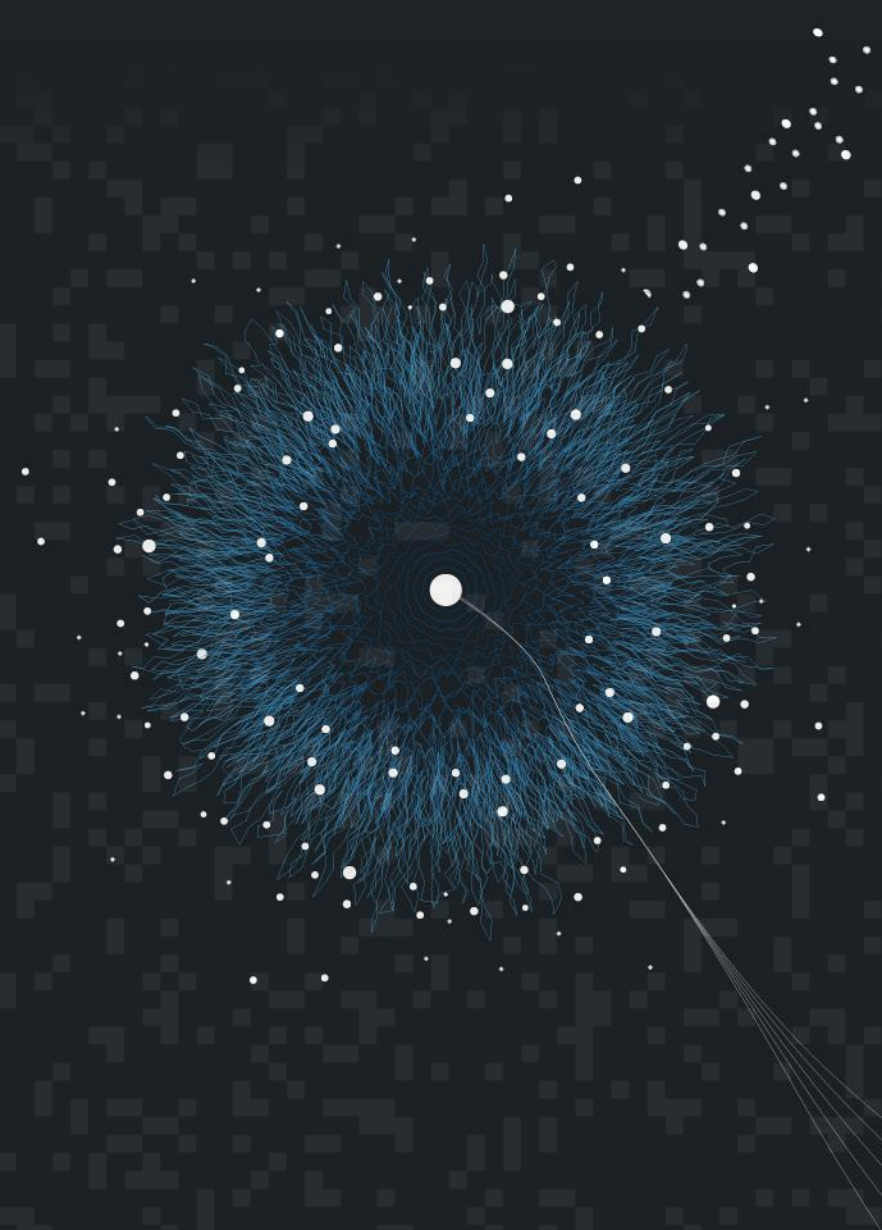


EEMCS - SCS

# CELLULAR AUTOMATA, FORMAL LANGUAGES AND MODEL CHECKING

FMT COLLOQUIA – MARCH 12, 2026

LUCA MARIOT



# ABOUT ME



**Dr. Luca Mariot**

Office: ZI 2035

E-Mail: [l.mariot@utwente.nl](mailto:l.mariot@utwente.nl)

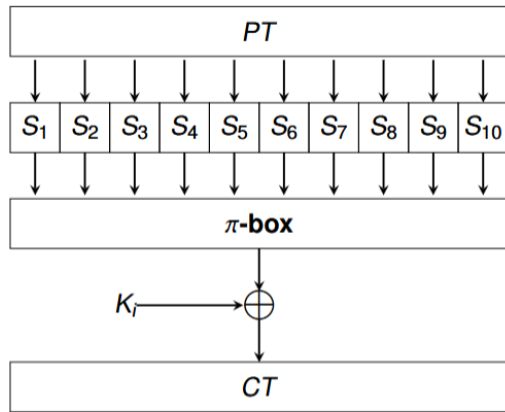
**Assistant Professor** in the Semantics, Cybersecurity and Services group (SCS)

**Research Interests:**

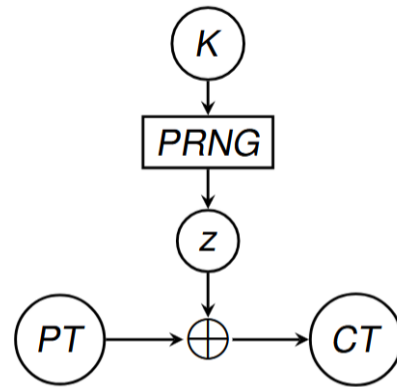
- Cryptography
- Cybersecurity
- Artificial Intelligence
- Cellular Automata
- Evolutionary Computing

**Website:** <https://lucamariot.org>

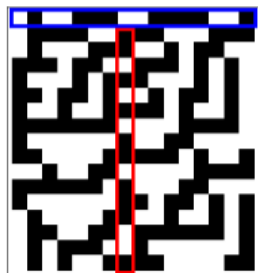
# AI-DRIVEN DESIGN OF CRYPTO PRIMITIVES



(b) Block cipher



(a) Stream cipher



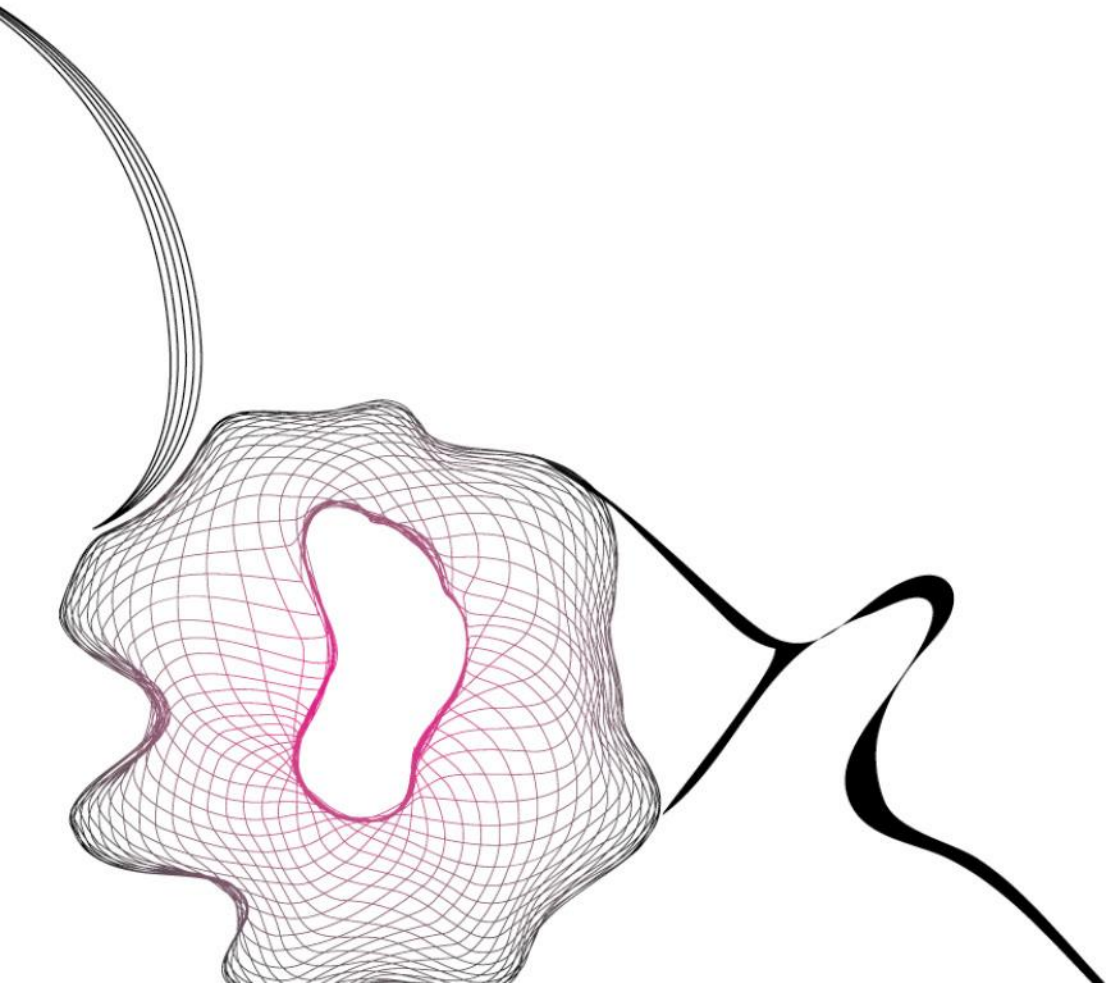
$$\Downarrow F : \{0, 1\}^n \rightarrow \{0, 1\}^m$$



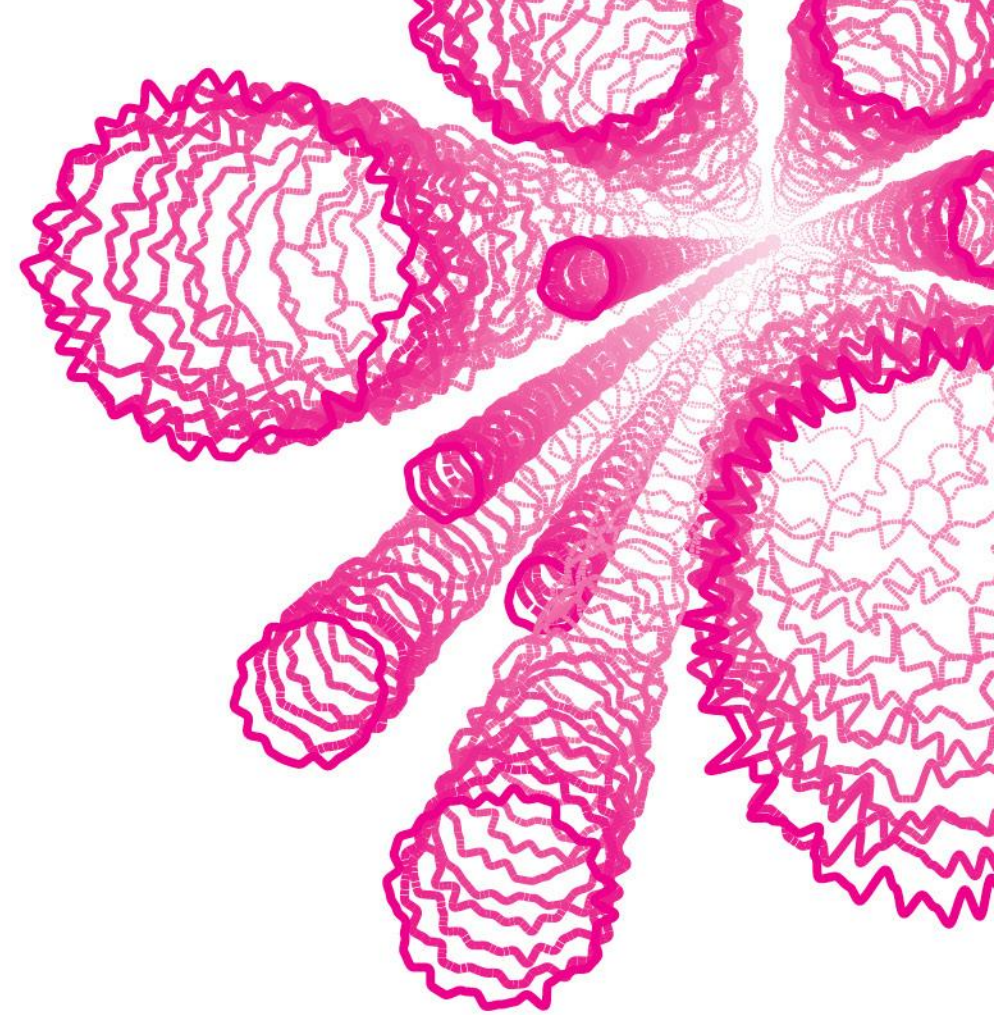
- **Symmetric Cryptography:** design of low-level primitives (PRNG, block ciphers, ...)
- **AI-driven approach** [14]: support the design phase with:
  - Metaheuristic optimization techniques (**Evolutionary Algorithms** [3], ...)
  - Nature-inspired computational models (**Cellular Automata** [12], ...)

# SUMMARY

- Intro to Cellular Automata
- Model checking of CA properties
- Algebraic Language Theory in CA-based Crypto Applications



# INTRODUCTION TO CELLULAR AUTOMATA



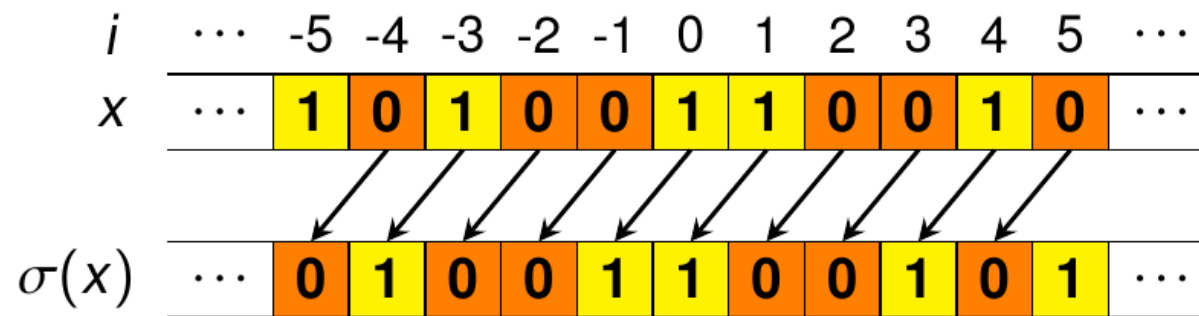
# WHAT ARE CELLULAR AUTOMATA (CA)?

- Pioneered by **John Von Neumann** [21] and **Stanislaw Ulam** [22] in the 1950s
- Initially conceived to study **self-replication phenomena**
- **In essence:** a grid of identical cells that evolve synchronously using the same local rule
- Most famous example: **Game of Life** by **John H. Conway** (1970s) [5]



# SHIFT-INVARIANT TRANSFORMATIONS

- Let  $\Sigma$  be a **finite alphabet** (e.g.  $\Sigma = \{0, 1\}$ ) and  $x \in \Sigma^{\mathbb{Z}}$  a **bi-infinite word** over  $\Sigma$
- **Shift operator**  $\sigma : \Sigma^{\mathbb{Z}} \rightarrow \Sigma^{\mathbb{Z}}$ : defined as  $\sigma(x)_i = x_{i+1}$ , for all  $x \in \{0, 1\}^{\mathbb{Z}}$ ,  $i \in \mathbb{Z}$



- A mapping  $F : \Sigma^{\mathbb{Z}} \rightarrow \Sigma^{\mathbb{Z}}$  is **shift-invariant** [10] if it commutes with  $\sigma$  i.e.:

$$F(\sigma(x)) = \sigma(F(x)), \text{ for all } x \in \Sigma^{\mathbb{Z}}$$

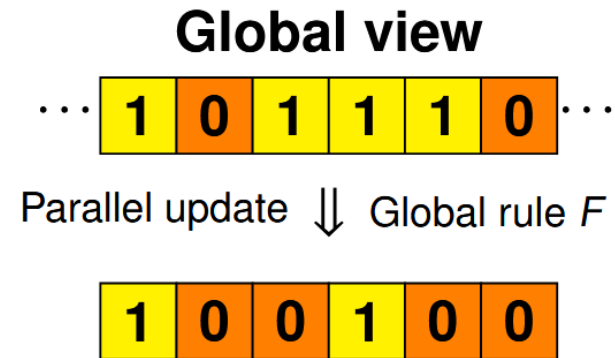
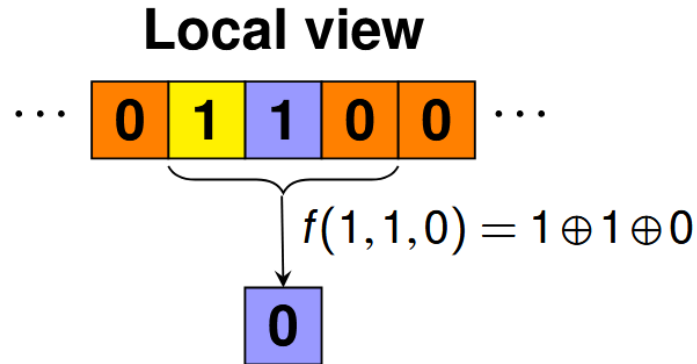
# INFINITE 1-D CA

- A map  $F : \Sigma^{\mathbb{Z}} \rightarrow \Sigma^{\mathbb{Z}}$  defined by a **local rule**  $f : \Sigma^d \rightarrow \Sigma$  of **diameter**  $d = m + a + 1$  such that, for all  $x \in \Sigma^{\mathbb{Z}}$  and  $i \in \mathbb{Z}$ ,

$$F(x)_i = f(x_{i-m}, \dots, x_i, \dots, x_{i+a})$$

- If  $m = a = r$ : symmetric neighborhood ( $r$  is the **radius**)

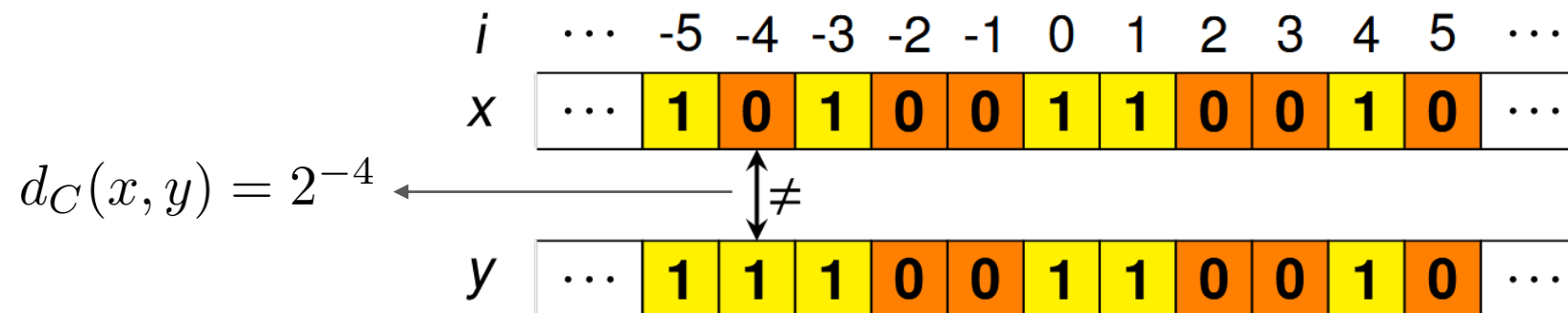
Example:  $d = 3$ ,  $r = 1$ ,  $f(x_{i-1}, x_i, x_{i+1}) = x_{i-1} \oplus x_i \oplus x_{i+1}$  (rule 150)



# INFINITE 1-D CA – CHARACTERIZATION

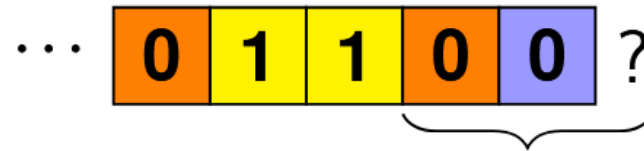
- **Curtis-Hedlund-Lyndon Theorem** [7]: a map  $F : \Sigma^{\mathbb{Z}} \rightarrow \Sigma^{\mathbb{Z}}$  is a CA if and only if it is shift-invariant and (uniformly) continuous
- **Continuity**: defined by equipping the full-shift space  $\Sigma^{\mathbb{Z}}$  with the **Cantor distance**:

$$\forall x, y \in \Sigma^{\mathbb{Z}}, d_C(x, y) = \begin{cases} 0 & , \text{ if } x = y \\ 2^{-i} & , \text{ if } x \neq y, i = \max\{j \in \mathbb{N} : x_{-j} \neq y_{-j} \vee x_j \neq y_j\} . \end{cases}$$

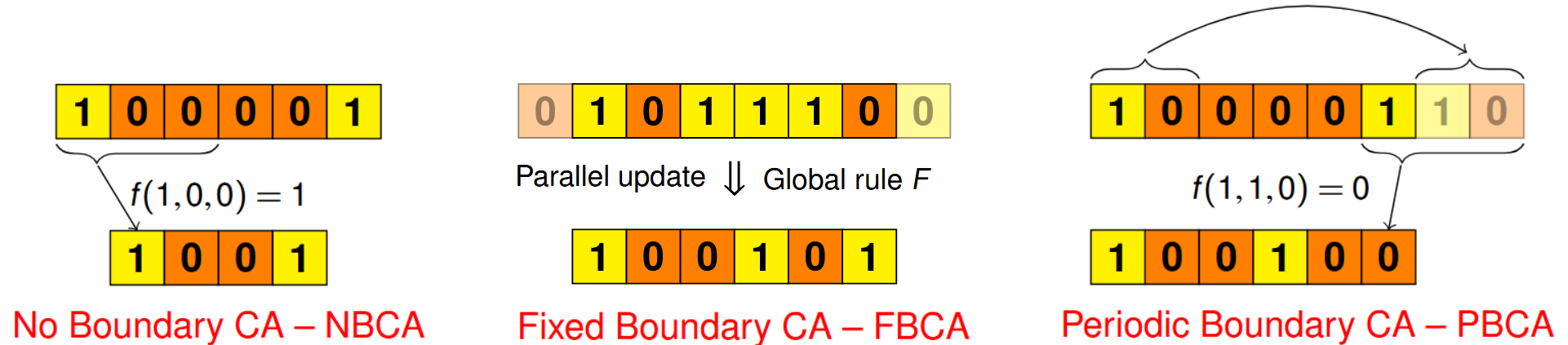


# FINITE CA

- The only CA we care about in actual implementations/simulations
- Same definition as in the infinite case, **but...**



- ... Here we have to deal with **boundary conditions** [12]:

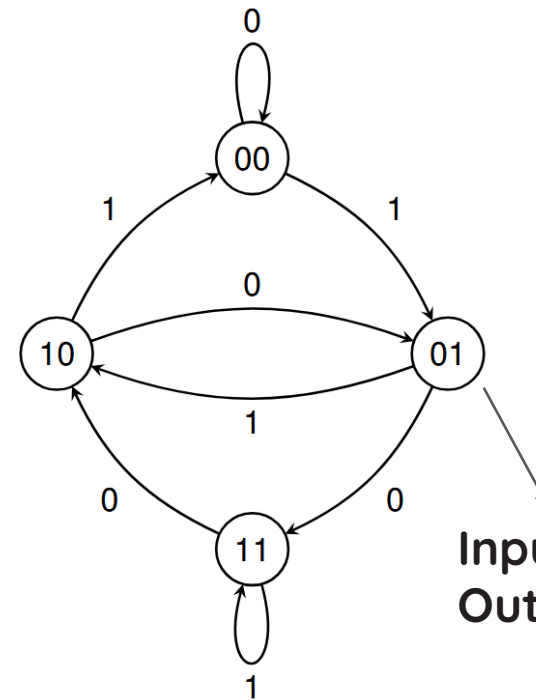


# CA REPRESENTATIONS

- **Truth table:** most straightforward representation
- **De Bruijn graph** [18]: useful to express global properties (injectivity, surjectivity, ...)

$x_1$	$x_2$	$x_3$	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

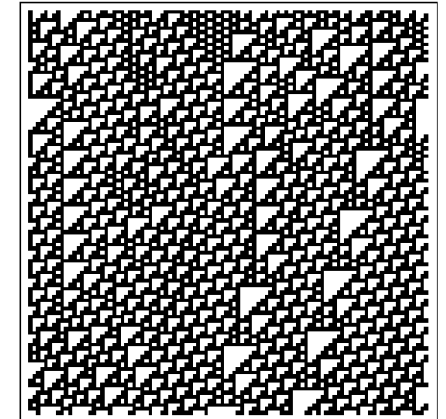
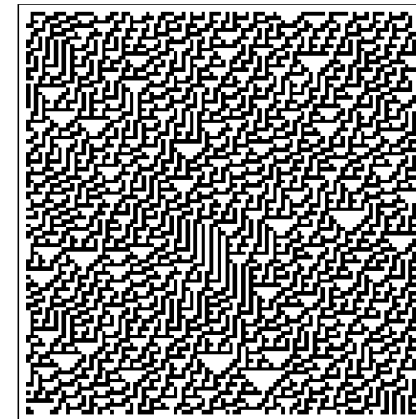
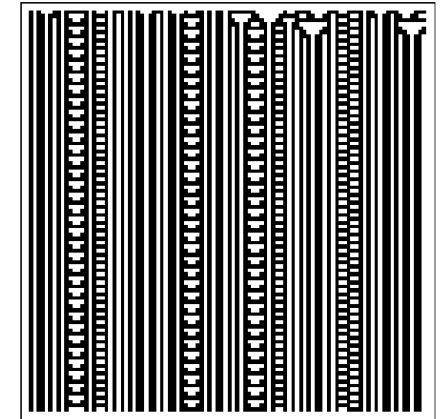
01101001<sub>10</sub> = 150  
(Wolfram code)



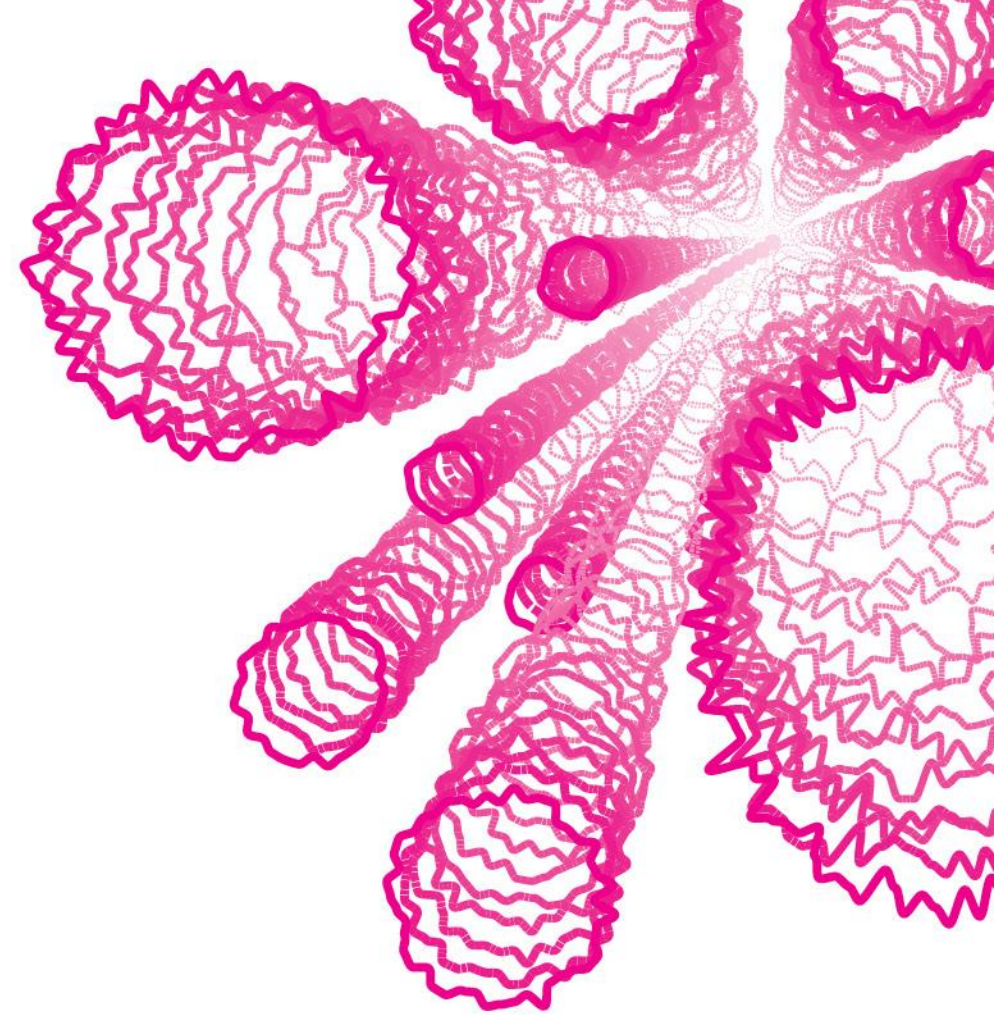
**Input:** path on the vertices  
**Output:** path on the edges

# WOLFRAM'S CLASSIFICATION OF CA

- **Wolfram** [23] gave an *empirical* classification for **elementary CA** ( $r = 1$ )
- **Class 1:** rapidly converge to a fixed point (eg: rule 168)
- **Class 2:** give rise to periodic structures (eg: rule 94)
- **Class 3:** exhibit chaotic behavior (eg: rule 30)
- **Class 4:** generate complex interacting patterns (eg: rule 110)



# MODEL CHECKING OF CA PROPERTIES



# CLASSIFICATION OF CA DYNAMICS

- Wolfram's classification is based on **visual patterns**
- Other approaches exist, for example based on:
  - **Topology** [10]
  - **Measure Theory** [8]
- **Goal:** find out which CA properties are **decidable**, and what is their **computational complexity**
- Most approaches focus on **long-term dynamics**

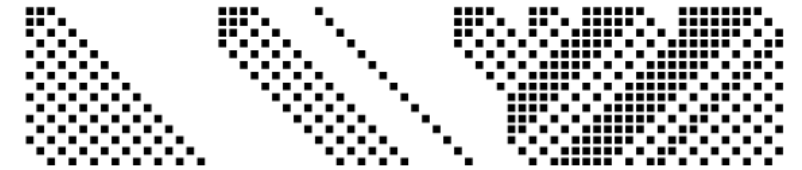


Figure 14: The traffic rule ECA184

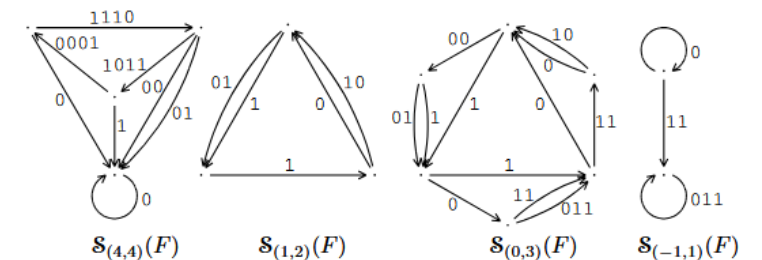
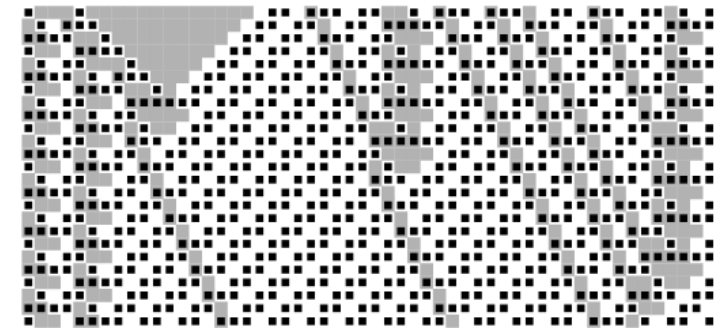


Figure 15: ECA 62 and its signal subshifts

Image credits: [10]

# MODEL-THEORETIC APPROACH

- **Idea:** focus on the *short-term* behavior of CA instead [20]
- **First-Order (FO) structure** of a 1-D CA defined by a local rule  $f : \Sigma^d \rightarrow \Sigma$ :

$$\mathcal{C}_f = \langle C, \rightarrow \rangle$$

where:

- $C$  is the space of all CA configurations (e.g.  $C = \Sigma^{\mathbb{Z}}$  for infinite CA)
- $\rightarrow$  is the binary predicate for the application of the global rule  $F$
- **Goal:** given a FO sentence  $\varphi$ , we want to determine if  $\mathcal{C}_f \models \varphi$

# FIRST-ORDER THEORY OF FINITE CA

- Example of property definable in  $\mathcal{C}_f$ : **injectivity**  $\varphi \equiv \forall x, y, z (x \rightarrow z \wedge y \rightarrow z \Rightarrow x = y)$
- How to check it: construct an automaton  $\mathcal{A}_f$  that given  $X, Y \in \Sigma^n$  tests if  $X \rightarrow Y$

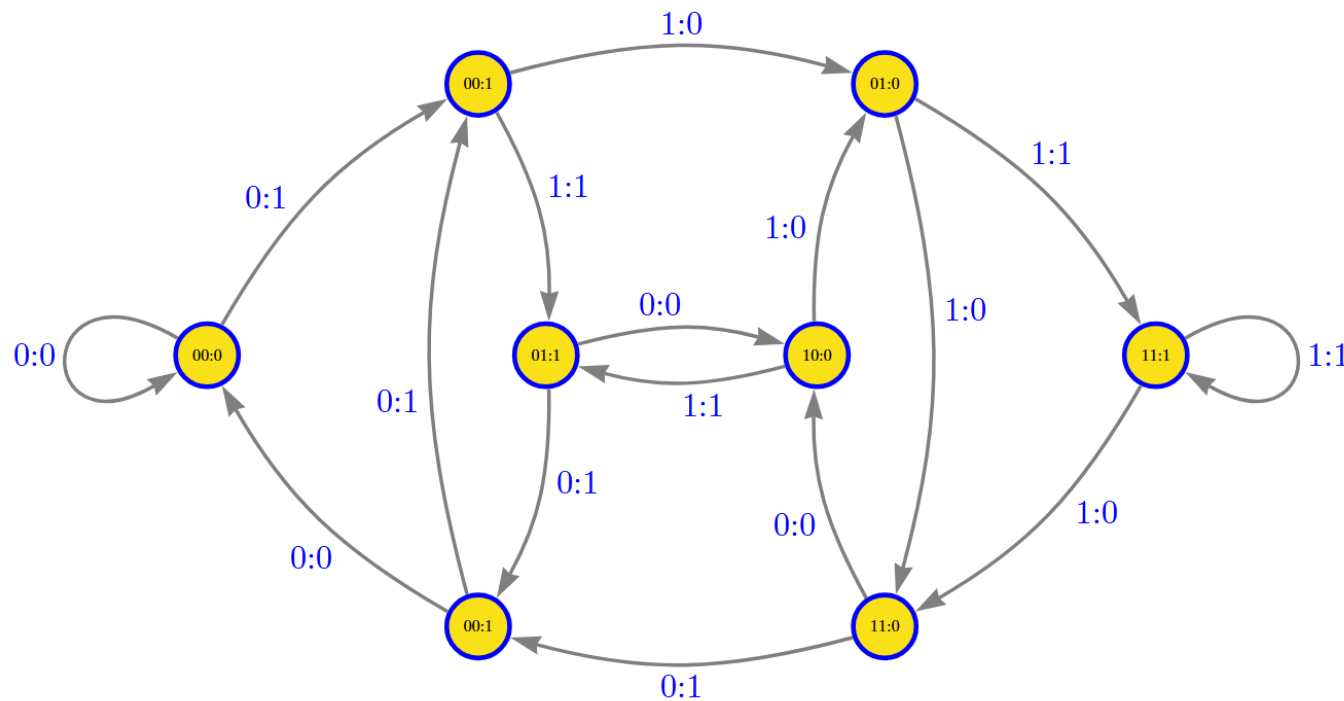


Image credits: [20]

- State set:  $\Sigma^{2r} \times \Sigma^r$
- Transitions:
 
$$\begin{array}{c} a_1, \dots, a_{2r} \\ b_1, \dots, b_r \end{array} \xrightarrow{a:b} \begin{array}{c} a_2, \dots, a_{2r}, a \\ b_2, \dots, b_r, b \end{array}$$

# DECIDABILITY RESULTS

- **Theorem:** FO logic for both *finite* and *infinite* 1-D CA is decidable [19]
- Infinite case: resort to **Büchi automata** (slightly modified for bi-infinite words)

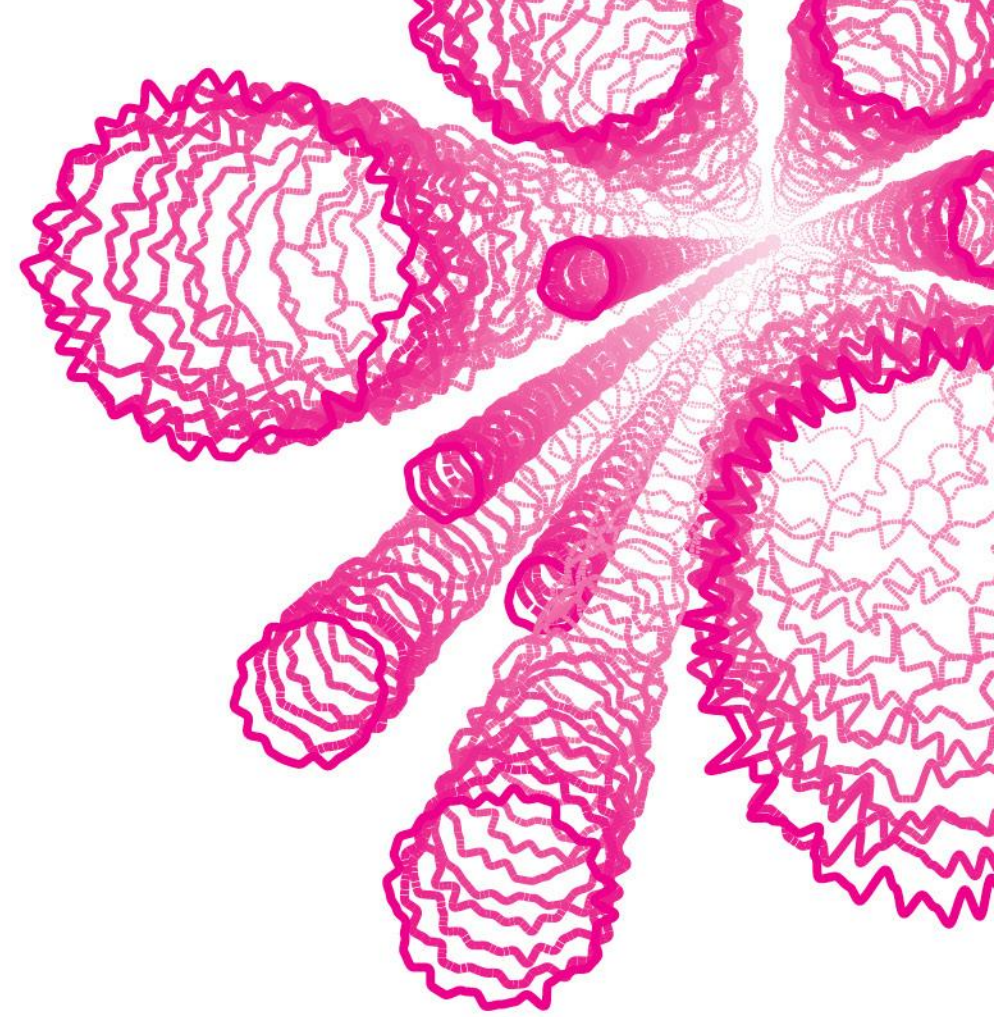
## What we *can* check:

- Surjectivity
- Reversibility
- “There are *exactly* 2 fixed points”

## What we *can't* check:

- *Nilpotency*: all configurations converge to a quiescent state
- Sentences related to orbits of the CA
- Need to include additional predicates

# ALGEBRAIC LANGUAGE THEORY FOR CA-BASED CRYPTO APPLICATIONS



# CRYPTO APPLICATION: SECRET SHARING

- Suppose a *dealer* holds a secret value  $s \in \{0, 1\}^\ell$  and wants to share it with  $n$  parties  $P_1, P_2, \dots, P_n$
- *Secret sharing*: the dealer gives *shares* of  $s$  to each participant
- At a later time, the participants need to pool their shares to *reconstruct* the secret
- Classic scenario: nuclear command & control

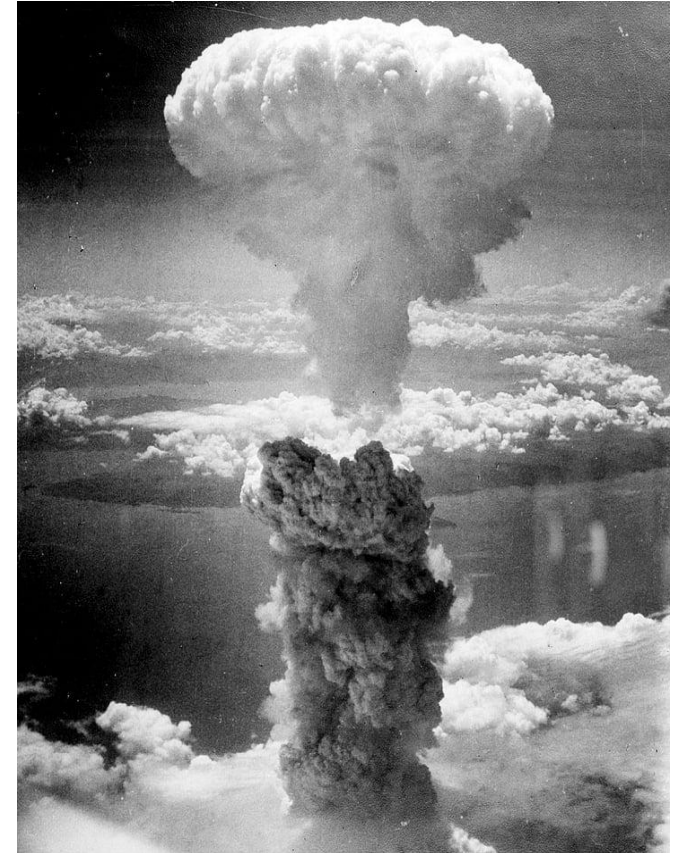
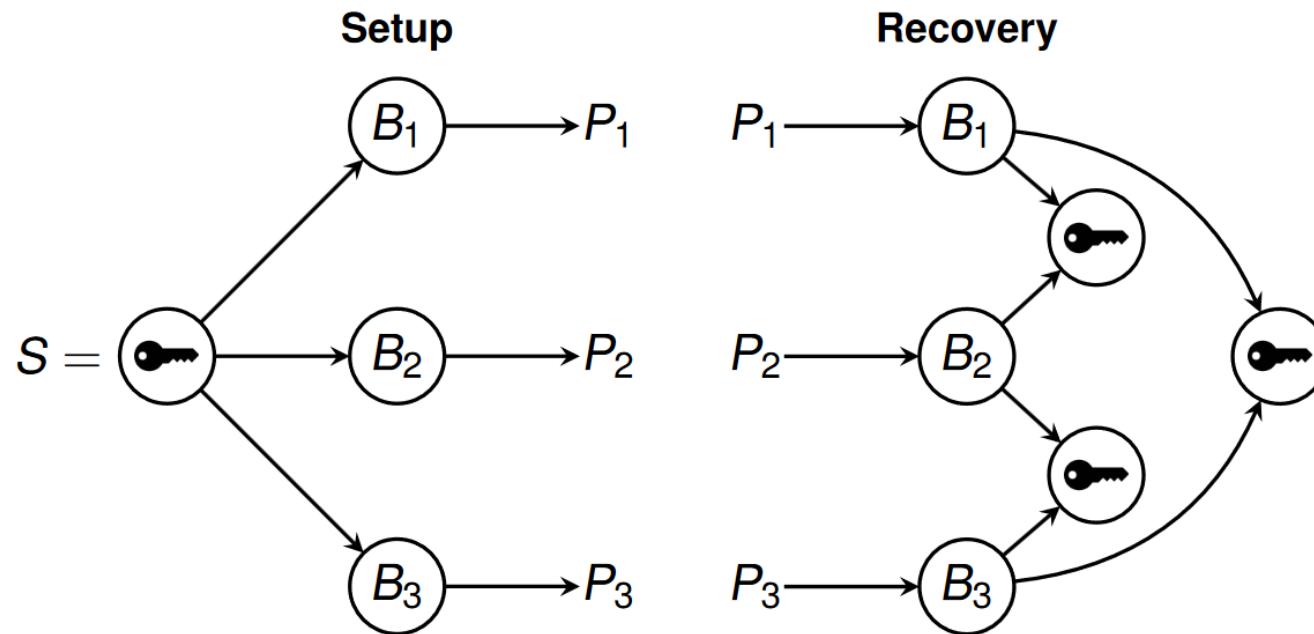


Image credits: Wikipedia

# THRESHOLD SECRET SHARING

- $(t, n)$ -threshold secret sharing scheme: at least  $t$  out of  $n$  players need to combine their shares to recover the secret

Example:  $(2, 3)$ -scheme



- **Example:** Shamir's secret sharing [15]

# LATIN SQUARES AND SECRET SHARING

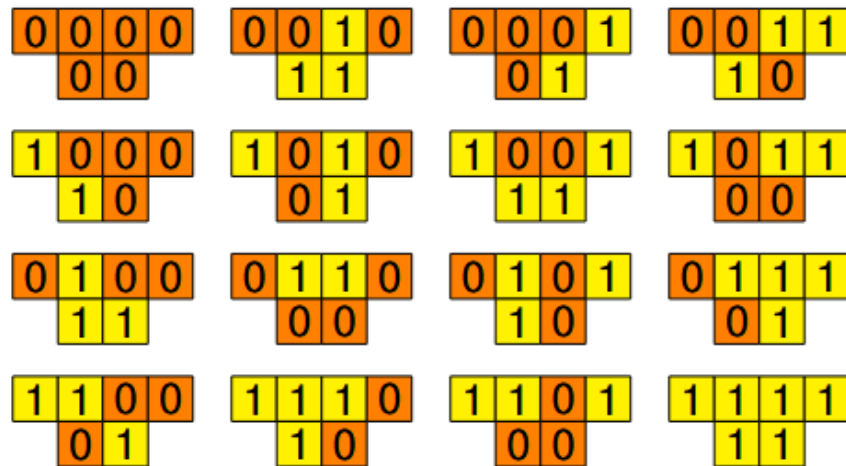
- **Latin square:**  $n \times n$  matrix where all rows and columns are permutations of  $[n] = \{1, \dots, n\}$
- **Orthogonal Latin Squares:** their *superposition* yields all pairs  $(x, y) \in [n] \times [n]$

1	3	4	2	1	4	2	3	1	3	4	2
4	2	1	3	3	2	4	1	4	2	1	3
2	4	3	1	4	1	3	2	2	4	3	1
3	1	2	4	2	3	1	4	3	1	2	4

- **n Mutually Orthogonal Latin Squares (n-MOLS)** are equivalent to  $(2, n)$  threshold secret sharing schemes [17]

# LATIN SQUARES AND CA

- A **No-Boundary CA** (NBCA) with *bipermutive* local rule  $f : \Sigma^d \rightarrow \Sigma$  can be used to generate a  $N \times N$  Latin square, where  $N = |\Sigma|^{2(d-1)}$
- **Example:**  $F : \{0, 1\}^4 \rightarrow \{0, 1\}^2$  with  $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$  (rule 150)



(a) Rule 150 on 4 bits

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

(b) Latin square  $L_{150}$

**Encoding:**

00  $\mapsto$  1  
 10  $\mapsto$  2  
 01  $\mapsto$  3  
 11  $\mapsto$  4

# ORTHOGONAL LATIN SQUARES AND CA

- Linear bipermutive rule:  $f(x) = x_1 \oplus a_2 x_2 \oplus \dots \oplus a_{d-1} x_{d-1} \oplus x_d$
- Associated polynomial:  $P_f(X) = 1 + a_2 X + \dots + a_{d-1} X^{d-2} + X^{d-1}$
- **Theorem:** two linear bipermutive CA generates orthogonal Latin squares iff their polynomials are *coprime* [11, 13]

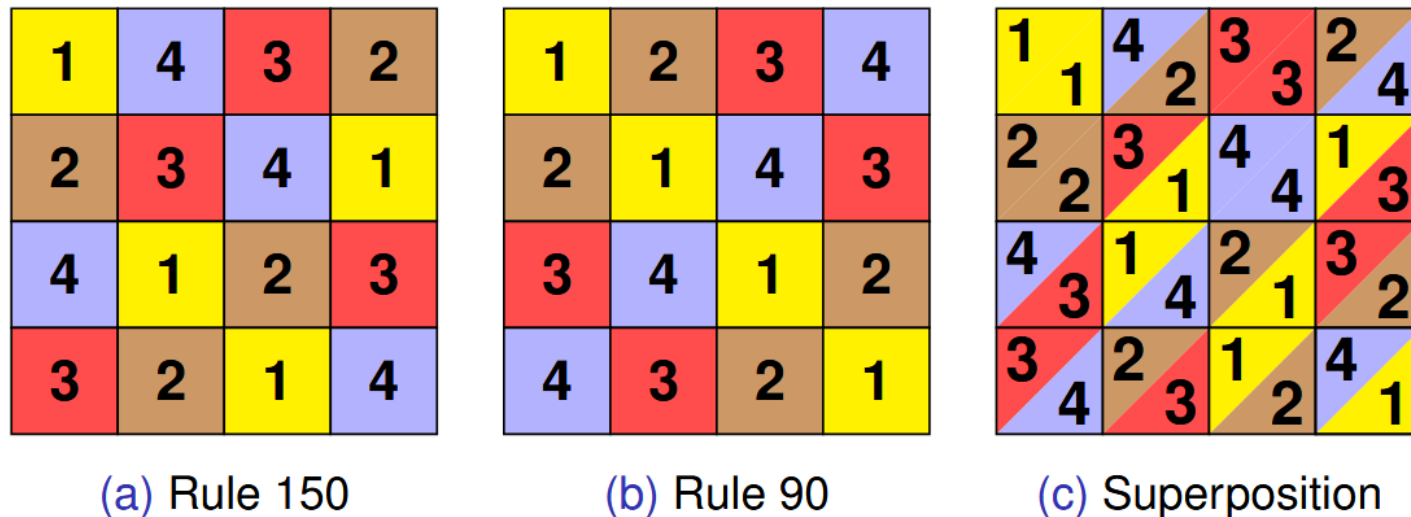


Figure:  $P_{150}(X) = 1 + X + X^2$ ,  $P_{90}(X) = 1 + X^2$  (coprime)

# COUNTING POLYNOMIALS

- **Question 1:** How many pairs of binary coprime polynomials of degree  $n$  with *nonzero constant term* are there?

$$f(x) = 1 + a_1x + \cdots + a_{n-1}x^{n-1} + x^n ,$$

$$g(x) = 1 + b_1x + \cdots + b_{n-1}x^{n-1} + x^n .$$

- **Question 2:** How to *enumerate* all such pairs, for a fixed degree  $n$ ?
- **Strategy:** use Euclid's algorithm "in reverse" (dilcuE's algorithm [1])

## Euclid's algorithm

$$\begin{aligned} (x^4 + x^2, x^4 + x^3 + 1) &\xrightarrow{1} (x^4 + x^3 + 1, x^3 + x^2 + 1) \xrightarrow{x} \\ (x^3 + x^2 + 1, x + 1) &\xrightarrow{x^2} (x + 1, 1) \xrightarrow{x+1} (1, 0) \end{aligned}$$

## dilcuE's algorithm

$$\begin{aligned} (0, 1) &\xrightarrow{x+1} (1, x + 1) \xrightarrow{x^2} (x + 1, x^3 + x^2 + 1) \xrightarrow{x} \\ (x^3 + x^2 + 1, x^4 + x^3 + 1) &\xrightarrow{1} (x^4 + x^3 + 1, x^4 + x^2) = (f, g) \end{aligned}$$

# PROBLEM STRUCTURE

- **Idea:** characterize the *sequences* of quotients in dilcuE's algorithm that gives coprime pairs with nonzero constant terms when starting from (1,0)

$$\begin{array}{r}
 \begin{array}{l}
 \text{degrees} \\
 \underbrace{\hspace{2cm}} \\
 q_1 \rightarrow x^{d_1} + \underbrace{q_{1,d_1-1}x^{d_1-1} + \dots + q_{1,1}x}_{\text{middle terms}} + \underbrace{s_1}_{\text{constant terms}} \\
 q_2 \rightarrow x^{d_2} + q_{2,d_2-1}x^{d_2-1} + \dots + q_{2,1}x + s_2 \\
 \vdots \rightarrow \vdots + \vdots + \dots + \vdots + \vdots \\
 q_k \rightarrow x^{d_k} + q_{k,d_k-1}x^{d_k-1} + \dots + q_{k,1}x + s_k
 \end{array}
 \end{array}
 \begin{array}{l}
 \boxed{\begin{array}{c} s_1 \\ s_2 \\ \vdots \\ s_k \end{array}} \xrightarrow{\text{Let's focus only on these...}}
 \end{array}$$

- **Notation:**  $r_i, r_{i+1} \rightarrow$  consecutive remainders in Euclid's at step i. Step i+1:

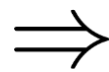
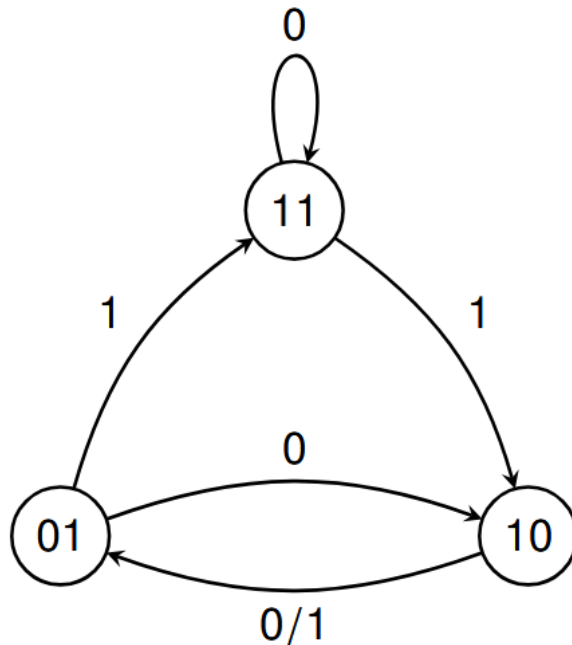
$$r_i(x) = q_{i+1}(x)r_{i+1}(x) + r_{i+2}(x)$$

# FINITE STATE AUTOMATON OF REMAINDERS

- $(c_i, c_{i+1})$ : constant terms of  $r_i, r_{i+1}$
- $X_{i+1}$ : constant term of  $q_{i+1}$
- $\delta((c_i, c_{i+1}), X_{i+1})$ : next pair  $(c_{i+1}, c_{i+2})$

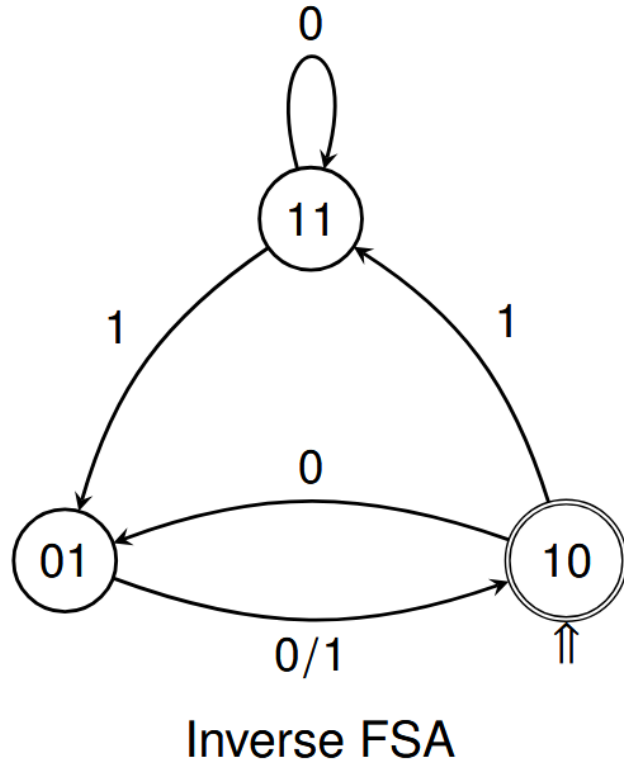
$(c_i, c_{i+1})$	$X_{i+1}$	$\delta((c_i, c_{i+1}), X_{i+1})$
(1, 1)	0	(1, 1)
(1, 1)	1	(1, 0)
(1, 0)	0	(0, 1)
(1, 0)	1	(0, 1)
(0, 1)	0	(1, 0)
(0, 1)	1	(1, 1)

**Remark:** the pair (0, 0) *never* occurs



The sequences of constant terms form a **regular language** [4]

# THE LANGUAGE OF CONSTANT TERMS SEQUENCES



- The FSA is the **de Bruijn** graph over the set  $\{11, 10, 01\}$
- FSA is **permutative**: for dilcuE's, simply reverse the arrows
- **Initial State**: 10
- **Final state**: 11 (but we can use 10)

**Regular Expression of the Language:**

$$L = (0(0 + 1) + (10^*1(0 + 1)))^*$$

# COUNTING CONSTANT TERMS SEQUENCES

- **Enumeration:** Visit the FSA with DFS up to depth  $n$
- **Counting:** use the *Chomsky-Schutzemberger enumeration theorem* [2]
- Transform  $L = (0(0 + 1) + (10^*1(0 + 1)))^*$  as follows:
  - ▶  $0, 1 \Rightarrow X$
  - ▶  $+, \cdot \Rightarrow +, \cdot$
  - ▶  $* \Rightarrow \frac{1}{1-X}$

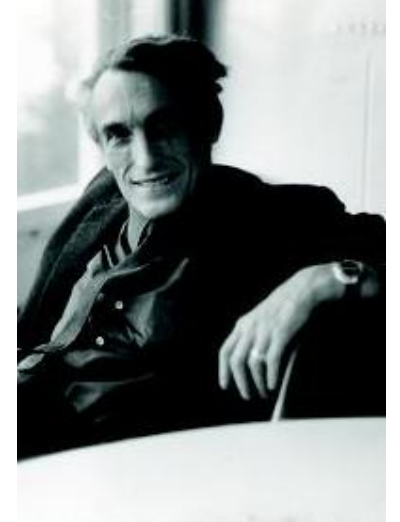
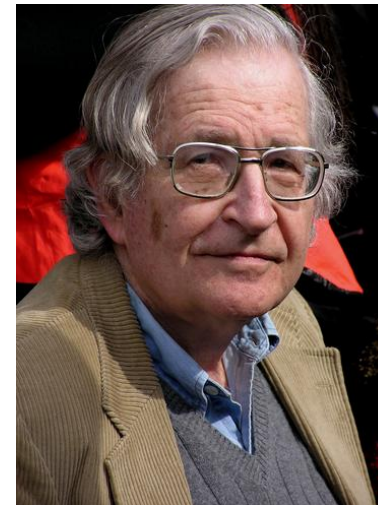


Image credits: Wikipedia

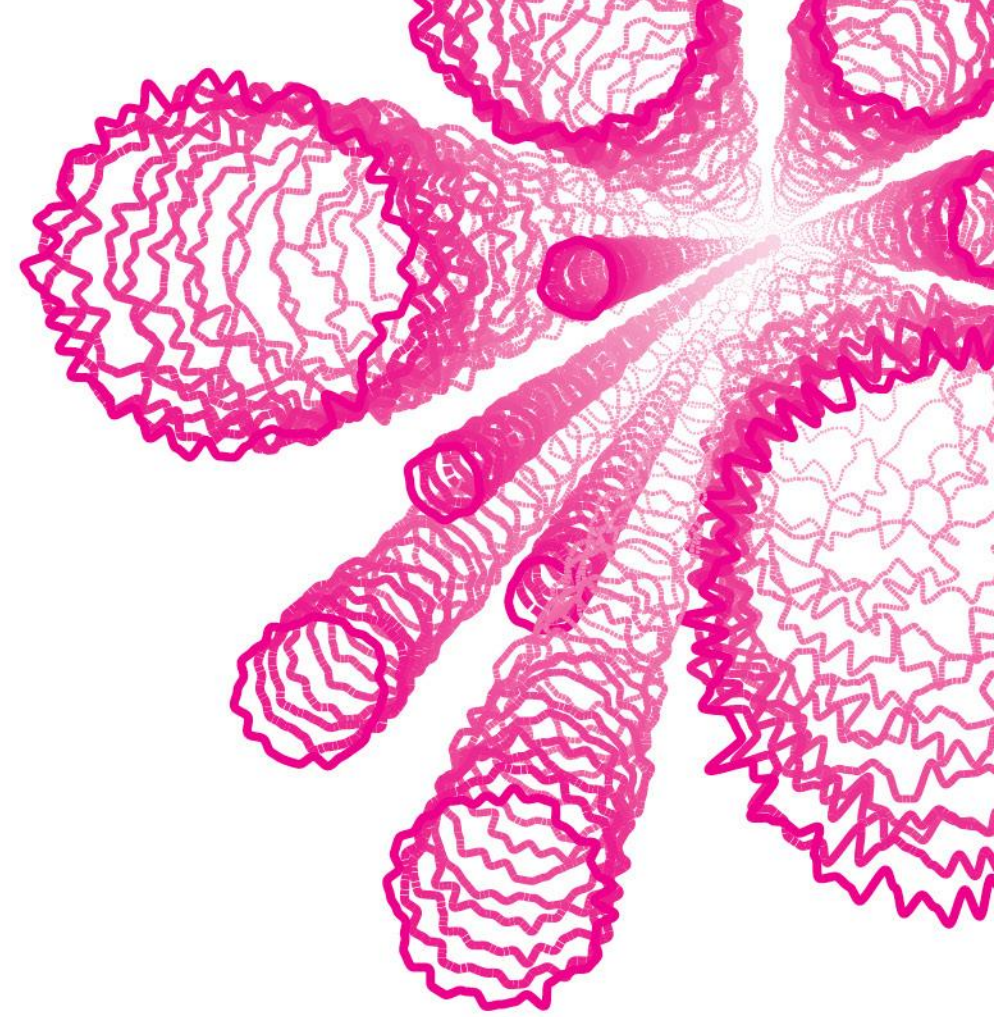
**Generating Function:**

$$\sum_{n=0}^{\infty} a_n \cdot X^n = \frac{1-X}{1-X-2X^2},$$

**Closed Form:**

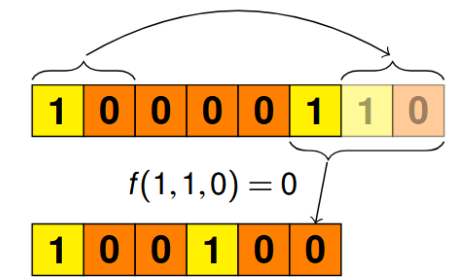
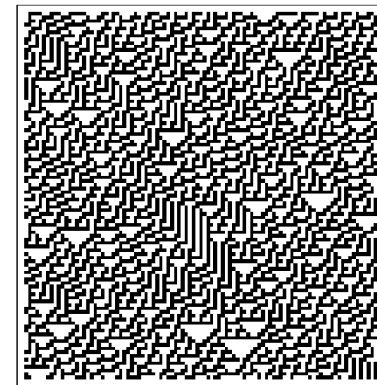
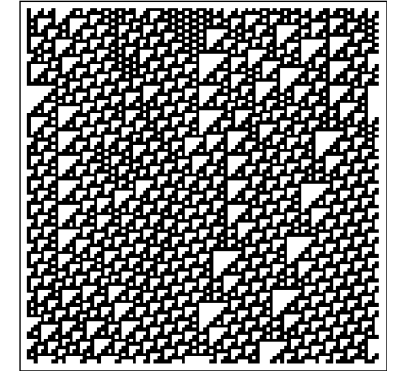
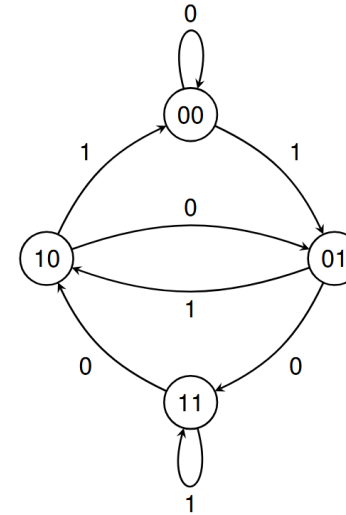
$$a_n = \frac{2^n + 2 \cdot (-1)^n}{3}$$

# CONCLUSIONS



# TALK SUMMARY

- CA have been investigated extensively wrt to their long-term dynamical properties
- Model-checking gives an interesting approach to classify short-term properties in terms of their decidability
- Formal languages arise unexpectedly in combinatorial applications of CA

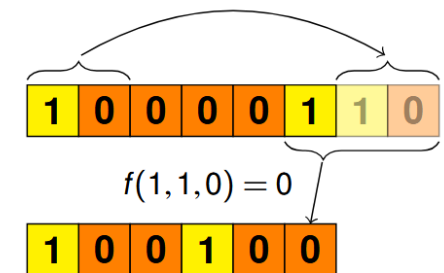
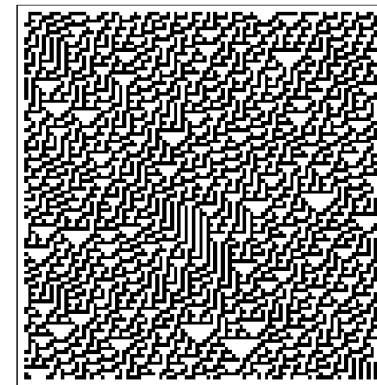
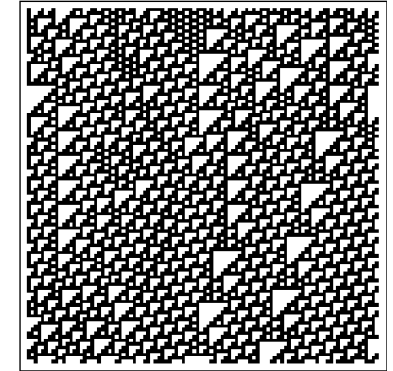
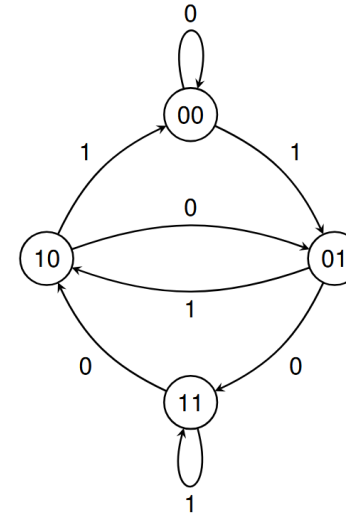


Periodic Boundary CA – PBCA

# OTHER DIRECTIONS/TOPICS

There's much more!

- Classification of 2-D and n-D CA (most stuff is undecidable!) [9]
- Using CA to recognize formal languages “in real time” [16]
- Generalization to **Automata Networks** [6]



Periodic Boundary CA – PBCA

**THANKS!  
QUESTIONS?**

# REFERENCES

1. A. T. Benjamin, C.D. Bennett: The probability of relatively prime polynomials. *Mathematics Magazine* 80(3): 196-202 (2007)
2. N. Chomsky, M.P. Schützenberger: The algebraic theory of context-free languages. In *Studies in Logic and the Foundations of Mathematics*, vol. 26, pp. 118-161 (1959)
3. M. Djurasevic, D. Jakobovic, L. Mariot, S. Picek: A survey of metaheuristic algorithms for the design of cryptographic Boolean functions. *Cryptogr. Commun.* 15(6): 1171-1197 (2023)
4. E. Formenti, L. Mariot: An Enumeration Algorithm for Binary Coprime Polynomials with Nonzero Constant Term. *CoRR* abs/2207.00406 (2022)
5. M. Gardner: Mathematical Games: The fantastic combinations of John Conway's new solitaire game "Life". *Scientific American*, vol. 223, pp. 120-123 (1970)
6. E. Goles, P. Montealegre, M. Ríos-Wilson, G. Theyssier: On the complexity of freezing automata networks of bounded pathwidth. *Nat. Comput.* 25(1): 1 (2026)
7. G. A. Hedlund: Endomorphisms and automorphisms of the shift dynamical systems. *Math. Syst. Theory* 3(4): 320-375 (1969)
8. S. Ishii: Measure Theoretic Approach to the Classification of Cellular Automata. *Discret. Appl. Math.* 39(2): 125-136 (1992)
9. J. Kari: Theory of cellular automata: A survey. *Theor. Comput. Sci.* 334(1-3): 3-33 (2005)
10. P. Kurka: Topological dynamics of cellular automata. In *Encyclopedia of Complexity and Systems Science*, pp. 9246-9268 (2009)
11. L. Mariot, E. Formenti, A. Leporati: Constructing Orthogonal Latin Squares from Linear Cellular Automata. In: *Exploratory papers of AUTOMATA 2016* (2016)
12. L. Mariot, S. Picek, A. Leporati, D. Jakobovic: Cellular automata based S-boxes. *Cryptogr. Commun.* 11(1): 41-62 (2019)
13. L. Mariot, M. Gadouleau, E. Formenti, A. Leporati: Mutually orthogonal latin squares based on cellular automata. *Des. Codes Cryptogr.* 88(2): 391-411 (2020)
14. L. Mariot, D. Jakobovic, T. Bäck, J.C. Hernandez-Castro: Artificial Intelligence for the Design of Symmetric Cryptographic Primitives. *Security and Artificial Intelligence*, pp. 3-24 (2022)
15. A. Shamir: How to Share a Secret. *Commun. ACM* 22(11): 612-613 (1979)
16. A. R. Smith III: Real-Time Language Recognition by One-Dimensional Cellular Automata. *J. Comput. Syst. Sci.* 6(3): 233-253 (1972)
17. D. R. Stinson: *Combinatorial designs - constructions and analysis*. Springer (2004)
18. K. Sutner: De Bruijn graphs and linear cellular automata. *Complex Syst.*, vol. 5(1): 19-30 (1991)
19. K. Sutner: Model Checking One-Dimensional Cellular Automata. *J. Cell. Autom.* 4(3): 213-224 (2009)
20. K. Sutner: Linear cellular automata and decidability. In *Automata, Universality, Computation: Tribute to Maurice Margenstern*, pp. 259-276 (2015)
21. J. Von Neumann, *Theory of self-reproducing automata*. Edited by A. W. Burks, University of Illinois Press (1966)
22. S. Ulam: Random processes and transformations. *Proceedings of the International Congress on Mathematics*, vol. 2, pp. 264-275 (1952)
23. S. Wolfram: Statistical mechanics of cellular automata. *Rev. Mod. Phys.* 55(3): 601-644 (1983)