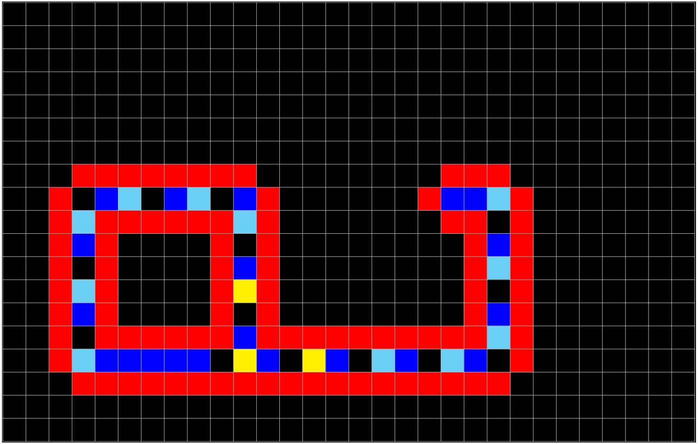
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

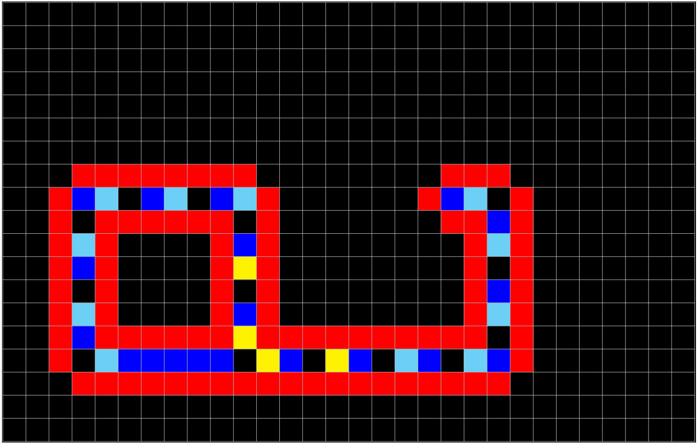
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

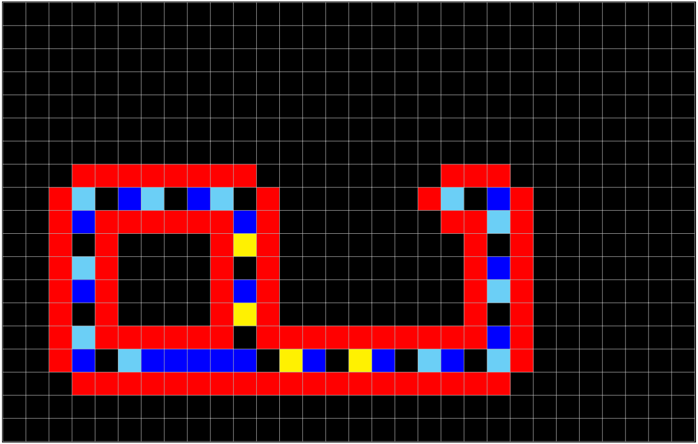
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

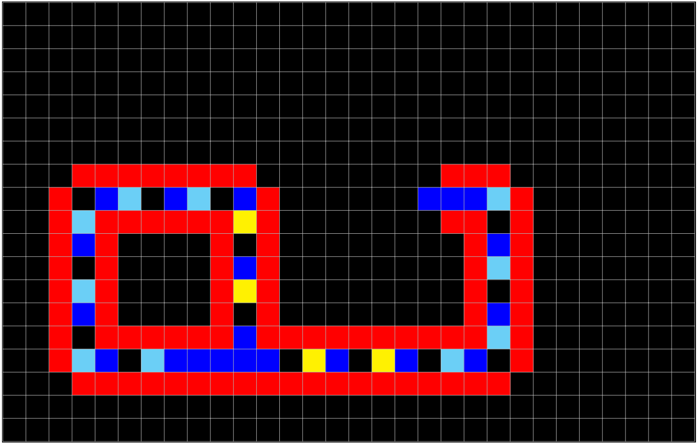
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

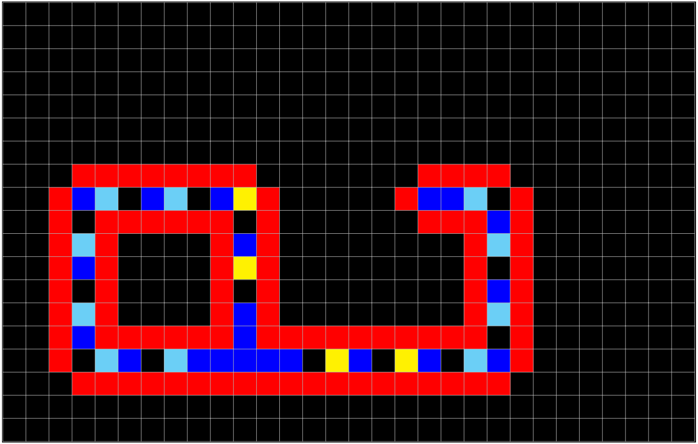
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

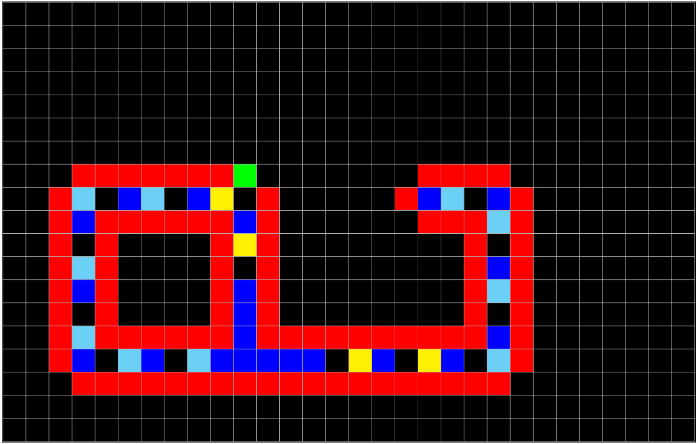
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

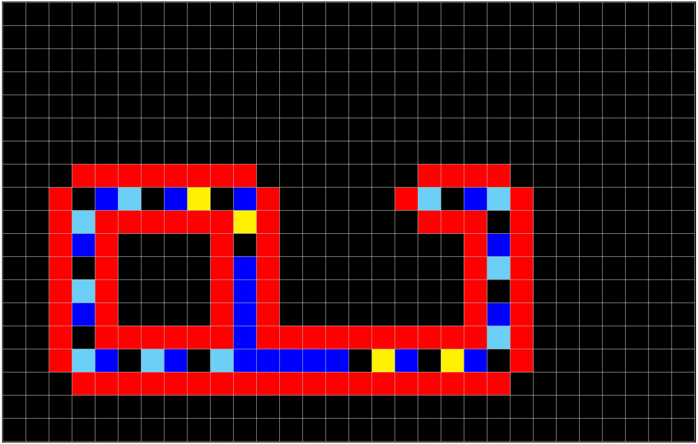
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

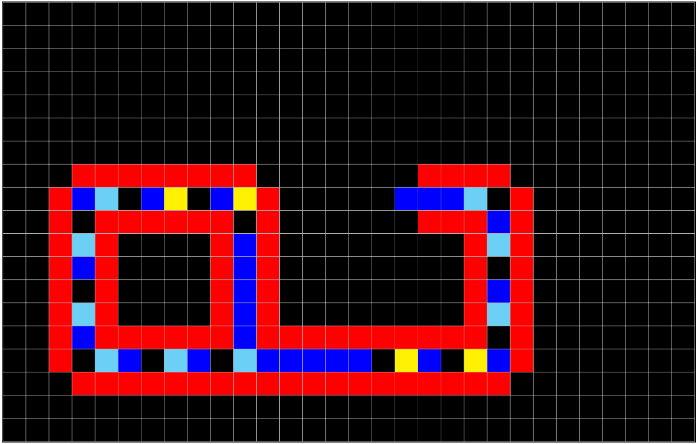
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

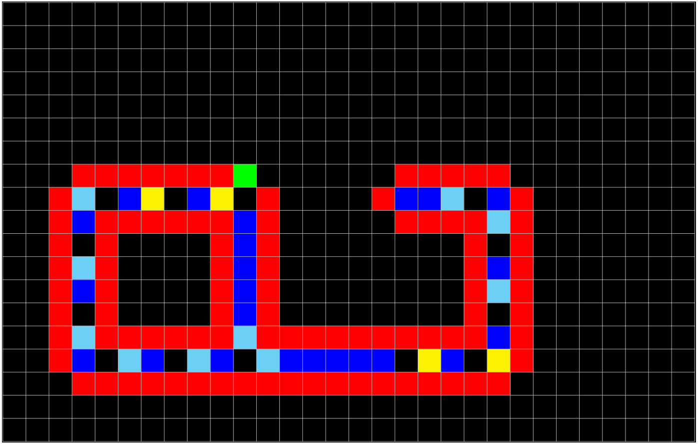
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

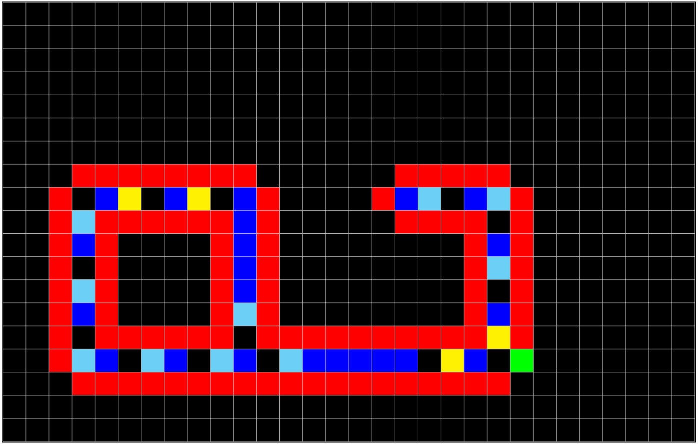
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

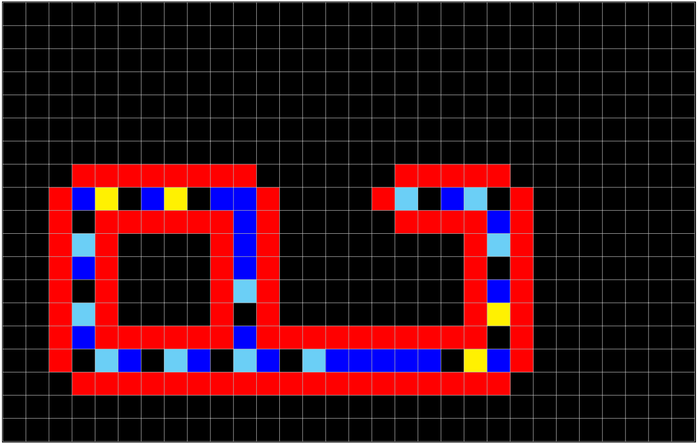
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

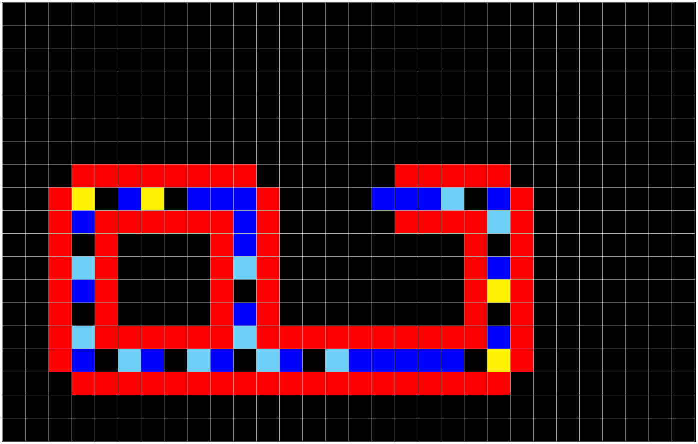
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

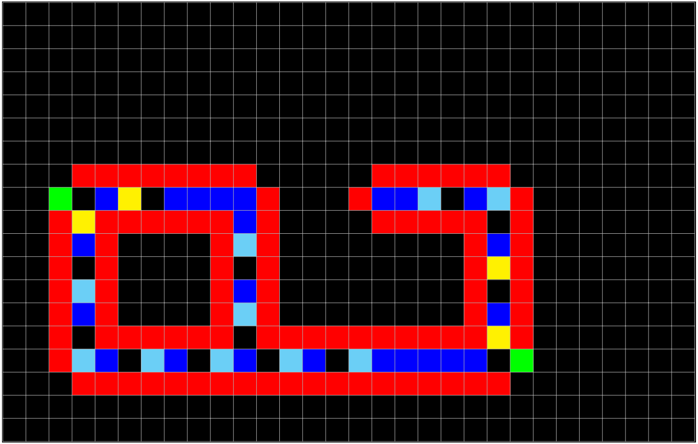
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

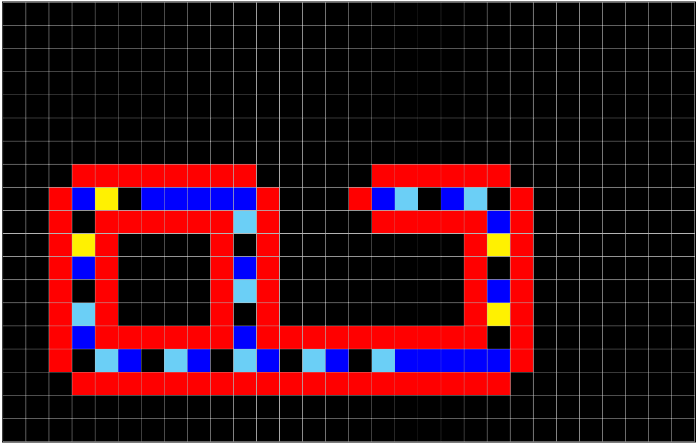
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

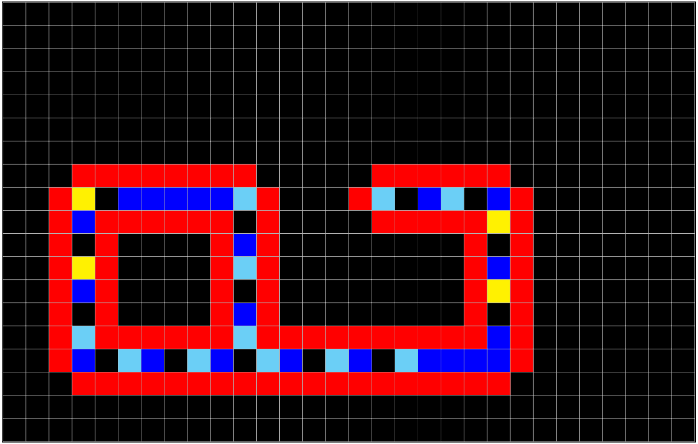
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

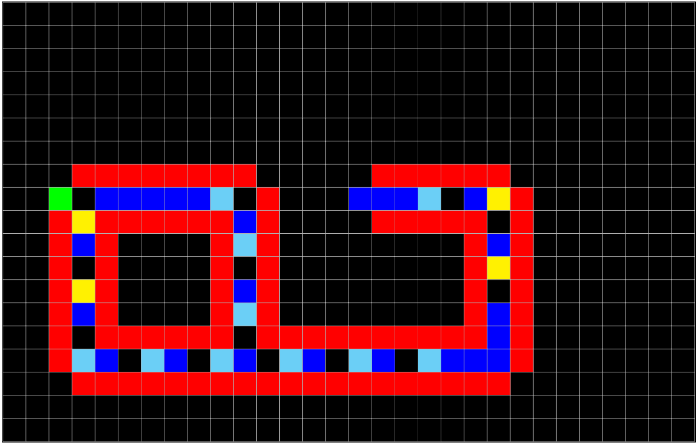
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

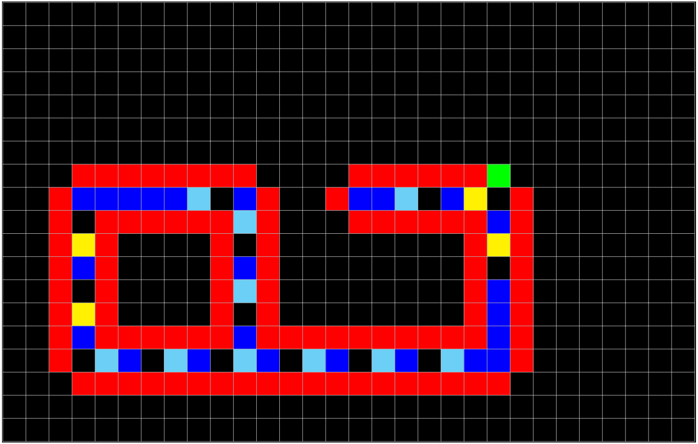
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

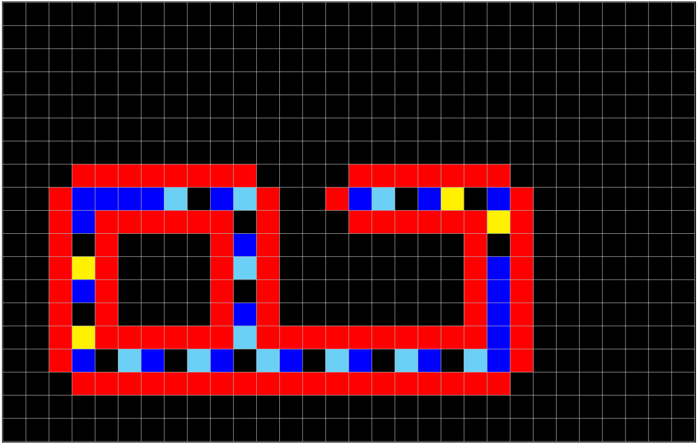
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

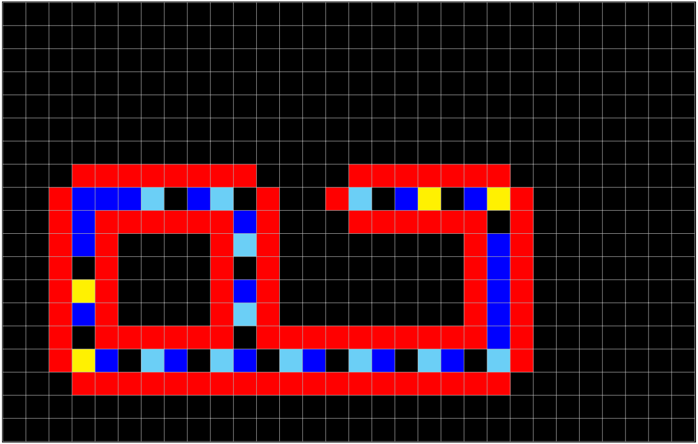
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

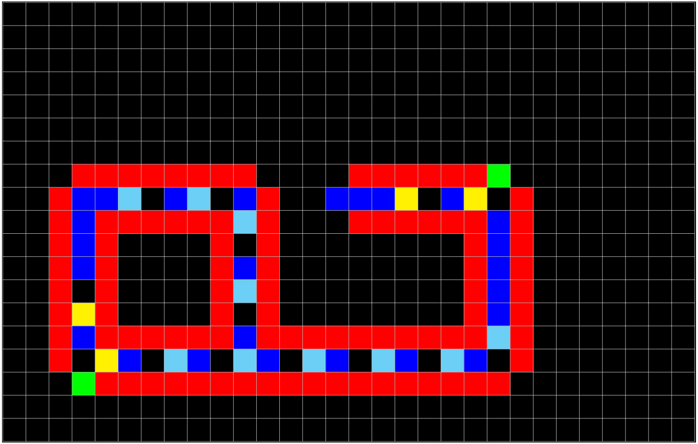
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

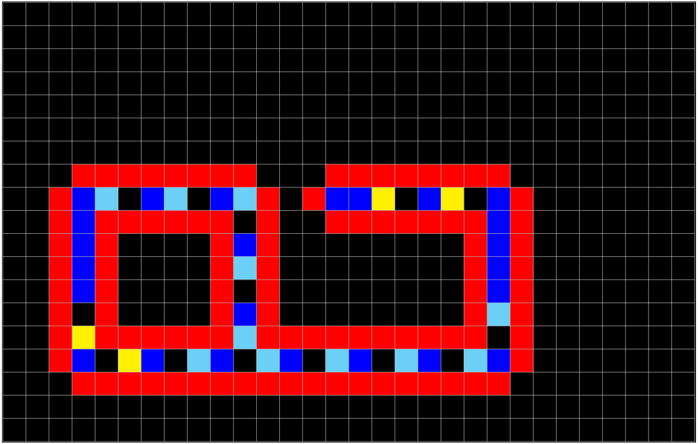
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

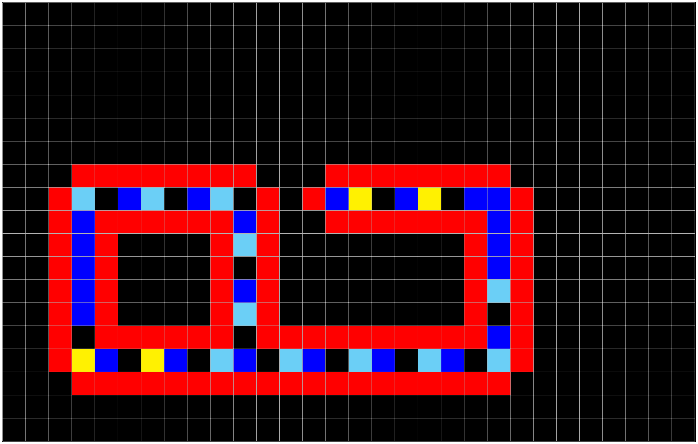
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

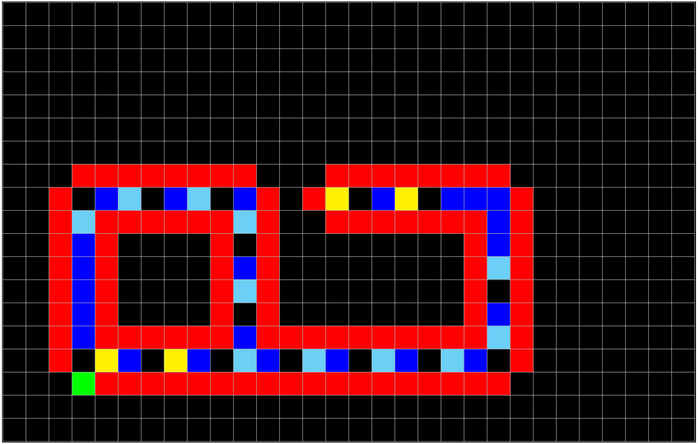
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

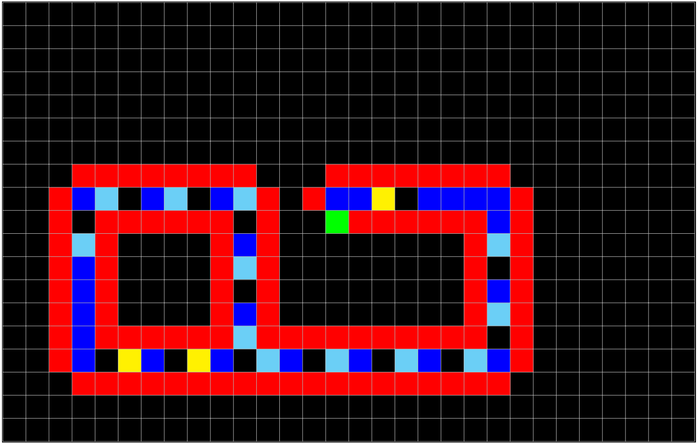
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

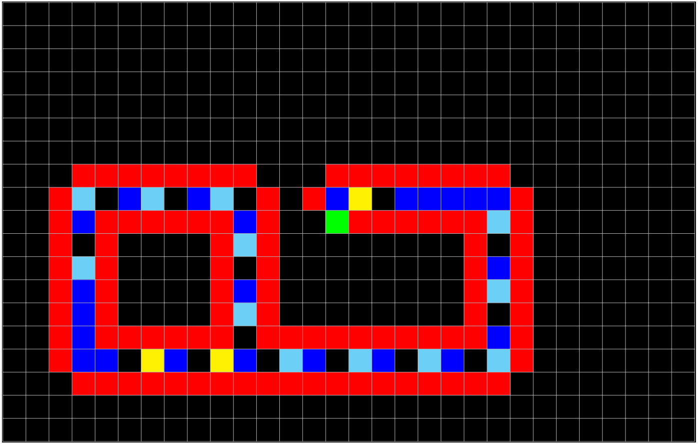
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

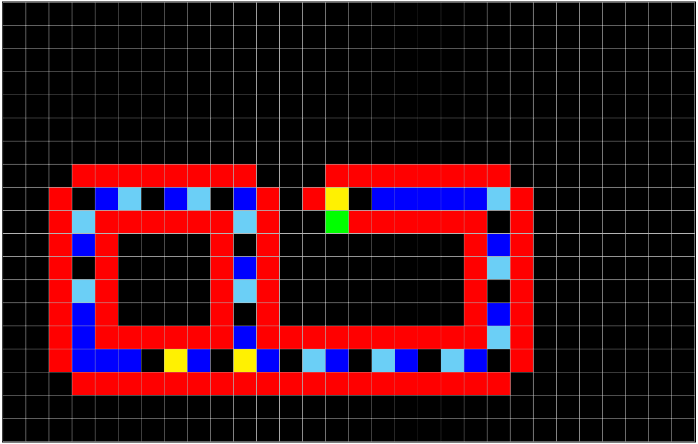
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

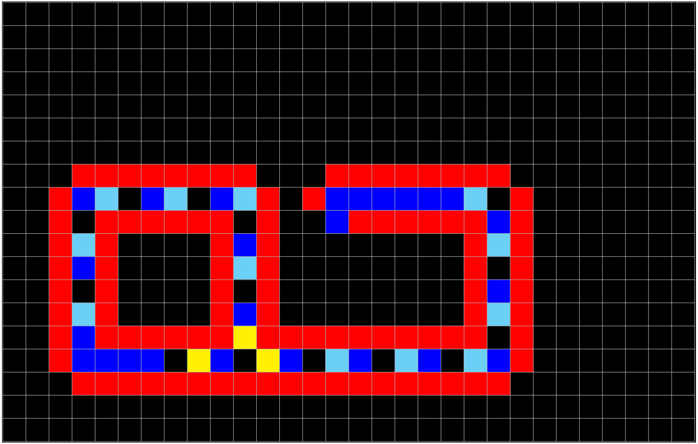
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

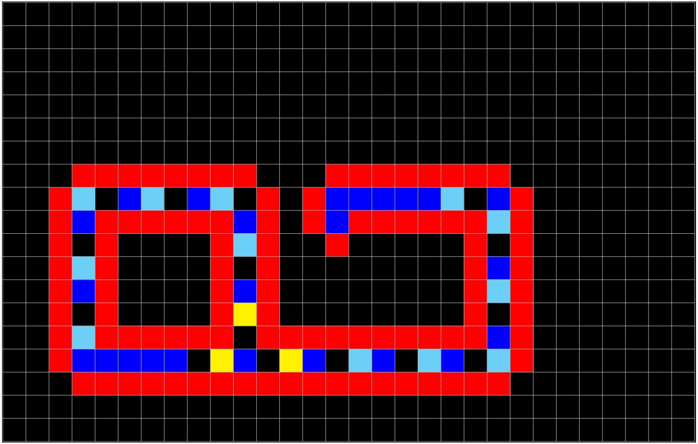
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

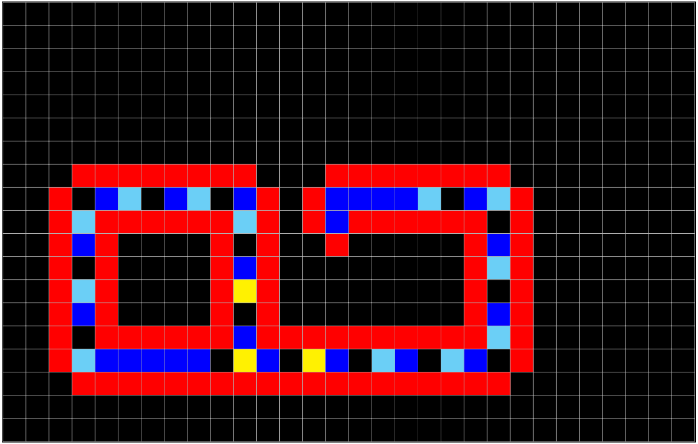
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

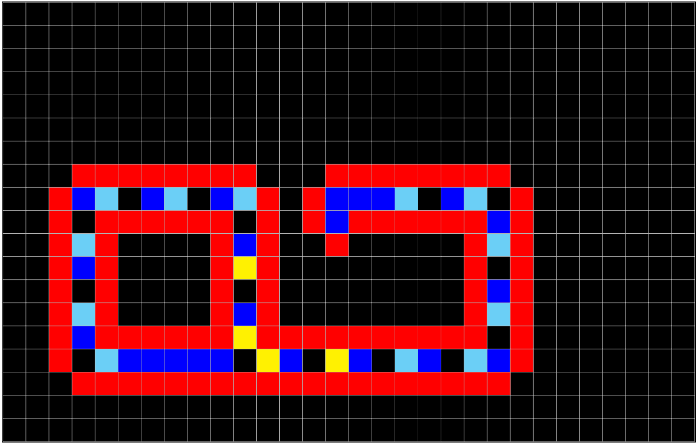
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

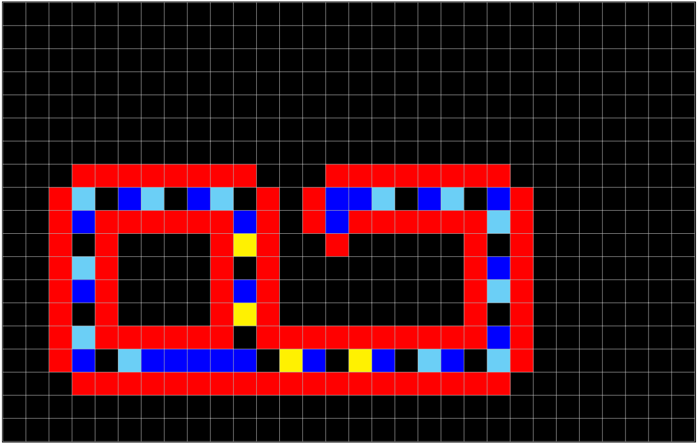
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

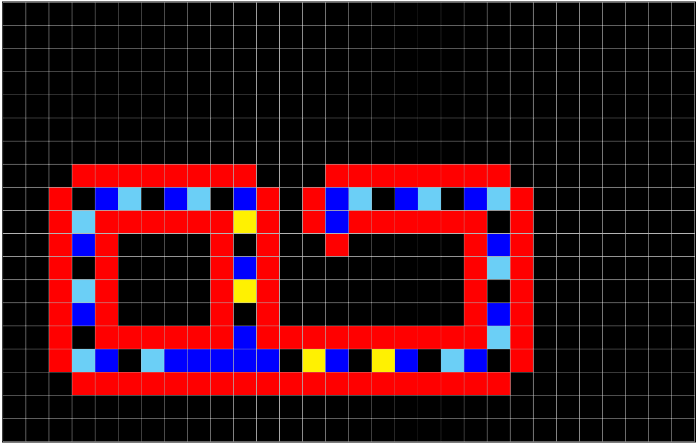
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

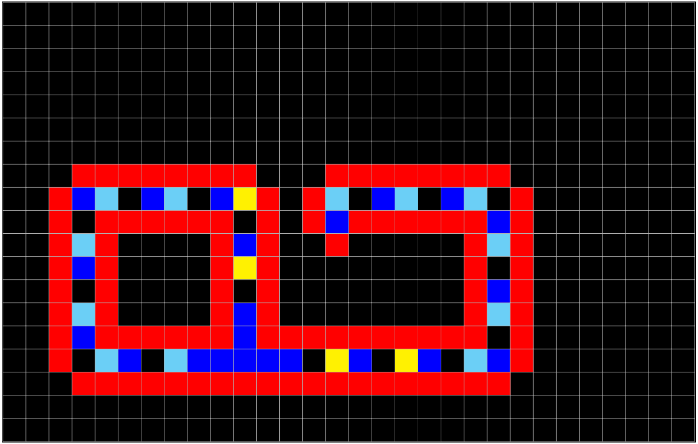
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

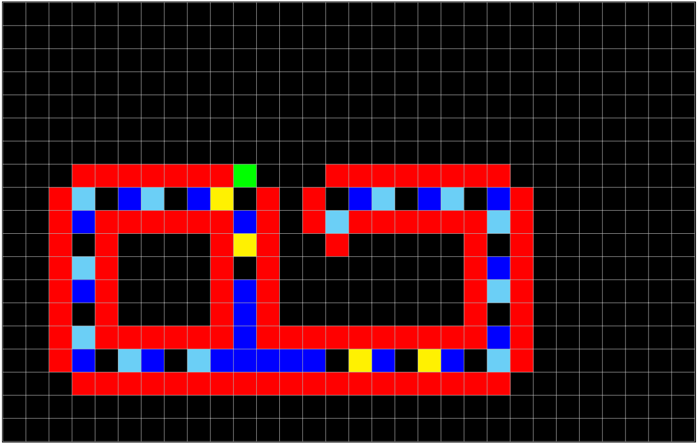
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

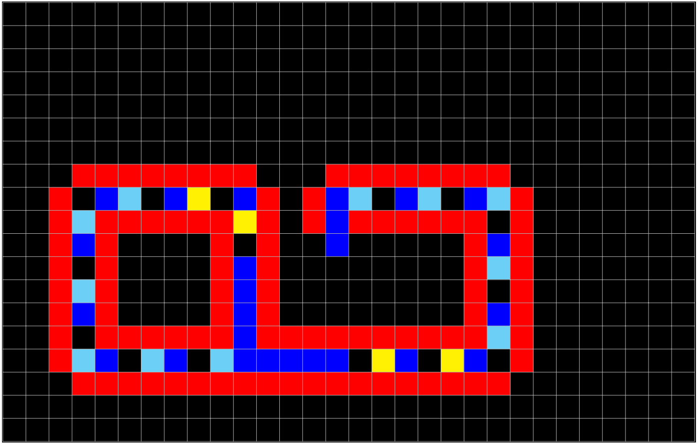
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

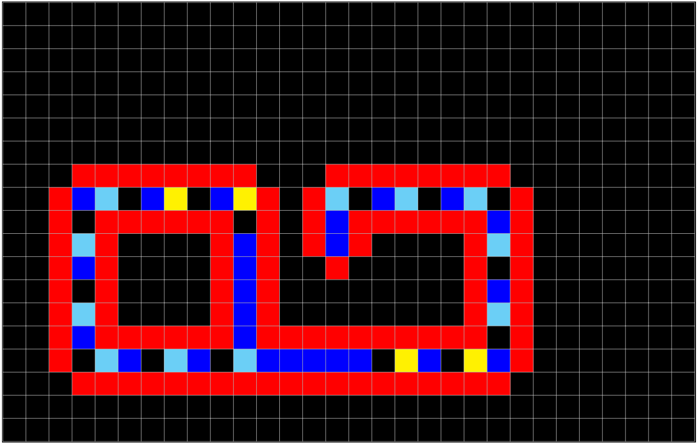
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

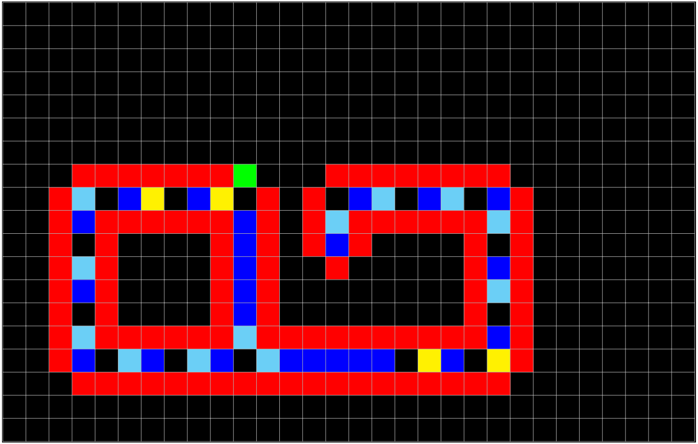
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

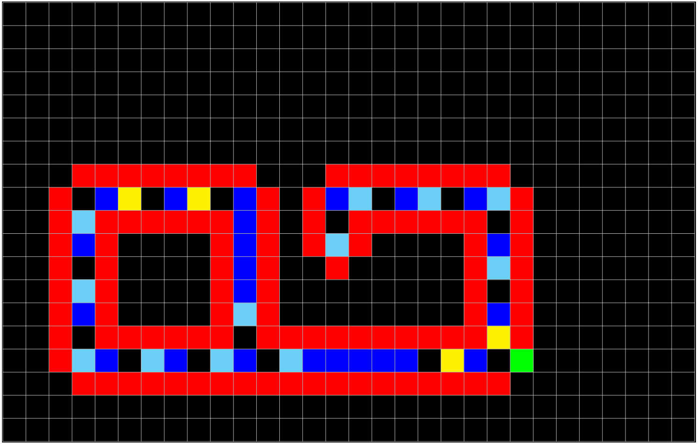
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

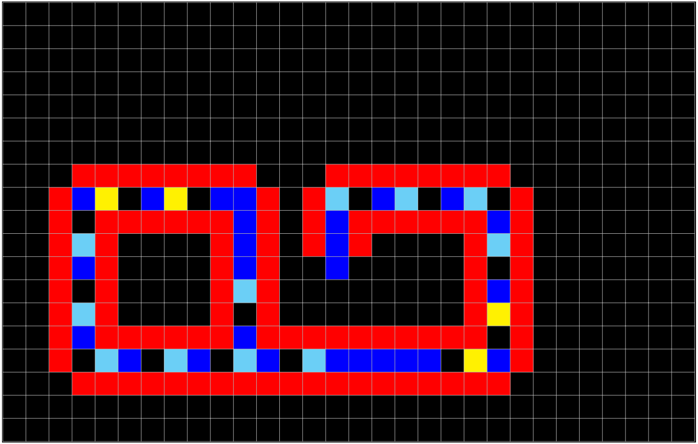
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

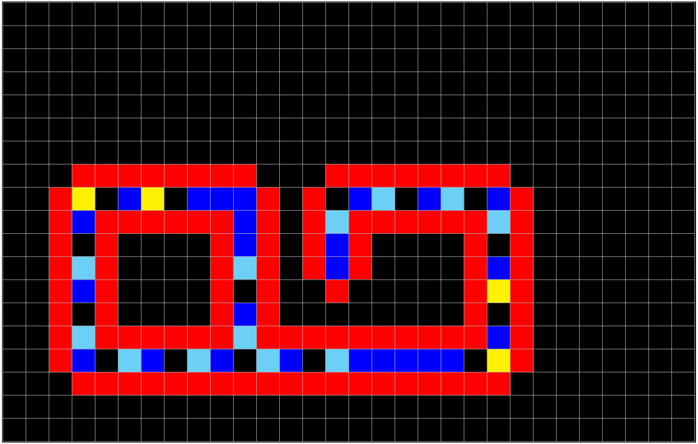
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

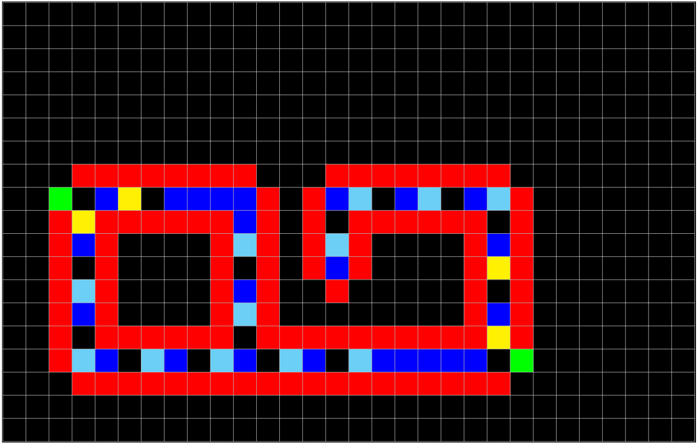
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

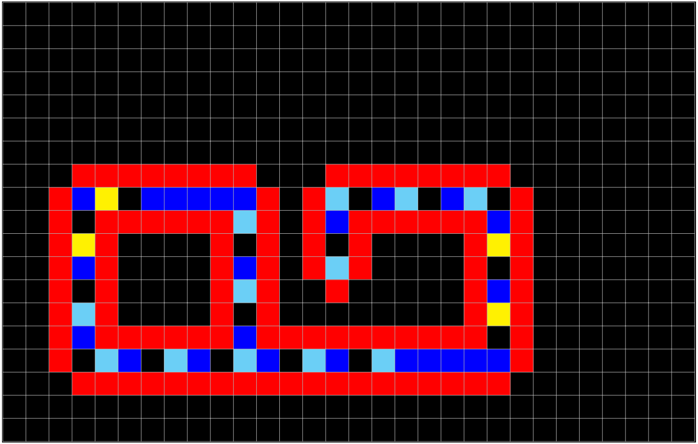
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

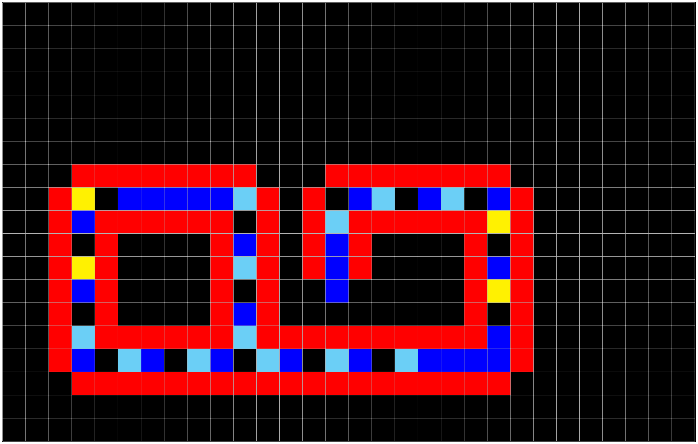
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

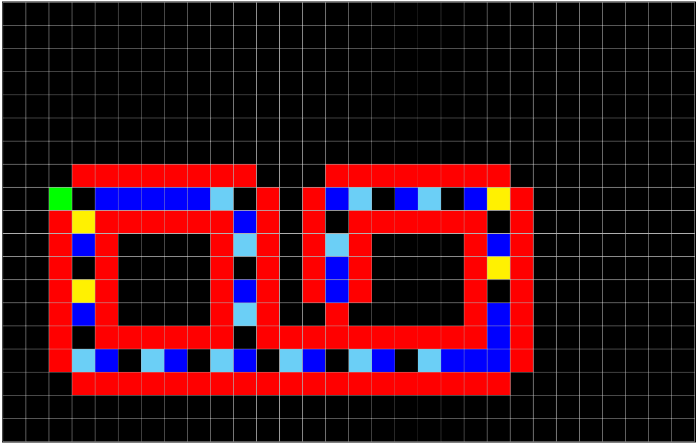
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

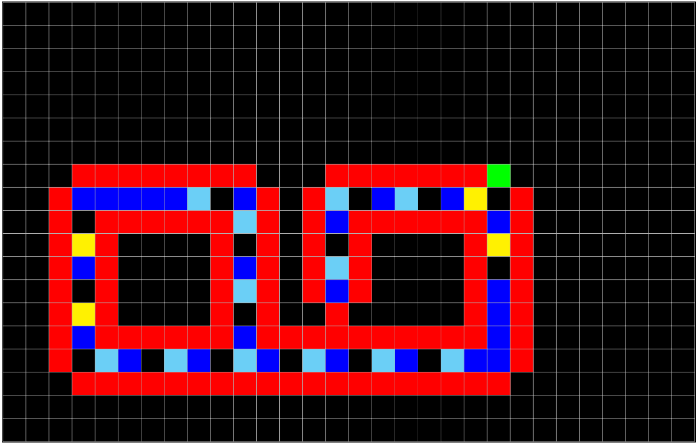
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

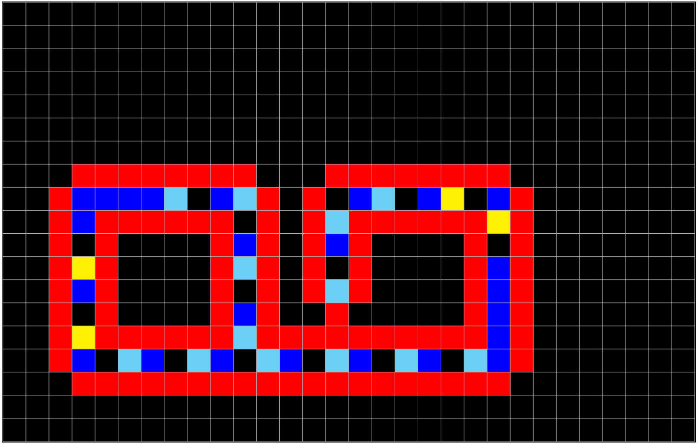
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

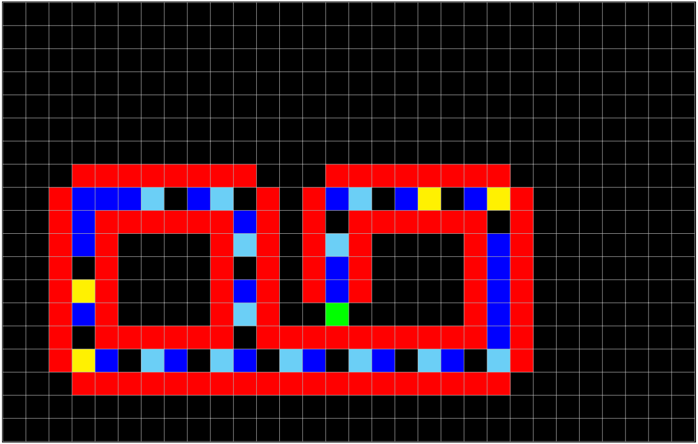
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

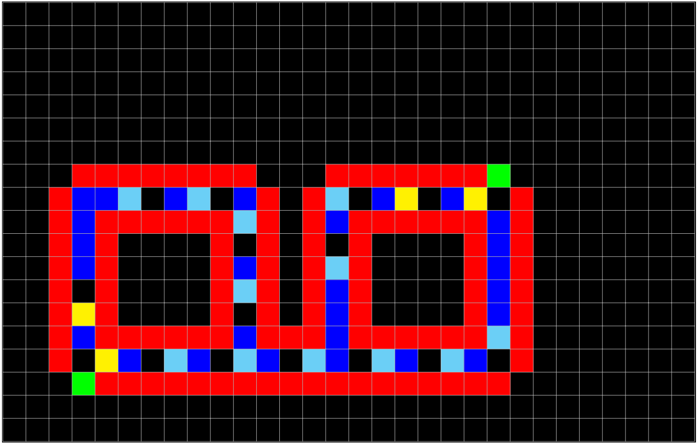
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

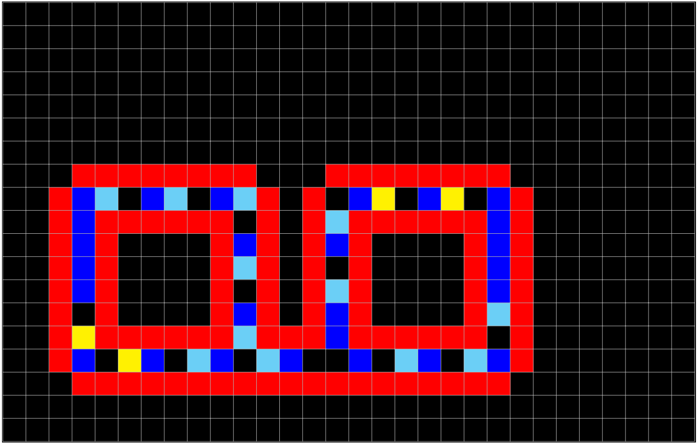
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

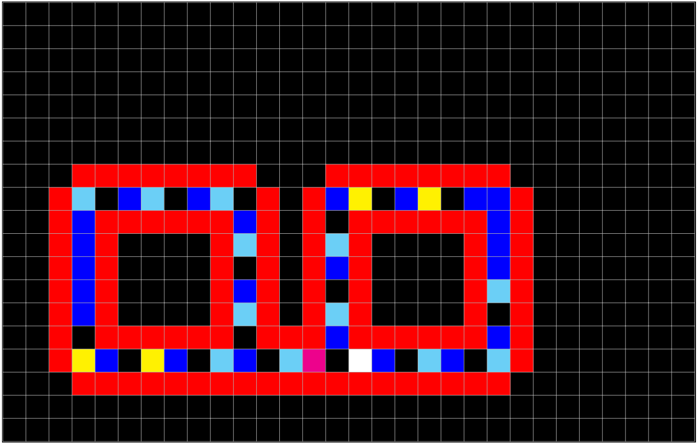
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

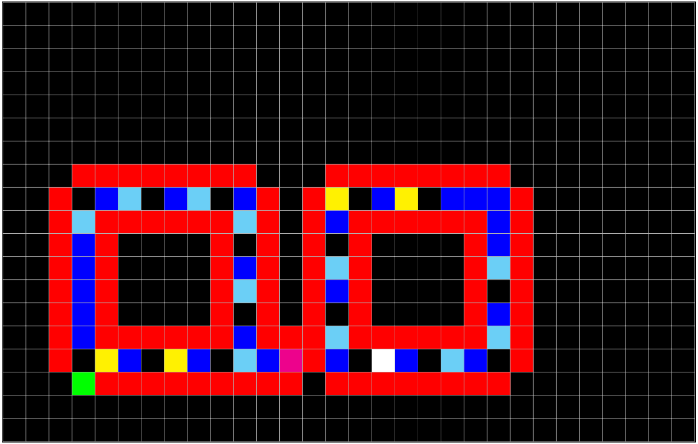
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

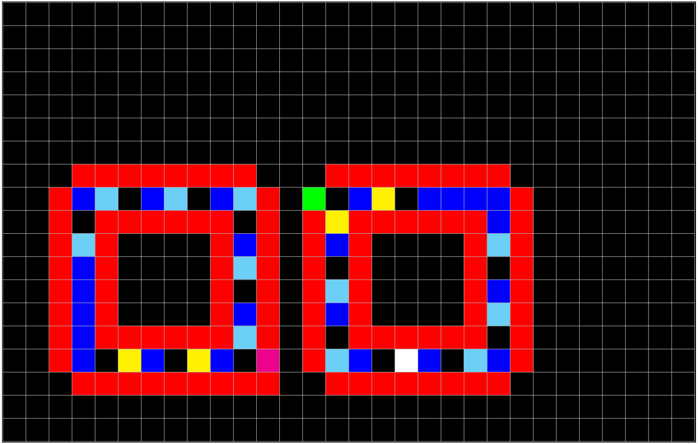
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

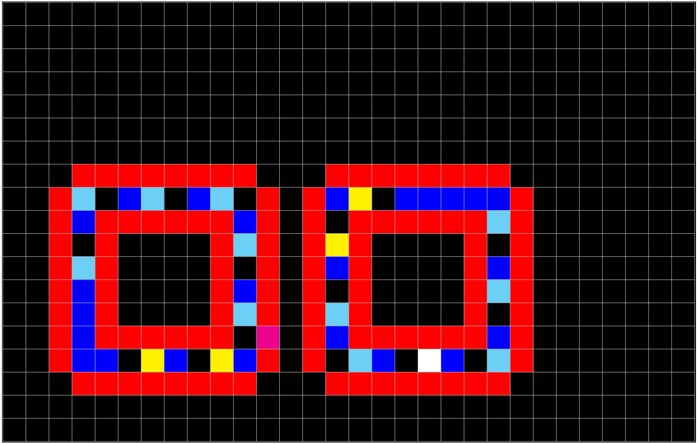
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

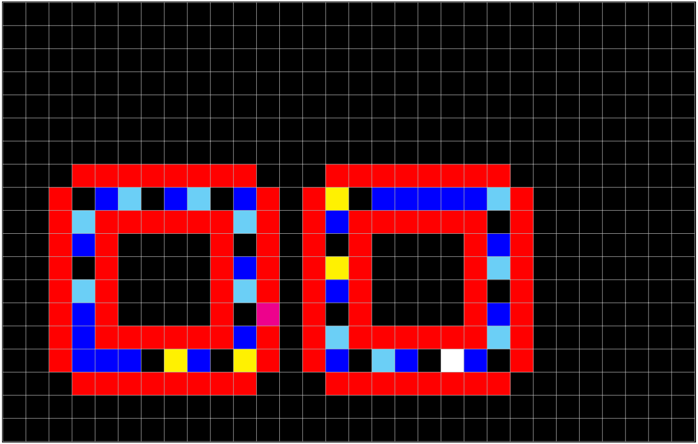
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

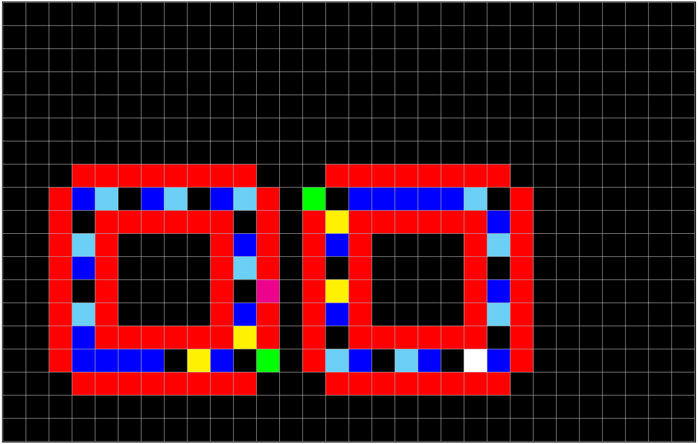
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

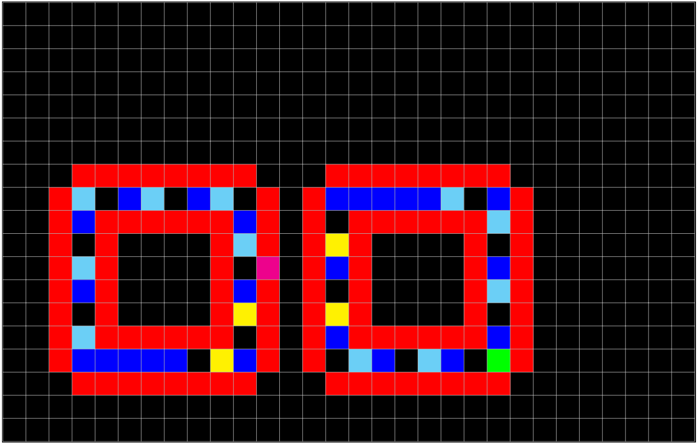
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

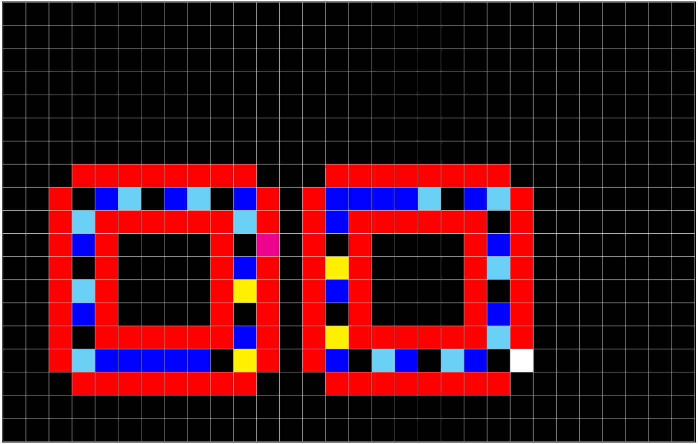
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

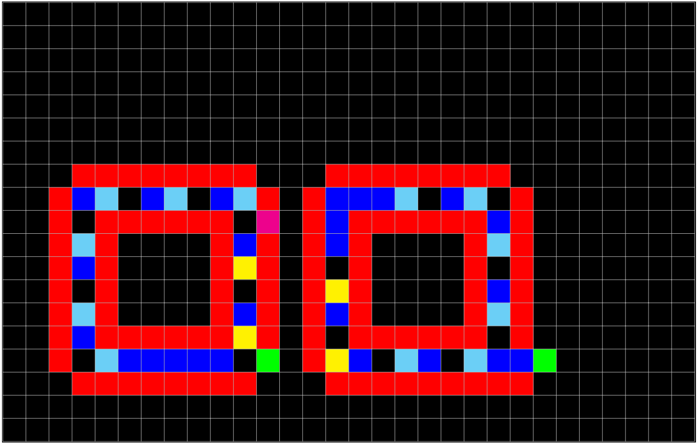
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

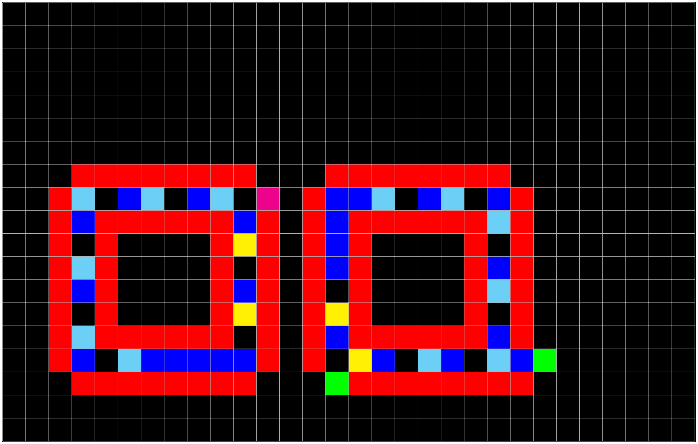
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

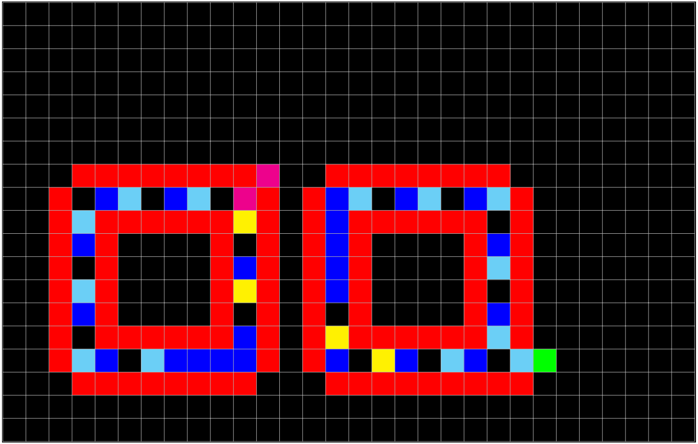
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

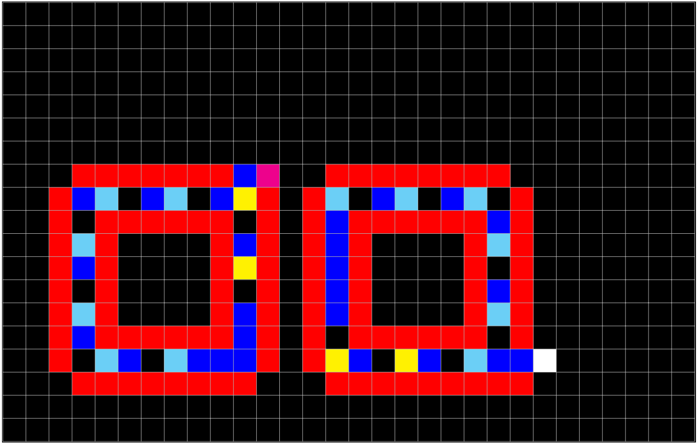
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

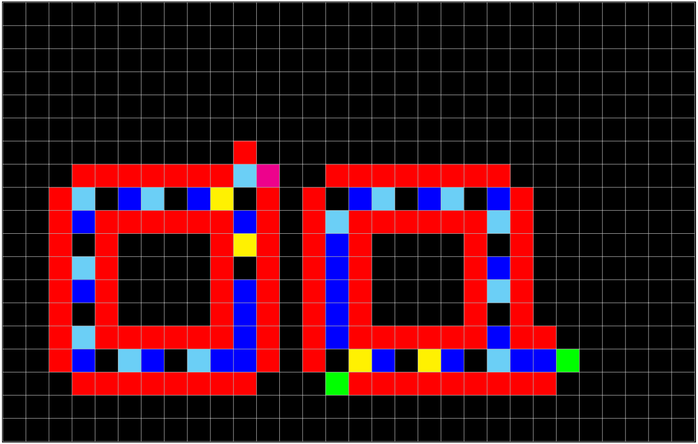
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

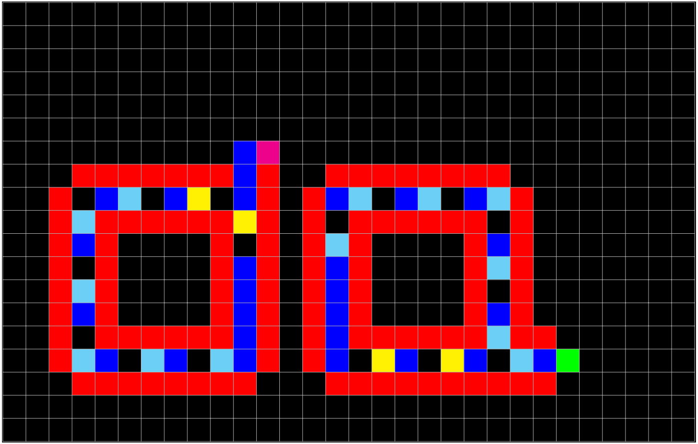
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

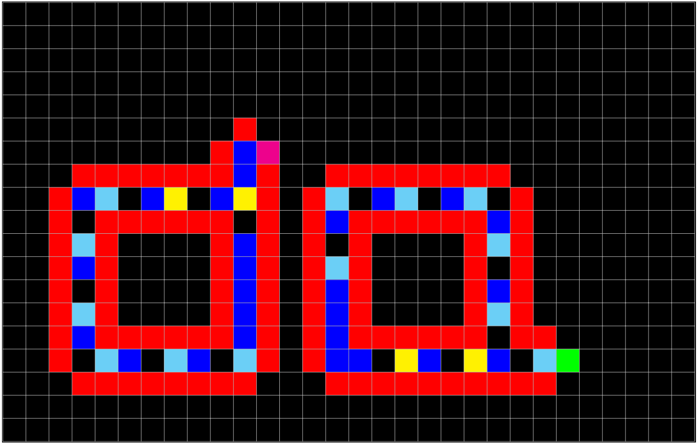
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

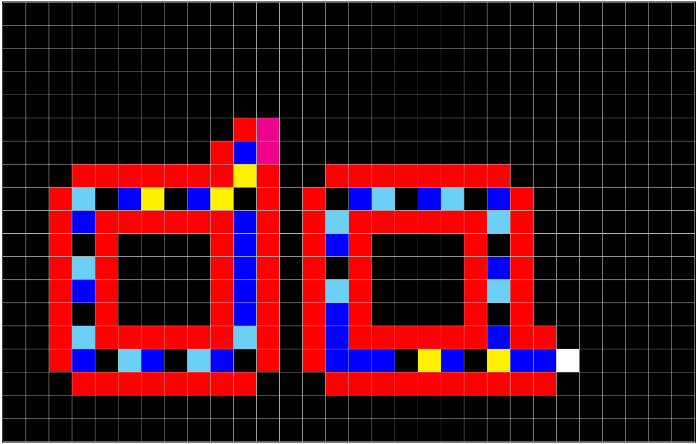
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

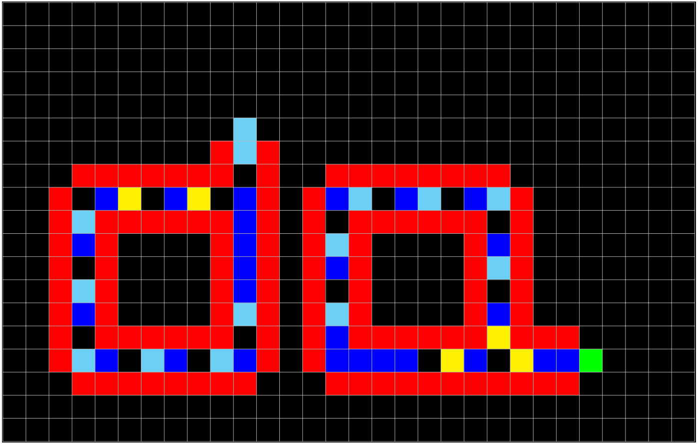
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

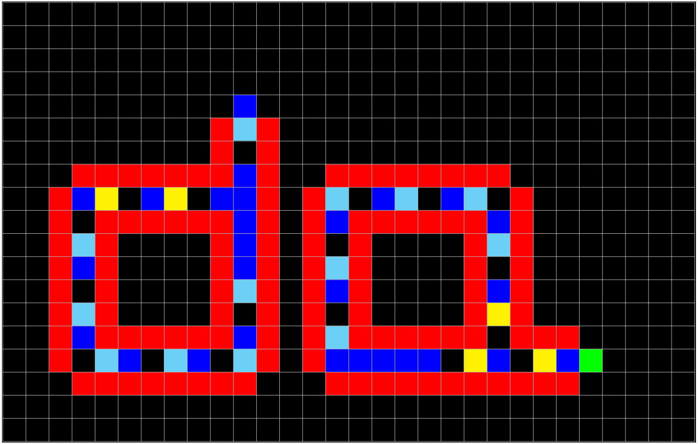
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

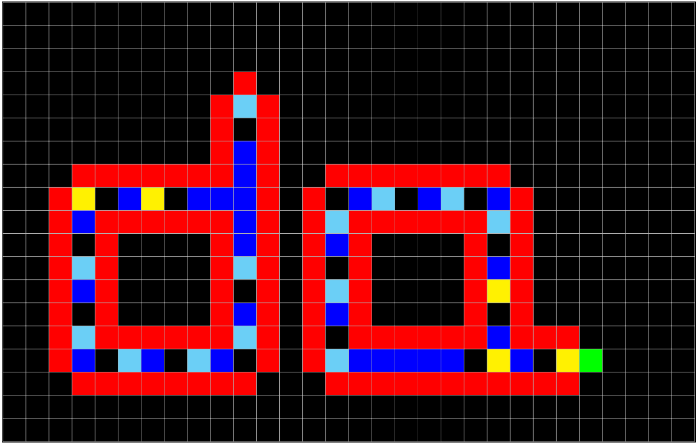
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

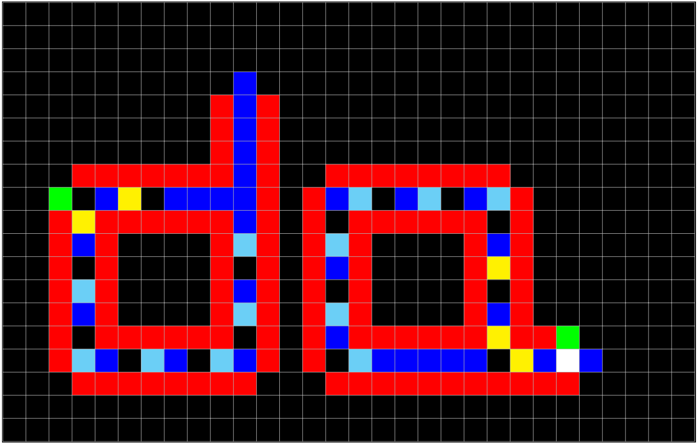
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

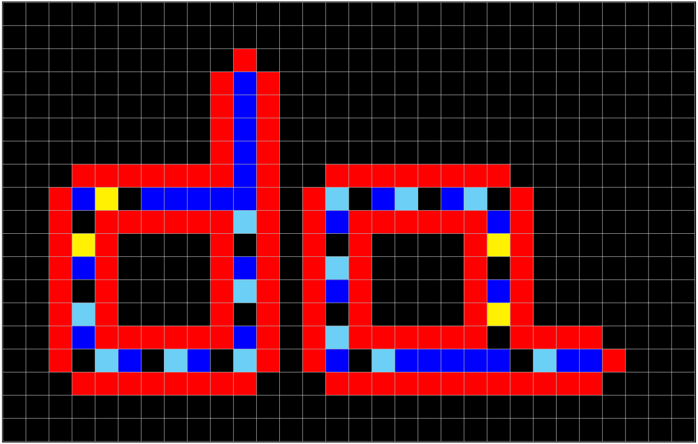
# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

*(Langton, 1984)*

# Langton self-reproducing loops



Langton modifies Codd rule to permit **non universal rotation invariant** self-reproduction by 86 cells in 151 steps.

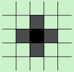
*(Langton, 1984)*

# The XOR rule

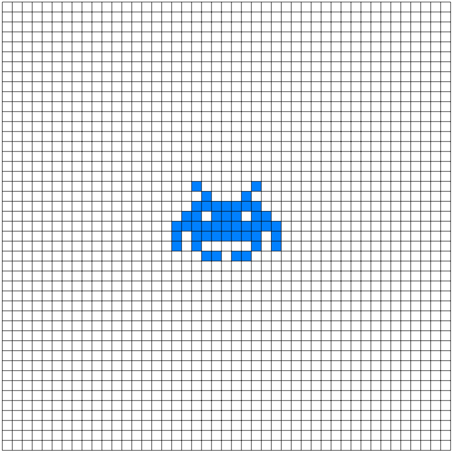
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



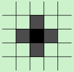
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

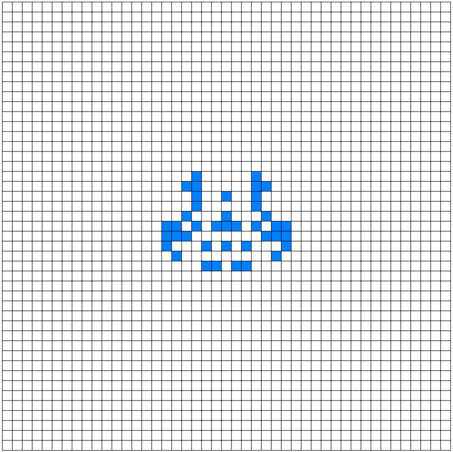
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



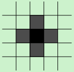
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

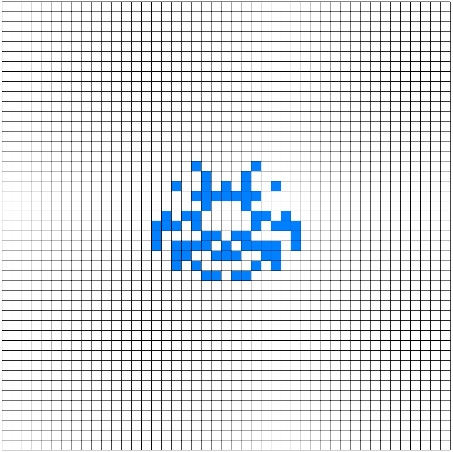
What is a **formal definition** of a self-reproducing CA?

**The XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



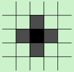
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

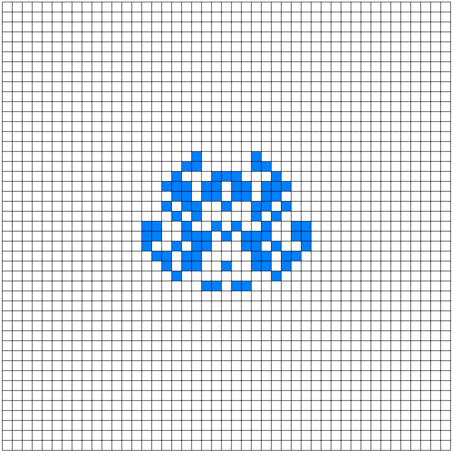
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



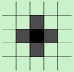
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

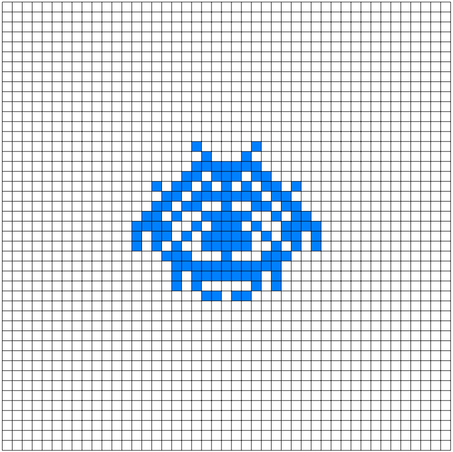
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



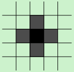
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

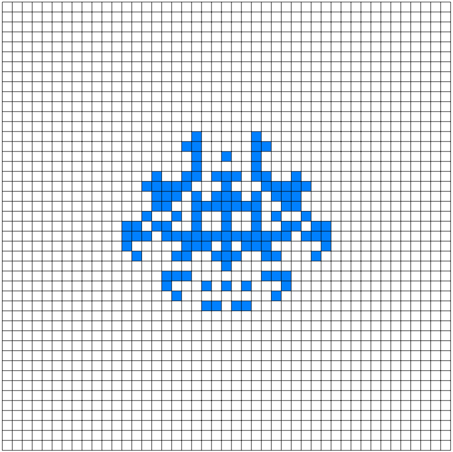
What is a **formal definition** of a self-reproducing CA?

**The XOR CA**

$$S = \mathbb{Z}_2$$

$$N =$$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



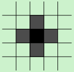
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

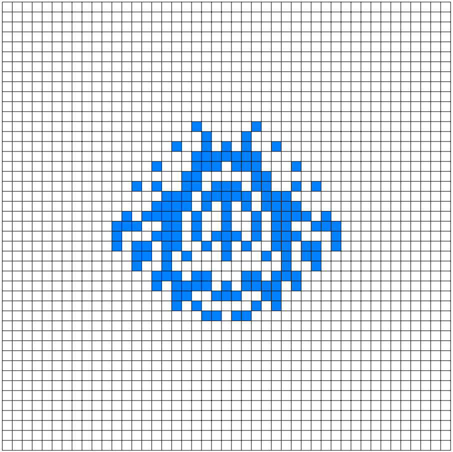
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



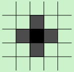
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

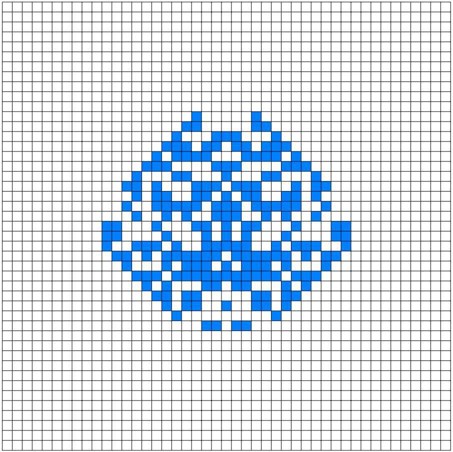
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



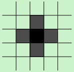
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

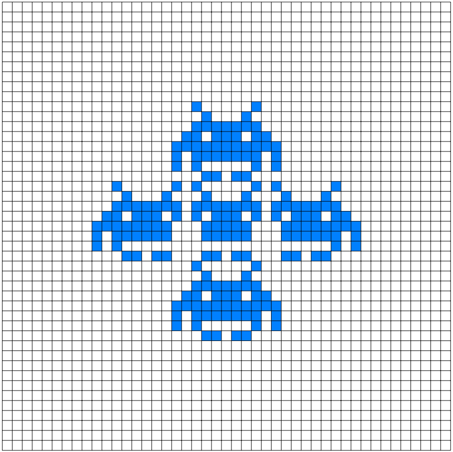
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



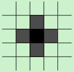
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

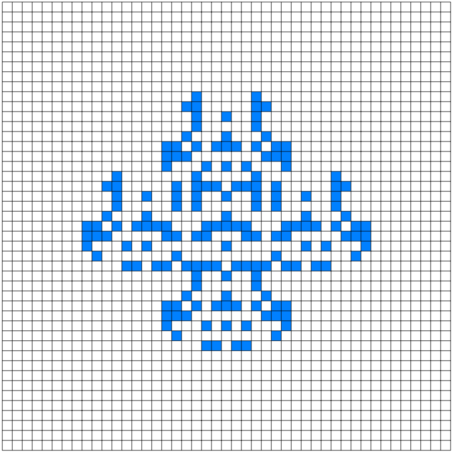
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



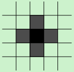
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

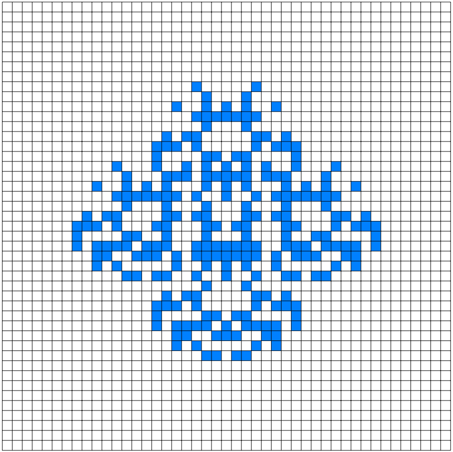
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



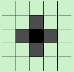
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

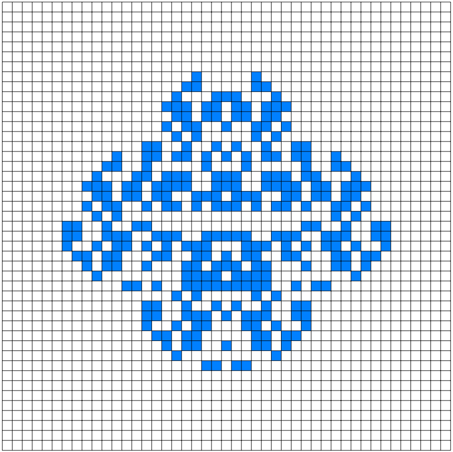
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



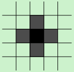
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

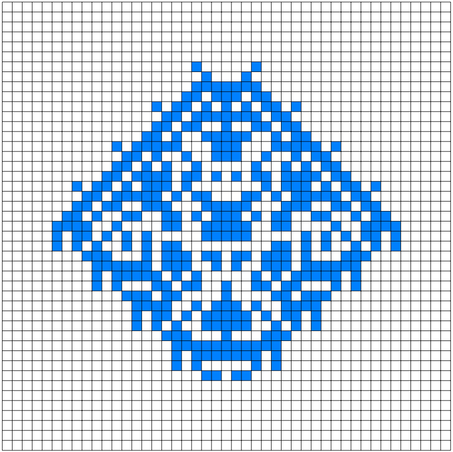
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



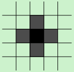
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

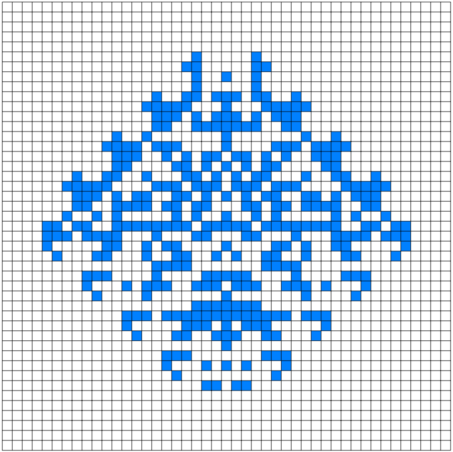
What is a **formal definition** of a self-reproducing CA?

**The XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



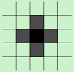
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

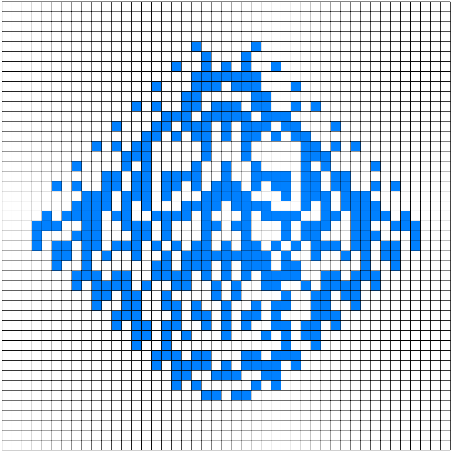
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



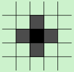
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

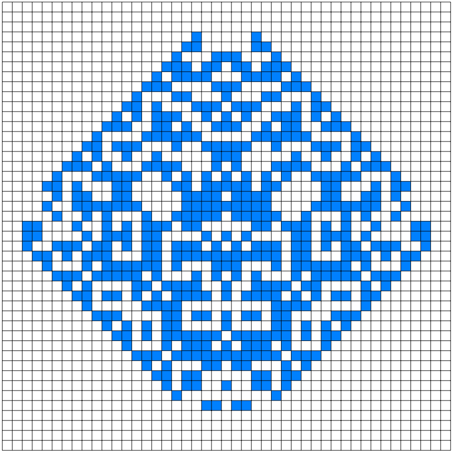
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



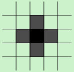
Is the XOR CA a **fair example** of a self-reproducing CA?

# The XOR rule

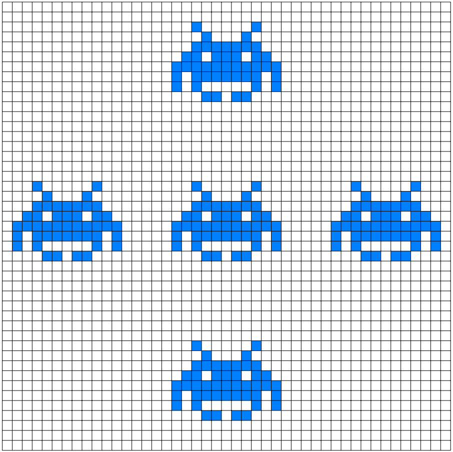
What is a **formal definition** of a self-reproducing CA?

The **XOR CA**

$$S = \mathbb{Z}_2$$

$N =$ 

$$f(x_i) = \sum_i x_i \pmod 2$$



Is the XOR CA a **fair example** of a self-reproducing CA?

# Outline of the tutorial

**Part I Computing inside cellular space**

***Engineering*** *CA and configurations to achieve computational tasks. Universalities. Massively parallel computing.*

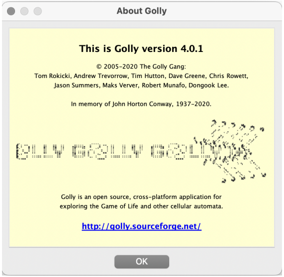
**Part II Computing properties of cellular automata**

***Analyzing*** *given CA to decide both immediate and dynamical properties. Classical results.*

**Part III Computation and reduction: undecidability results**

***Reducing*** *instances of undecidable problems to CA and configurations to prove undecidability results. Lots of properties of CA are undecidable.*

# Going further


**Section I Cellular Automata**

http://golly.sf.net/

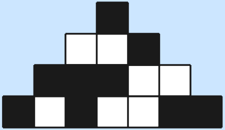
https://lstu.fr/cirm24

# Part I

# Computing
# inside the cellular space

# Part I
# Computing
# inside the cellular space

**1. Cellular automata**



2. A universal model of computation

3. A model of parallel computation

# Cellular automata

**Definition** A **CA** is a tuple $(d, S, N, f)$ where $S$ is a **finite set of states**, $N \subseteq_{\text{finite}} \mathbb{Z}^d$ is the finite **neighborhood** and $f : S^N \to S$ is the **local rule** of the cellular automaton.

A **configuration** $c \in S^{\mathbb{Z}^d}$ is a coloring of $\mathbb{Z}^d$ by $S$.

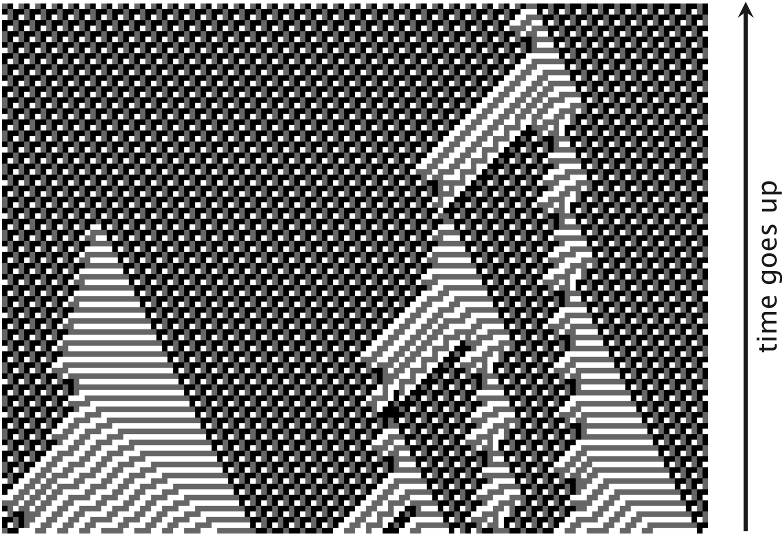
The **global map** $G : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ applies $f$ uniformly and locally:

$$\forall c \in S^{\mathbb{Z}}, \forall z \in \mathbb{Z}^d, \qquad F(c)(z) = f(c_{|z+N}).$$

A **space-time diagram** $\Delta \in S^{\mathbb{Z}^d \times \mathbb{N}}$ satisfies, for all $t \in \mathbb{N}$,

$$\Delta(t+1) = F(\Delta(t)).$$

# Space-time diagram



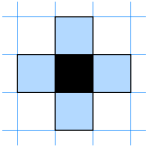

$S = \{0, 1, 2\}$, $r = 1$, $f(x, y, z) = \lfloor 6450288690466/3^{9x+3y+z} \rfloor \pmod 3$

# 2D cellular automata

Typical **high dimension CA** in this tutorial: $c \in S^{\mathbb{Z}^2}$.

Classical **von Neumann** neighborhood :

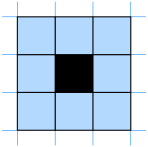
$$N_{\mathsf{vN}} = \{0\} \times \{-1,0,1\} \cup \{-1,0,1\} \times \{0\}$$



von Neumann

Classical **Moore** neighborhood :

$$N_{\mathsf{Moore}} = \{-1,0,1\} \times \{-1,0,1\}$$



Moore

# 1D cellular automata

More restricted **low dimension CA**, easier to analyze: $c \in S^{\mathbb{Z}}$.

Classical **first neighbors** neighborhood :

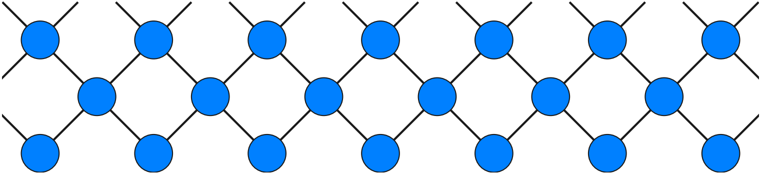
$$N_{\text{first}} = \{-1, 0, 1\}$$

1D

Classical **one-way** neighborhood :

$$N_{\text{OCA}} = \{0, 1\}$$

OCA



# Configurations

The set of configurations, $S^{\mathbb{Z}^d}$, is **uncountable**. What reasonnable **countable subset** can we consider?

**Recursive configurations** are useless, **undecidability** is everywhere (Rice theorem).

**Finite configurations** with a quiescent state.

**Periodic configurations** are ultimately periodic.

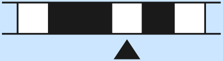
**Ultimately periodic configurations** compromise.

Thanks to **locality**, one can also consider **partial space-time diagrams** to study all configurations.

# Part I
# Computing
# inside the cellular space

1. Cellular automata

**2. A universal model of computation**



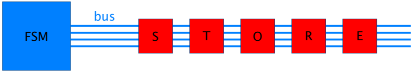
3. A model of parallel computation

# Universality in higher dimensions

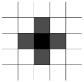
Construction of universal CA appeared with CA as a tool to embed computation into the CA world. First, for **2D CA**

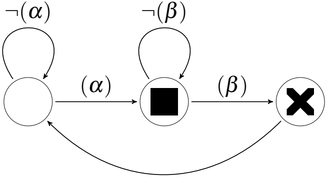
| | | | |
|---|---|---|---|
| 1966 | von Neumann | 5 | 29 |
| 1968 | Codd | 5 | 8 |
| 1970 | Conway | 8 | 2 |
| 1970 | Banks | 5 | 2 |

A natural idea in 2D is to emulate **universal boolean circuits** by embedding ingredients into the CA space: **signals**, **wires**, **turns**, **fan-outs**, **gates**, **delays**, **clocks**, *etc.*



# Copper

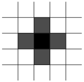
$\left(\mathbb{Z}^2, \{\square, \blacksquare, \boldsymbol{\times}\},\right.$  $\left., \delta\right)$

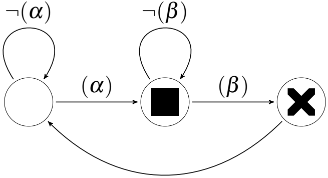



($\alpha$) both north and south, or east and west, neighbors in state $\boldsymbol{\times}$ or $\blacksquare$;

($\beta$) at least two neighbors in state $\boldsymbol{\times}$ or $\blacksquare$ and either exactly one neighbor in state $\boldsymbol{\times}$ or exactly one neighbor in state $\blacksquare$.

# Copper

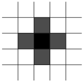
$\Big(\mathbb{Z}^2, \{\square, \blacksquare, \boldsymbol{\times}\},$  $, \delta\Big)$

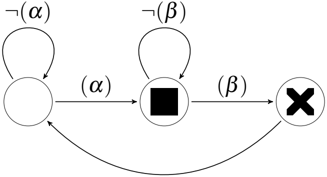



($\alpha$) both north and south, or east and west, neighbors in state $\boldsymbol{\times}$ or $\blacksquare$ ;

($\beta$) at least two neighbors in state $\boldsymbol{\times}$ or $\blacksquare$ and either exactly one neighbor in state $\boldsymbol{\times}$ or exactly one neighbor in state $\blacksquare$.

# Copper

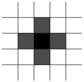
$$\left(\mathbb{Z}^2, \{\square, \blacksquare, \boldsymbol{\times}\},\right.$$

$$\left., \delta\right)$$

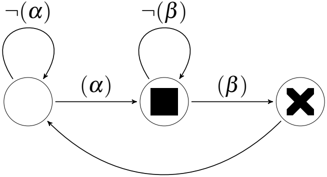



($\alpha$) both north and south, or east and west, neighbors in state $\boldsymbol{\times}$ or $\blacksquare$ ;

($\beta$) at least two neighbors in state $\boldsymbol{\times}$ or $\blacksquare$ and either exactly one neighbor in state $\boldsymbol{\times}$ or exactly one neighbor in state $\blacksquare$.

# Copper

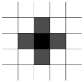
$\Big(\mathbb{Z}^2, \{\square, \blacksquare, \boldsymbol{\times}\},$  $, \delta\Big)$

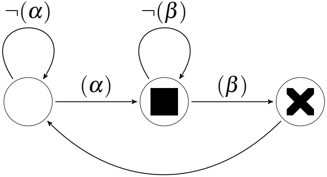



$(\alpha)$ both north and south, or east and west, neighbors in state $\boldsymbol{\times}$ or $\blacksquare$;

$(\beta)$ at least two neighbors in state $\boldsymbol{\times}$ or $\blacksquare$ and either exactly one neighbor in state $\boldsymbol{\times}$ or exactly one neighbor in state $\blacksquare$.

# Copper

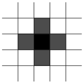
$$\left(\mathbb{Z}^2, \{\square, \blacksquare, \boldsymbol{\times}\},\right.$$  $$\left., \delta\right)$$

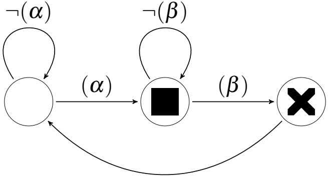



($\alpha$) both north and south, or east and west, neighbors in state $\boldsymbol{\times}$ or $\blacksquare$ ;

($\beta$) at least two neighbors in state $\boldsymbol{\times}$ or $\blacksquare$ and either exactly one neighbor in state $\boldsymbol{\times}$ or exactly one neighbor in state $\blacksquare$.

# Copper

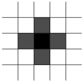
$\Big(\mathbb{Z}^2, \{\square, \blacksquare, \boldsymbol{\times}\},$  $, \delta\Big)$

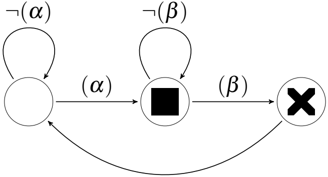



($\alpha$) both north and south, or east and west, neighbors in state $\boldsymbol{\times}$ or $\blacksquare$ ;

($\beta$) at least two neighbors in state $\boldsymbol{\times}$ or $\blacksquare$ and either exactly one neighbor in state $\boldsymbol{\times}$ or exactly one neighbor in state $\blacksquare$.

# Copper

$\left(\mathbb{Z}^2, \{\square, \blacksquare, \boldsymbol{\times}\},\right.$  $\left., \delta\right)$

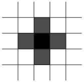


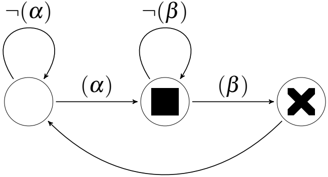

($\alpha$) both north and south, or east and west, neighbors in state ✖ or ■ ;

($\beta$) at least two neighbors in state ✖ or ■ and either exactly one neighbor in state ✖ or exactly one neighbor in state ■.

■ ✖■✖ ■

# Copper

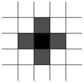
$$\left(\mathbb{Z}^2, \{\square, \blacksquare, \boldsymbol{\times}\},\right.$$  $$\left., \delta\right)$$

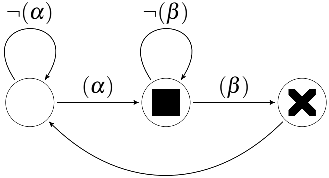



($\alpha$) both north and south, or east and west, neighbors in state $\boldsymbol{\times}$ or $\blacksquare$;

($\beta$) at least two neighbors in state $\boldsymbol{\times}$ or $\blacksquare$ and either exactly one neighbor in state $\boldsymbol{\times}$ or exactly one neighbor in state $\blacksquare$.

# Copper

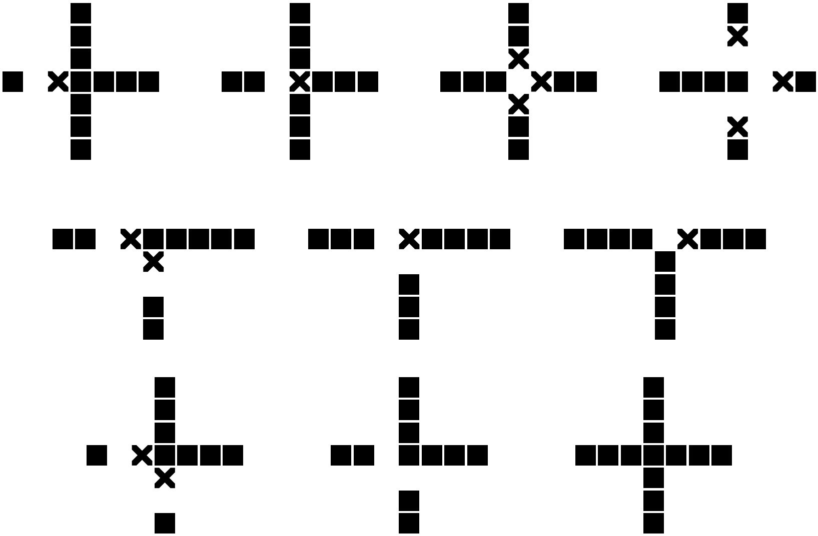
$$\left(\mathbb{Z}^2, \{\square, \blacksquare, \boldsymbol{\times}\},\right.$$  $$\left., \delta\right)$$



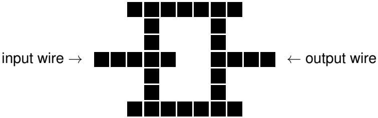

($\alpha$) both north and south, or east and west, neighbors in state ✖ or ■ ;

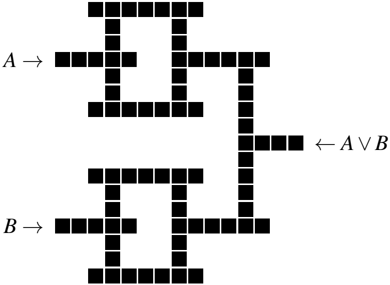
($\beta$) at least two neighbors in state ✖ or ■ and either exactly one neighbor in state ✖ or exactly one neighbor in state ■.

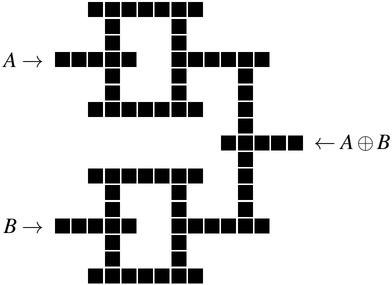
# Copper: Intersections



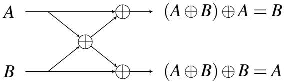
# Copper: Gates

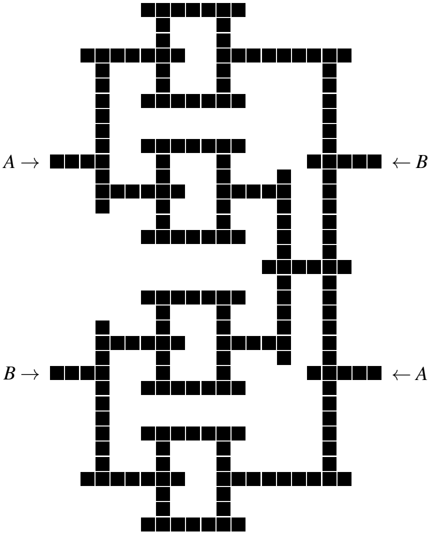


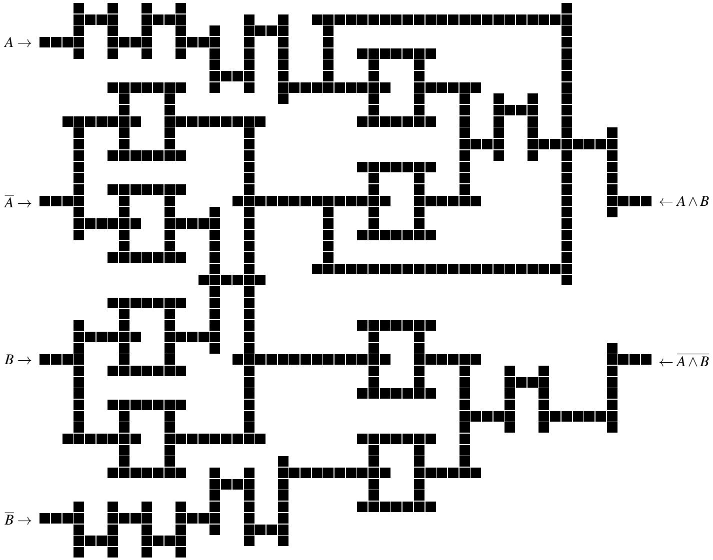



# Copper: Xing


$A$
$B$
$(A \oplus B) \oplus A = B$
$(A \oplus B) \oplus B = A$

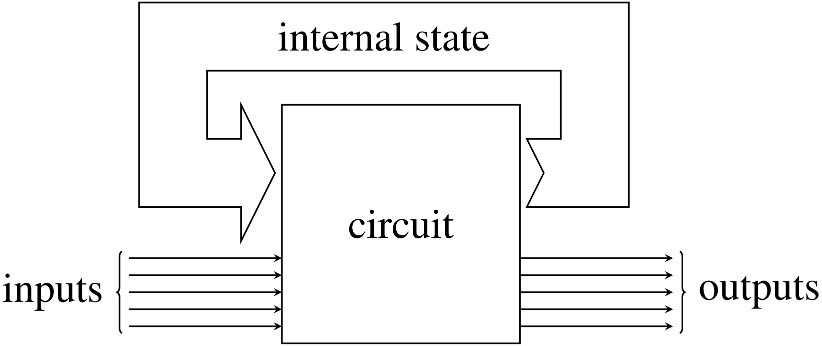

$A \rightarrow$
$\leftarrow B$
$B \rightarrow$
$\leftarrow A$

# Copper: AND



$A \rightarrow$
$\overline{A} \rightarrow$
$B \rightarrow$
$\overline{B} \rightarrow$
$\leftarrow A \wedge B$
$\leftarrow \overline{A \wedge B}$

# Copper: FSM



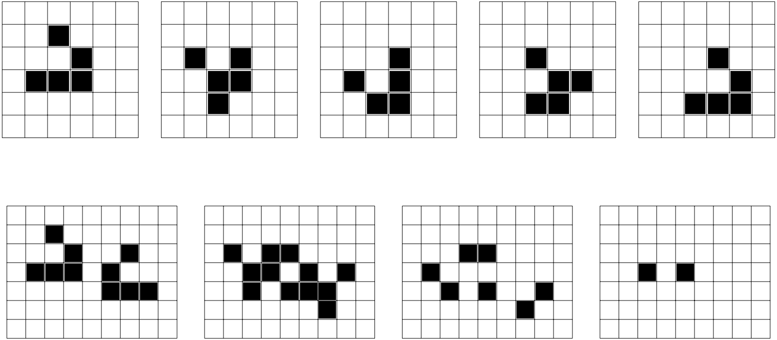

**Theorem** Copper is **universal for boolean circuits**.

Simulating a universal device requires an **ultimately periodic configuration** of **infinitely many** non quiescent cells.

# The Game of Life

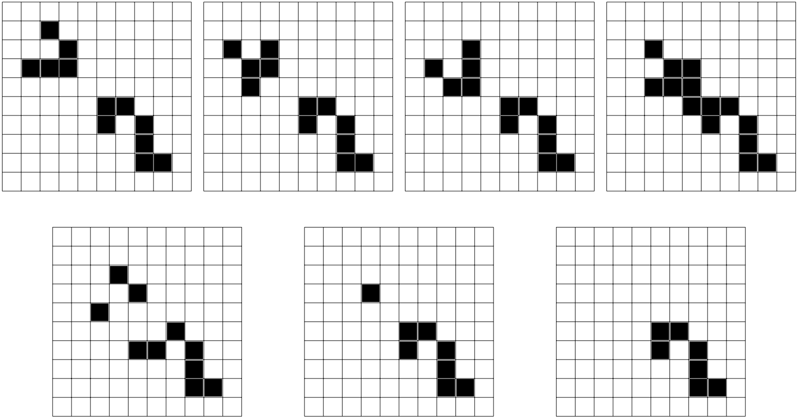
**Theorem** GoL is **universal for boolean circuits**.

The construction uses **gliders** as signals.

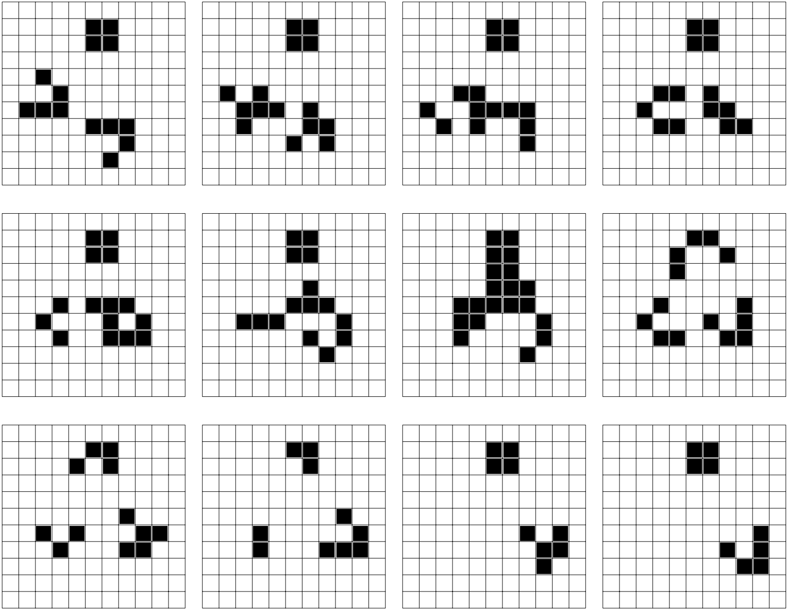


(Conway *et al.*, Winning Ways Vol. 2., 1971)

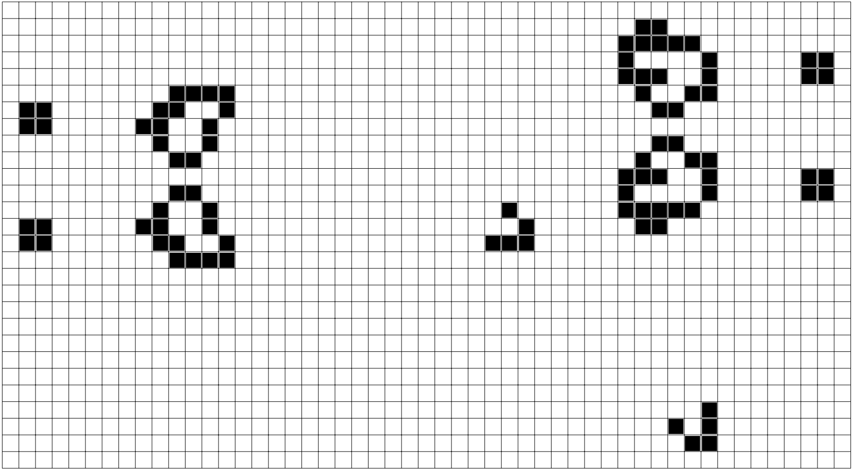
# GoL: Eater



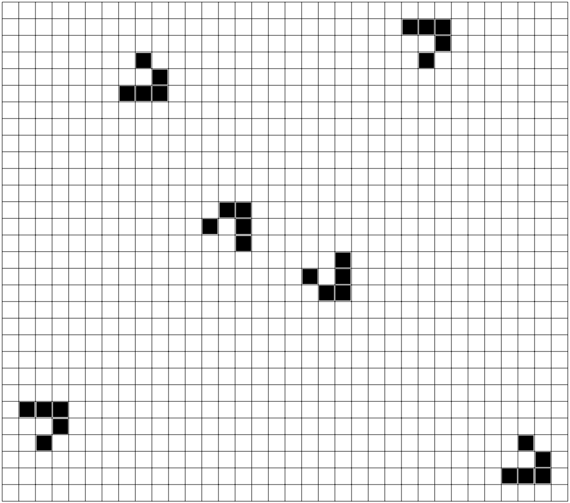
# GoL: Duplicator



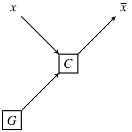
# GoL: Gosper's p46 Gun

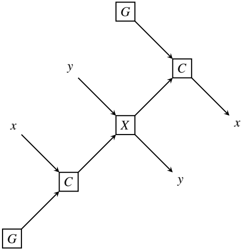


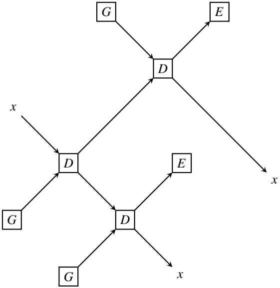
# GoL: Xing

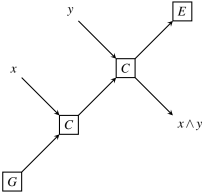


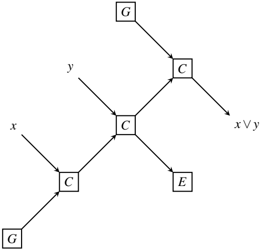
# GoL: Combining



# Universality in 1D

**Remark** Boolean circuits are **less intuitive** to simulate in 1D, but it is easy to simulate **sequential models of computation** like Turing machines.

(A. R. Smith III, Simple computation-universal cellular spaces, 1971)

| | | |
|---|---|---|
| 1971 | Smith III | 18 |
| 1987 | Albert & Culik II | 14 |
| 1990 | Lindgren & Nordhal | 7 |
| 2004 | Cook | 2 |

A cellular automaton is **Turing-universality** if

# Universality in 1D

**Remark** Boolean circuits are **less intuitive** to simulate in 1D, but it is easy to simulate **sequential models of computation** like Turing machines.

(A. R. Smith III, Simple computation-universal cellular spaces, 1971)

| | | |
|---|---|---|
| 1971 | Smith III | 18 |
| 1987 | Albert & Culik II | 14 |
| 1990 | Lindgren & Nordhal | 7 |
| 2004 | Cook | 2 |

A cellular automaton is **Turing-universality** if... What exactly is the **formal definition**?

# Universality in 1D

**Remark** Boolean circuits are **less intuitive** to simulate in 1D, but it is easy to simulate **sequential models of computation** like Turing machines.

(A. R. Smith III, Simple computation-universal cellular spaces, 1971)

| | | |
|---|---|---|
| 1971 | Smith III | 18 |
| 1987 | Albert & Culik II | 14 |
| 1990 | Lindgren & Nordhal | 7 |
| 2004 | Cook | 2 |

A cellular automaton is **Turing-universality** if... What exactly is the **formal definition**? What is a **non universal** CA?

# Universality in 1D

**Remark** Boolean circuits are **less intuitive** to simulate in 1D, but it is easy to simulate **sequential models of computation** like Turing machines.

(A. R. Smith III, Simple computation-universal cellular spaces, 1971)

| | | |
|---|---|---|
| 1971 | Smith III | 18 |
| 1987 | Albert & Culik II | 14 |
| 1990 | Lindgren & Nordhal | 7 |
| 2004 | Cook | 2 |

A cellular automaton is **Turing-universality** if... What exactly is the **formal definition**? What is a **non universal** CA?

A consensual yet **formal definition** is **unknown** and seems difficult to achieve. (Durand & Roka, 1999)

# TM *à la* Smith III

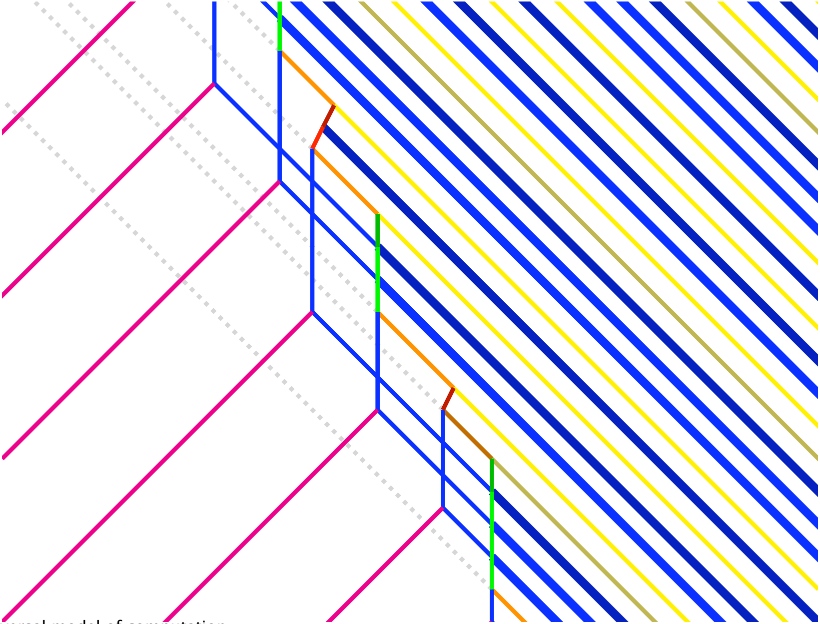
| | | | | |
|---|---|---|---|---|
| $B$ | $B$ | $(q_0, a)$ | $a$ | $B$ |
| $B$ | $B$ | $a$ | $(q_1, B)$ | $B$ |
| $B$ | $B$ | $(q_1, b)$ | $B$ | $B$ |
| $B$ | $(q_0, B)$ | $b$ | $B$ | $B$ |
| $B$ | $B$ | $(q_0, a)$ | $B$ | $B$ |

*Time always moves upwards!*

# TM *à la* Lindgren & Nordhal

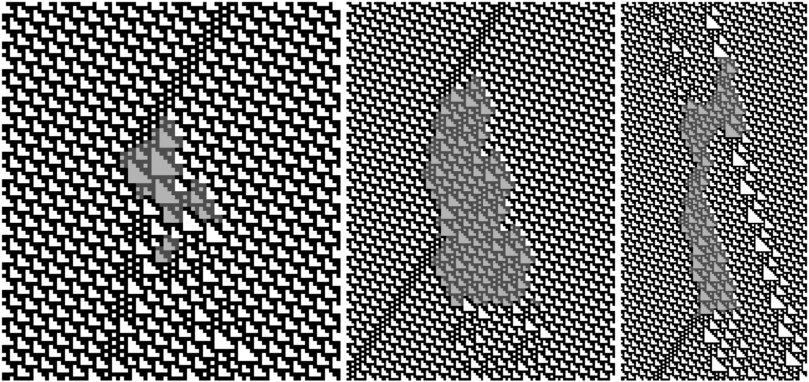
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $B$ | $\bullet$ | $B$ | $\bullet$ | $a$ | $q_0$ | $\bullet$ | $a$ | $\bullet$ | $B$ |
| $B$ | $\leftrightarrow$ | $B$ | $\leftrightarrow$ | $a$ | $\leftrightarrow$ | $q_0$ | $a$ | $\leftrightarrow$ | $B$ |
| $B$ | $\bullet$ | $B$ | $\bullet$ | $a$ | $\bullet$ | $q_1$ | $B$ | $\bullet$ | $B$ |
| $B$ | $\leftrightarrow$ | $B$ | $\leftrightarrow$ | $a$ | $q_1$ | $\leftrightarrow$ | $B$ | $\leftrightarrow$ | $B$ |
| $B$ | $\bullet$ | $B$ | $\bullet$ | $q_1$ | $b$ | $\bullet$ | $B$ | $\bullet$ | $B$ |
| $B$ | $\leftrightarrow$ | $B$ | $q_1$ | $\leftrightarrow$ | $b$ | $\leftrightarrow$ | $B$ | $\leftrightarrow$ | $B$ |
| $B$ | $\bullet$ | $B$ | $q_0$ | $\bullet$ | $b$ | $\bullet$ | $B$ | $\bullet$ | $B$ |
| $B$ | $\leftrightarrow$ | $B$ | $\leftrightarrow$ | $q_0$ | $b$ | $\leftrightarrow$ | $B$ | $\leftrightarrow$ | $B$ |
| $B$ | $\bullet$ | $B$ | $\bullet$ | $q_0$ | $a$ | $\bullet$ | $B$ | $\bullet$ | $B$ |

# Universality of Rule 110 *à la* Cook



# Cook: Details...

Uses **huge** particles and collisions...



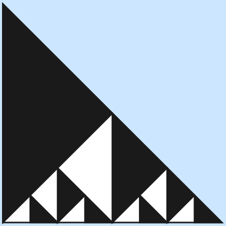
**Theorem** Rule 110 is **Turing-universal**.

# Part I
# Computing
# inside the cellular space

1. Cellular automata

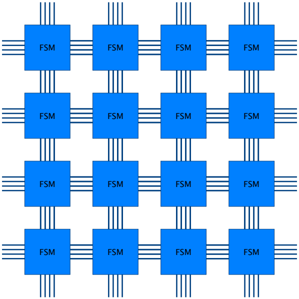
2. A universal model of computation

**3. A model of parallel computation**



# Another path to universality

**Remark** Boolean circuits are not sequential and can also simulate **parallel models of computation**.




This leads to a stronger notion of **intrinsic universality** on CA, the ability to simulate any CA.

# Bulking classifications

**Idea** define a **quasi-order** on cellular automata, **equivalence classes** capturing behaviors.

**Definition** A CA $\mathcal{A}$ is **algorithmically simpler** than a CA $\mathcal{B}$ if all the space-time diagrams of $\mathcal{A}$ are space-time diagrams of $\mathcal{B}$ *(up to uniform state renaming)*.

Formally, $\mathcal{A} \subseteq \mathcal{B}$ if there exists $\varphi : S_{\mathcal{A}} \to S_{\mathcal{B}}$ injective such that $\overline{\varphi} \circ G_{\mathcal{A}} = G_{\mathcal{B}} \circ \overline{\varphi}$.

That is, the following diagram commutes:

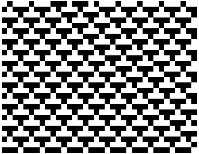
$$\begin{tikzcd} C \arrow[r, "\varphi"] \arrow[d, "G_{\mathcal{A}}"'] & \overline{\varphi}(C) \arrow[d, "G_{\mathcal{B}}"] \\ G_{\mathcal{A}}(C) \arrow[r, "\varphi"'] & \overline{\varphi}(G_{\mathcal{A}}(C)) \end{tikzcd}$$

# Bulking quasi-order

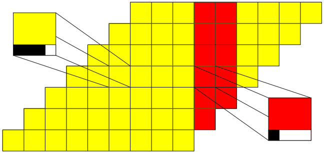
Quotient the set of CA by **discrete affine transformations**, the only geometrical transformations preserving CA.

The $\langle m, n, k \rangle$ transformation of $\mathcal{A}$ satisfies:

$$G_{\mathcal{A}^{\langle m,n,k \rangle}} = \sigma^k \circ o^m \circ G_{\mathcal{A}}^n \circ o^{-m} \ .$$



$\mathcal{A}$

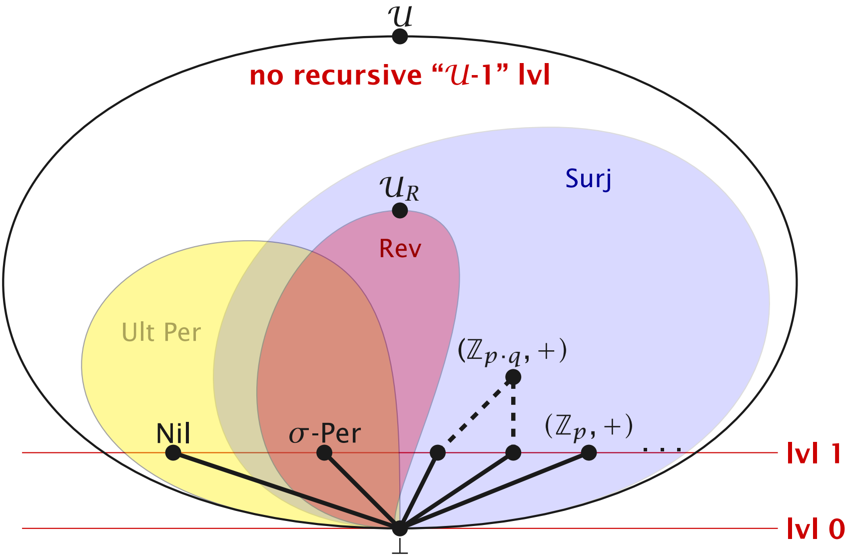


$\mathcal{A}^{\langle 4,4,1 \rangle}$

**Definition** The **bulking quasi-order** is defined by $\mathcal{A} \leqslant \mathcal{B}$ if there exists $\langle m, n, k \rangle$ and $\langle m', n', k' \rangle$ such that

$$\mathcal{A}^{\langle m,n,k \rangle} \subseteq \mathcal{B}^{\langle m',n',k' \rangle} \ .$$

# The big picture



# Intrinsic universality

**Definition** A CA $\mathcal{U}$ is **intrinsically universal** if it is maximal for $\leqslant$, *i.e.* for all CA $\mathcal{A}$, there exists $\alpha$ such that $\mathcal{A} \subseteq \mathcal{U}^{\alpha}$.

**Theorem** There exists **Turing universal** CA that are not intrinsically universal.

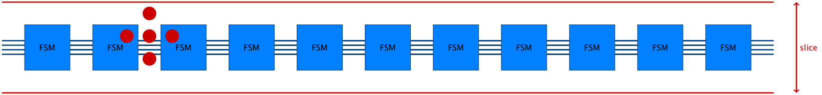
where Turing universality is obtained in a very classical way to ensure compatibility with your own definition.

**Theorem** **Boolean circuit universal** 2D CA are also **intrinsically universal**.

(Delorme et al., 2011)

# Using boolean circuits

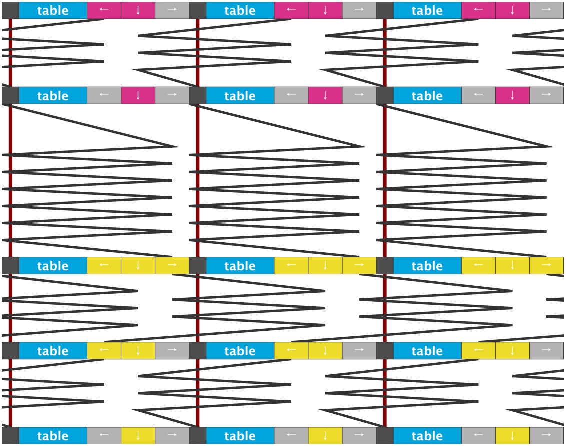
Every **2D intrinsically universal** CA can be converted to a **1D intrinsically universal** CA **[Banks 1970]**.




Cut **slices** of a periodic configuration, catenate them **horizontally**, use the **adequate neighborhood**.

The neighborhood can be transformed into radius 1 at the cost of **increase of the number of states**.

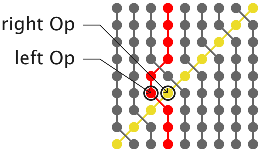
# Using highly parallel Turing machines

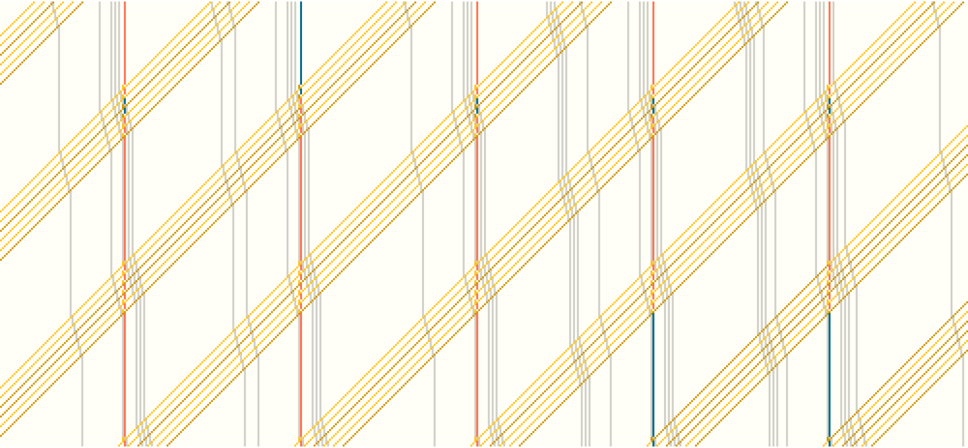



Use one **Turing-like head** per macro-cell, the **moving sequence** being **independent** of the computation.

# More intricate: 6 states

A **6 states** intrinsically universal CA of radius 1 embedding **boolean circuits** into the line.



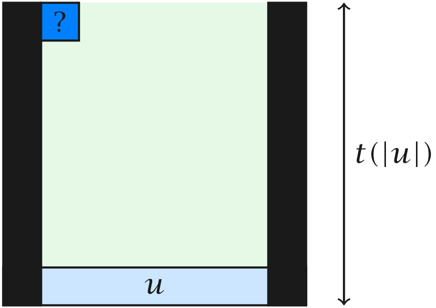
# Parallel language recognition

In the **70s** CA have been studied as **a model of massive parallelism**, in particular as **language recognizers**.

A CA $(S, N, f)$ **recognizes** a language $L \subseteq \Sigma^*$ in time $t(n)$ with border state $\#$ and accepting states $Y \subseteq S$ if for each $u \in \Sigma^*$, starting from the configuration ${}^{\omega}\#u\#^{\omega}$, at time $t(|u|)$ the state of cell $0$ is in $Y$ if and only if $u \in L$.

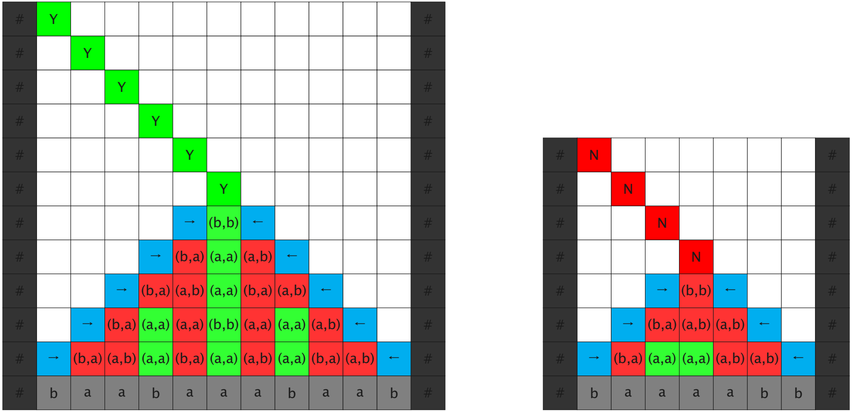
**Real time**: $t(n) = n$

**Linear time**: $t(n) = \alpha n$



# Example: palindromes

Recognizing **palindromes** in real time with 16 states.



# Real time vs Linear time

**Theorem** Every language recognized in time $t(n) = n + T(n)$ is recognized in time $t(n) = n + T(n)/k$ for all $k$. The cost of **acceleration** is paid using more states.

Same techniques as for Turing machine **acceleration**.

**Theorem[Ibarra]** Every language recognized in **linear time** is recognized in **real time** if and only if the class of real time langages is **closed by mirror image**.

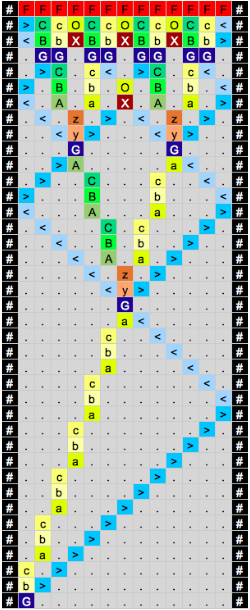
**Open Pb** Does **Real time** = **Linear time**?

# Firing Squad Synchronization Pb

CA can also solve **purely parallel tasks**.

**Definition** A CA solves the **FSSP** if for all $n > 0$ starting from $\#GB^{n-1}\#$ it eventually enters $\#F^n\#$ and the fire state $F$ never appears before.

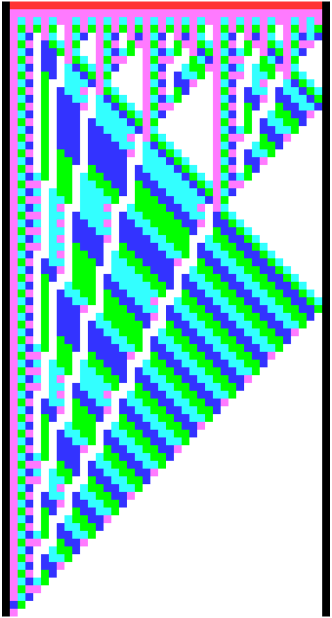
**Theorem[Minksy]** A **CA solves FFSP** in time $3n - 1$ with 15+1 states.



# Mazoyer's solution

**Remark** No CA can solve the FSSP in time less than $2n - 2$.

**Theorem[Mazoyer 1984]** A **CA solves FSSP** in **optimal time** with 6+1 states.



# Part II

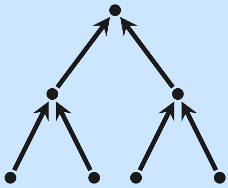
# Computing properties of CA

# Part II
# Computing properties of CA

**4. Discrete dynamical systems**

5. Immediate properties

6. Dynamical properties



# Cellular automata

**Definition** A **CA** is a tuple $(d, S, r, f)$ where $S$ is a **finite set of states**, $r \in \mathbb{N}$ is the **neighborhood radius** and $f : S^{(2r+1)^d} \to S$ is the **local rule** of the cellular automaton.

A **configuration** $c \in S^{\mathbb{Z}^d}$ is a coloring of $\mathbb{Z}^d$ by $S$.

The **global map** $G : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ applies $f$ uniformly and locally:

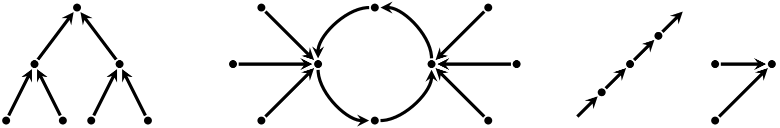
$$\forall c \in S^{\mathbb{Z}}, \forall z \in \mathbb{Z}^d, \qquad F(c)(z) = f(c(z-r), \ldots, c(z+r)).$$

A **space-time diagram** $\Delta \in S^{\mathbb{Z}^d \times \mathbb{N}}$ satisfies, for all $t \in \mathbb{N}$,

$$\Delta(t+1) = F(\Delta(t)).$$

# Discrete dynamical systems

**Definition** A **DDS** is a pair $(X, F)$ where $X$ is a topological space and $F : X \to X$ is a continuous map.



**Definition** The **orbit** of $x \in X$ is the sequence $(F^n(x))$ obtained by iterating $F$.

In this tutorial, $X = S^{\mathbb{Z}}$ is endowed with the **Cantor topology** (product of the discrete topology on $S$), and $F$ is a continuous map **invariant by translation**.

# Topology

**Definition** A **topological space** is a pair $(E, \mathcal{O})$ where $\mathcal{O} \subseteq \mathcal{P}(E)$ is the set of **open** subsets satisfying:

- $\mathcal{O}$ contains both $\varnothing$ and $E$;
- $\mathcal{O}$ is closed under union;
- $\mathcal{O}$ is closed under finite intersection.

$S$ is endowed with the **discrete topology**: $\mathcal{O} = \mathcal{P}(S)$.

$S^{\mathbb{Z}^d}$ is endowed with the **Cantor topology**: the product topology of the discrete topology.

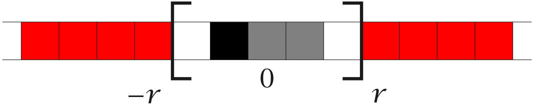
$$\mathcal{O} = \left\{ \prod X_i \,\middle|\, X_i \subseteq S \wedge \mathrm{Card}(\{i | X_i \neq S\}) < \omega \right\}$$

Cantor topology is **metric** and **compact**.

# Cylinders

**Definition** The cylinder $[m] \subseteq S^{\mathbb{Z}^d}$ with radius $r \geqslant -1$ generated by the pattern $m \in S^{[-r,r]^d}$ is

$$[m] = \left\{ c \in S^{\mathbb{Z}^d} \,\middle|\, \forall p \in \mathbb{Z}^d, ||p||_\infty \leqslant r \Rightarrow c(p) = m(p) \right\}$$



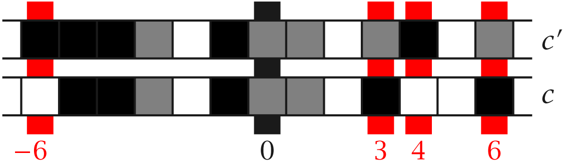

**Proposition** Cylinders are a countable **clopen generating set**.

**Notation** $[m] \prec [m']$ means $[m]$ is a **sub-cylinder** of $[m']$, *i.e.* $[m'] \subset [m]$.

# Metric

**Proposition** Cantor topology is **metric**

$$\forall c, c' \in S^{\mathbb{Z}^d}, \quad \delta(c, c') = 2^{-\min\left\{\|p\|_\infty \,\middle|\, c_p \neq c'_p\right\}}$$




$$\delta(c, c') = 1/8$$

**Remark** Open balls of $\delta$ exactly correspond to cylinders.

# Compact

**Proposition** Every sequence of configurations $(c_i) \in S^{\mathbb{Z}^d}{}^{\mathbb{N}}$ admits a converging subsequence.

**Proof** by **extraction**:

By recurrence, let $\left(c_i^0\right) = (c_i)$.

It is alway possible to find:

- a cylinder $[m_n]$ of radius $n$ and
- an infinite subsequence $\left(c_i^{n+1}\right)$ of $\left(c_{i+1}^n\right)$

such that for all $i \in \mathbb{N}$, $c_i^{n+1} \in [m_n]$.

By construction $[m_{n+1}] \subset [m_n]$ and $\left(c_0^{i+1}\right)$ is a converging subsequence of $(c_i)$ (to $\bigcap [m_i]$): $\delta(c_0^{n+1}, c_0^{n+2}) \leqslant 2^{-n}$.

# König trees

**Remark** Cantor topology is essentially **combinatorial**.

**Remark** Main properties can be obtained using **extraction**.

**König's Lemma** Every infinite tree with finite branching admis an infinite branch.

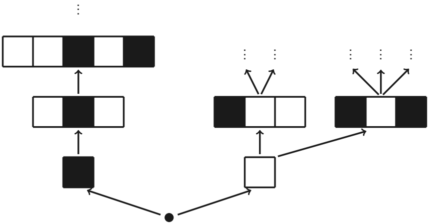
**Definition** The **König tree** $\mathcal{A}_C$ of a set of configurations $C \subseteq S^{\mathbb{Z}^d}$ is the tree $(V_C, E_C)$ where

$$\begin{aligned} V_C &= \{[m] | C \cap [m] \neq \emptyset\} \\ E_C &= \left\{([m],[m']) \middle| [m] \prec [m'] \wedge \mathrm{r}([m']) = \mathrm{r}([m]) + 1\right\} \end{aligned}$$

The root of the tree is the cylinder $[] = S^{\mathbb{Z}^d}$ of radius $-1$.

# Toppings

The König tree of a **non empty** set of configurations is an **infinite tree** with finite branching.



To each infinite branch $([m_i])$ is associated a **unique configuration** $\bigcap [m_i]$.

**Definition** The **topping** $\overline{\mathcal{A}_C}$ of a König tree is the set of configurations associated to its infinite branches.

# König topology

The **König topology** is defined by its closed sets: toppings of König trees.

The **complementary of a closed set** is the **union of cylinders** that are not nodes of the tree.

**Theorem** **Cantor** and **König** topologies are the **same**.

Most **topological concepts** can be explained using **trees**:

- dense sets;
- closed sets with non empty interior;
- compacity;
- Baire's theorem.

# Continuity

**Proposition** clopen sets are finite unions of cylinders.

**Definition** A mapping $G : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ is local in $p \in \mathbb{Z}^d$ if there exists a radius $r$ such that:

$$\forall c, c' \in S^{\mathbb{Z}^d}, \quad \left[c_{|r}\right] = \left[c'_{|r}\right] \Rightarrow G(c)_p = G(c')_p \quad .$$

**Proposition** A mapping $G : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ is continuous if and only if it is local in every point.

# Curtis-Hedlund-Lyndon Theorem

**Definition** The **translation** $\sigma_k : S^{\mathbb{Z}^d} \to S^{\mathbb{Z}^d}$ with vector $k \in \mathbb{Z}^d$ satisfies:

$$\forall c \in S^{\mathbb{Z}^d}, \forall p \in \mathbb{Z}^d, \quad \sigma_k(c)_p = c_{p-k} \quad .$$

**Theorem[Hedlund 1969]** Continuous mapping commuting with translations are exactly global maps of CA.

Thus **CA** can be given by their **global map**.

**Remark** CA have a **dual nature**: discrete dynamical systems with a description as finite automata.

# Symbolic dynamics

A central object in **symbolic dynamics** is **subshift**.

**Definition** A **subshift** of $S^{\mathbb{Z}^d}$ is a set of configurations both **closed** and **invariant by translation**.

**Ex** $\ldots abaababaaa \ldots$

$$X = \left\{ c \in \{a,b\}^{\mathbb{Z}} \,\middle|\, \forall p \in \mathbb{Z}, c_p = b \Rightarrow c_{p+1} = a \right\}$$

**Remark** **Subshifts** are also very natural when studying **CA**.

# Langage of a subshift

**Definition** The **language** $L(X)$ of a **subshift** $X$ is the set of finite patterns appearing in $X$.

**Proposition** A **subshift** is characterized by its **language**.

$$\overline{L} = \left\{ c \in S^{\mathbb{Z}^d} \,\middle|\, \forall r \geqslant 0, \forall m \in S^{[-r,r]^d}, m \prec c \Rightarrow m \in L \right\}$$

**Warning** It might be that $L(\overline{L}) \neq L$.

# Forbidden words

**Proposition** A subshift is characterized by the set of its **forbidden words**: the complementary of its language.

**Proposition** Subshifts are in bijection with **minimal sets of forbidden words** (for set inclusion).

**Ex** $X = S_{\{bb\}}$

# SFT, tilings, soficity

**Definition** A **subshift of finite type** (**SFT**) is defined by a finite set of forbidden words.

**Proposition** The **set of SFT** is invariant by **CA preimage**.

**Remark** **SFT** correspond to **tilings**: colorings with local uniform constraints.

**Definition** A **sofic subshift** is the image of a SFT by a CA.

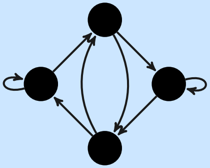
**Proposition** **1D sofic subshifts** are subshifts that admit a **regular language** of forbidden words.

# Part II
# Computing properties of CA

4. Discrete dynamical systems

**5. Immediate properties**

6. Dynamical properties



# Configurations

**Definition** A configuration $c \in S^{\mathbb{Z}^d}$ is **periodic**, with period $(p_i) \in {\mathbb{N}^*}^d$, if

$$\forall p \in \mathbb{Z}^d, \forall (k_i) \in \mathbb{Z}^d, \quad c(p) = c(p + (k_1 p_1, \ldots, k_d p_d)) \quad .$$

**Notation.** $G_p$ denotes $G$ restricted to periodic configurations.

**Definition** A configuration $c \in S^{\mathbb{Z}^d}$ is **$s$-finite** if $s$ is quiescent ($f(s, \ldots, s) = s$) and

$$\mathrm{Card}\left(\left\{p \in \mathbb{Z}^d \,\middle|\, c(p) \neq s\right\}\right) < \omega \quad .$$

**Notation.** $G_f$ denotes $G$ restricted to $s$-finite configurations.

# Immediate properties

**Definition** A CA $G : C \to C$ is:

- **injective** if $\forall x, y \in C, F(x) \neq F(y)$;
- **surjective** if $\forall x \in C, F^{-1}(x) \neq \emptyset$;
- **bijective** if both injective and surjective.

**Definition** A bijective CA $G$ is **reversible** if there exists a CA $H$ such that $H = G^{-1}$.

**Corollary** Every **bijective** CA is **reversible**.

# Garden of Eden and orphans

**Definition** A configuration of a CA is a **garden of Eden** if it has no preimage.

**Proposition** A CA is **surjective** if and only if it has **no garden of Eden**.

**Definition** Given a CA, a pattern $m \in S^{[-r,r]^d}$ is an **orphan** if $m$ has no preimage.

**Proposition** A CA is **surjective** if and only if it has **no orphan**.

# Moore-Myhill

**Theorem[Moore 1962]** $G$ surjective $\Rightarrow$ $G_f$ injective.

**Theorem[Myhill 1963]** $G_f$ injective $\Rightarrow$ $G$ surjective.

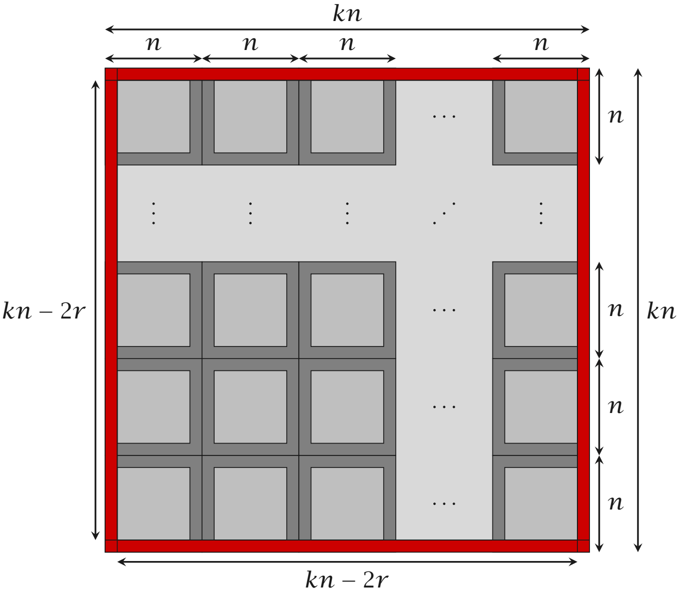
**Corollary** $G$ injective $\Rightarrow$ $G$ bijective.

Let's prove it!

**Lemma** For all $d, n, N, r \in \mathbb{N}$, there exists $k$ big enough so that

$$\left(N^{n^d} - 1\right)^{k^d} < N^{(kn-2r)^d}$$

# Key picture



# $G$ surjective $\Rightarrow G_f$ injective

Let $G$ be a CA with $N$ states and radius $r$ that is not injective on $s$-finite configurations. There exists two patterns $p_1$ and $p_2$ of size $n^d$ bordered by $s$ on a width $r$ with a same image. Replacing $p_1$ by $p_2$ in any configuration does not change its image.

Consider square patterns of side $kn$. Their images are patterns of side $kn - 2r$. There exists $N^{(kn-2r)^d}$ possible images for at most $(N^{n^d} - 1)^{k^d}$ preimages. By previous lemma, there is an orphan thus $G$ is not surjective.

# $G_f$ injective ⇒ $G$ surjective

Let $G$ be a non surjective CA with radius $r$. It admits an orphan of size $n^d$.

Consider the set of $s$-finite configurations the non-quiescent pattern of which is of side $kn - 2r$. There are $N^{(kn-2r)^d}$ such patterns. Their $s$-finite images have patterns of side $kn$. At most $(N^{n^d} - 1)^{k^d}$ of them are not orphans. Thus two of the configurations have a same image, $G$ is not injective on finite configurations.

# The big picture

$$
\begin{array}{ccccc}
G \text{ bijective} & \Longleftrightarrow & G \text{ réversible} & \Longleftrightarrow & G \text{ injective} \\
\Downarrow \;\; \not\Uparrow & & & & \Downarrow \;\; \Uparrow^{\triangle} \\
G_f \text{ bijective} & \begin{array}{c} \not\Longrightarrow \\ {}^{\square}\Longleftarrow \end{array} & G_p \text{ injective} & \Longleftrightarrow & G_p \text{ bijective} \\
\Updownarrow & & & & \Downarrow \;\; \not\Uparrow \\
G_f \text{ surjective} & & \begin{array}{c} \Longrightarrow^{\square} \\ \not\Longleftarrow \end{array} & & G_p \text{ surjective} \\
\Downarrow \;\; \not\Uparrow & & & & \Downarrow \;\; \Uparrow^{\square} \\
G_f \text{ injective} & \Longleftrightarrow & \Lambda_G = S^{\mathbb{Z}^d} & \Longleftrightarrow & G \text{ surjective}
\end{array}
$$

$\triangle$ means true for $d = 1$, false for $d \geqslant 2$

$\square$ means true for $d = 1$, **open** for $d \geqslant 2$

# Examples

**Proposition** $G_p$ injective $\Rightarrow$ $G_p$ bijective.

**Key** CA preserve periods.

**Proposition** $G_f$ surjective $\Rightarrow$ $G_f$ injective.

**Key** Finite configurations are dense + Moore-Myhill.

# Counter-examples

**Proposition** $\exists G$ surjective $\not\Rightarrow G_f$ surjective.

**XOR** rule: $S = \mathbb{Z}_2$, $f(x, y) = x + y \pmod 2$.

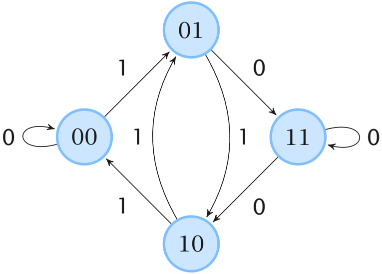
**Proposition** $\exists G$, $G_f$ surjective $\not\Rightarrow G$ injective.

**controlled-XOR** rule: $S = \{0, 1\} \times \mathbb{Z}_2$,
$f((1, x), (\_, y)) = (1, x + y \pmod 2))$ and
$f((0, x), \_) = (0, x)$.

# De Bruijn Graph

**Definition** The **De Bruijn graph** of a 1D CA $(S, r, f)$ is the labelled graph $(V, E)$ where:

- $V = S^{2r}$;
- $(u, s, v) \in E$ if $f(s_0, \ldots, s_{2r}) = s$ where $u = (s_0, \ldots, s_{2r-1})$ and $v = (s_1, \ldots, s_{2r})$.



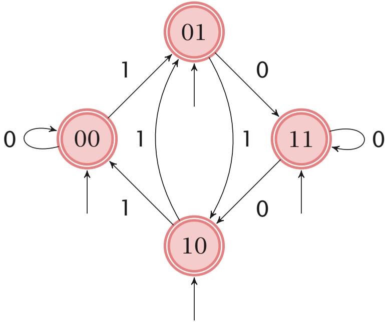

A convenient tool to decide properties of 1D CA.

# Deciding surjectivity in 1D

**Remark** A CA $G$ is **surjective** if and only if $G\left(S^{\mathbb{Z}^d}\right) = S^{\mathbb{Z}^d}$.

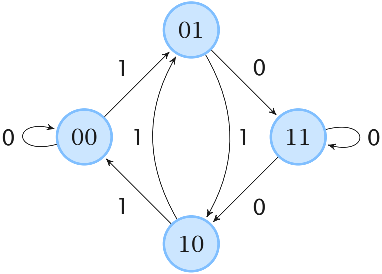
**Lemma** A 1D CA $G$ is **surjective** iff $L\left(G\left(S^{\mathbb{Z}}\right)\right) = S^*$.

**Theorem[Amoroso, Patt 1972]** Surjectivity 1D is decidable.



# Deciding injectivity in 1D

A 1D CA is **not injective** iff its De Bruijn graph contains **two distinct paths with the same biinfinite word**.




If such a pair of paths exist, there exists one with **ultimately periodic** paths.

**Theorem[Amoroso, Patt 1972]** Injectivity is decidable in 1D.

# Part II
# Computing properties of CA

4. Discrete dynamical systems

5. Immediate properties

**6. Dynamical properties**

# Symbolic dynamical systems

**Definition** A **SDS** is a DDS $(X, F)$ where $X$ is a **subshift**.

$x \in X$ is **periodic** with period $n$ if $F^n(x) = x$.

$x \in X$ is **ultimately periodic** with transitory $m$ if $F^m(x)$ is periodic.

**Definition** $Y \subseteq X$ is **invariant** if $F(Y) \subseteq Y$.

**Definition** $Y \subseteq X$ is **strongly invariant** if $F(Y) = Y$.

# Attractors

**Definition** The **limit set** of an invariant clopen $V \subseteq X$ is

$$\Lambda_F(V) = \bigcap_{n \in \mathbb{N}} F^n(V)$$

**Definition** An **attractor** is the limit set of a non-empty invariant clopen.

**Definition** The **bassin** of an attractor $Y$ is

$$\mathcal{B}_F(Y) = \left\{ x \in X \,\middle|\, \lim_{n \to \infty} \delta(F^n(x), Y) = 0 \right\}$$

**Definition** A **minimal attractor** is an attractor with no strict subset which is also an attractor.

# Limit set

**Definition** The **limit set** of a CA $\left(S^{\mathbb{Z}^d}, F\right)$ is the set of configurations than can appear at all time:

$$\Lambda_F = \bigcap_{n \in \mathbb{N}} F^n\left(S^{\mathbb{Z}^d}\right) \quad .$$

**Proposition** The **limit set** is a **non empty subshift**.

$F^n\left(S^{\mathbb{Z}^d}\right)$ is a non empty subshift and $F^{n+1}\left(S^{\mathbb{Z}^d}\right) \subseteq F^n\left(S^{\mathbb{Z}^d}\right)$.

**Proposition** The **limit set** is the **maximal attractor**.

# Biinfinite space-time diagrams

**Definition** A **biinfinite space-time diagram** $\Delta \in S^{\mathbb{Z}^{d+1}}$ satisfies:

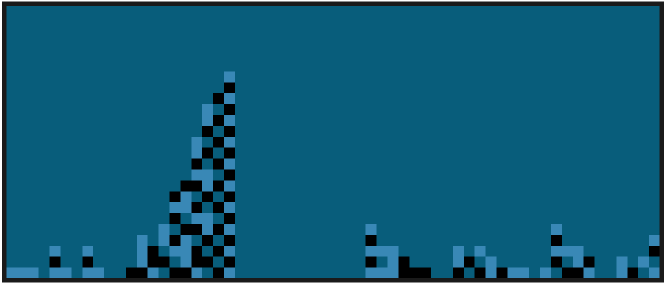
$$\forall t \in \mathbb{Z}, \quad \Delta(t+1) = F(\Delta(t)) \quad .$$

**Proposition** The **limit set** is the set of configurations of **biinfinite space-time diagrams**.

Every element $x$ of the limit set admits an infinite chain of predecessors $x = x_0, x_0 = F(x_1), \ldots, x_n = F(x_{n+1}), \ldots$

# Nilpotency

**Definition** A CA with quiescent state $q$ is **nilpotent** if every configuration converges in finite time to the $q$-monochromatic configuration.

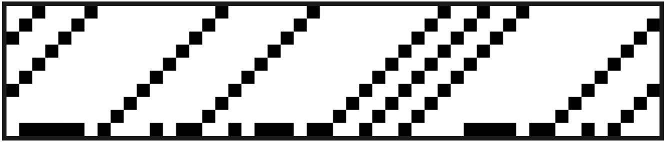


**Proposition** A CA is **nilpotent** if and only if its limit set is a **singleton**.

# SFT example

Let $F$ be the 1D CA with radius 1 and local rule

$$f(x,y,z) = \begin{cases} 1 & \text{si } (x,y,z) = (1,0,0) \\ 0 & \text{sinon} \end{cases} .$$

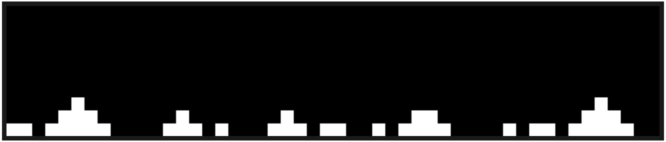


$$\Lambda_F = F\left(S^{\mathbb{Z}}\right) = S_{\{11,101\}}$$

# Max example

Let $F$ be the 1D CA with radius 1 and local rule

$$f(x, y, z) = \max(x, y, z) \quad .$$



$$\Lambda_F = {}^{\omega}0^{\omega} + {}^{\omega}1^{\omega} + {}^{\omega}01^{\omega} + {}^{\omega}10^{\omega} + {}^{\omega}10^{*}1^{\omega}$$

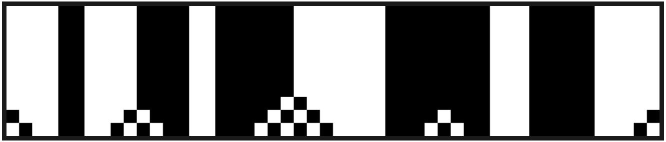
$$\Lambda_F = S_{\{01^n0 | n \in \mathbb{N}\}}$$

$\Lambda_F$ is **countable** and **not SFT**.

# Majority example

Let $F$ be the 1D CA with radius 1 and local rule

$$f(x, y, z) = \mathrm{maj}(x, y, z) \quad .$$



**Exercise** What is $\Lambda_F$?

**Hint** Consider $01 \cdot 00^+ \, (11^+00^+)^* \, 00^+ \cdot 10$.

# Language

**Proposition** For every CA, $L(\Lambda_F) = \bigcap_{n\in\mathbb{N}} L\left(F^n\left(S^{\mathbb{Z}^d}\right)\right)$.

**Corollary** If $\Lambda_F$ is a **SFT** then $\exists n\, \Lambda_F = F^n\left(S^{\mathbb{Z}^d}\right)$.

Consider the first **time of appearance** of each **minimal forbidden word**.

**Proposition** If $\Lambda_F = F^n\left(S^{\mathbb{Z}^d}\right)$ then $\Lambda_F$ is **sofic**.

If $F$ is a CA, so is $F^n$.

# Cardinality

**Proposition[CPY89]** If $\Lambda_F$ contains **two distinct elements** then it contains a **non spatially periodic** element.

**Corollary** A **limit set** is either a **singleton** either an **infinite** set.

# Recursivity

**Proposition** $L(\Lambda_F)$ is **co-recursively enumerable**.

Orphans of $F^n$ can be tested thus enumerated.

**Proposition[Hurd90]** For every co-recursively enumerable language $L \subseteq \Sigma^*$ there exists a CA $F$, a rational language $R \subseteq S^*$ and a morphism $\varphi : S^* \to \Sigma^*$ such that

$$\varphi(L(\Lambda_F) \cap R) = L \quad .$$

**Corollary** There exists CA with **non recursive limit sets**.

# Non rational context free

**Proposition[Hurd87]** There exists a 1D CA whose limit set has a **non rational context free** language.

**Exercise** Build one!

**Hint** Consider bouncing particles and walls.

# Context sensitive non context free

**Proposition[Hurd87]** There exists a 1D CA whose limit set has a **non context free context sensitive** language.

**Exercise** Build one!

**Hint** Complexify previous example.

# Non recursive 2D

**Proposition** There exists a 2D CA whose limit set has a **non recursive** language.

**Exercise** Build one!

**Hint** Consider space-time diagrams of Turing machines.

# Non recursive 1D

**Proposition[CPY89]** There exists a 1D CA whose limit set has a **non recursive** language.

**Exercise** Build one!

**Hint** Consider a CA that can simulate every CA.

# Surjectivity and injectivity

**Notation** $F_\Lambda$ is the restriction of $F$ to $\Lambda_F$.

**Proposition** $F_\Lambda$ is **surjective**.

**Proposition** If $\forall n\, \Lambda_F \neq F^n\left(S^{\mathbb{Z}^d}\right)$ then

$$\forall n \exists c \quad \forall i < n\, F^i(c) \notin \Lambda_F \;\wedge\; F^n(c) \in \Lambda_F \quad .$$

**Proposition[Taati 2008]** If $F_\Lambda$ is **injective** then $\exists n\, \Lambda_F = F^n\left(S^{\mathbb{Z}^d}\right)$.

# Part III

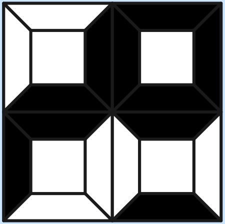
# Computation and reduction: undecidability results

# Part III
# Computation and reduction: undecidability results

**7. Tilings**

8. Undecidability results in 2D

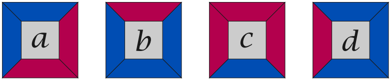
9. Undecidability results in 1D

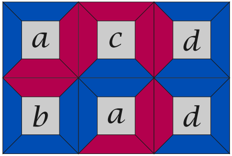


# The Domino Problem (DP)

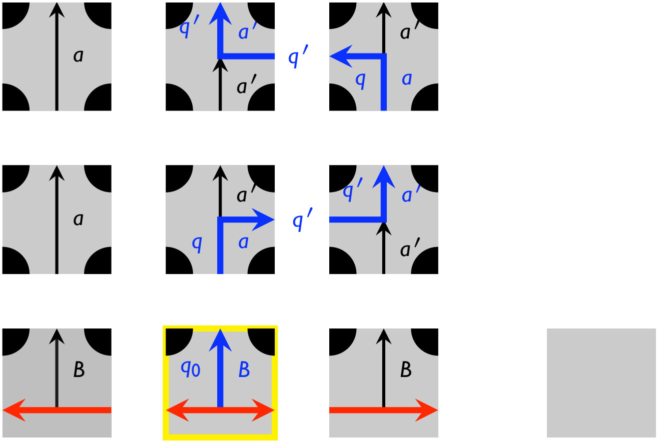
*"Assume we are* ***given a finite set of square plates*** *of the same size with* ***edges colored****, each in a different manner. Suppose further there are infinitely many copies of each plate (plate type). We are* ***not permitted to rotate or reflect a plate****. The question is to find an effective procedure by which we can* ***decide****, for each given finite set of plates,* ***whether we can cover up the whole plane*** *(or, equivalently, an infinite quadrant thereof)* ***with copies of the plates subject to the restriction that adjoining edges must have the same color****."*

*(Wang, 1961)*

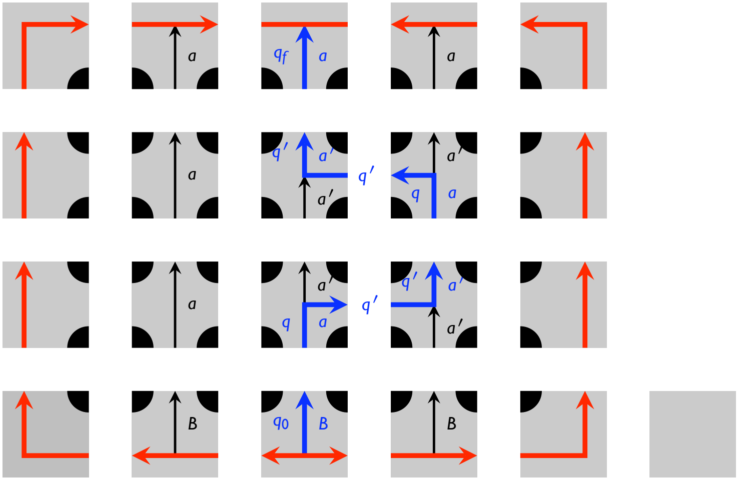


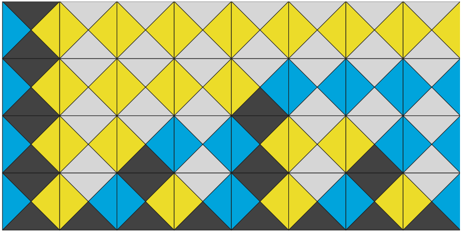
# Tiling with a fixed tile

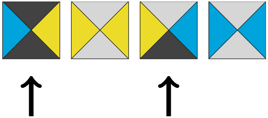


# Finite tiling



# Tiling with diagonal constraint





# Undecidability of DP

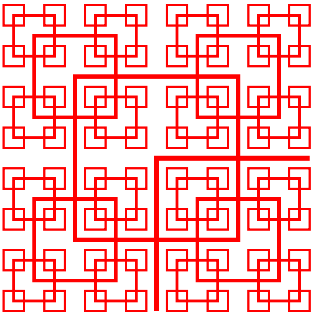
**Theorem[Berger64]** **DP** is **recursively undecidable**.

**Remark** To prove it one needs **aperiodic** tile sets.

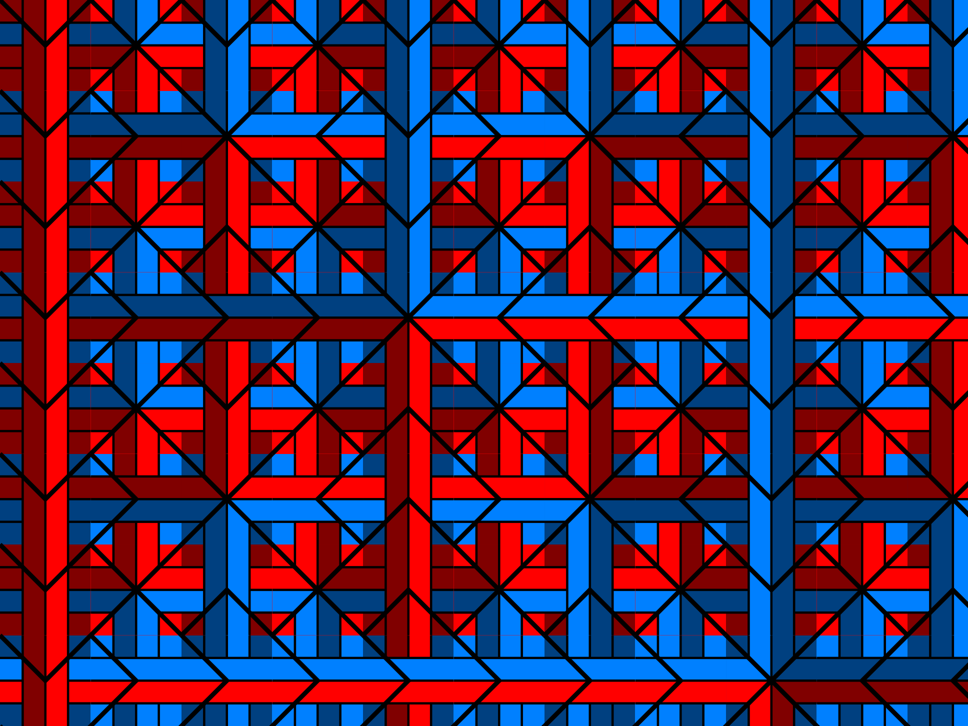
**Idea of the proof**

Enforce an (aperiodic) **self-similar structure** using local rules.

Insert a **Turing machine** computation **everywhere** using the structure.



**Remark** Plenty of different proofs!

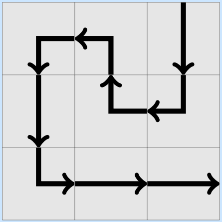


# Part III
# Computation and reduction: undecidability results

7. Tilings

**8. Undecidability results in 2D**

9. Undecidability results in 1D



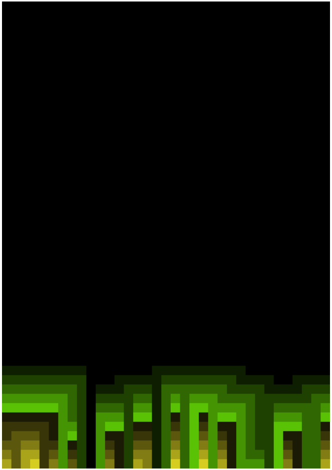
# The nilpotency problem (Nil)

**Definition** A DDS is **nilpotent** if
$\exists z \in X, \forall x \in X, \exists n \in \mathbb{N}, F^n(x) = z$.

Given a recursive encoding of the DDS, can we **decide** nilpotency?

A DDS is **uniformly nilpotent** if
$\exists z \in X, \exists n \in \mathbb{N}, \forall x \in X, F^n(x) = z$.

Given a recursive encoding of the DDS, can we **bound recursively** $n$?



# Nilpotency and limit set

**Definition** The limit set of a CA $F$ is the non-empty subshift

$$\Lambda_F = \bigcap_{n \in \mathbb{N}} F^n\left(S^{\mathbb{Z}}\right)$$

**Remark** $\Lambda_F$ is the set of configurations appearing in **biinfinite space-time diagrams** $\Delta \in S^{\mathbb{Z}\times\mathbb{Z}}$ such that $\forall t \in \mathbb{Z}, \Delta(t+1) = F(\Delta(t))$.

**Lemma** A CA is nilpotent iff its limit set is a **singleton**.

# 2D Nilpotency

**2D Nilpotency**
**Input:** a CA $(S, N, f)$.
**Question:** Is $F$ nilpotent?

**Theorem[CPY89]** Nilpotency is **undecidable** in 2D.

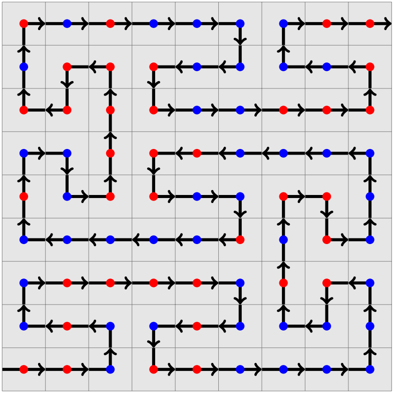
Prove than $\overline{\mathbf{DP}} \leqslant_m \mathbf{Nil2D}$.

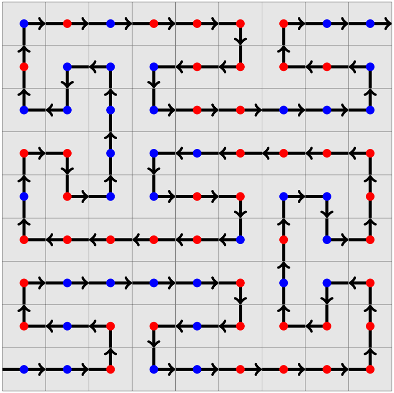
Given a set of Wang tiles $\tau$, build a CA with alphabet $\tau \cup \{\perp\}$ where $\perp$ is a spreading tiling error state.

# Surjectivity/Injectivity 2D

**Theorem[Kari 1990]** Both injectivity and surjectivity are **undecidable** in 2D.

For surjectivity, using Moore-Myhill, prove that injectivity on finite is undecidable in 2D.



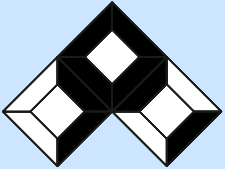


# Part III
# Computation and reduction: undecidability results

7. Tilings

8. Undecidability results in 2D

**9. Undecidability results in 1D**

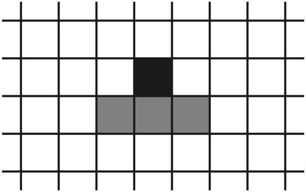


# Nilpotency 1D

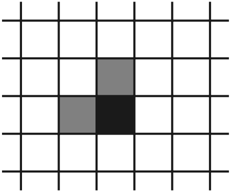
A state $\perp \in S$ is **spreading** if $f(N) = \perp$ when $\perp \in N$.

A CA with a spreading state $\perp$ is not nilpotent iff it admits a biinfinite space-time diagram without $\perp$.

**A tiling problem** Find a coloring $\Delta \in (S \setminus \{\perp\})^{\mathbb{Z}^2}$ satisfying the tiling constraints given by $f$.



$\equiv$



**Theorem[Kari92] NW-DP $\leqslant_m$ Nil**

# Revisiting DP

**Theorem[Kari92]** **NW-DP** is **recursively undecidable**.

**Remark** Reprove of undecidability of **DP** with the additionnal determinism constraint!

**Corollary** **Nil** is **recursively undecidable**.

# More on limit sets

**Theorem[Kari 1994]** The set of CA whose limit sets satisfy a non trivial property is never recursive.

**Theorem[Guillon Richard 2010]** Still true with a fixed alphabet!

**Definition** A CA $F$ is **weakly nilpotent** if

$$\forall c \forall p \exists t_0 \forall t > t_0 \quad F^t(c)_p = q \quad .$$

**Theorem[Guillon Richard 2008]** A CA is **weakly nilpotent** if and only if it is **nilpotent**.

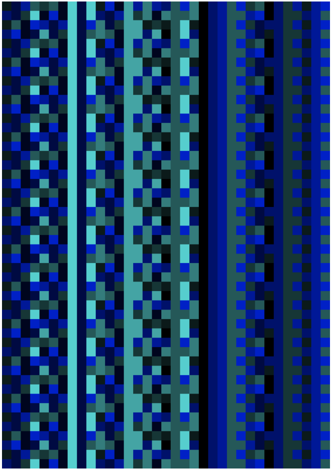
# The periodicity problem (Per)

**Definition** A DDS is **periodic** if
$\forall x \in X, \exists n \in \mathbb{N}, F^n(x) = x$.

Given a recursive encoding of the DDS, can we **decide** periodicity?

A DDS is **uniformly periodic** if
$\exists n \in \mathbb{N}, \forall x \in X, F^n(x) = x$.

Given a recursive encoding of the DDS, can we **bound recursively** $n$?



# Undecidability results

**Theorem** Both **Nil** and **Per** are **recursively undecidable**.

The proofs inject **computation** into **dynamics**.

Undecidability is not necessarily a negative result:
it is a **hint of complexity**.

**Remark** Due to **universe configurations** both nilpotency and periodicity are uniform.

The bounds grow **faster than any recursive function**: there exists simple nilpotent or periodic CA with huge bounds.

# The Immortality Problem (IP)

*"$(T_2)$ To find an effective method, which for every Turing-machine $M$ decides whether or not, for all tapes $I$ (finite and infinite) and all states $B$, $M$ will eventually halt if started in state $B$ on tape $I$"* *(Büchi, 1962)*

**Theorem[Hooper66]** **IP** is **recursively undecidable**.

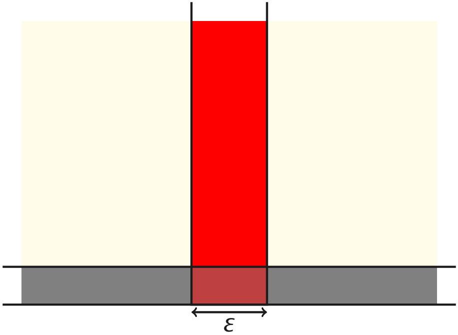
**Theorem[KO2008]** **R-IP** $\leqslant_m$ **TM-Per** $\leqslant_m$ **Per**

**Theorem[KO2008]** **R-IP** is **recursively undecidable**.

# Open Problem

**Definition** A CA $F$ is **positively expansive** if

$$\exists \varepsilon > 0,\ \forall x \neq y,\ \exists n \geqslant 0,\ d\left(F^n(x), F^n(y)\right) \geqslant \varepsilon$$




**Question** Is positive expansivity **decidable**?

## Table of Contents