Universalities in Cellular Automata

N. Ollinger (LIF, Aix-Marseille Université, CNRS, France)

JAC 2008
Uzès, April 24th

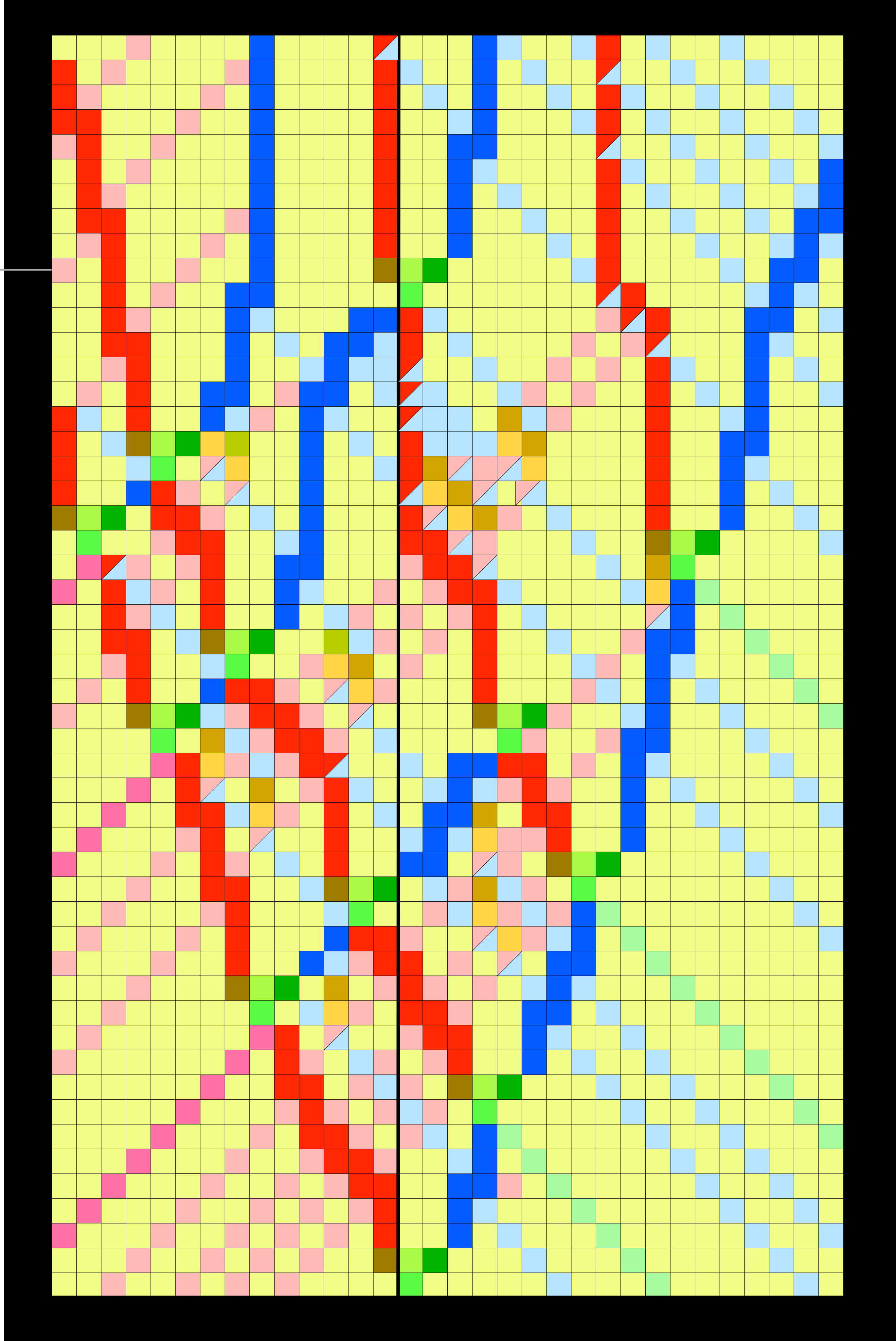
Universalities in Cellular Automata

N. Ollinger (LIF, Aix-Marseille Université, CNRS, France)

JAC 2008
Uzès, April 24th

Universal CA...

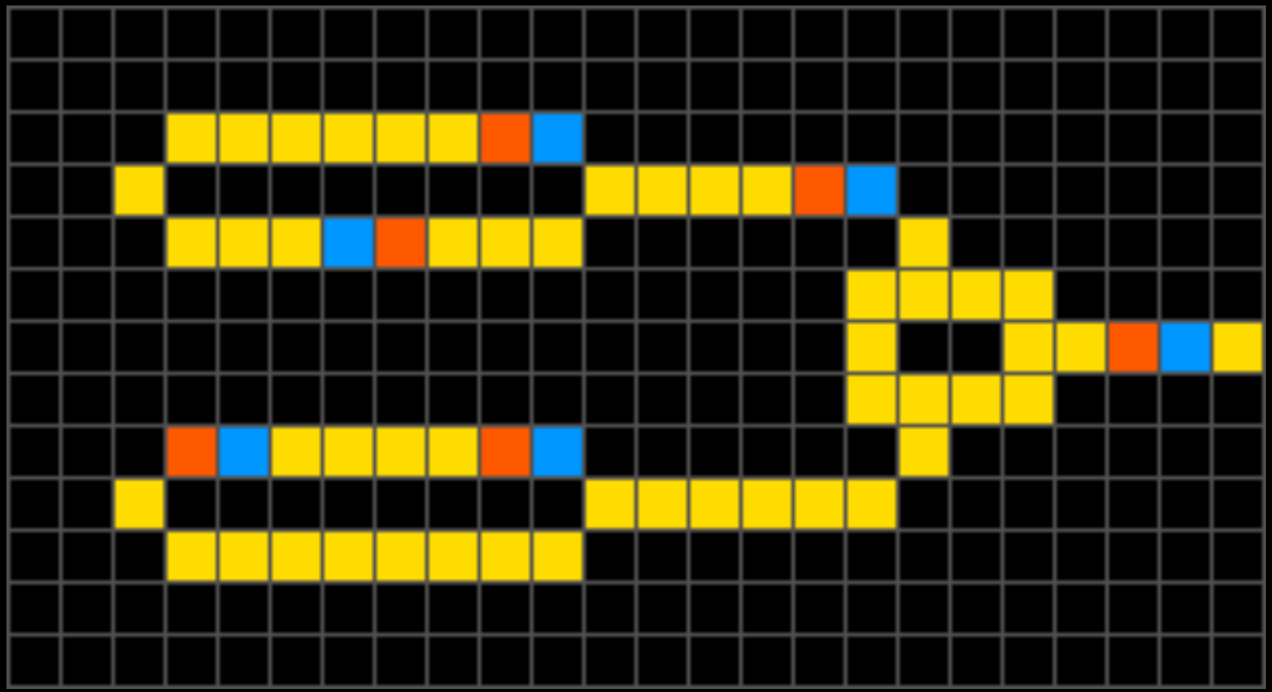
1. **à la Mazoyer?** (Tuesday morning)



Universal CA...

1. **à la Mazoyer?** (Tuesday morning)

2. **Boolean Circuit Simulator?**



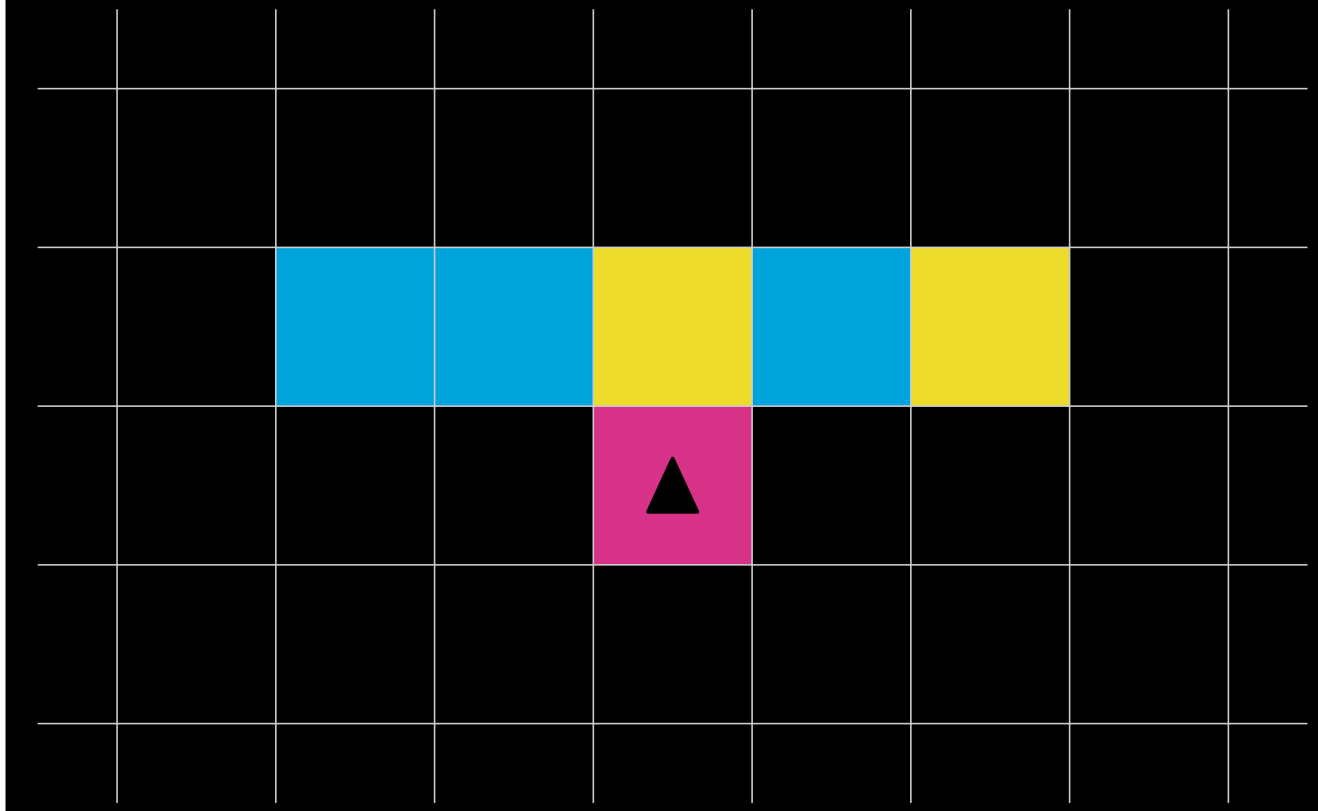
Wikipedia Commons animation by T. Schoch

Universal CA...

1. **à la Mazoyer?** (Tuesday morning)

2. **Boolean Circuit Simulator?**

3. **Computation Universality?**



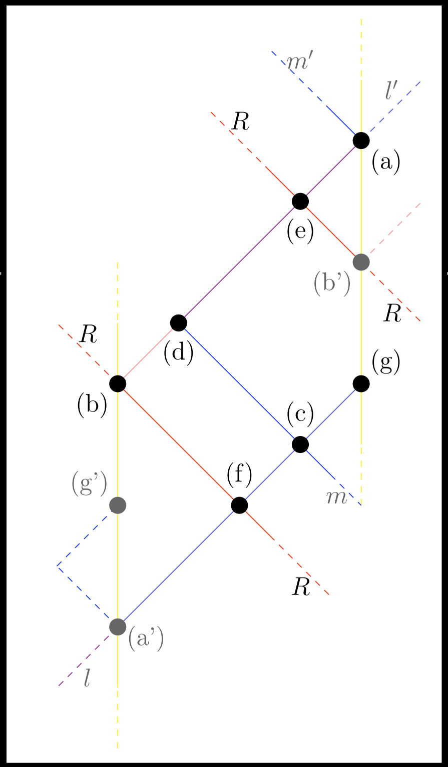
Universal CA...

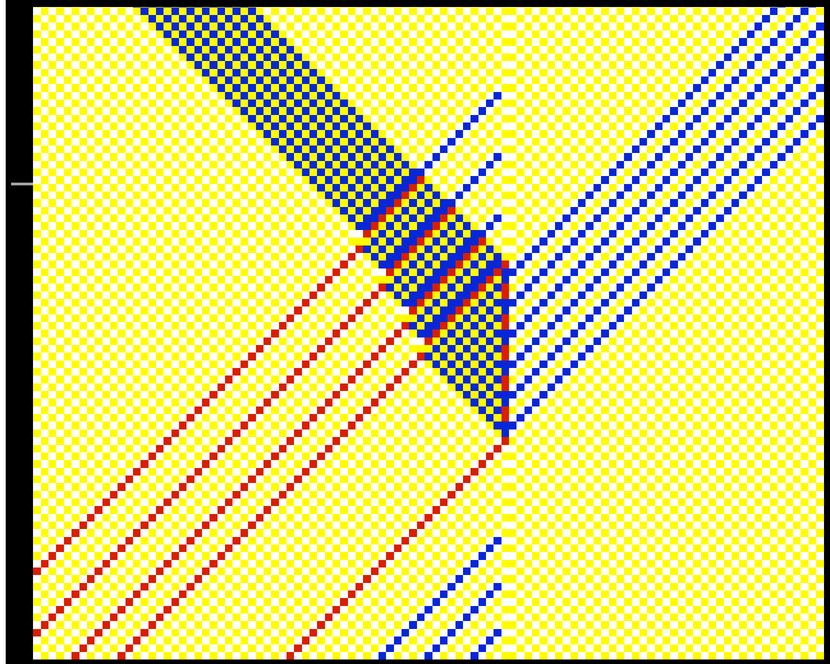
1. à la Mazoyer? (Tuesday morning)

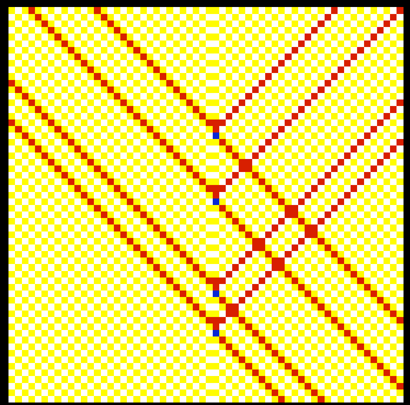
2. Boolean Circuit Simulator?

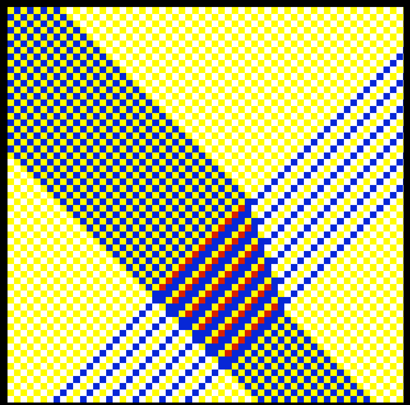
3. Computation Universality?

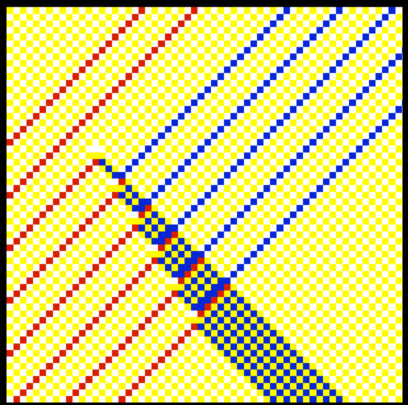
4. Intrinsic Universality?

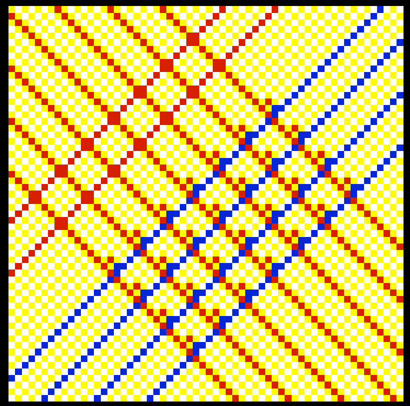


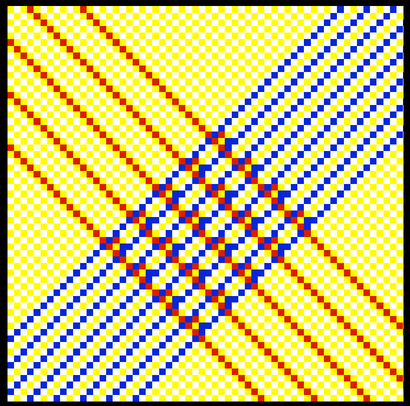


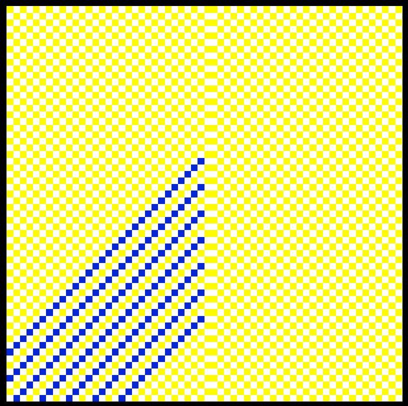












Universal CA...

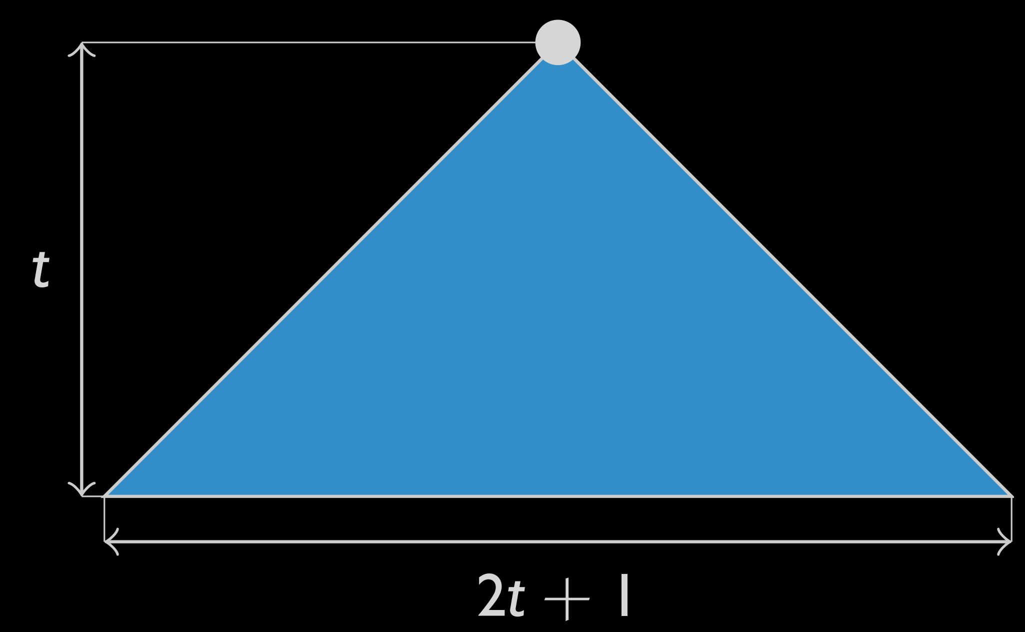
1. **à la Mazoyer?** (Tuesday morning)

2. **Boolean Circuit Simulator?**

3. **Computation Universality?**

4. **Intrinsic Universality?**

5. **P-complete prediction?**



Universal CA...

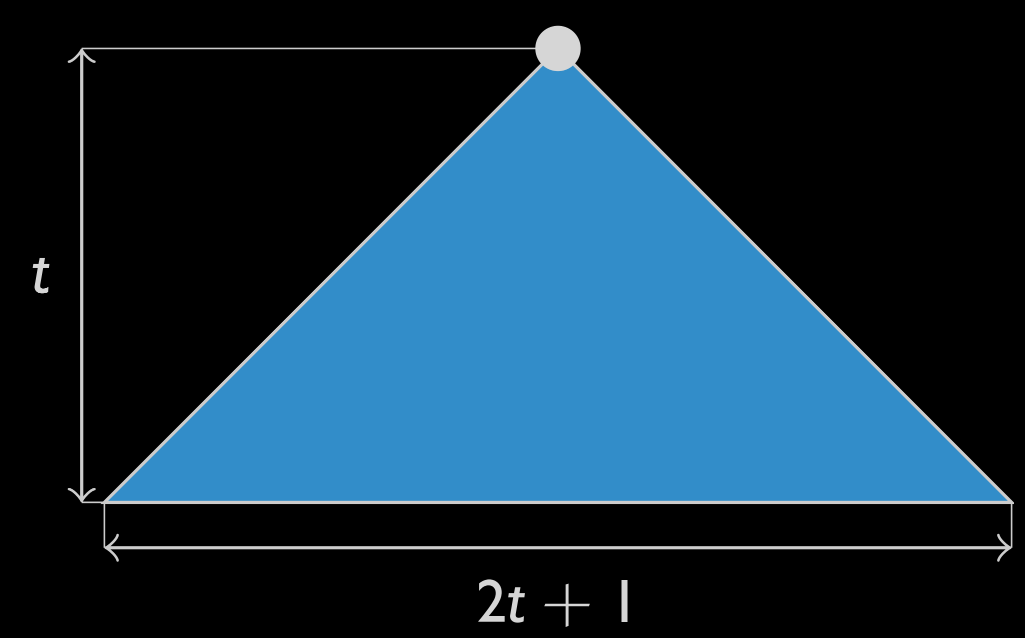
~~1. à la Mazoyer?~~ ~~(Tuesday morning)~~

2. Boolean Circuit Simulator?

3. Computation Universality?

4. Intrinsic Universality?

~~5. P-complete prediction?~~



(i) Higher Dimensions

2D CA

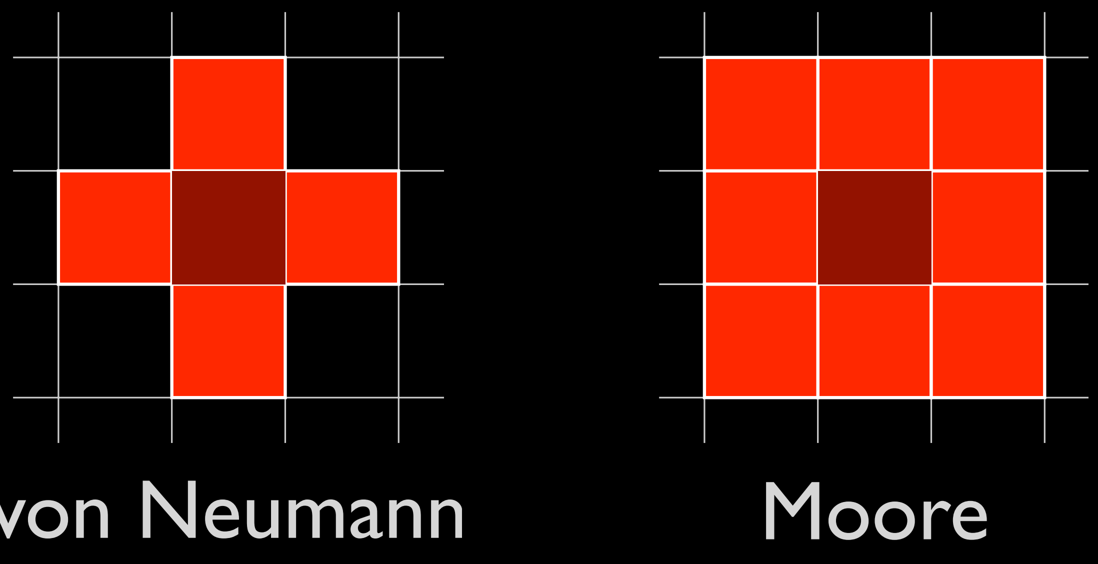
Definition A 2D CA is a triple (S, N, f) where S is the finite set of states, $N \subseteq_{\text{finite}} \mathbb{Z}^2$ is the neighborhood and $f : S^N \to S$ is the local rule of the CA.

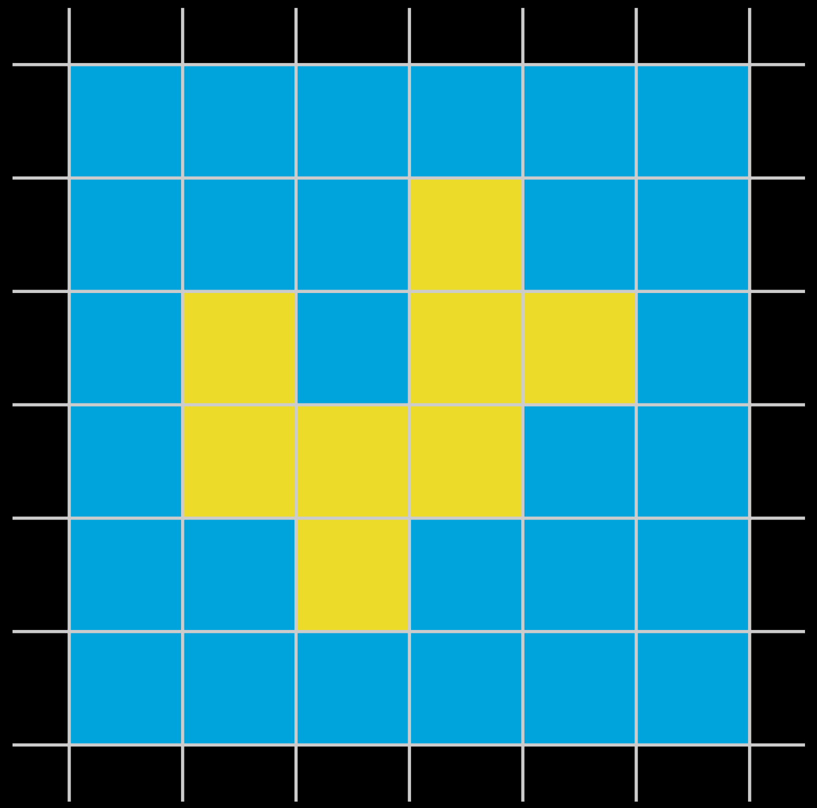
A *configuration* is a mapping $c \in S^{\mathbb{Z}^2}$.

The *global rule* $G : S^{\mathbb{Z}^2} \to S^{\mathbb{Z}^2}$ applies the local rule uniformly:

$$G(c)(i) = f(c(i+v_1), \ldots, c(i+v_k))$$

where $V = \{v_1, \ldots, v_k\}$.





2D CA

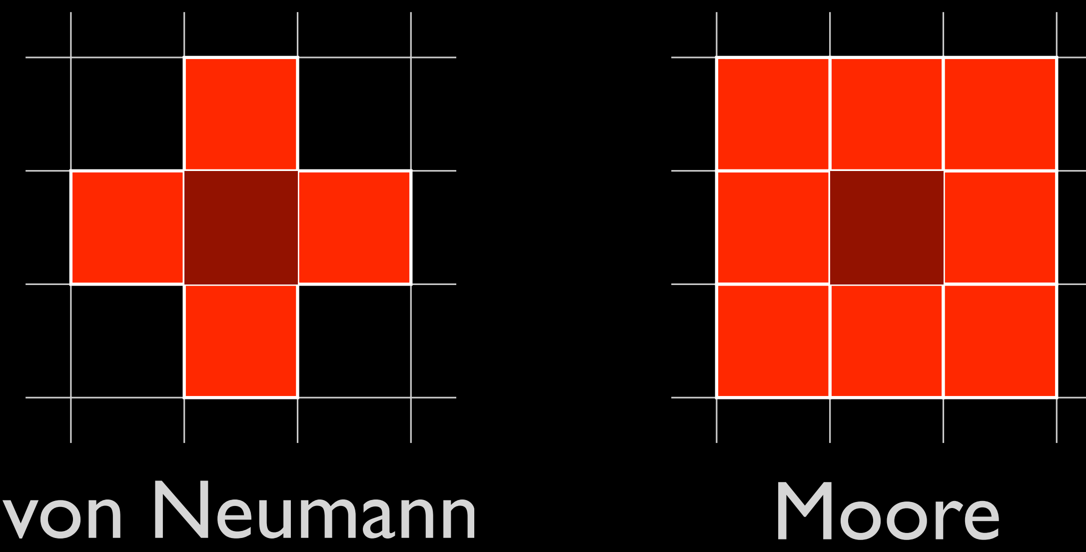
Definition A 2D CA is a triple (S, N, f) where S is the finite set of states, $N \subseteq_{\text{finite}} \mathbb{Z}^2$ is the neighborhood and $f : S^N \to S$ is the local rule of the CA.

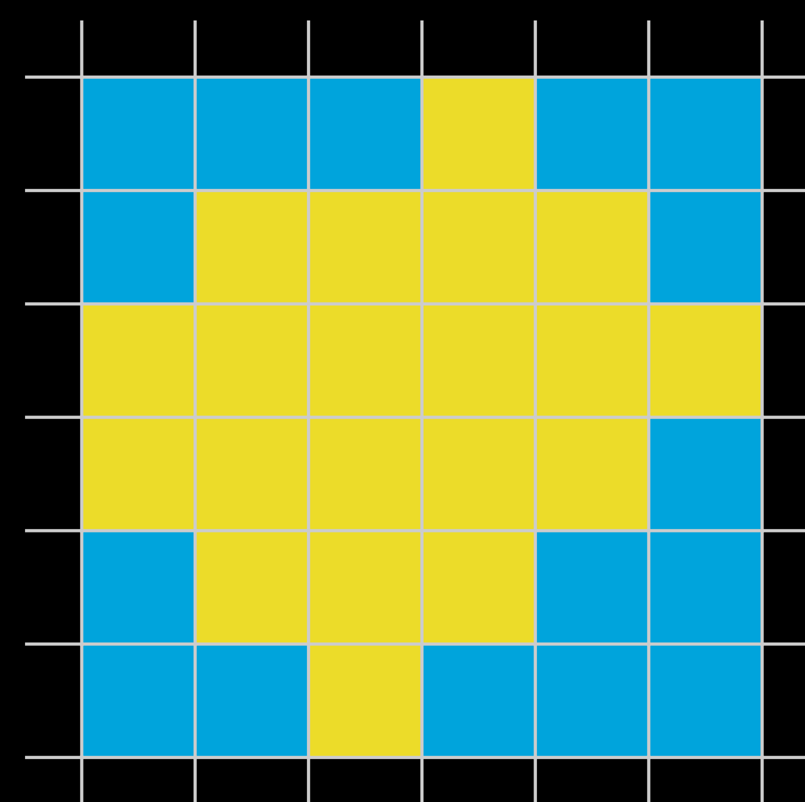
A *configuration* is a mapping $c \in S^{\mathbb{Z}^2}$.

The *global rule* $G : S^{\mathbb{Z}^2} \to S^{\mathbb{Z}^2}$ applies the local rule uniformly:

$$G(c)(i) = f(c(i+v_1), \ldots, c(i+v_k))$$

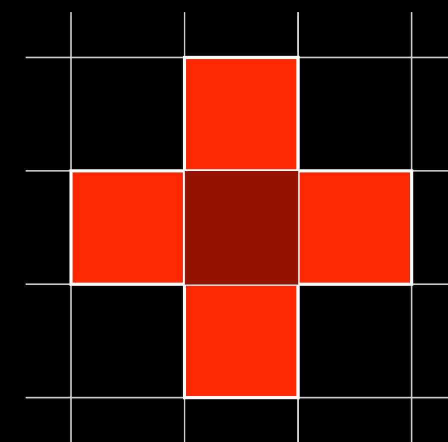
where $V = \{v_1, \ldots, v_k\}$.



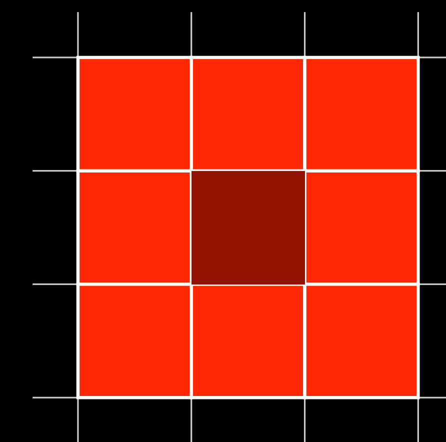


2D CA

Definition A 2D CA is a triple $(S, \mathrm{N}, \mathrm{f})$ where S is the finite set of states, $\mathrm{N} \subseteq_{\text{finite}} \mathbb{Z}^2$ is the neighborhood and $\mathrm{f} : S^{\mathrm{N}} \to S$ is the local rule of the CA.



von Neumann



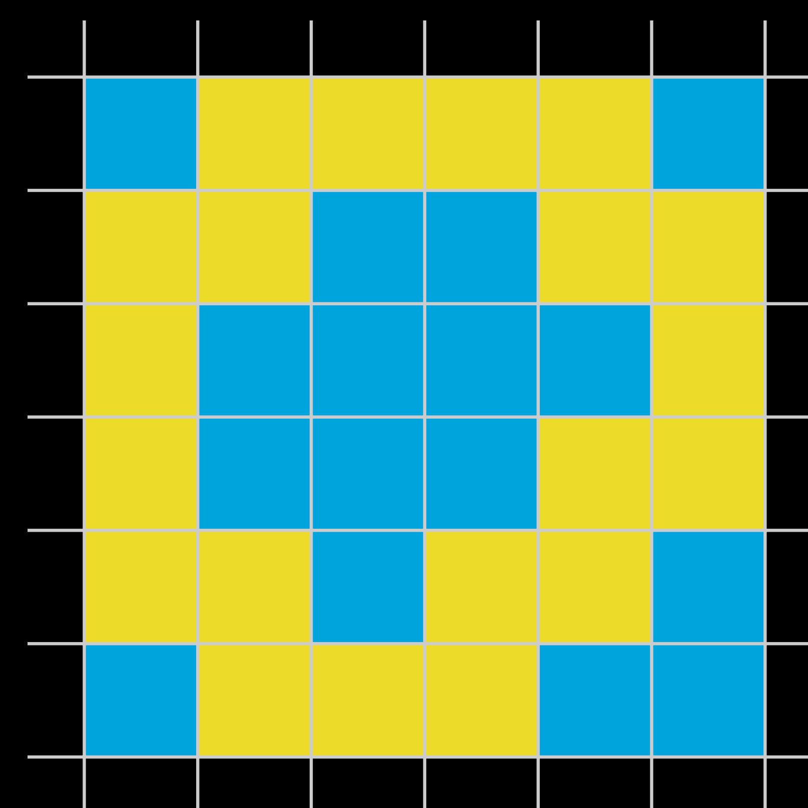
Moore

A *configuration* is a mapping $\mathrm{c} \in S^{\mathbb{Z}^2}$.

The *global rule* $\mathrm{G} : S^{\mathbb{Z}^2} \to S^{\mathbb{Z}^2}$ applies the local rule uniformly:

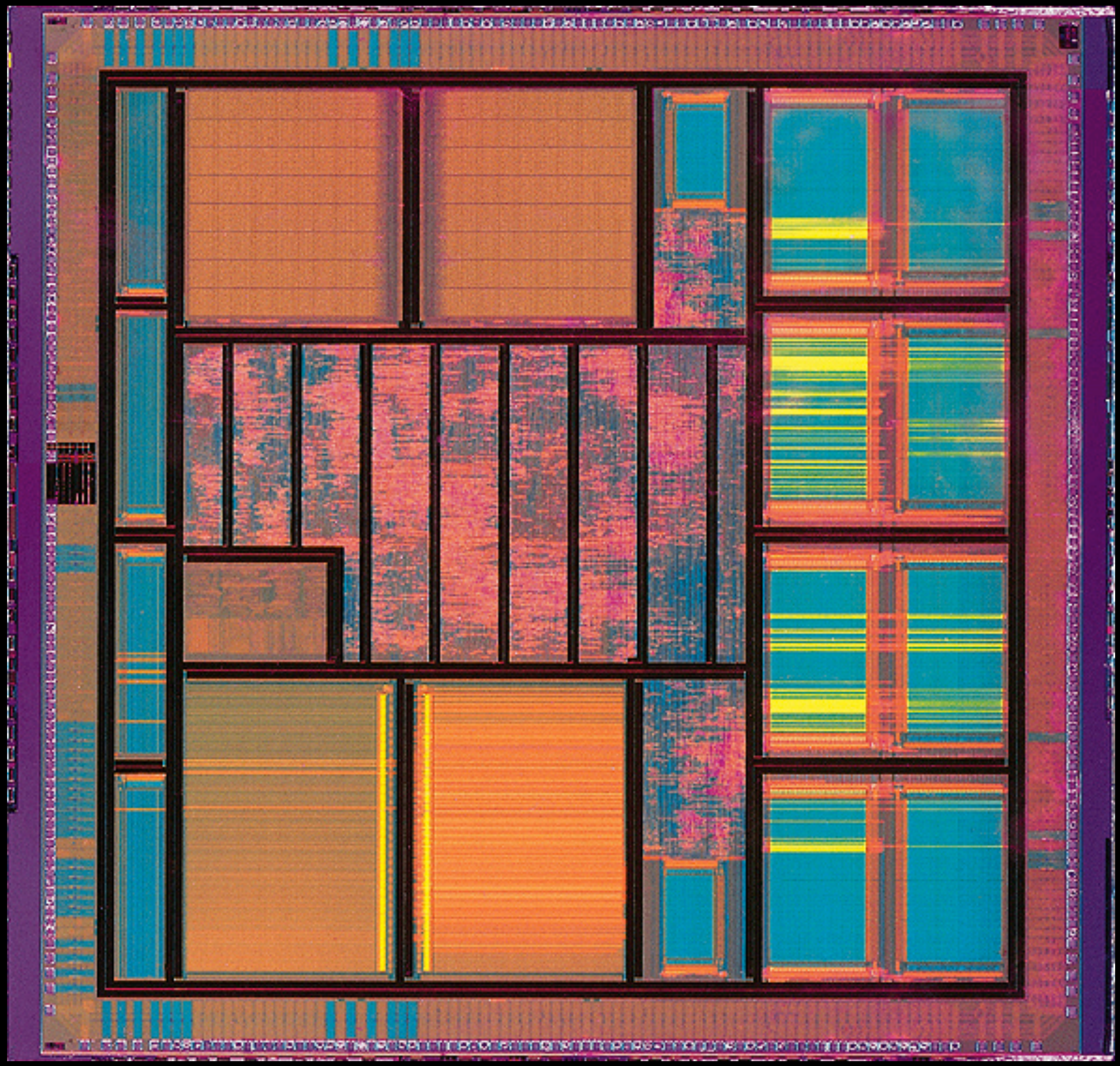
$$\mathrm{G}(\mathrm{c})(\mathrm{i}) = \mathrm{f}(\mathrm{c}(\mathrm{i}+v_1), \ldots, \mathrm{c}(\mathrm{i}+v_\mathrm{k}))$$

where $\mathrm{V} = \{v_1, \ldots, v_\mathrm{k}\}$.



Boolean Circuits

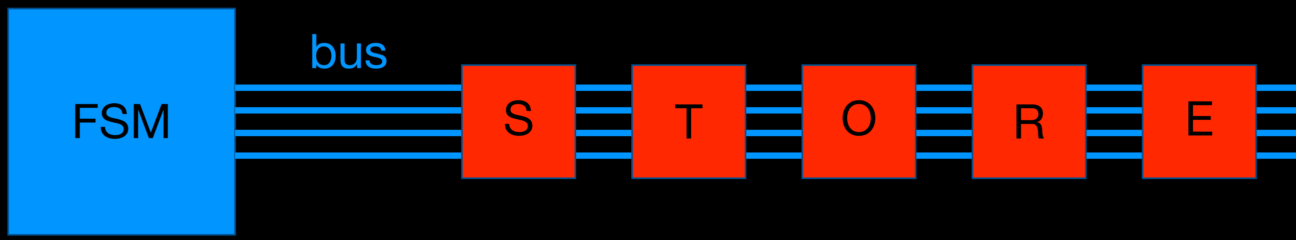
- **[Kleene 56]** boolean circuits = FSM = regular languages.
- Computers are build out of:
  - wires
  - boolean gates
  - delays/clocks
- CMOS and other technology uses 2D objects.



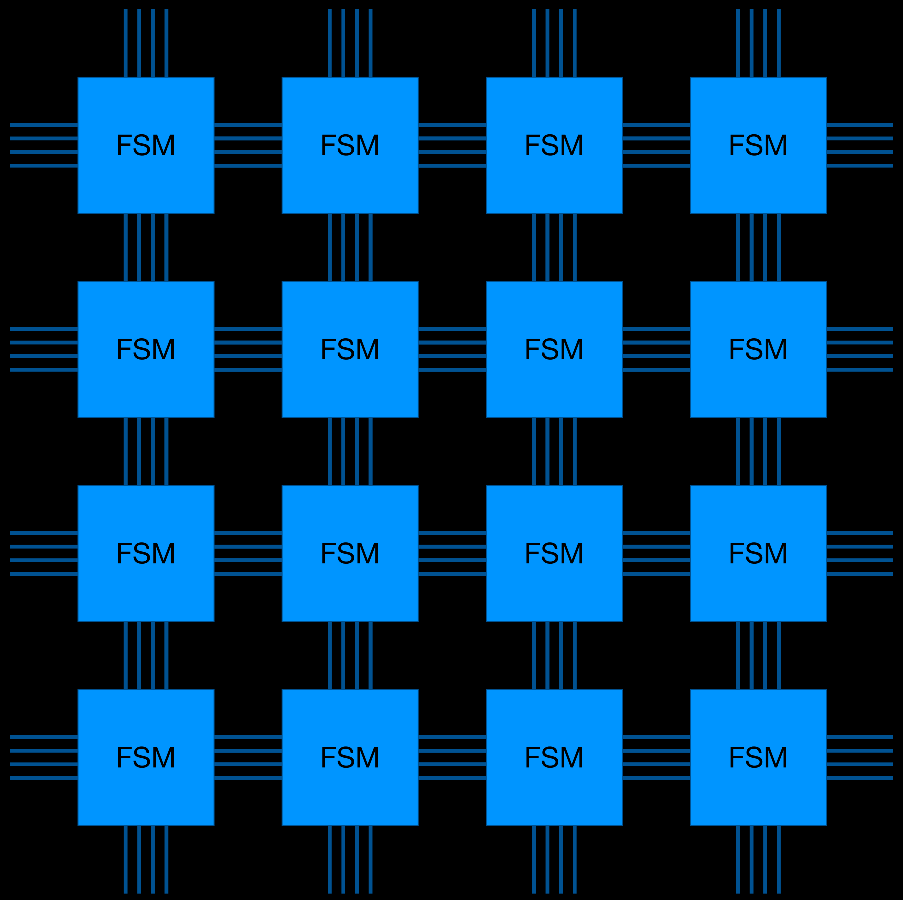
Wikipedia Commons GDL image

Universal Circuits

Boolean circuits can encode both FSM and secondary devices.



Turing Universality can be achieved using FSM (control) + Tape/Registers (storage).



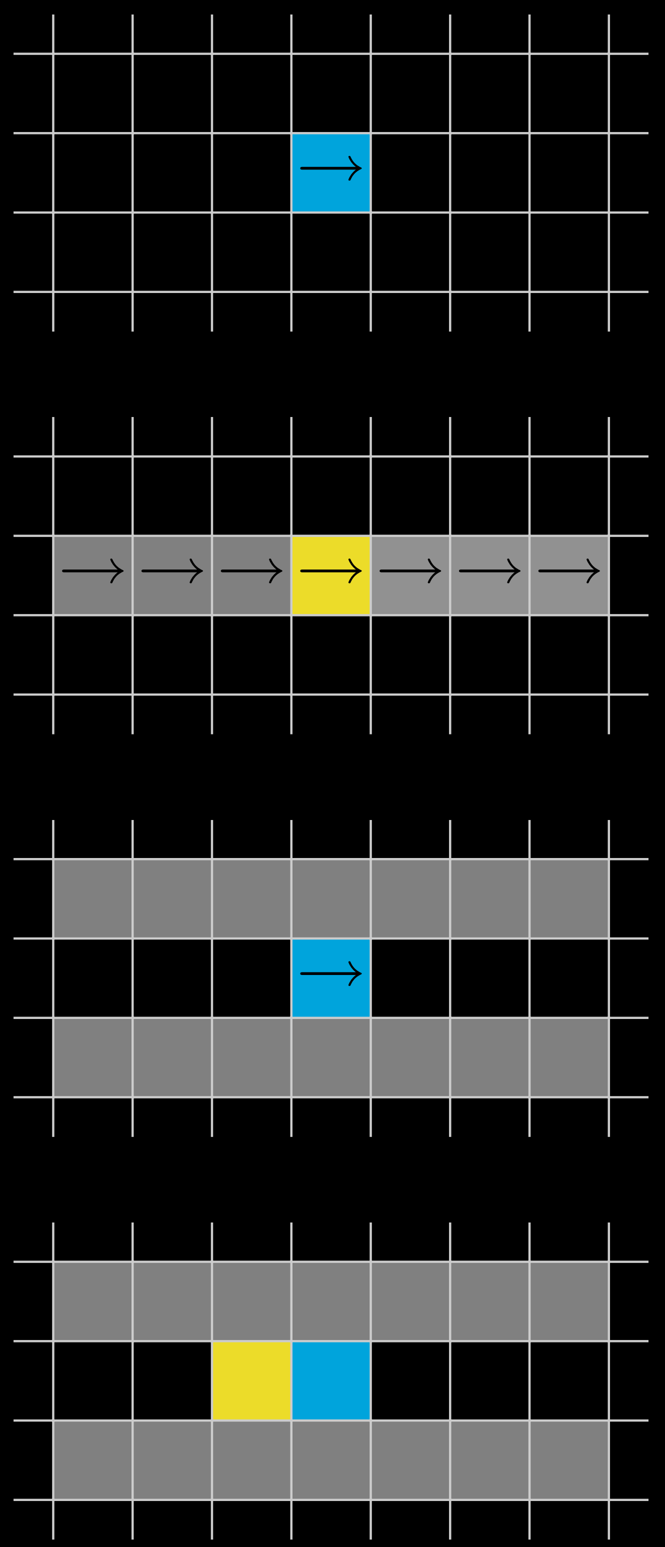
Intrinsic Universality can be achieved using one FSM (local rule) per cell + uniform wiring (transmission).

Transmitting Signals

- **Wires** made out of cells are path for the boolean signals with or without explicit wire, several encodings.

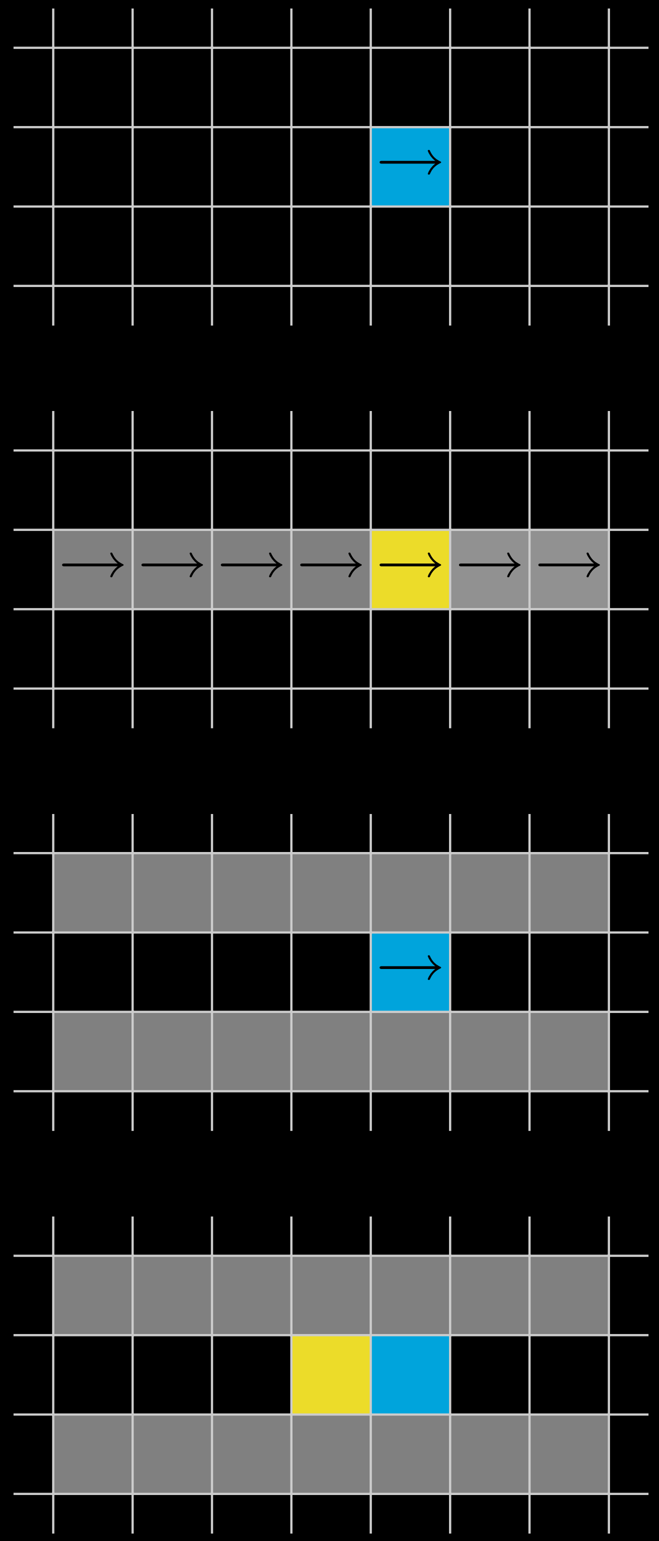
- **Turning** around to route any reasonable family of paths.

- **Fan-out** to route copies a same signal.



Transmitting Signals

- **Wires** made out of cells are path for the boolean signals with or without explicit wire, several encodings.
- **Turning** around to route any reasonable family of paths.
- **Fan-out** to route copies a same signal.

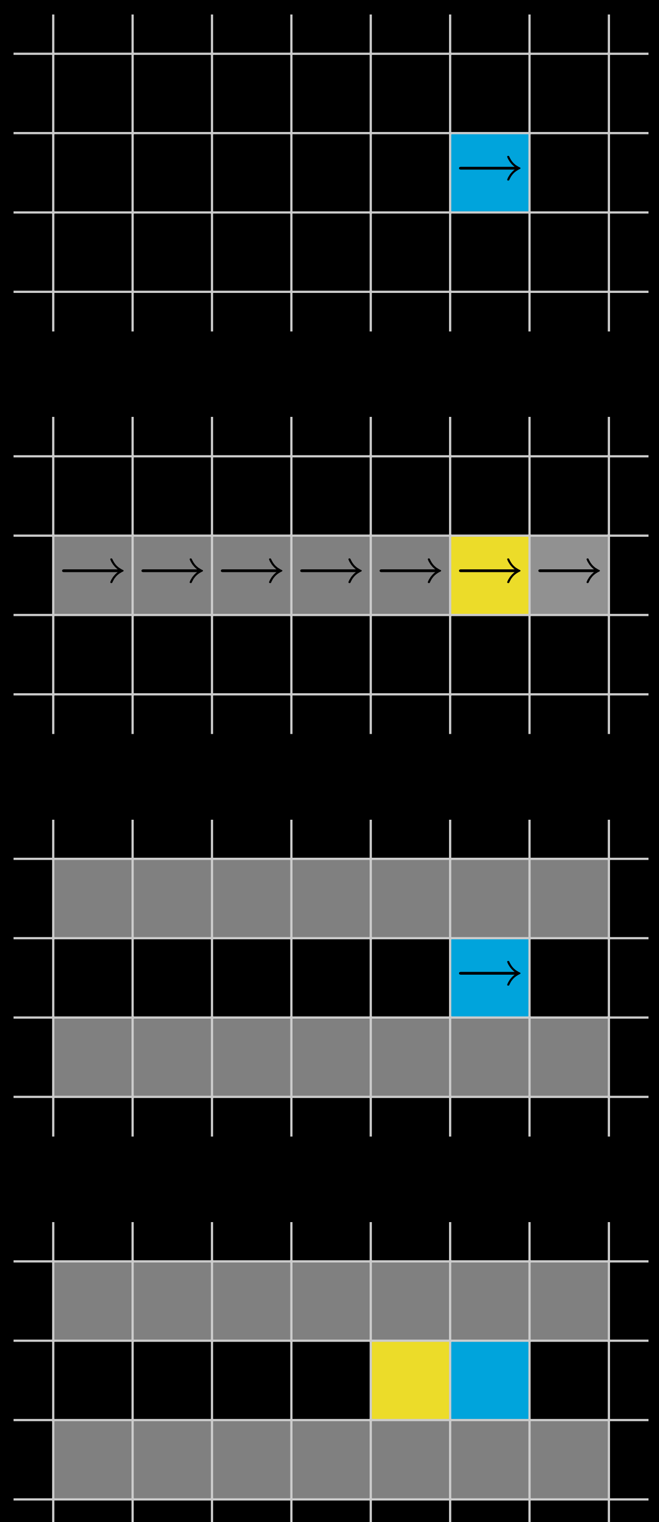


Transmitting Signals

- **Wires** made out of cells are path for the boolean signals with or without explicit wire, several encodings.

- **Turning** around to route any reasonable family of paths.

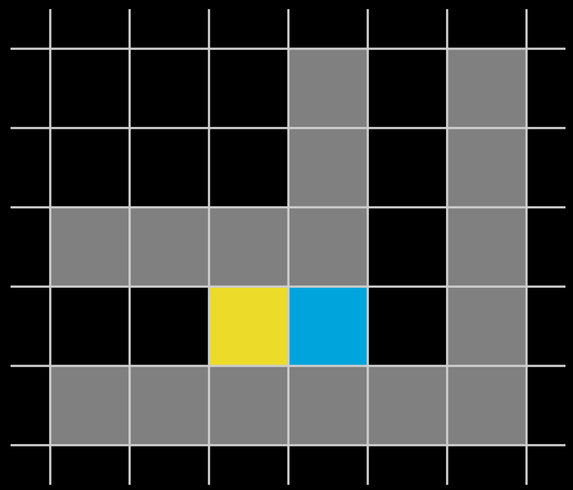
- **Fan-out** to route copies a same signal.

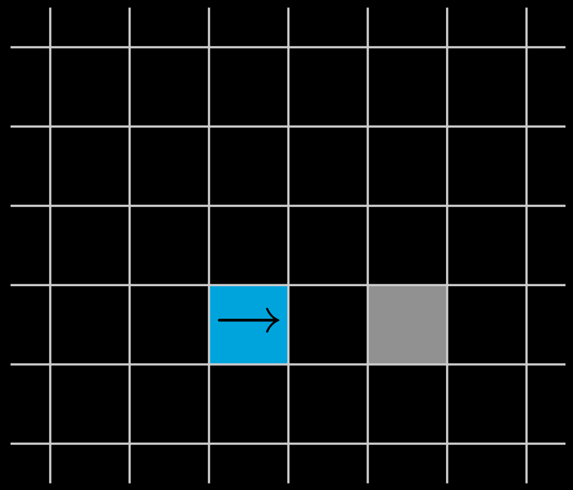


Transmitting Signals

- **Wires** made out of cells are path for the boolean signals with or without explicit wire, several encodings.

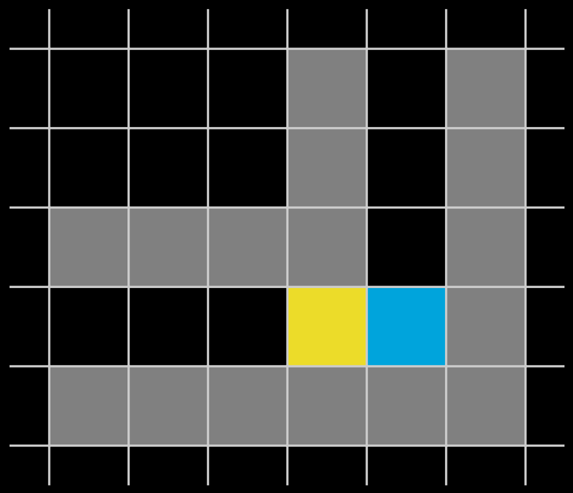
- **Turning** around to route any reasonable family of paths.

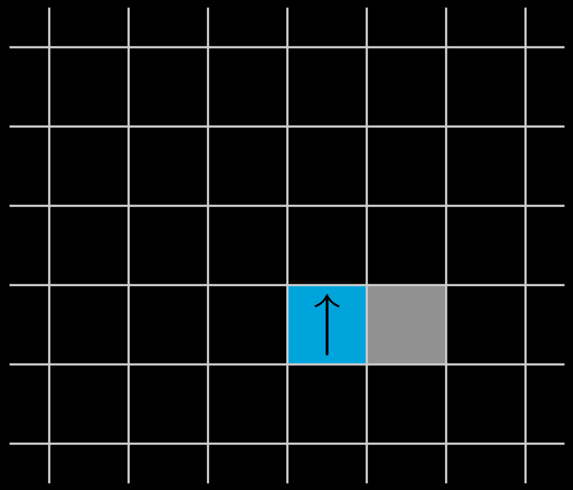




Transmitting Signals

- **Wires** made out of cells are path for the boolean signals with or without explicit wire, several encodings.
- **Turning** around to route any reasonable family of paths.

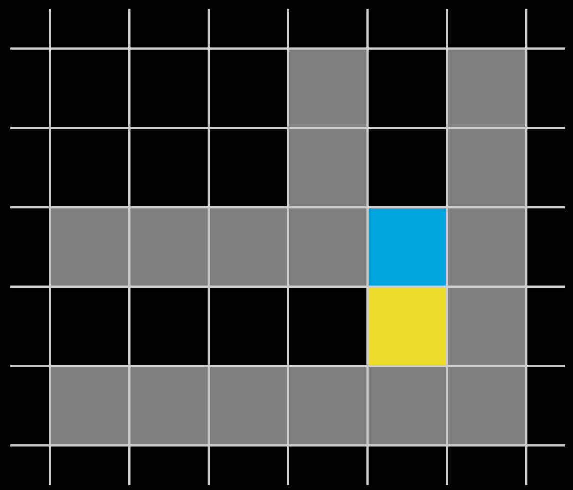


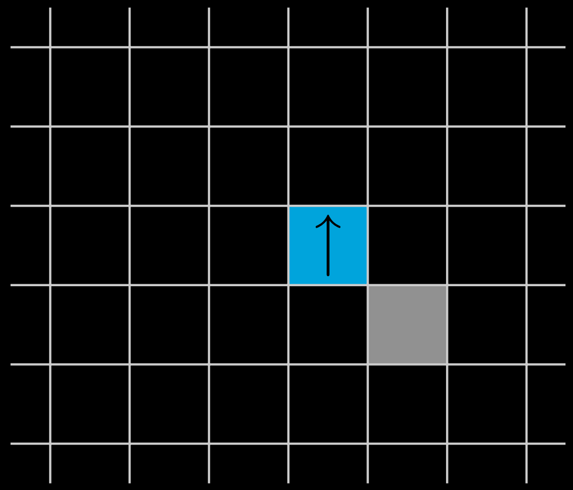


Transmitting Signals

- **Wires** made out of cells are path for the boolean signals with or without explicit wire, several encodings.

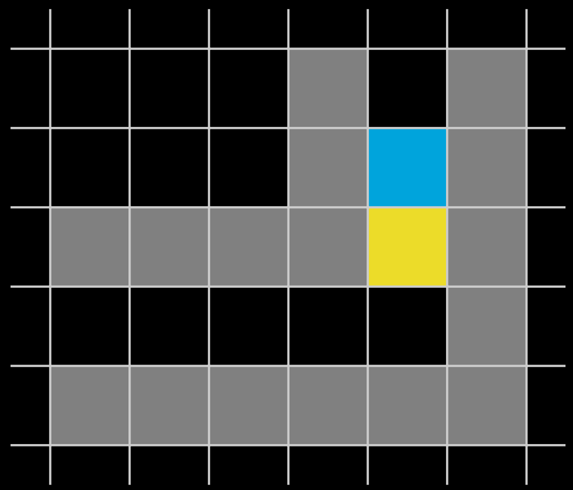
- **Turning** around to route any reasonable family of paths.

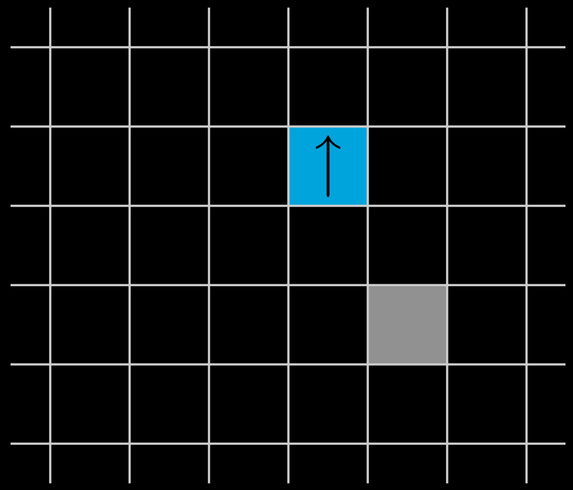




Transmitting Signals

- **Wires** made out of cells are path for the boolean signals with or without explicit wire, several encodings.
- **Turning** around to route any reasonable family of paths.



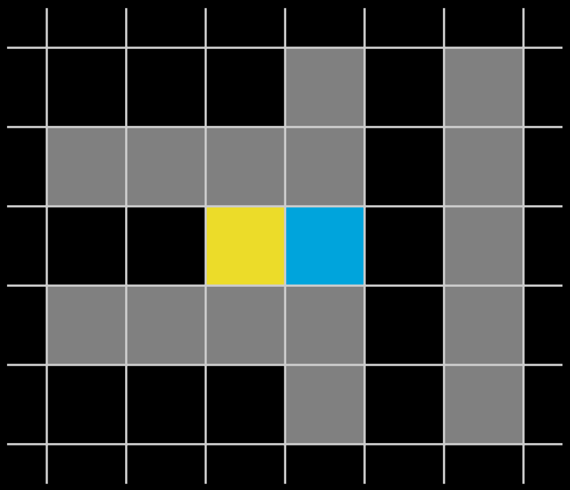


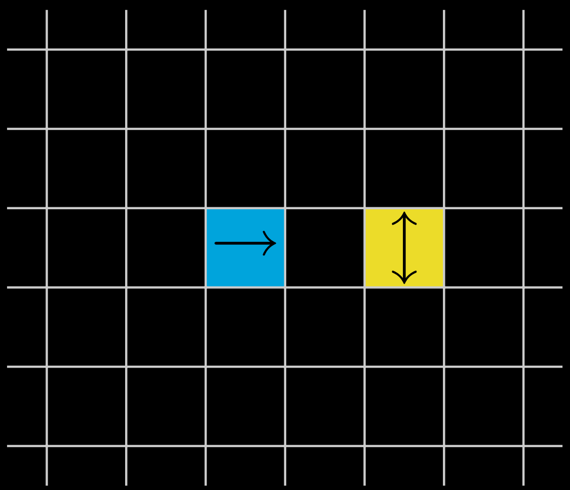
Transmitting Signals

- **Wires** made out of cells are path for the boolean signals with or without explicit wire, several encodings.

- **Turning** around to route any reasonable family of paths.

- **Fan-out** to route copies a same signal.



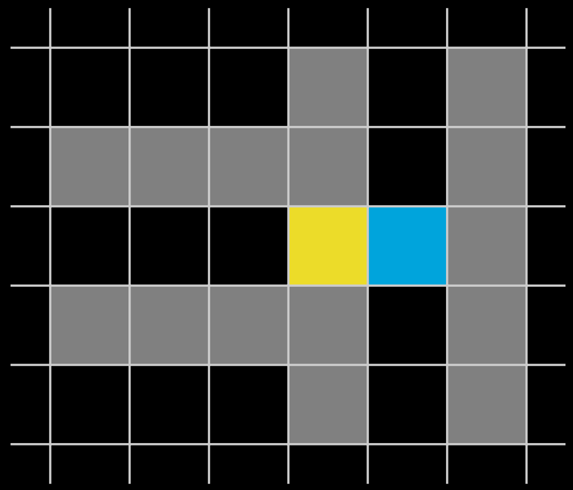


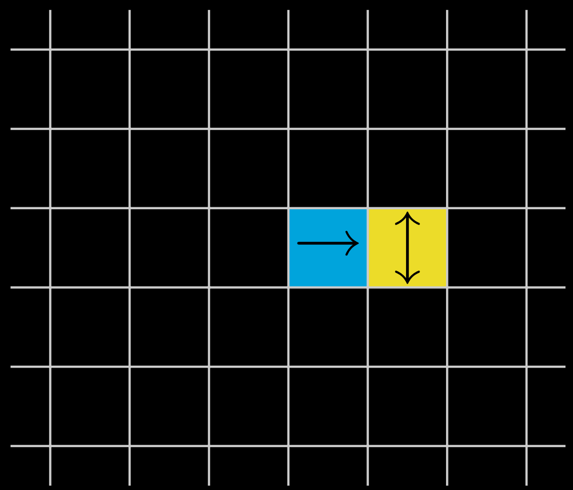
Transmitting Signals

- **Wires** made out of cells are path for the boolean signals with or without explicit wire, several encodings.

- **Turning** around to route any reasonable family of paths.

- **Fan-out** to route copies a same signal.



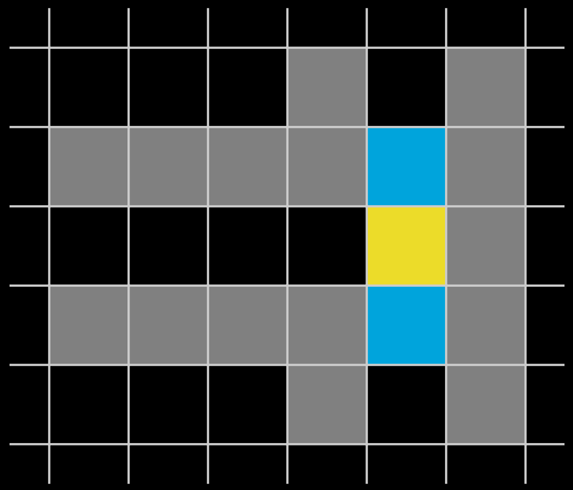


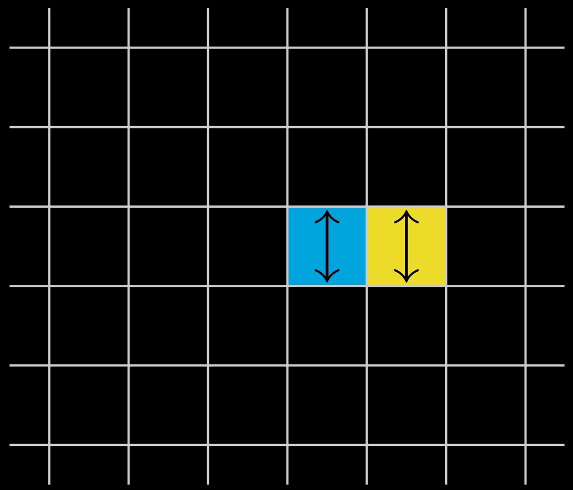
Transmitting Signals

- **Wires** made out of cells are path for the boolean signals with or without explicit wire, several encodings.

- **Turning** around to route any reasonable family of paths.

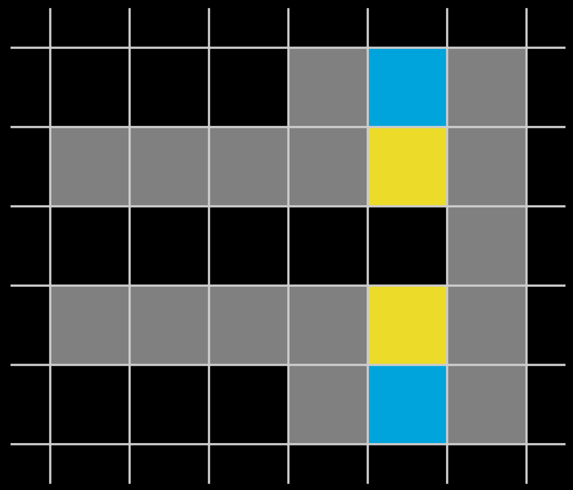
- **Fan-out** to route copies a same signal.





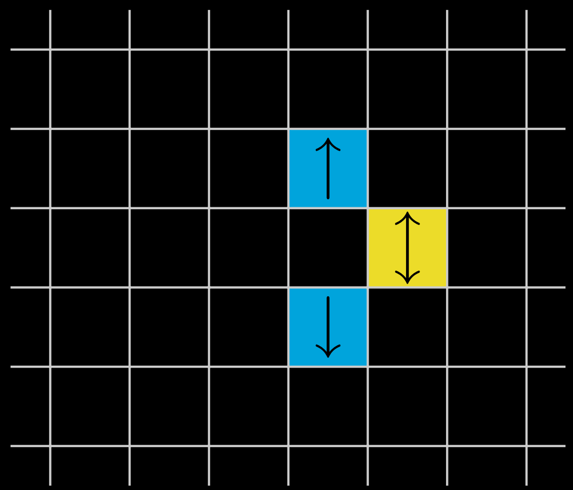
Transmitting Signals

- **Wires** made out of cells are path for the boolean signals with or without explicit wire, several encodings.



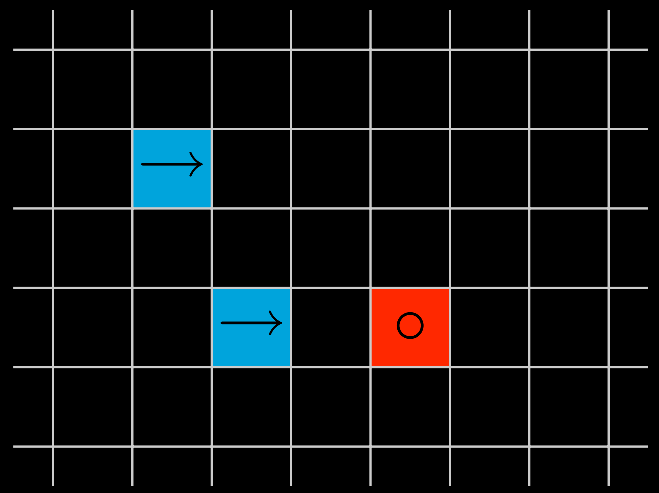
- **Turning** around to route any reasonable family of paths.

- **Fan-out** to route copies a same signal.



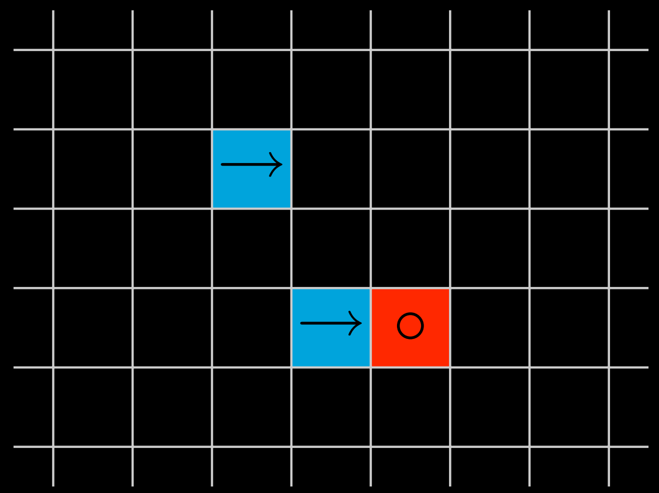
Composing Signals

- **Delays** to synchronize signal arrival at gate input (can be done by turning).
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).
- **Crossing** either explicit or implicit (delay trick or boolean coding).



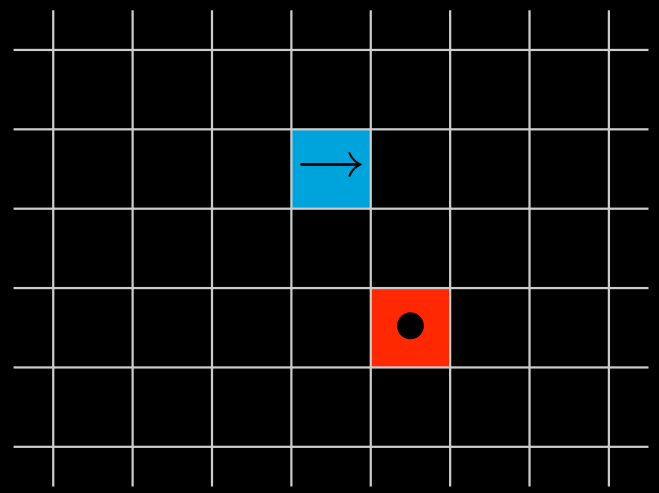
Composing Signals

- **Delays** to synchronize signal arrival at gate input (can be done by turning).
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).
- **Crossing** either explicit or implicit (delay trick or boolean coding).



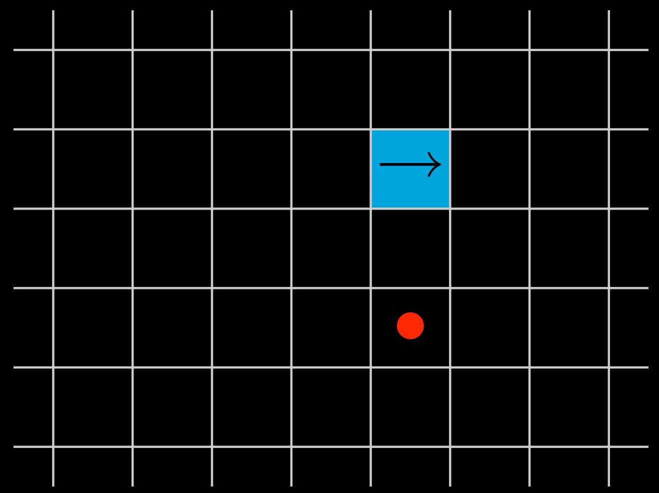
Composing Signals

- **Delays** to synchronize signal arrival at gate input (can be done by turning).
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).
- **Crossing** either explicit or implicit (delay trick or boolean coding).



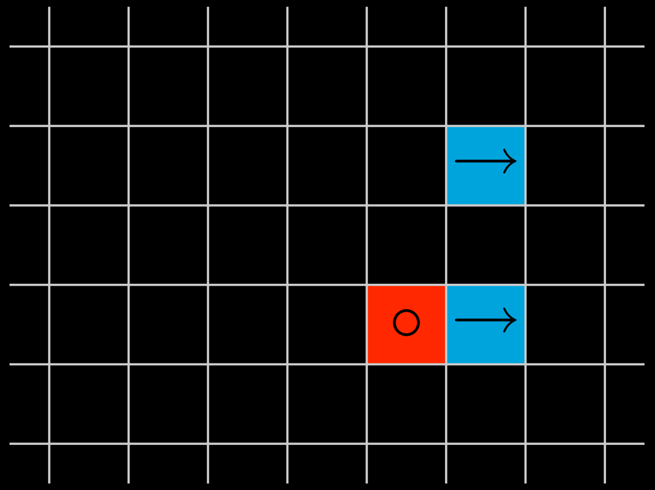
Composing Signals

- **Delays** to synchronize signal arrival at gate input (can be done by turning).
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).
- **Crossing** either explicit or implicit (delay trick or boolean coding).



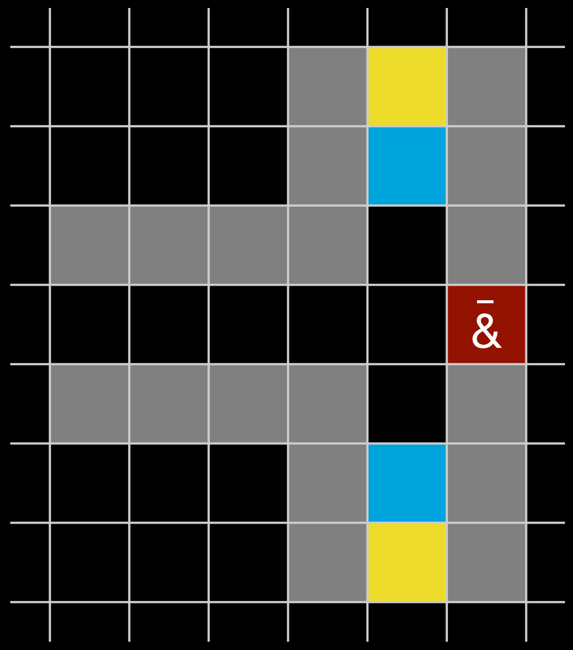
Composing Signals

- **Delays** to synchronize signal arrival at gate input (can be done by turning).
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).
- **Crossing** either explicit or implicit (delay trick or boolean coding).



Composing Signals

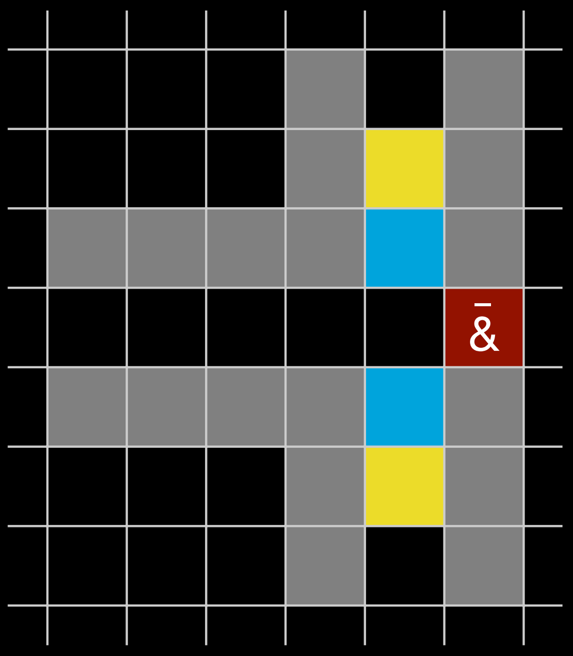
- **Delays** to synchronize signal arrival at gate input (can be done by turning).
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).



Composing Signals

- **Delays** to synchronize signal arrival at gate input (can be done by turning).

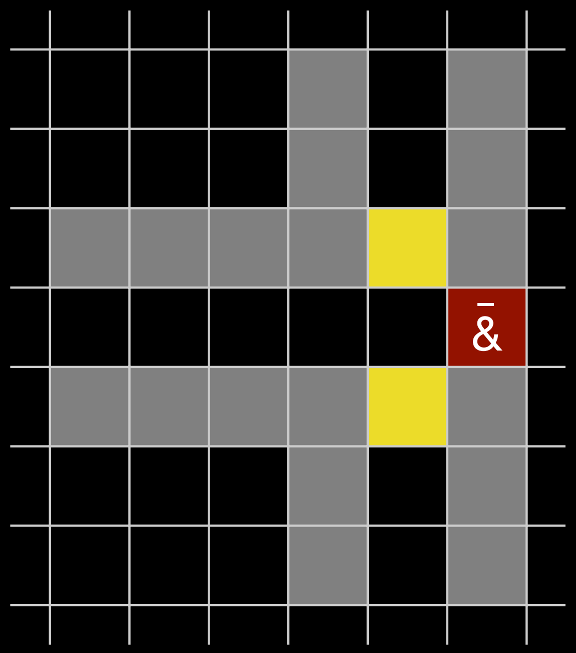
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).



Composing Signals

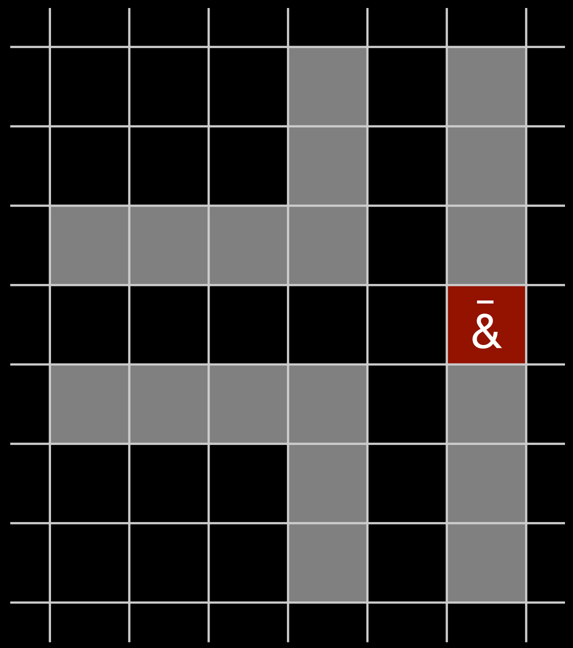
- **Delays** to synchronize signal arrival at gate input (can be done by turning).

- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).



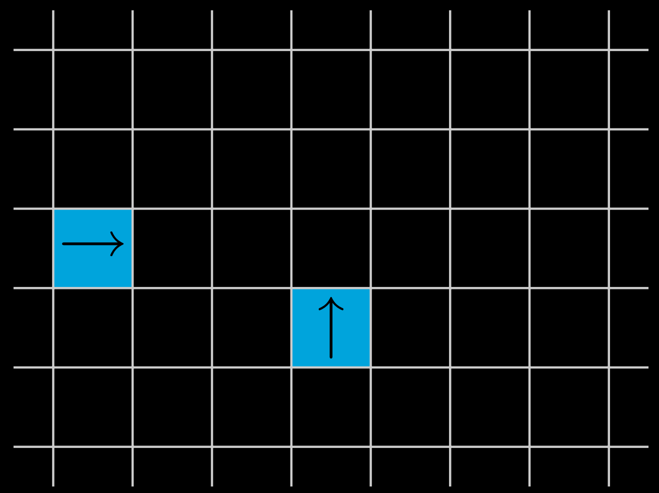
Composing Signals

- **Delays** to synchronize signal arrival at gate input (can be done by turning).
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).



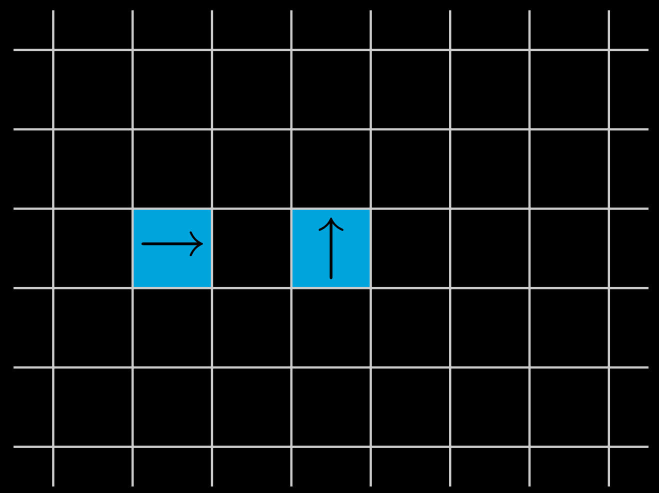
Composing Signals

- **Delays** to synchronize signal arrival at gate input (can be done by turning).
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).
- **Crossing** either explicit or implicit (delay trick or boolean coding).



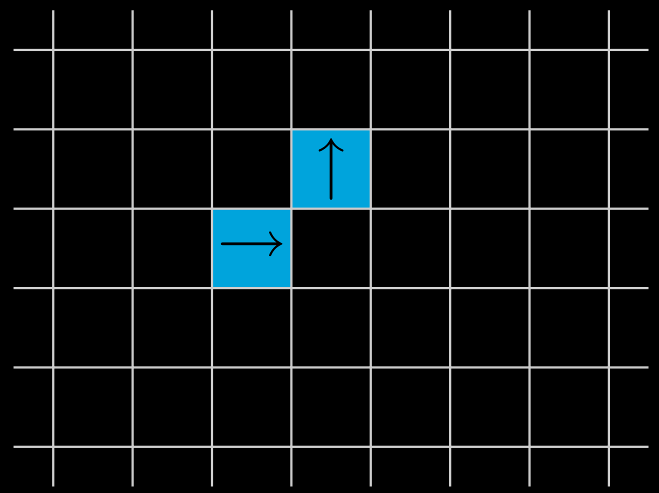
Composing Signals

- **Delays** to synchronize signal arrival at gate input (can be done by turning).
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).
- **Crossing** either explicit or implicit (delay trick or boolean coding).



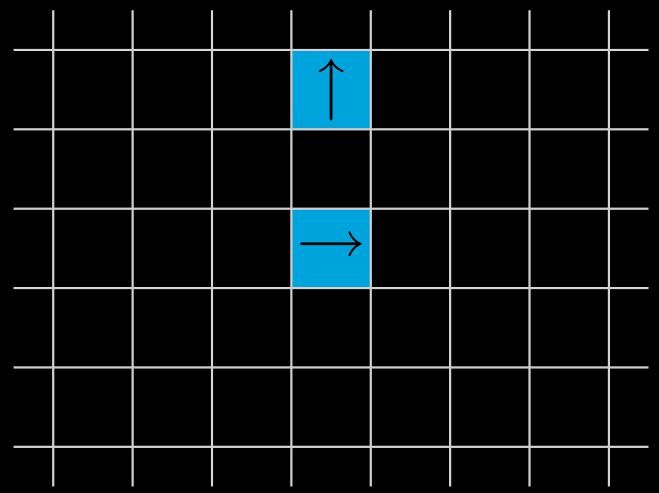
Composing Signals

- **Delays** to synchronize signal arrival at gate input (can be done by turning).
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).
- **Crossing** either explicit or implicit (delay trick or boolean coding).



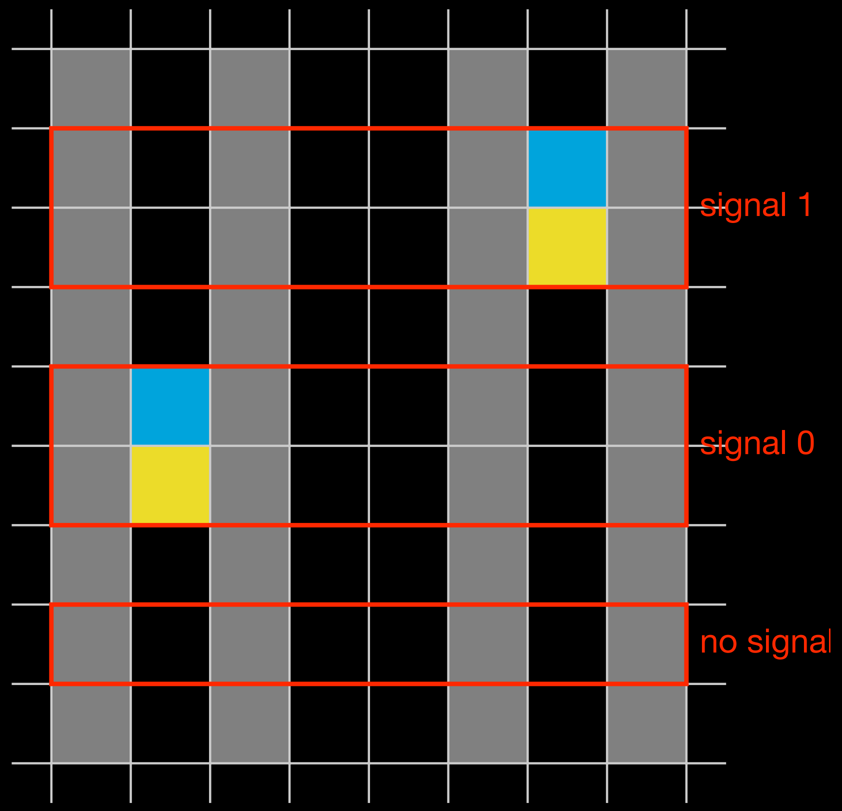
Composing Signals

- **Delays** to synchronize signal arrival at gate input (can be done by turning).
- **Gates** taken in a universal boolean family (like NAND or OR+NOT, constants allowed).
- **Crossing** either explicit or implicit (delay trick or boolean coding).



Clock?

- Three values on a wire:
  - No signal
  - Signal 0
  - Signal 1
- What is the behavior of a NAND gate?
- Either use a clock...
- ...or encode signals on two wires (with AND + OR + Xing).



Examples

von Neumann 1966. 29 states, *2 type of arrow paths + delays*

Codd 1968. 8 states, *explicit undirected wire + 5 signal types*

Banks 1970. 2 states, *trickier encoding of signals*

Conway 1970. Game of Life, 2 states (Moore neighborhood), gliders

(ii) Turing-Universality

1D CA

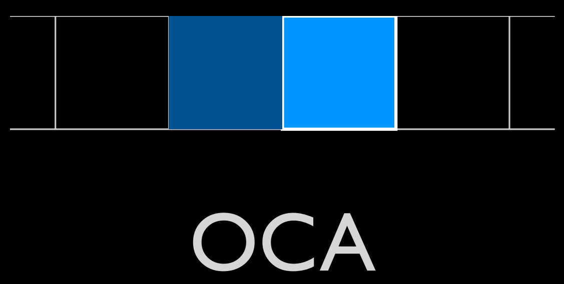
Definition A 1D CA is a triple (S, N, f) where S is the finite set of states, $N \subseteq_{\text{finite}} \mathbb{Z}$ is the neighborhood and $f : S^N \to S$ is the local rule of the CA.

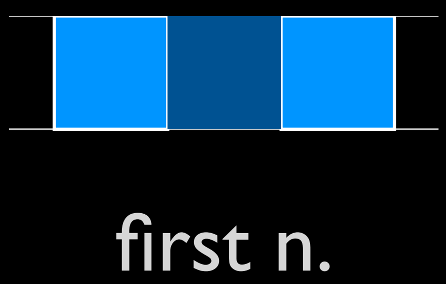
A *configuration* is a mapping $c \in S^{\mathbb{Z}}$.

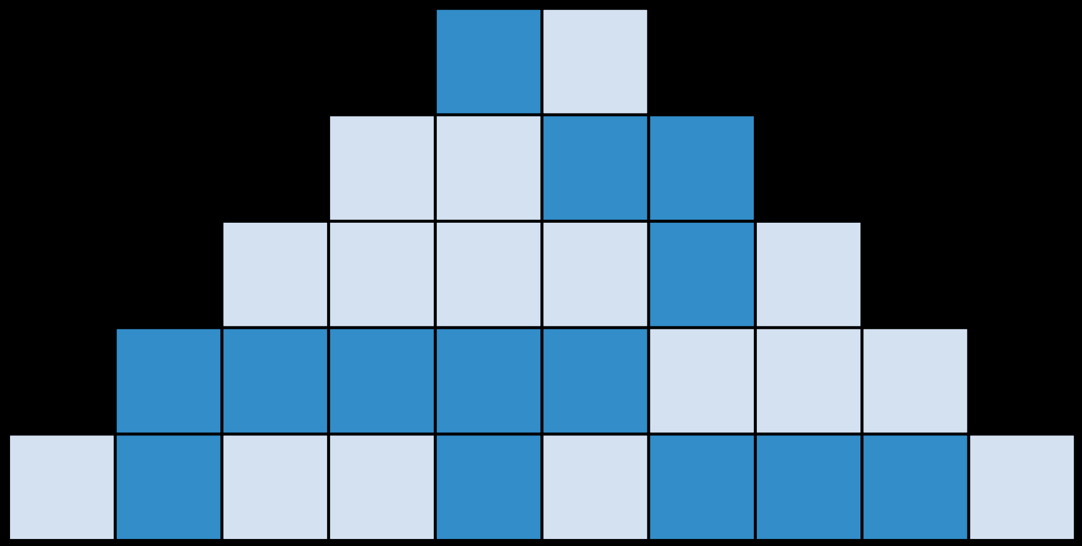
The *global rule* $G : S^{\mathbb{Z}} \to S^{\mathbb{Z}}$ applies the local rule uniformly:

$$G(c)(i) = f(c(i+v_1), \ldots, c(i+v_k))$$

where $V = \{v_1, \ldots, v_k\}$.







Space-time diagram

Turing-completeness

- In 1D boolean circuit are not that easy to simulate in space-time.

- For Turing Machines **[Turing 36]** introduces **the** Universal machine (with respect to a given enumeration of TM and pairs encoding).

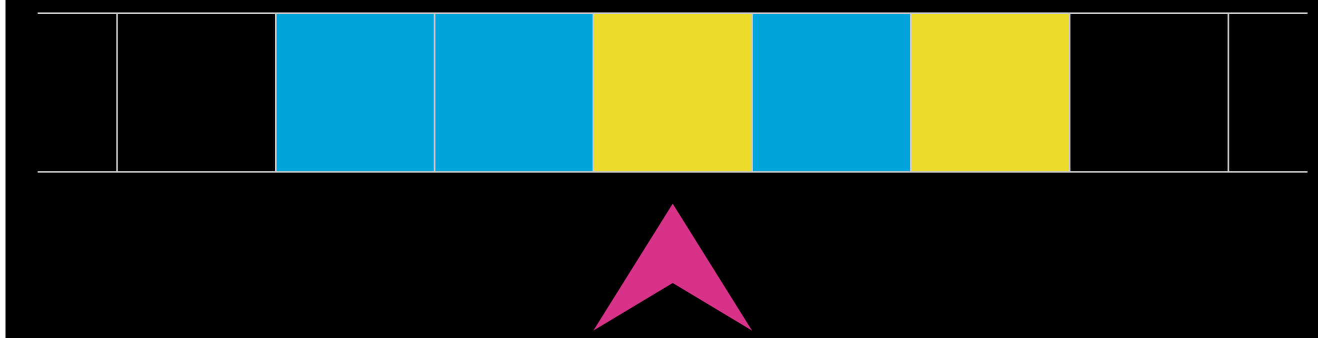
$$\forall i, j \quad \varphi_U(\langle i, j \rangle) = \varphi_i(j)$$

- Moreover, it is classical to express the power of models of computation by simulating well-known Turing-complete models.

- The *intuition* says “a CA is universal if it can **simulate** any Turing machine”... Or replace TM by any reasonable Model of Computation.

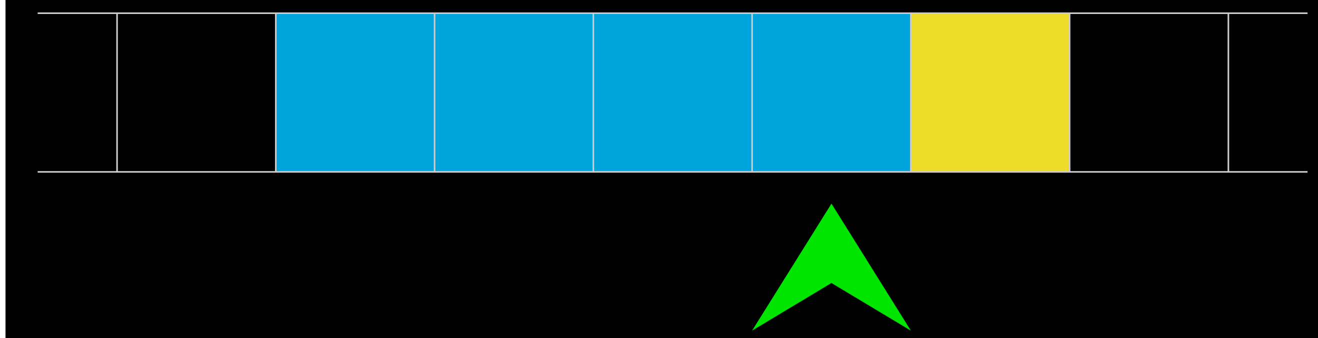
Turing Machines

- TM = FSM + biinfinite tape
- Actions: read, write, move
- Input on the tape
- Initial state
- Halting state
- Output on the tape



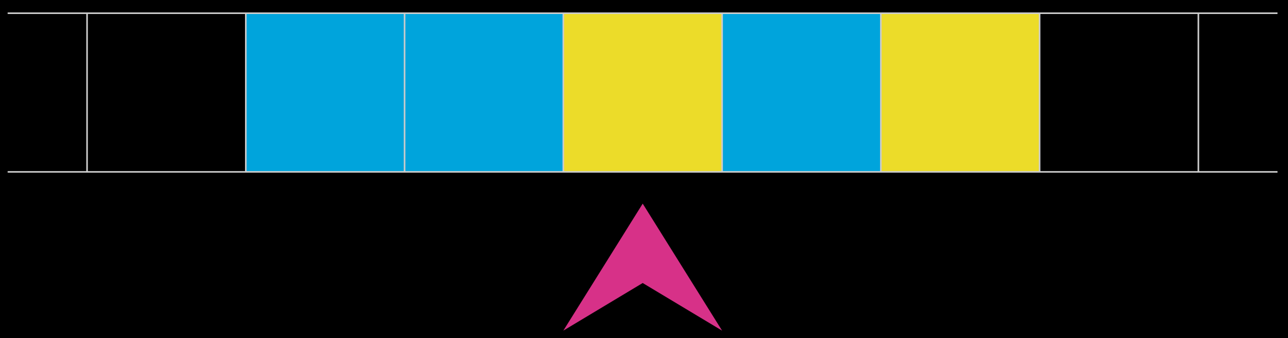
Turing Machines

- TM = FSM + biinfinite tape
- Actions: read, write, move
- Input on the tape
- Initial state
- Halting state
- Output on the tape



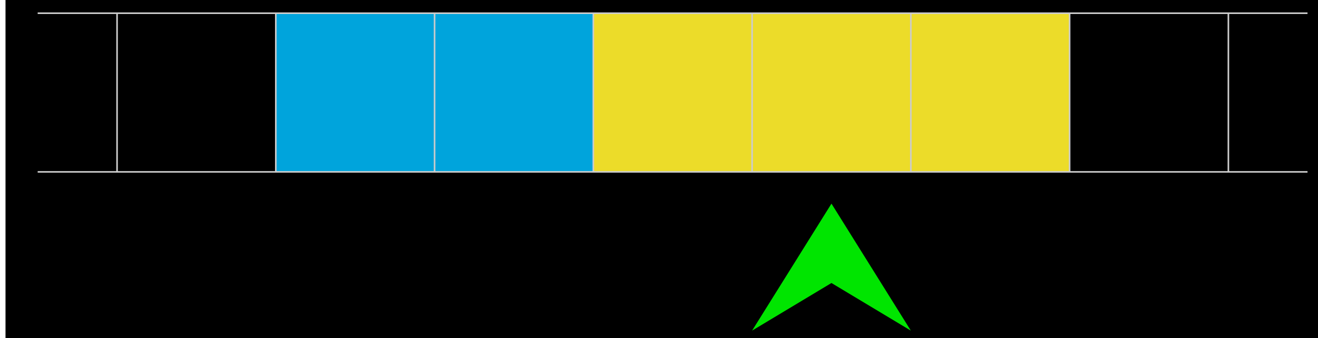
Turing Machines

- TM = FSM + biinfinite tape
- Actions: read, write, move
- Input on the tape
- Initial state
- Halting state
- Output on the tape



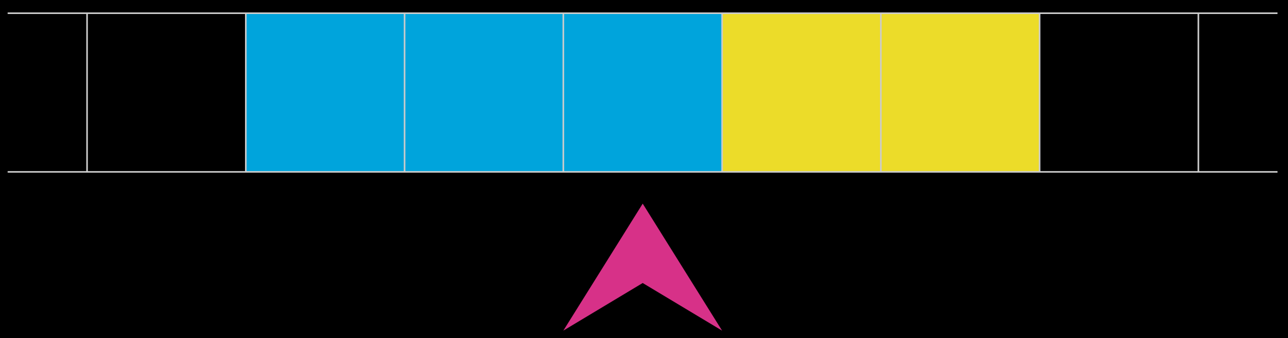
Turing Machines

- TM = FSM + biinfinite tape
- Actions: read, write, move
- Input on the tape
- Initial state
- Halting state
- Output on the tape



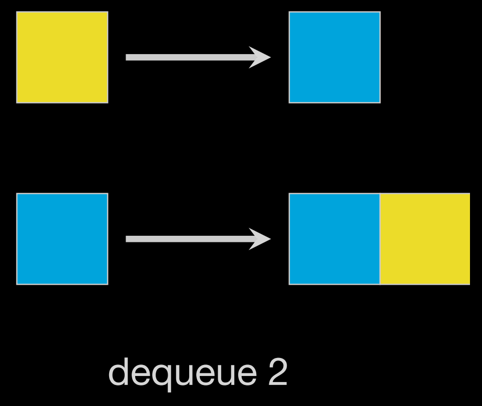
Turing Machines

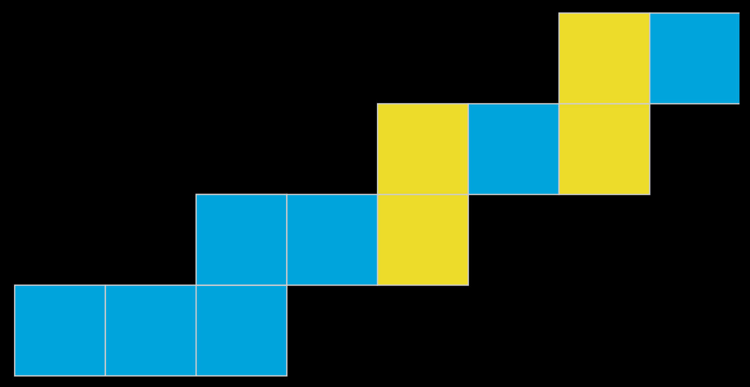
- TM = FSM + biinfinite tape
- Actions: read, write, move
- Input on the tape
- Initial state
- Halting state
- Output on the tape



Tag Systems

- Introduced by Post
- TS = FSM + queue
- Actions: enqueue, dequeue
- Input, halt, small output
- Canonical model: no state, constant dequeue + enqueue word depending on prefix.





Turing-Universality

[Durand and Róka 1996] formalization is needed to define the frontier between universal and non universal (and prove things) but there are several difficult problems:

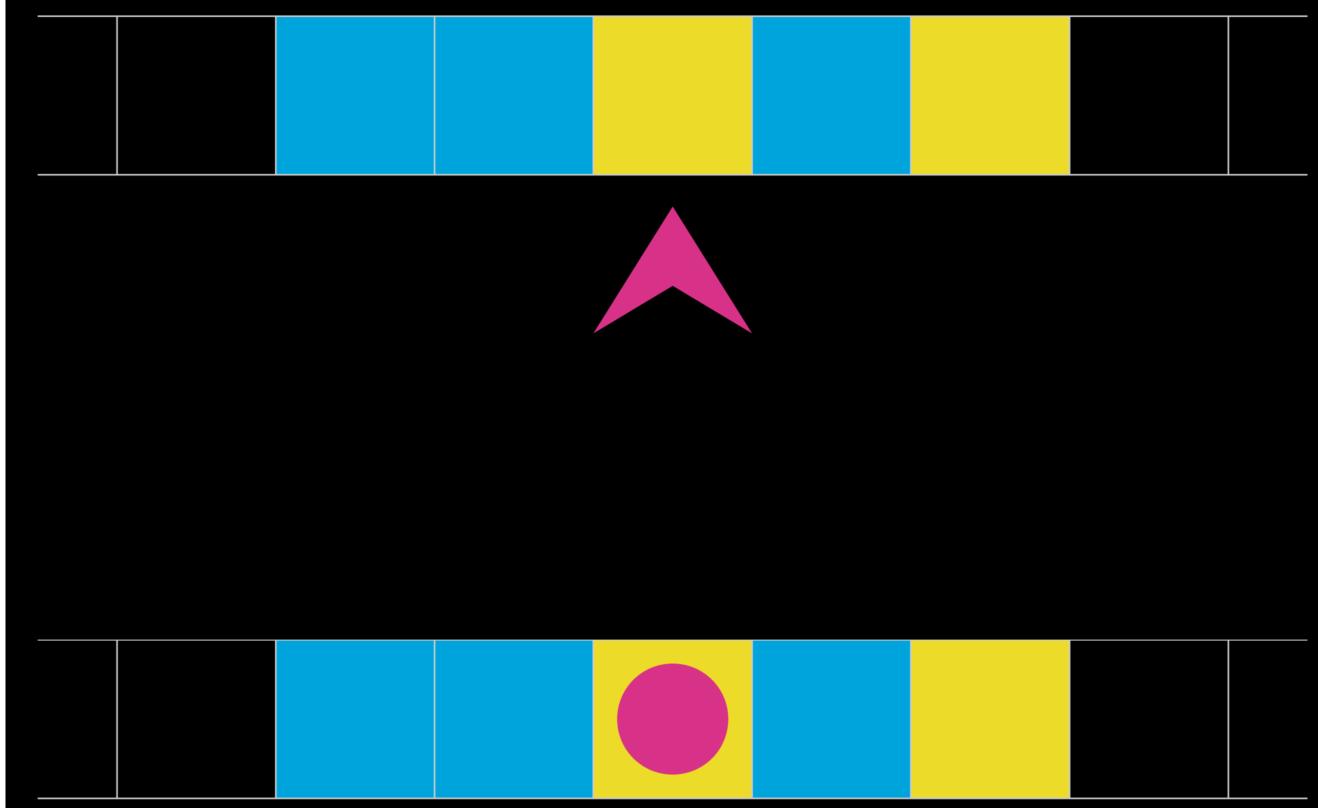
- When CA simulate an extrinsic model, how is it permitted to encode the input (infinite configuration)? what is a halting condition? how do we decode the output (infinite configuration)?

- More pragmatically, there seems to be no agreement on the definition of **a** universal TM or universal TS.

Having **no definition** is a major drawback of Turing-Universality.

à la Smith III

- The configuration encodes the tape.
- The cell pointed by the head also contains the state.

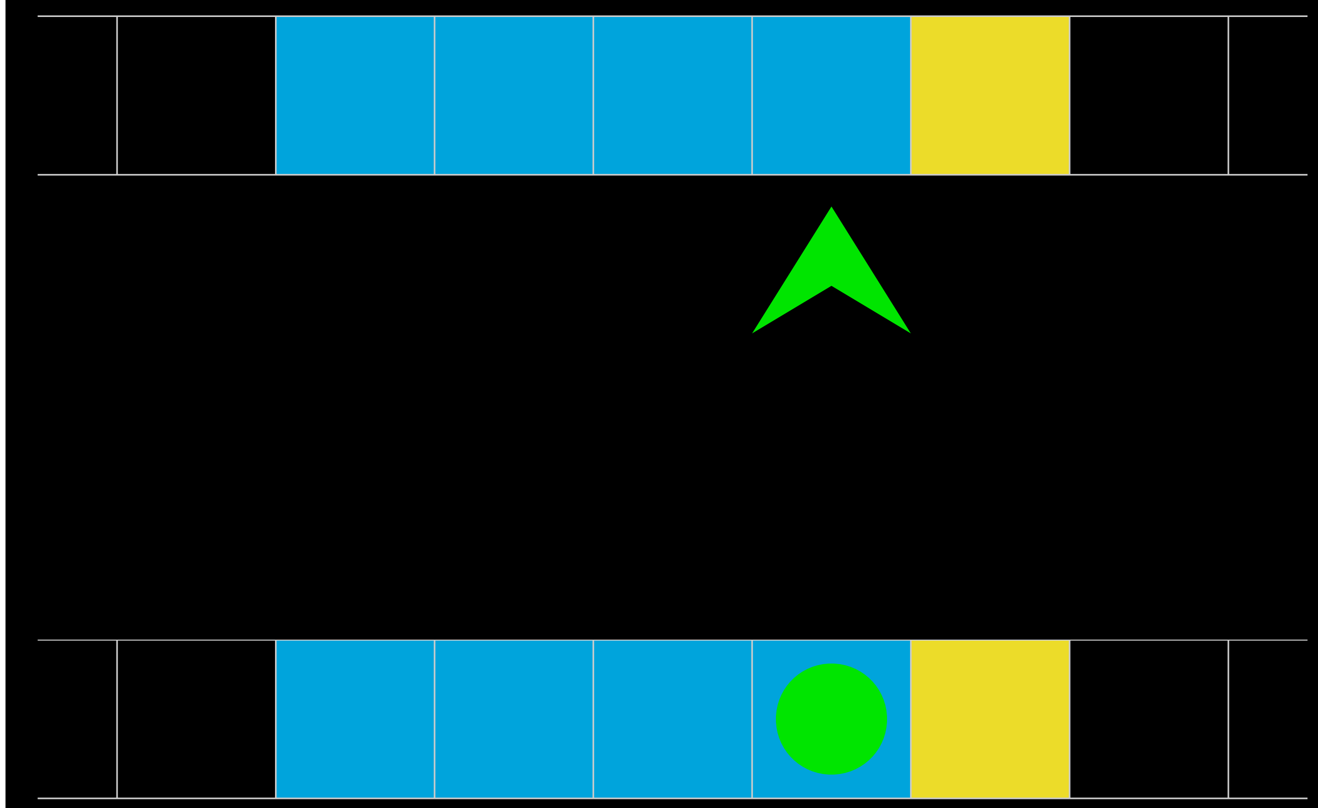


$$(\Sigma \cup S \times \Sigma, \{-1, 0, 1\}, f)$$

$m(n+1)$ states

à la Smith III

- The configuration encodes the tape.
- The cell pointed by the head also contains the state.



$$(\Sigma \cup S \times \Sigma, \{-1, 0, 1\}, f)$$

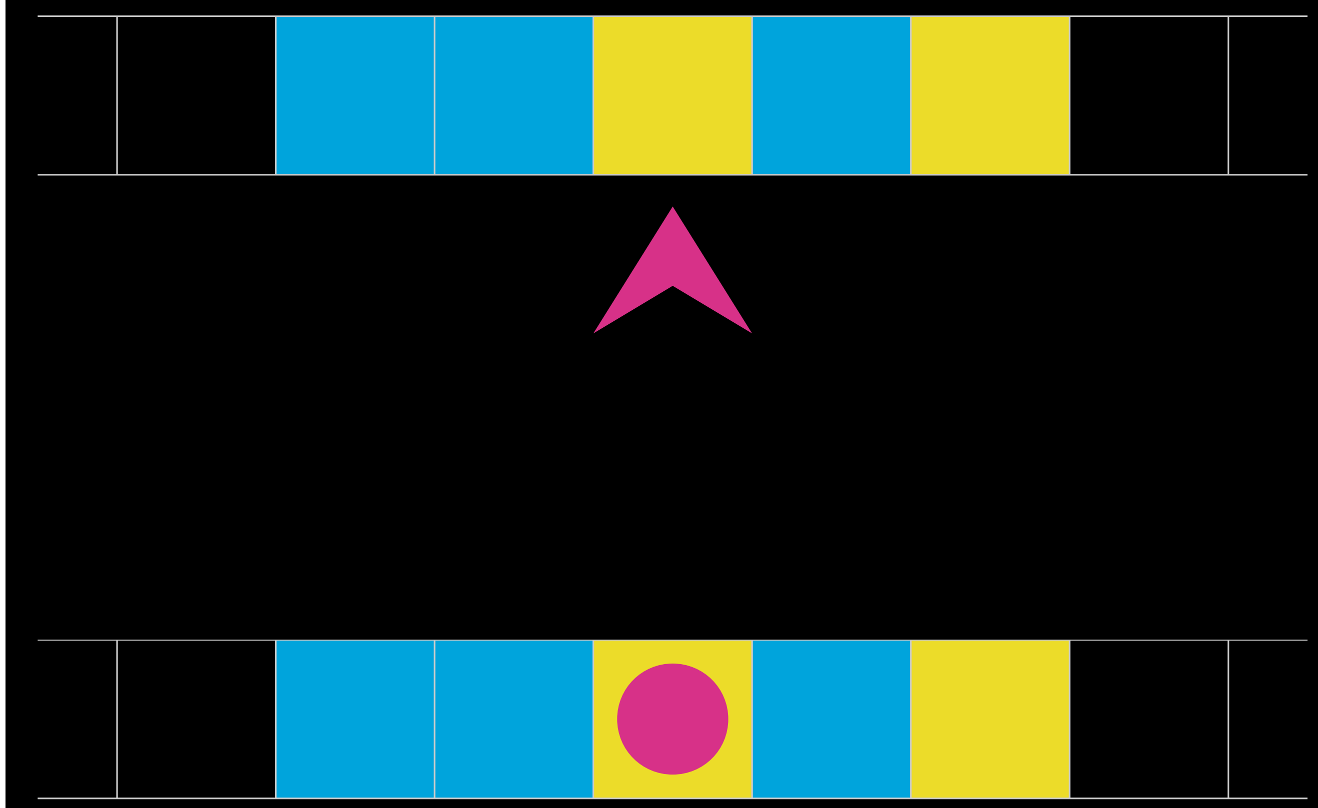
$m(n+1)$ states

à la Smith III

- The configuration encodes the tape.
- The cell pointed by the head also contains the state.

$$(\Sigma \cup S \times \Sigma, \{-1, 0, 1\}, f)$$

$m(n+1)$ states

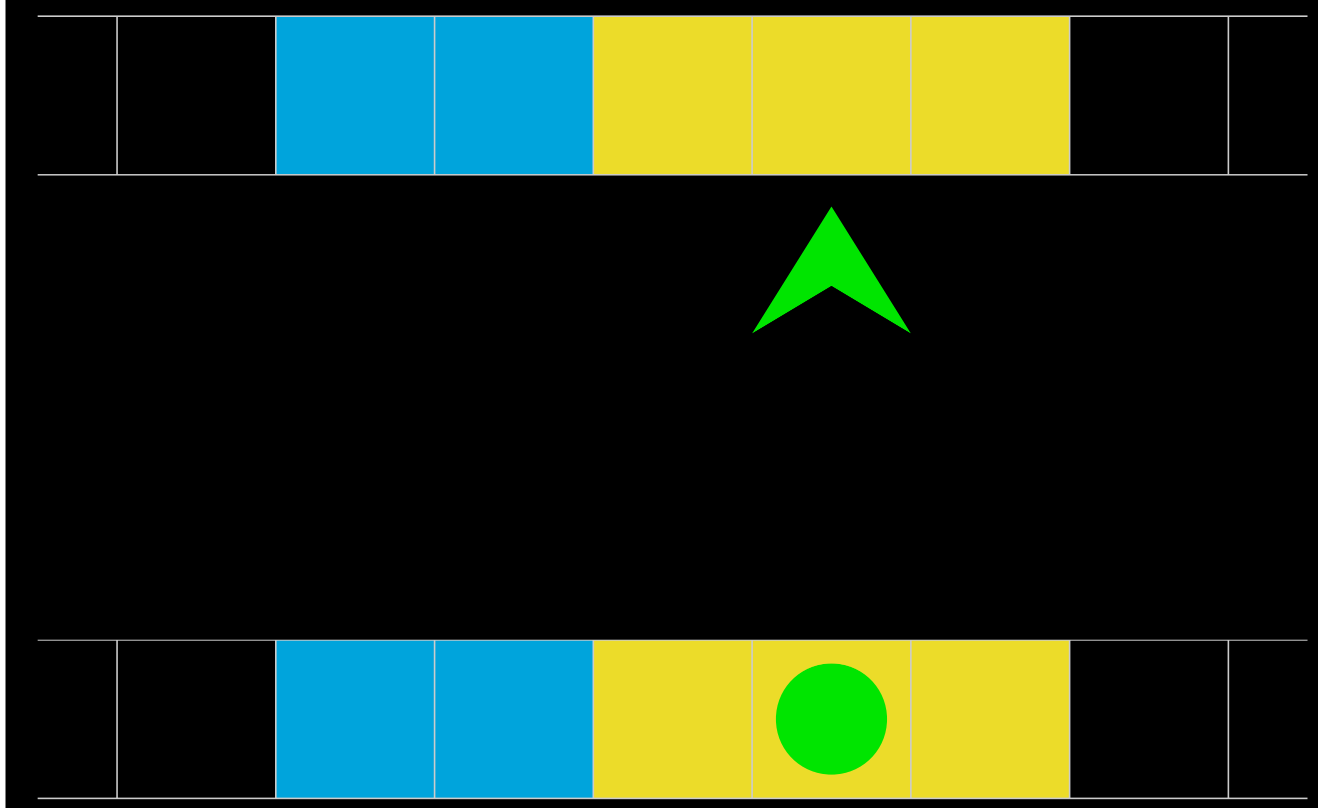


à la Smith III

- The configuration encodes the tape.
- The cell pointed by the head also contains the state.

$$(\Sigma \cup S \times \Sigma, \{-1, 0, 1\}, f)$$

$m(n+1)$ states

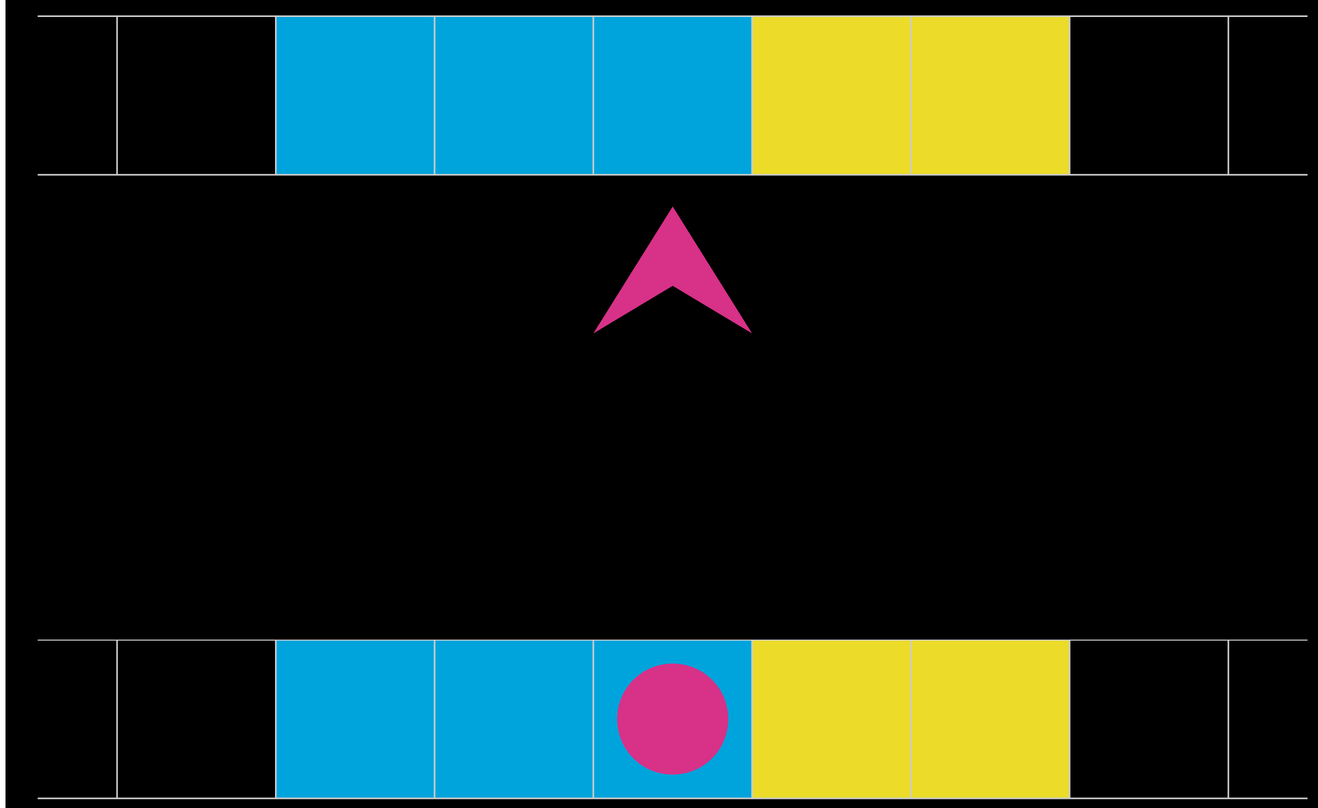


à la Smith III

- The configuration encodes the tape.
- The cell pointed by the head also contains the state.

$$(\Sigma \cup S \times \Sigma, \{-1, 0, 1\}, f)$$

$m(n+1)$ states



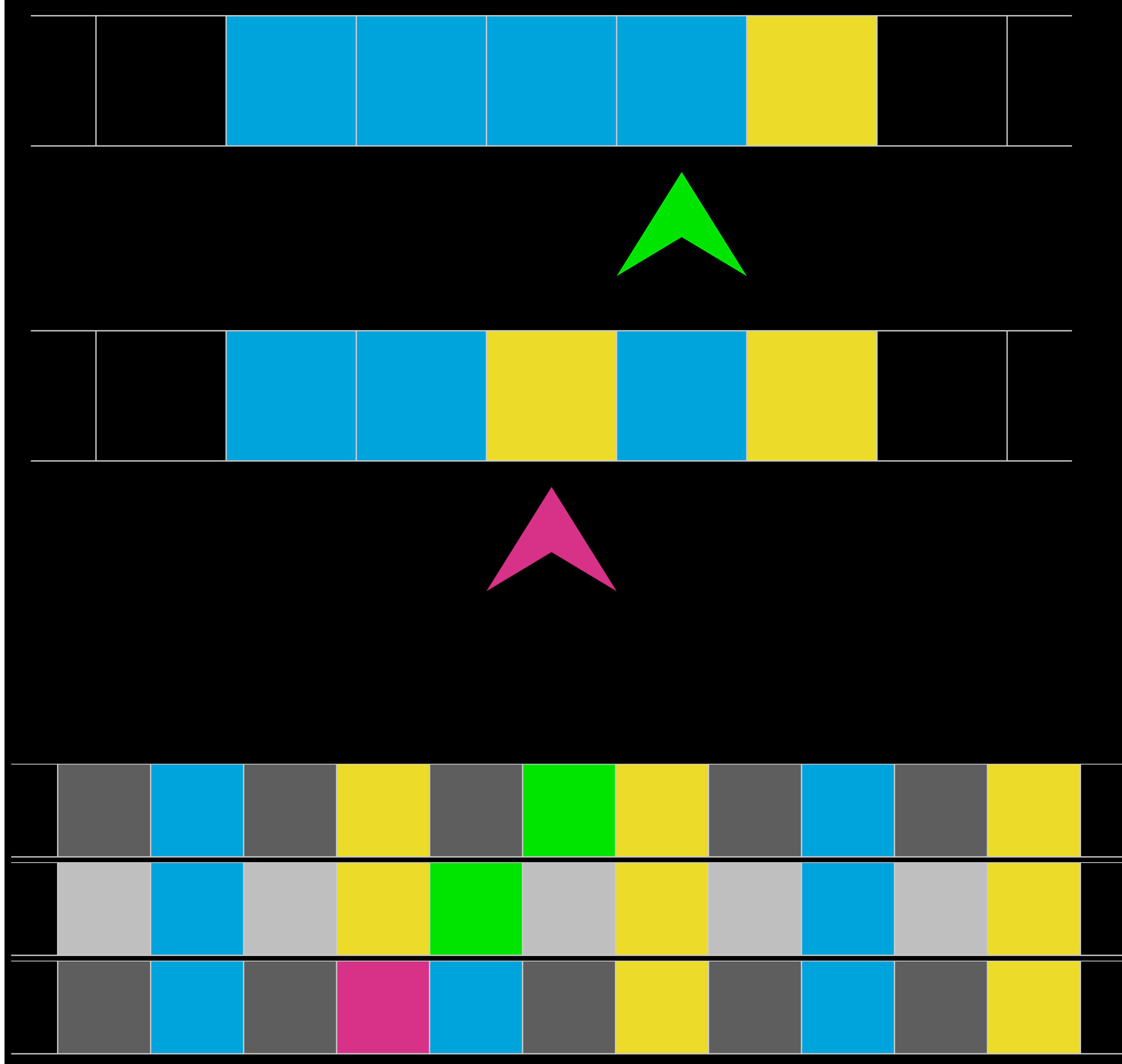
à la L&N

- Separate read/write from move.
- One step simulated in two steps.

$$(\Sigma \cup S \cup \{\bullet, \leftrightarrow\}, \{-1, 0, 1\}, f)$$

$m + n + 2$ states

[Lindgren & Nordhal 1990] plus encoding with signals.

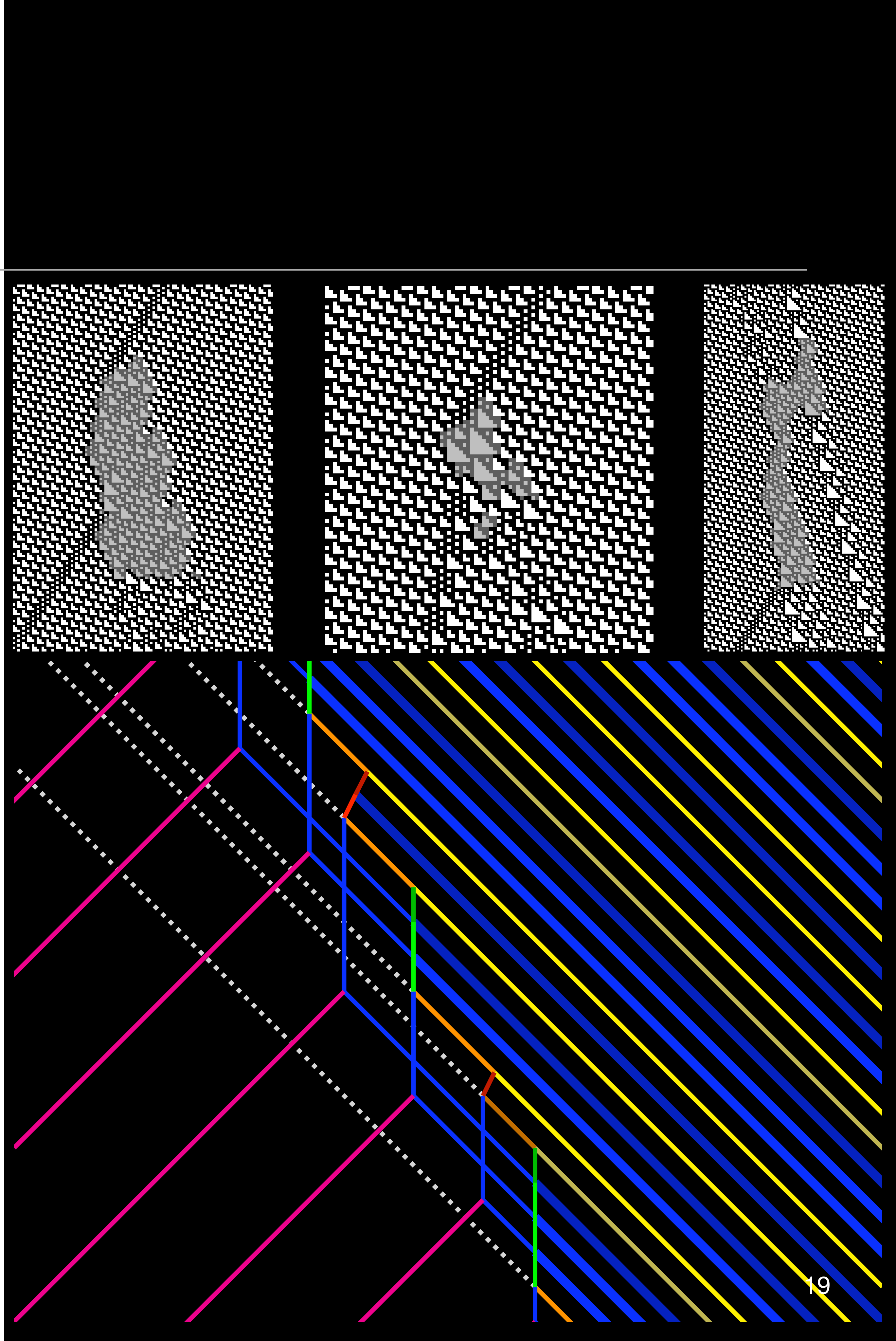


Cook 2004

[Cook 2004] Rule 110 is Turing-Universal

- Simulation of **Cyclic** Tag Systems

- See Gaétan's talk just after!



(iii) Intrinsic Universality

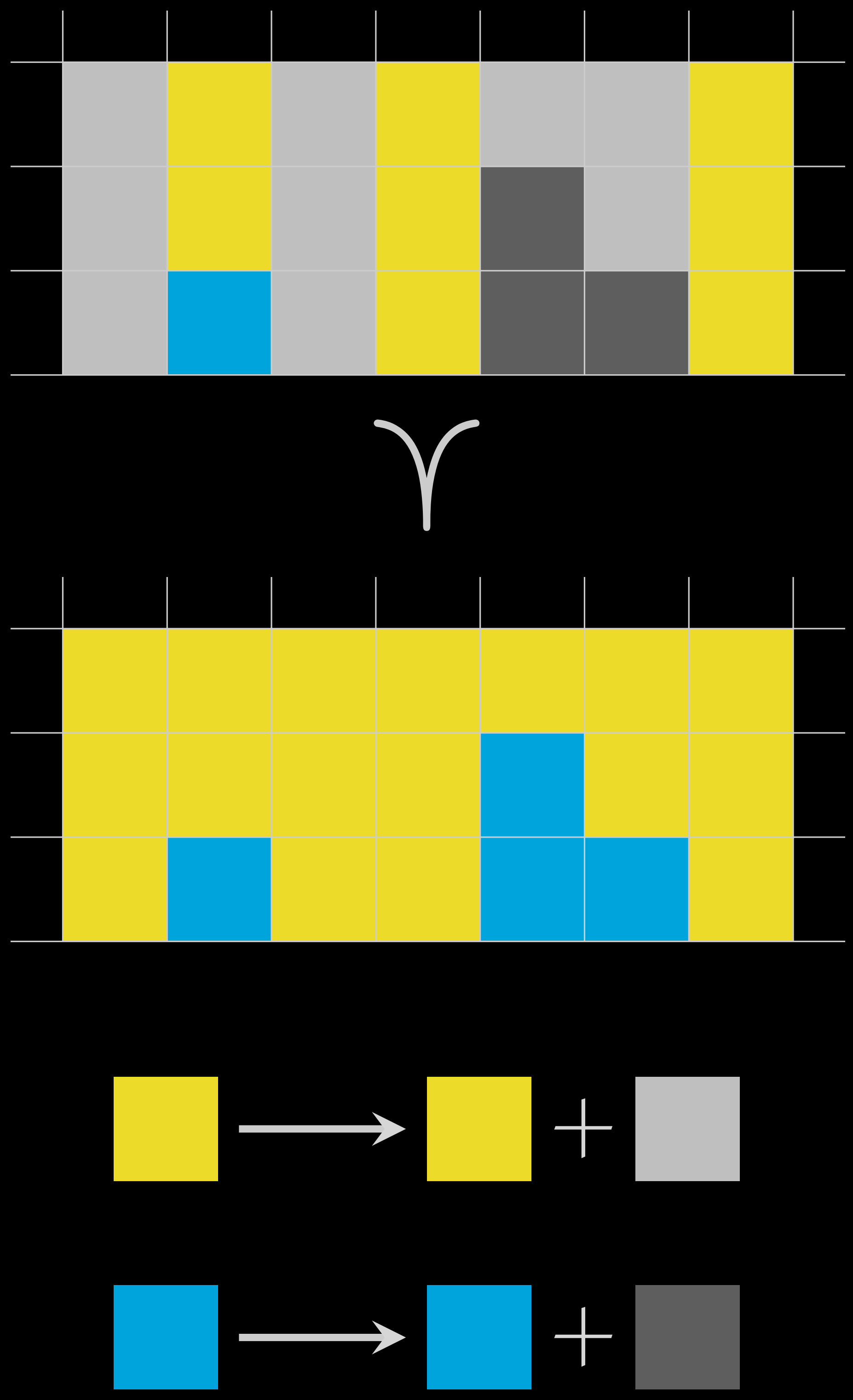
Intrinsic Simulation

- Universality can be seen as an intrinsic property of a model.
- An object is universal if it can **simulate** all other objects of the family.
- For simulation use something reasonable with respect to the initial model.
- For CA, reasonable simulation certainly means shift-invariance plus similar space-time diagrams.
- Nice formalization with Bulking (*aka* Grouping) **[Mazoyer & Rapaport 1999, NO 2001, Theysier 2005]**.

Direct Simulation

A cellular automaton $\mathcal{B}$ *directly simulates* a cellular automaton $\mathcal{A}$, denoted $G_{\mathcal{A}} \prec G_{\mathcal{B}}$, according to a mapping $\varphi : S_{\mathcal{A}} \to 2^{S_{\mathcal{B}}}$ if for any pair of states $a, b \in S_{\mathcal{A}}$, $\varphi(a) \cap \varphi(b) = \emptyset$ and for any configuration $c \in S_{\mathcal{A}}^{\mathbb{Z}}$,

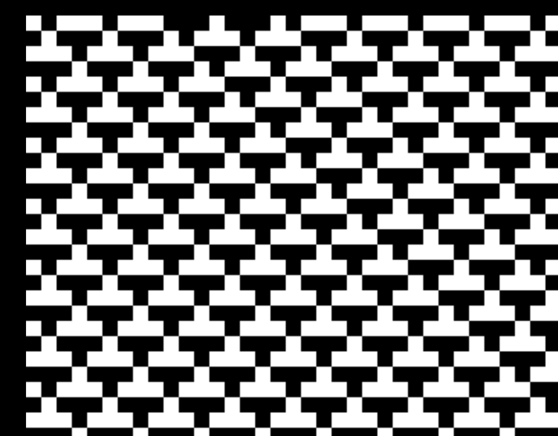
$$G_{\mathcal{B}}(\varphi(c)) \subseteq \varphi(G_{\mathcal{A}}(c)) \quad .$$

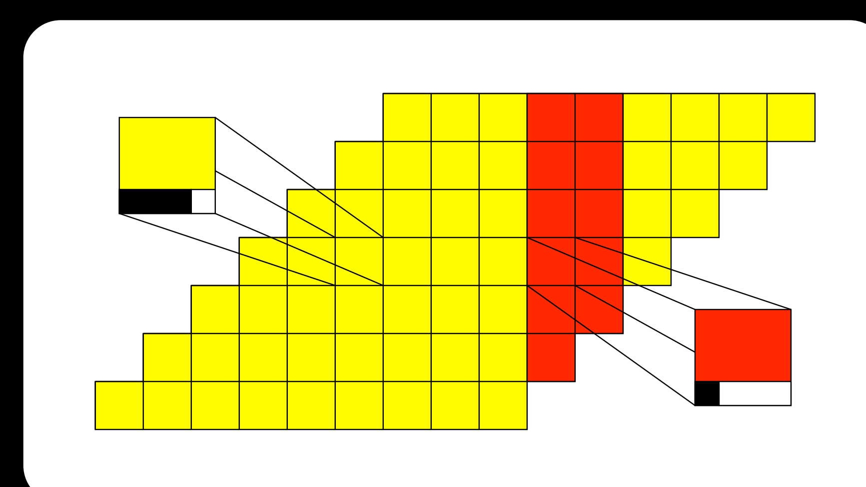


Geometric Transform

$\mathcal{U}$ is intrinsically universal if for each cellular automaton $\mathcal{A}$ there exists an unpacking map o_m, a positive integer $n \in \mathbb{N}$ and a translation vector $v \in \mathbb{Z}^d$ such that

$$G_{\mathcal{A}} \prec o_m^{-1} \circ G_{\mathcal{U}}^n \circ o_m \circ \sigma_v \quad .$$





Some Properties

Formal definition allow us to write proofs.

[folklore] Boolean Circuit Universality = IU for 2D+

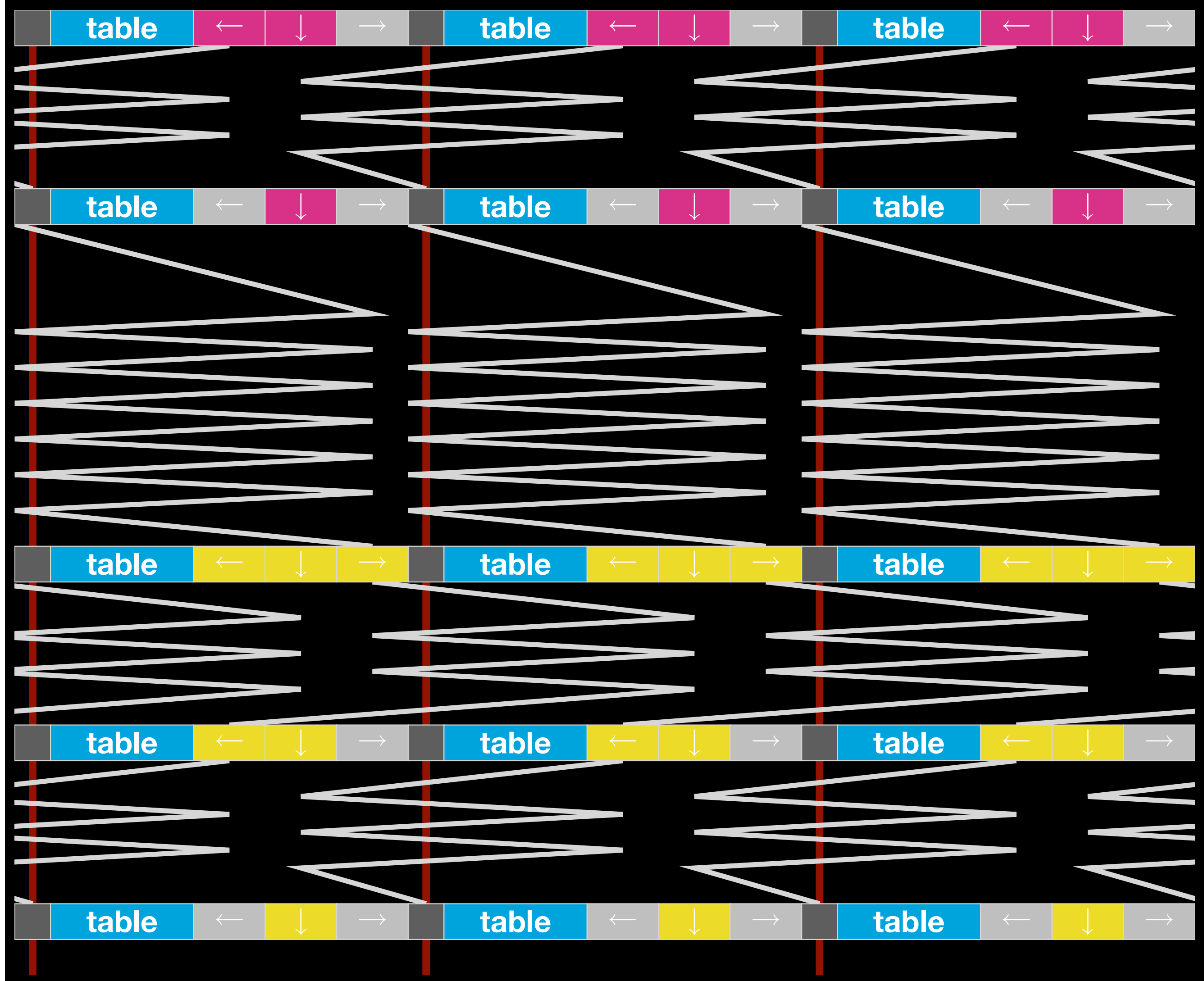
[Mazoyer & Rapaport 1999] No CA is IU in real-time.

[NO 2003] It is undecidable to know if a given CA is IU.

[NO 2002, Theyssier 2005] There exists TU CA that are not IU (infinitely far from IU).

Parallel TM style

- Comb-like infinite family of Turing heads, one per encoding meta-cell.
- All heads move the same, only the states and read/write differ.
- A meta-cell contains transition table + neighbors & self states.



Examples

Banks 1970. 18 states (converting 2D IU to 1D IU by slicing)

Albert & Čulik 1987. 14 states (totalistic OCA simulation)

NO 2002. 6 states (boolean circuits simulation)

Richard 2008. 4 states (totalistic OCA simulation using signals)

More details...

To learn more

- Read the survey in the proceedings
- Few pictures...
- ...but 86 bibliographic reference,
- Chronology,
- Tips and tricks.
- Everything to build small Universal CA by yourself.

That’s all folks!