

# Automaty Komórkowe

## Wykład 6

Witold Bołt, 10.04.2024

# Poprzednio omówiliśmy

- **Wykład 1:** Sprawy organizacyjne, motywację do zajmowania się CA, podstawowe pojęcia / definicje / intuicje.
- **Wykład 2:** Definicja (formalna) i podstawowe fakty o ECA. Reprezentacja Wolframa.
- **Wykład 3:** Symetrie w zbiorze ECA, relacje do ogólnej teorii układów dynamicznych, własności CA/ECA.
- **Wykład 4:** Alternatywne reprezentacje reguły lokalnej (wielomiany, wyrażenia logiczne), problem klasyfikacji gęstości (DCP).
- **Dwa tygodnie przerwy** 😓
- **Wykład 5 (zdalny):** Algorytmy ewolucyjne - poszukiwanie automatów komórkowych o określonych własnościach






# Stochastyczne Automaty Komórkowe

... czyli coś na co wszyscy czekali 🤪

# Automaty Komórkowe, które znamy i ❤️

- **Stan** komórki w chwili  $t + 1$  zależy od stanu komórki w chwili  $t$  oraz od stanów komórek sąsiednich w chwili  $t$ . Jest **skończenie wiele stanów**, które można przyjmować.
- Zmiana stanu jest deterministyczna.
- **Reguła lokalna** opisuje zmianę stanu komórki i jest (deterministyczną) funkcją.
- **Reguła globalna** opisuje zmianę stanów wszystkich komórek zgodnie z działaniem reguły lokalnej. Reguła lokalna jest (deterministyczną) funkcją.
- Jeśli automat ma skończenie wiele komórek, to w pewnym momencie ewolucja stanów staje się okresowa - **historia kołem się toczy** dosłownie.

# Automaty Komórkowe, które znamy i ❤️

- **Stan** komórki w chwili  $t + 1$  zależy od stanu komórki w chwili  $t$  oraz od stanów komórek sąsiednich w chwili  $t$ . Jest **skończenie wiele stanów**, które można przyjmować. 
- Zmiana stanu jest deterministyczna. 
- **Reguła lokalna** opisuje zmianę stanu komórki i jest (deterministyczną) funkcją. 
- **Reguła globalna** opisuje zmianę stanów wszystkich komórek zgodnie z działaniem reguły lokalnej. Reguła lokalna jest (deterministyczną) funkcją. 
- Jeśli automat ma skończenie wiele komórek, to w pewnym momencie ewolucja stanów staje się okresowa - **historia kołem się toczy** dosłownie. 

# Zanim pójdziemy dalej...

- **Nie jestem** ekspertem od teorii rachunku prawdopodobieństwa, statystyki, procesów stochastycznych itp.
- **Będę** kłamać, ale tylko trochę 😊 i naruszać czystą, poprawną, formalną terminologię - ale w słusznym celu!
- Wszystko będzie **intuicyjne** i praktyczne - zgodne raczej z tym jak pisać programy niż jak dowodzić twierdzenia.
- Damy radę! 💪



# Stochastyczne Automaty Komórkowe, które będziemy ❤️

- **Stan** komórki w chwili  $t + 1$  zależy od stanu komórki w chwili  $t$  oraz od stanów komórek sąsiednich w chwili  $t$ . Jest **skończenie wiele stanów**, które można przyjmować.
- Zmiana stanu **nie** jest deterministyczna.
- **Reguła lokalna** opisuje zmianę stanu komórki i jest **funkcją losową**.
- **Reguła globalna** opisuje zmianę stanów wszystkich komórek zgodnie z działaniem reguły lokalnej. Reguła lokalna jest *funkcją losową*.
- Stochastyczne Automaty Komórkowe = Stochastic Cellular Automata (SCAs)
- Niektórzy używają również nazw Probabilistic CAs (PCAs) albo Non-Deterministic CAs albo (bardzo rzadko) Random CAs.

# Funkcje losowe (uwaga teraz kłamię)

- **Funkcja losowa** jest tu rozumiana w sensie “programistycznym” - tzn. każde uruchomienie / realizacja / wyliczenie wartości funkcji losowej dla ustalonego zestawu argumentów może dawać różne wyniki należące do **dziedziny** tej funkcji losowej.
- Jeśli dziedzina funkcji losowej ma *skończenie wiele elementów* (a tylko takie sytuacje będą miały u nas miejsce), to mamy prostą sytuację bo aby zdefiniować funkcję losową wystarczy podać *prawdopodobieństwa* przyjęcia przez nią poszczególnych, możliwych wyników.
- W ramach ćwiczenia domowego możecie sobie wypracować formalnie poprawną i prawdziwie matematyczną definicję tej prostej koncepcji, ale nie będzie to nam potrzebne do niczego na zajęciach.



# Losowa reguła lokalna

- Niech  $f: \{0,1\}^3 \rightarrow \{0,1\}$  będzie **funkcją losową**. Aby zdefiniować  $f$  wystarczy podać wartości ośmiu **prawdopodobieństw**:

$$\begin{bmatrix} 1,1,1 & 1,1,0 & 1,0,1 & 1,0,0 & 0,1,1 & 0,1,0 & 0,0,1 & 0,0,0 \\ p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0 \end{bmatrix}$$

- Podane prawdopodobieństwa interpretujemy następująco:

$$\Pr(f(1,1,1) = 1) = p_7, \dots, \Pr(f(0,0,0) = 1) = p_0$$

- Oczywiście zachodzi wtedy:

$$\Pr(f(1,1,1) = 0) = 1 - p_7, \dots, \Pr(f(0,0,0) = 0) = 1 - p_0$$

- ... no i właśnie dlatego wystarczy zdefiniować jedynie osiem prawdopodobieństw.
- Powyższą tabelkę prawdopodobieństw będziemy nazywać **pLUT** (*probabilistic LUT*). Podobnie jak w przypadku zwykłego LUT, wystarczy podać sam wektor prawdopodobieństw  $(p_7, p_6, \dots, p_0)$ .

# Losowa reguła lokalna

- Zwróć uwagę, że reguła lokalna deterministycznego CA zadana przez LUT zawierający jedynie wartości 0 i 1 może być interpretowana jako pLUT i wszystko “będzie się zgadzać”.
- Mówiąc nieco bardziej formalnie, stochastyczne automaty komórkowe zdefiniowane przez losowe reguły lokalne jak na poprzednim slajdzie, są **rozszerzeniem** pojęcia deterministycznych automatów komórkowych.  
“ $CA \subset SCA$ ”

# SCAs

- Reguła globalna - funkcja losowa  $F: \{0,1\}^* \rightarrow \{0,1\}^*$
- Ewolucja SCA - łańcuch Markowa (bez pamięci)
- Miary probabilistyczne w przestrzeni konfiguracji... nie zajmujemy się tym
- Diagram czasoprzestrzenny? Rozkład na przestrzeni diagramów.
- Czy SCA zawsze jest “losowy”? Oczywiście nie. Jeśli część z prawdopodobieństw  $p_i \in \{0,1\}$  to SCA może czasem (lub niemal zawsze) zachowywać się deterministycznie.
- Jeśli  $p_i \in (0,1)$  dla każdego  $i$ , to wtedy dla dowolnego  $\mathbf{x} \in \{0,1\}^N$  zachodzi:

$$\forall_{\mathbf{y} \in \{0,1\}^N} \Pr( F(\mathbf{x}) = \mathbf{y} ) > 0,$$

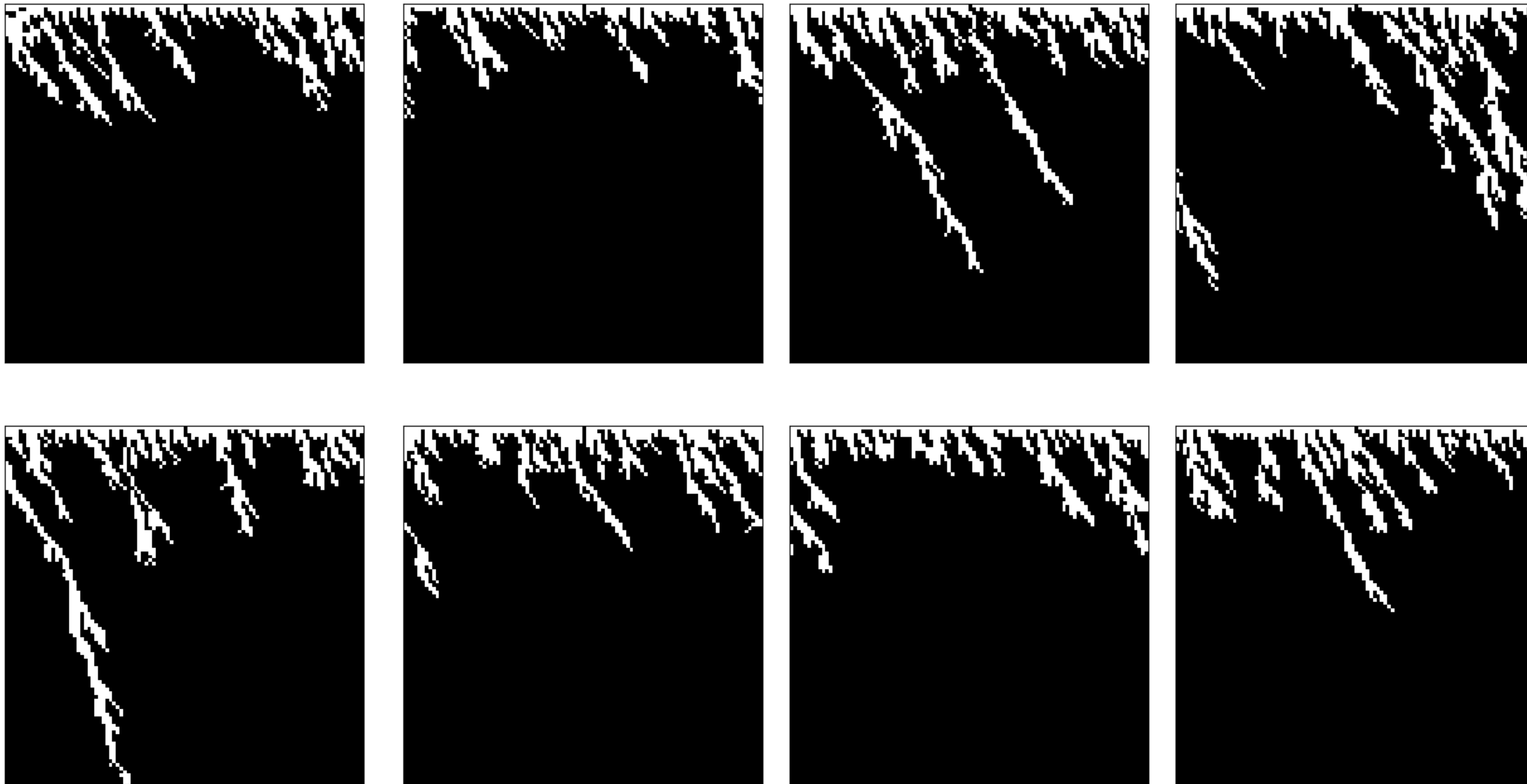
- czyli mówiąc potocznie “wszystko się może zdarzyć...”.

# Przykład

- Niech pLUT będzie następujący:

$$\begin{bmatrix} 1,1,1 & 1,1,0 & 1,0,1 & 1,0,0 & 0,1,1 & 0,1,0 & 0,0,1 & 0,0,0 \\ 1 & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix}$$

- Jak mogą wyglądać *space-time diagrams*?



100 komórek, konfiguracja początkowa: jedynka w środku, 100 kroków czasowych, 8 wybranych space-time diagram

# Przykład

- Niech pLUT będzie następujący:

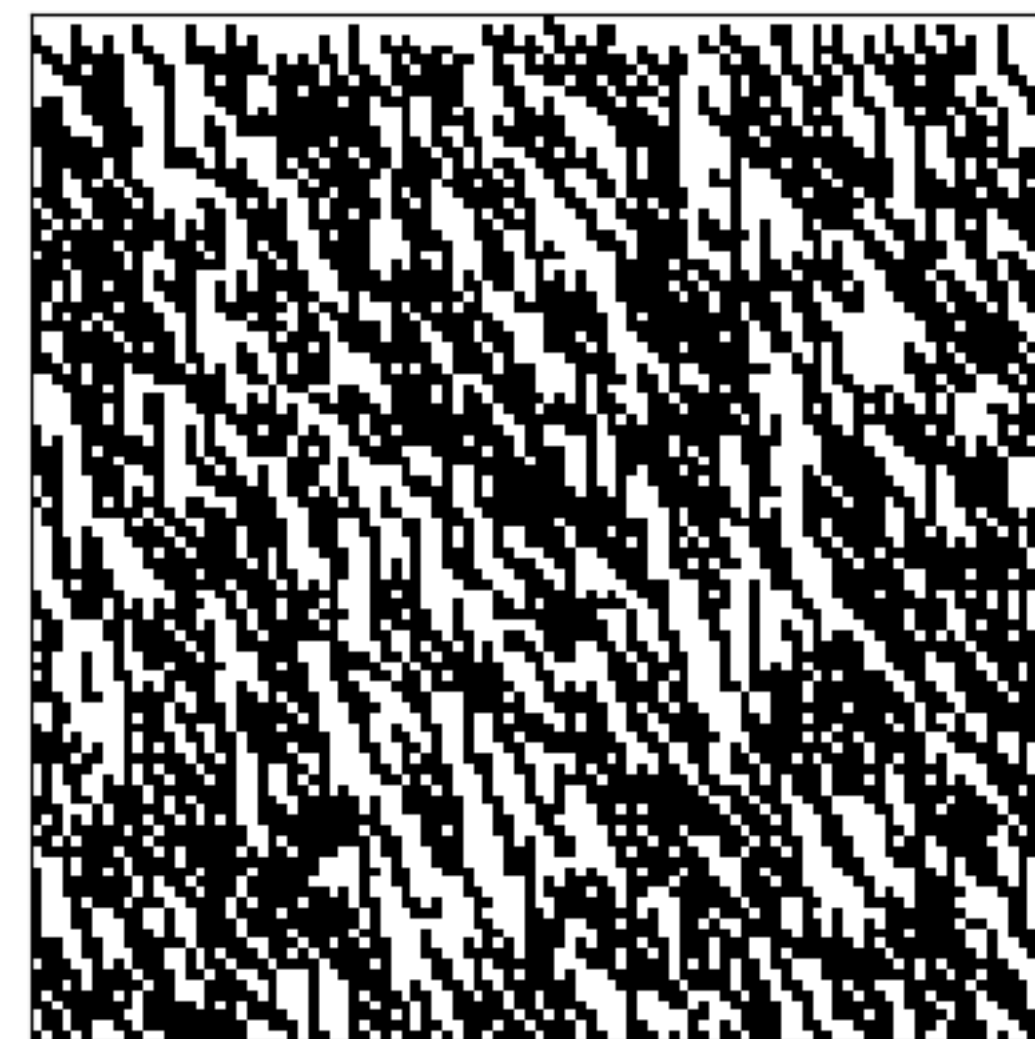
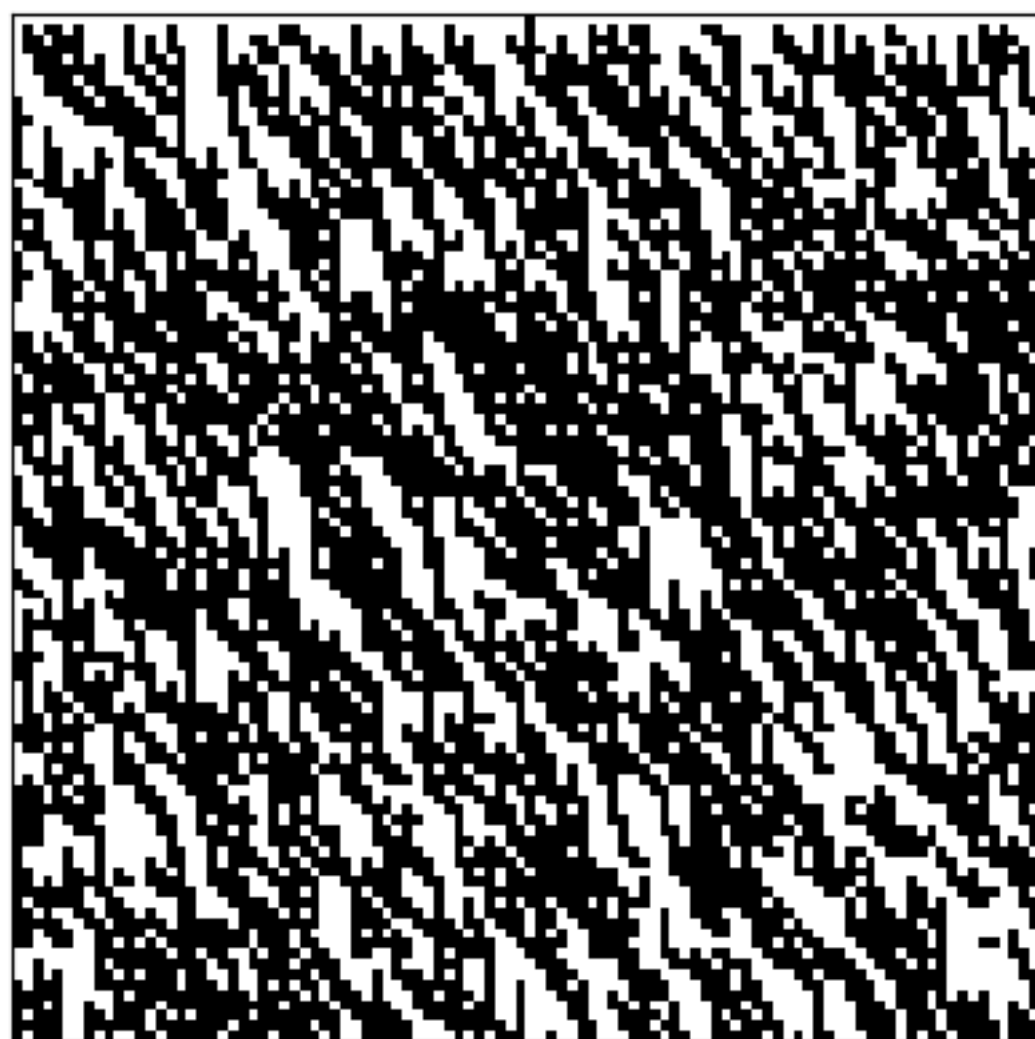
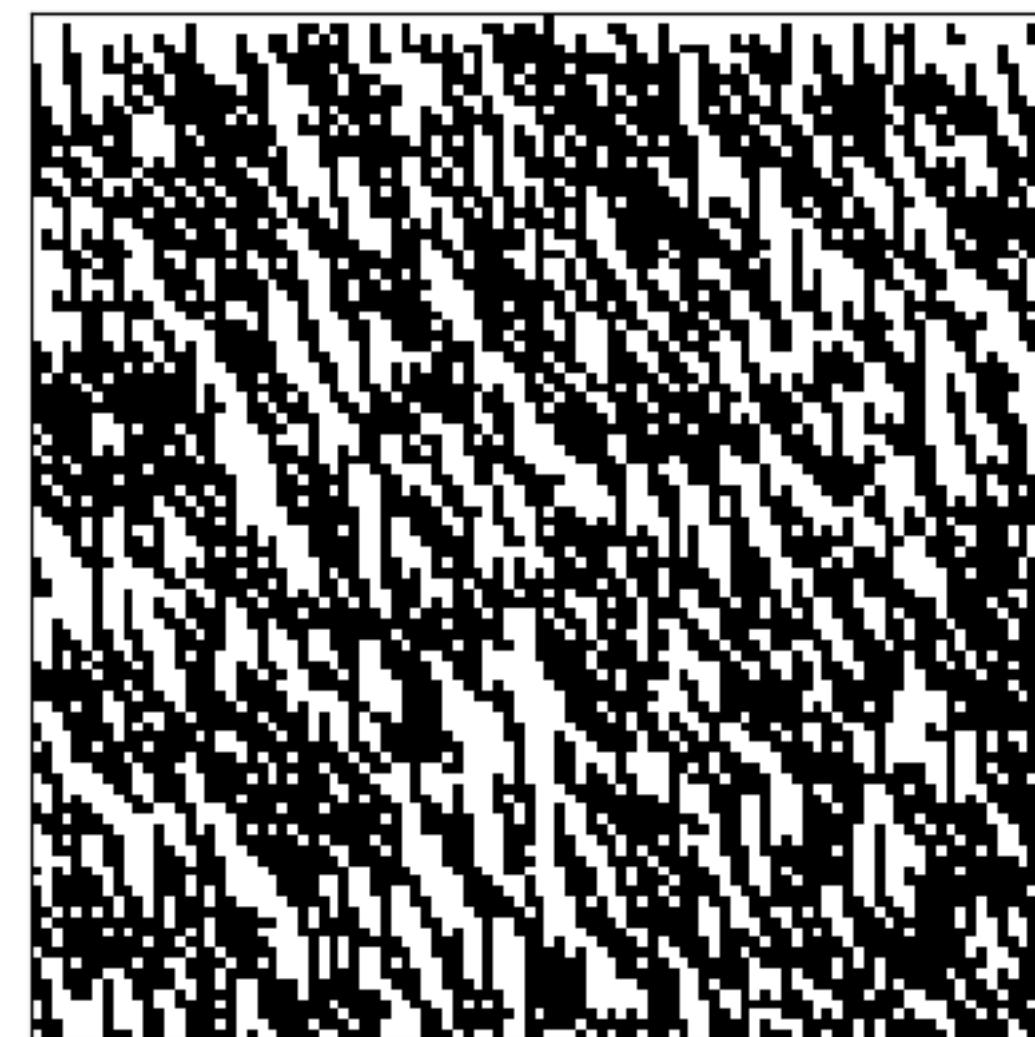
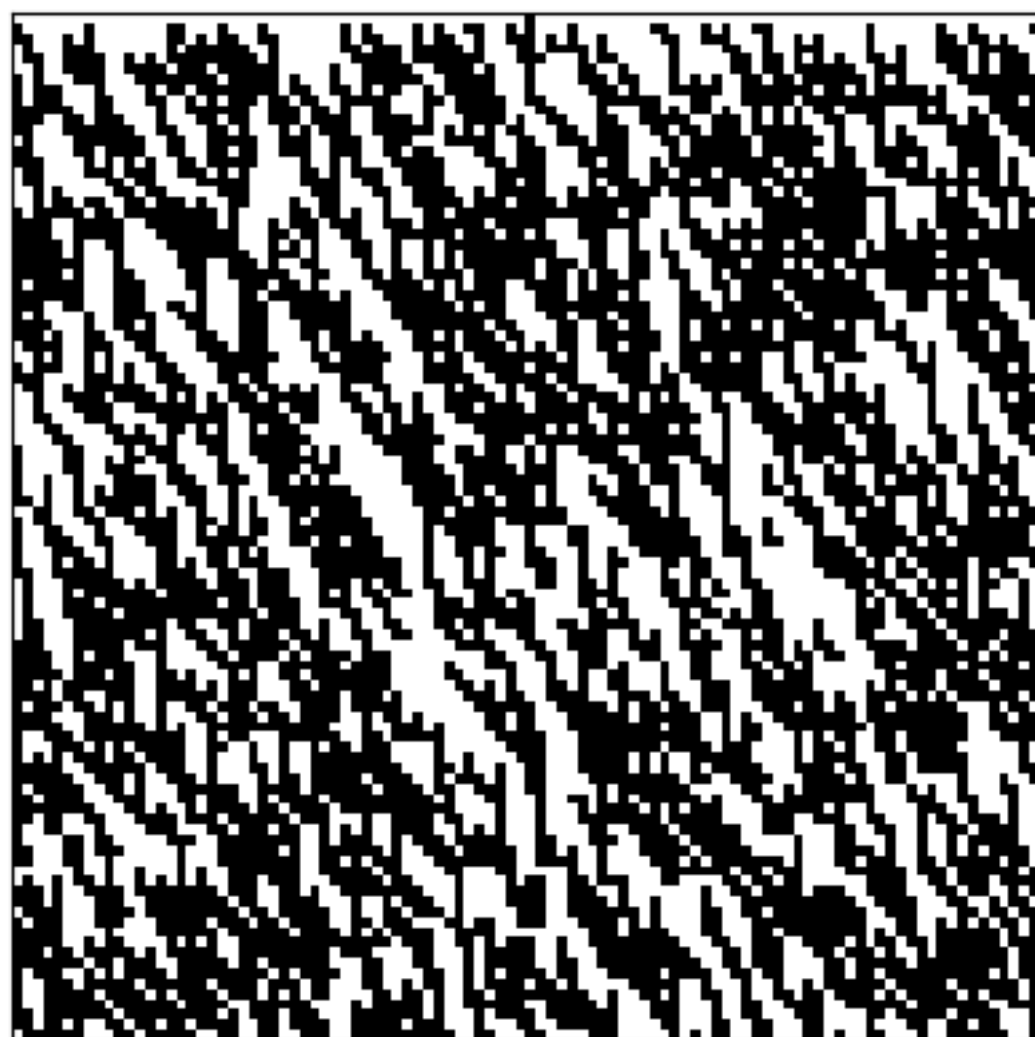
$$\begin{bmatrix} 1,1,1 & 1,1,0 & 1,0,1 & 1,0,0 & 0,1,1 & 0,1,0 & 0,0,1 & 0,0,0 \\ 1 & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix}$$

- Zmodyfikujmy lekko ten pLUT:

$$\begin{bmatrix} 1,1,1 & 1,1,0 & 1,0,1 & 1,0,0 & 0,1,1 & 0,1,0 & 0,0,1 & 0,0,0 \\ \mathbf{0.8} & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix}$$

- Jak teraz będzie wyglądać **space-time diagram**?





Zmieniliśmy tylko jeden wpis w pLUT!

# Przykład

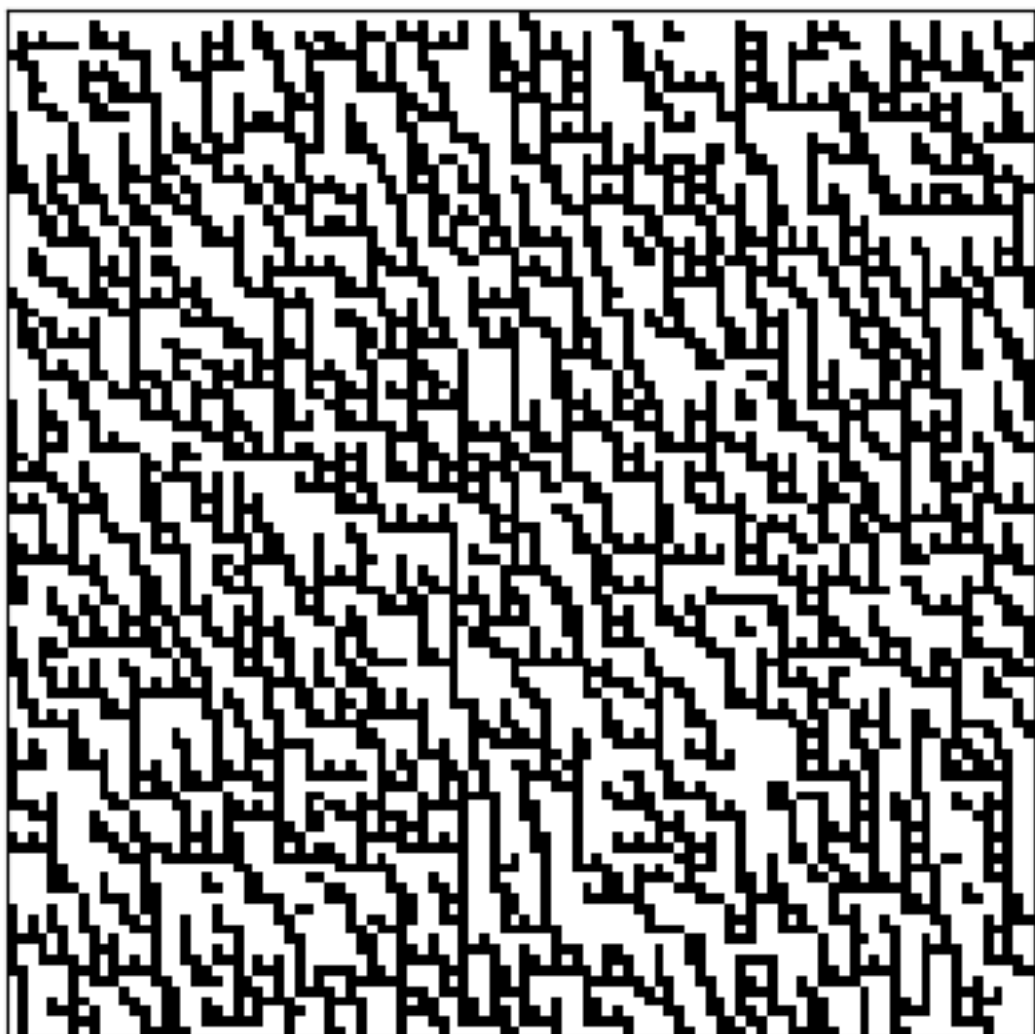
- Niech pLUT będzie następujący:

$$\begin{bmatrix} 1,1,1 & 1,1,0 & 1,0,1 & 1,0,0 & 0,1,1 & 0,1,0 & 0,0,1 & 0,0,0 \\ 1 & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix}$$

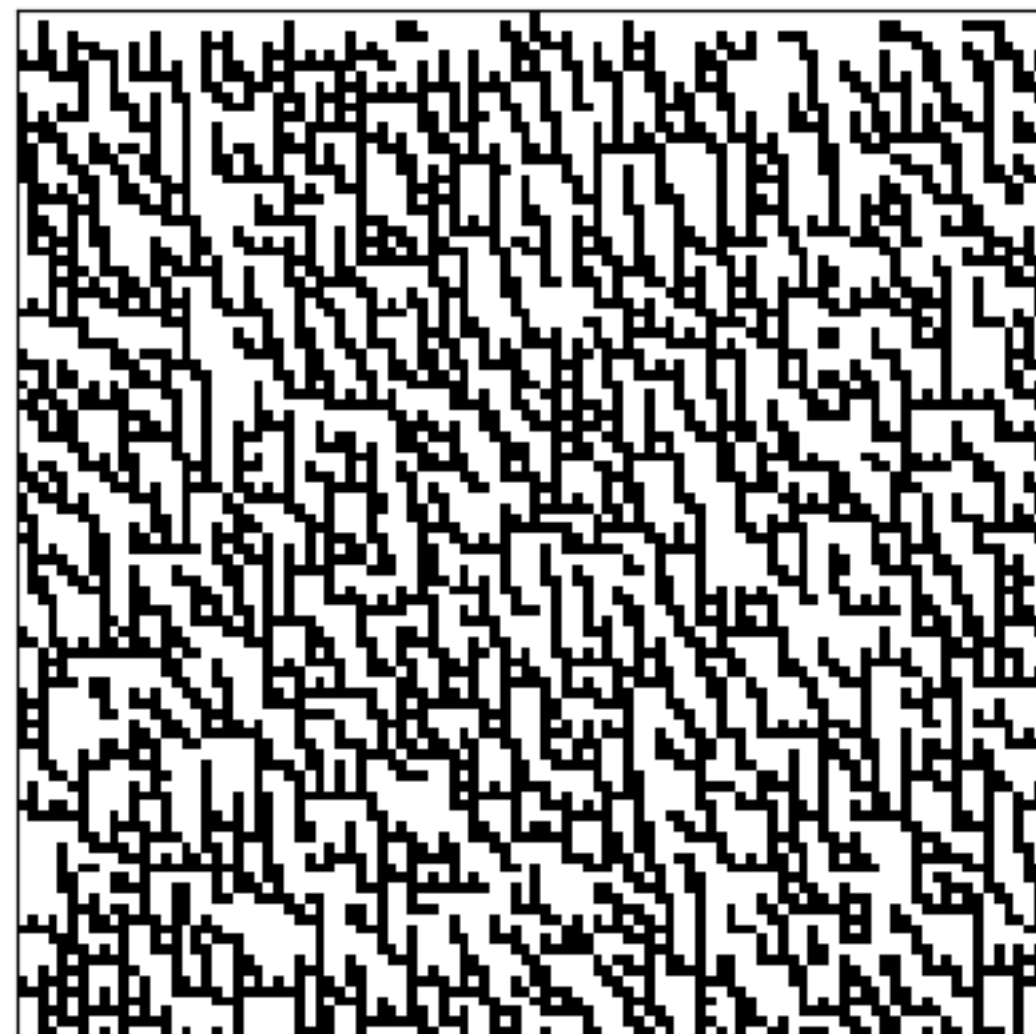
- Zmodyfikujmy lekko ten pLUT:

$$\begin{bmatrix} 1,1,1 & 1,1,0 & 1,0,1 & 1,0,0 & 0,1,1 & 0,1,0 & 0,0,1 & 0,0,0 \\ \mathbf{p} & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix}$$

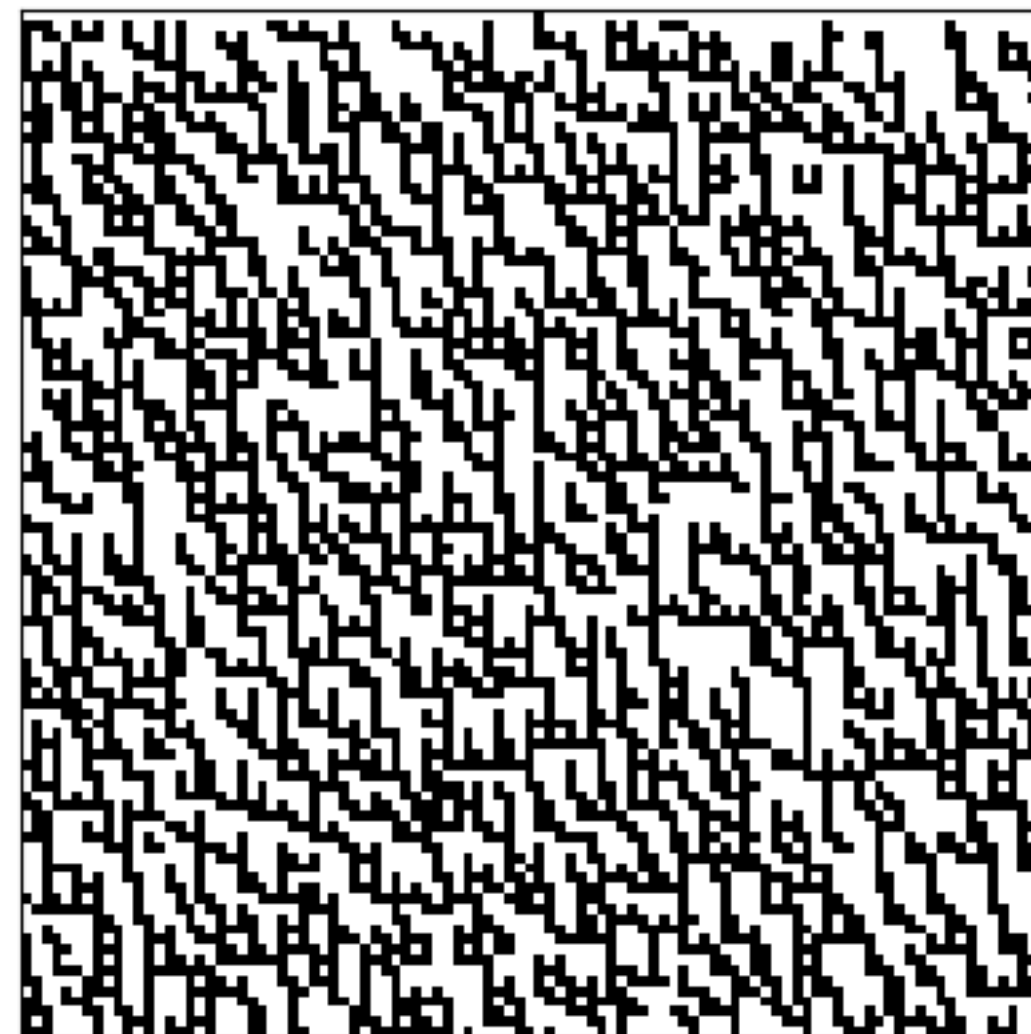
- Jak teraz będzie wyglądać **space-time diagram**?



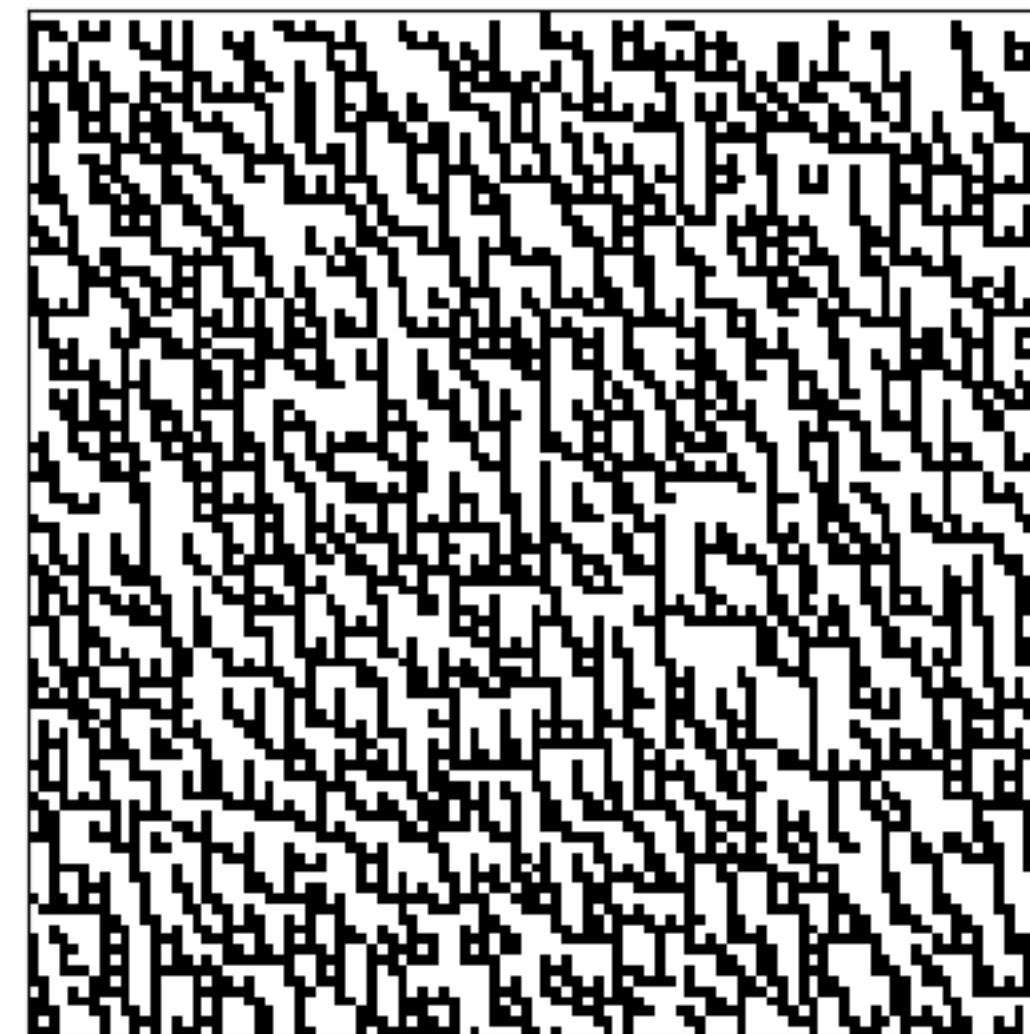
$p = 0.0$



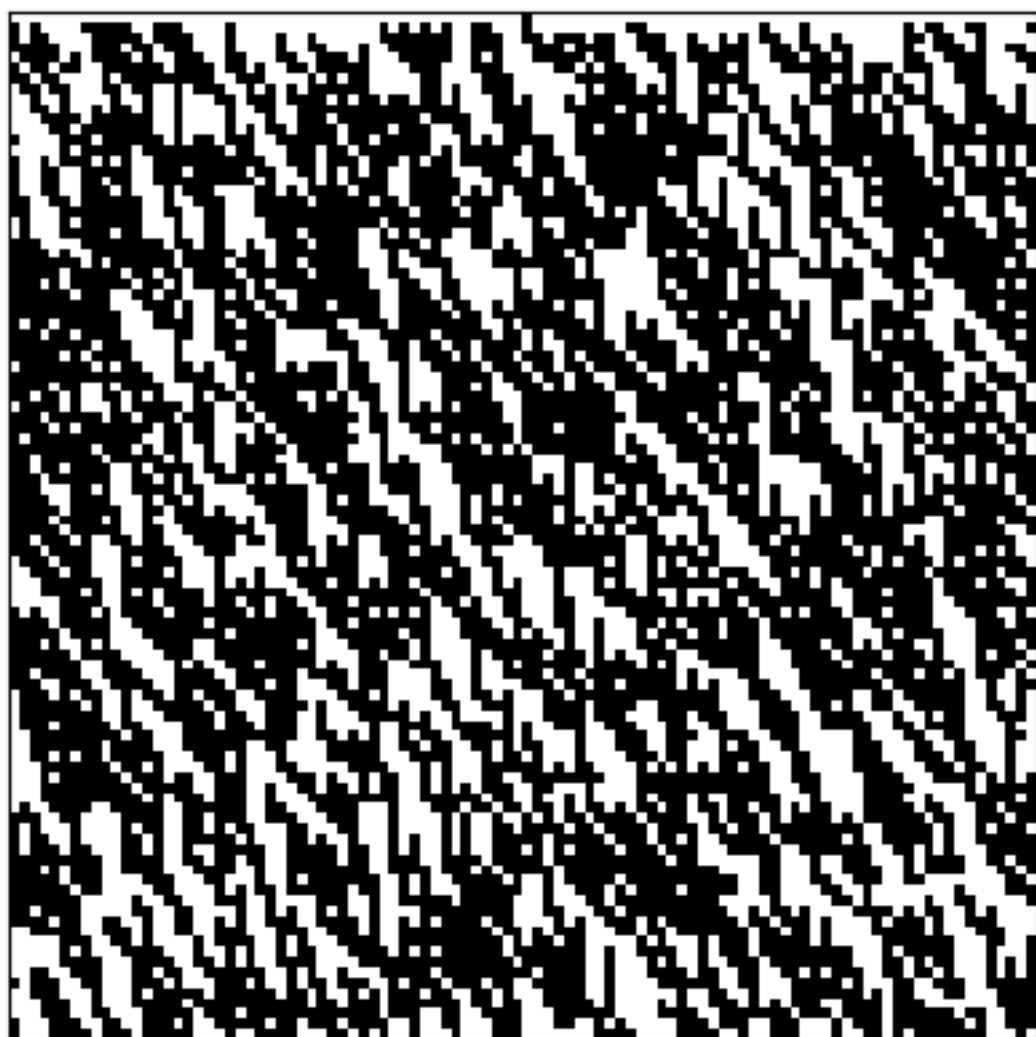
$p = 0.01$



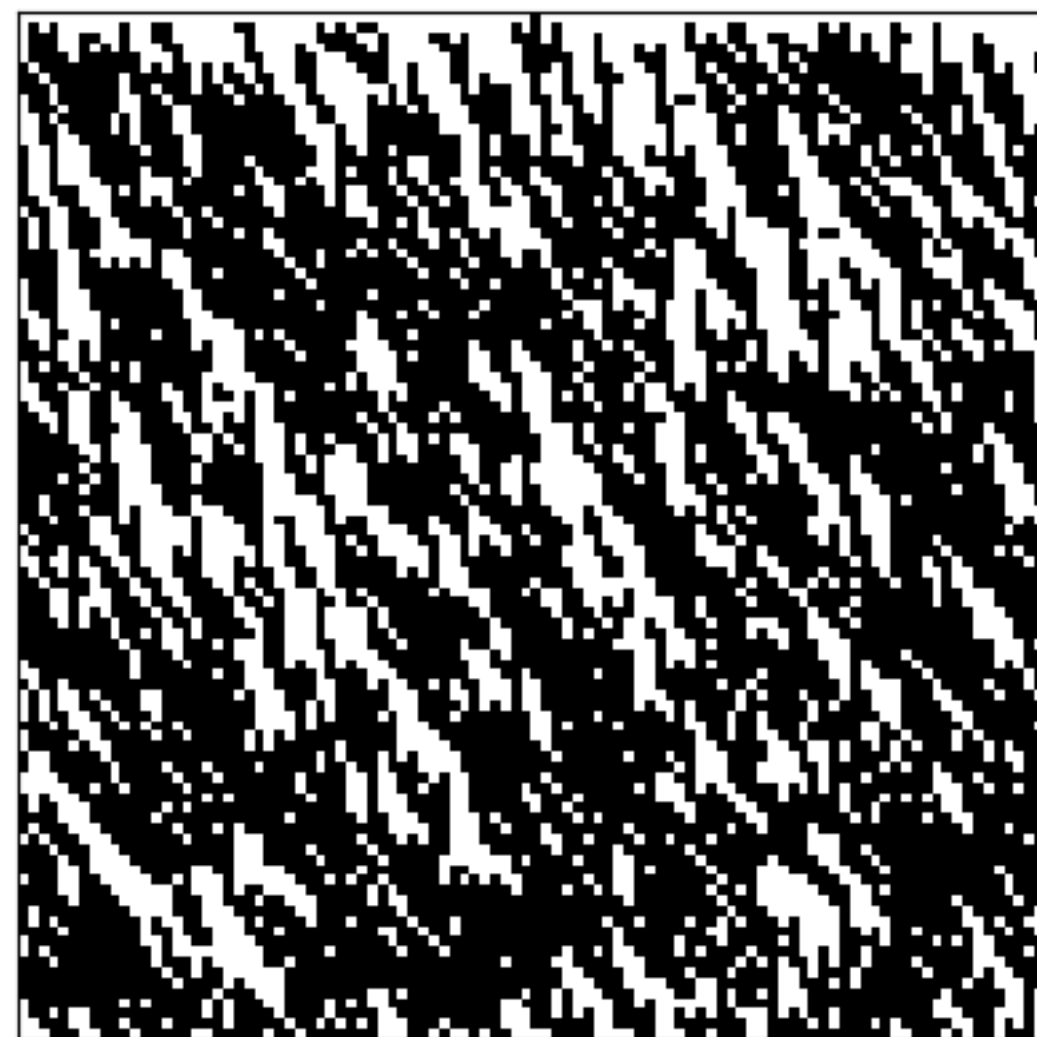
$p = 0.1$



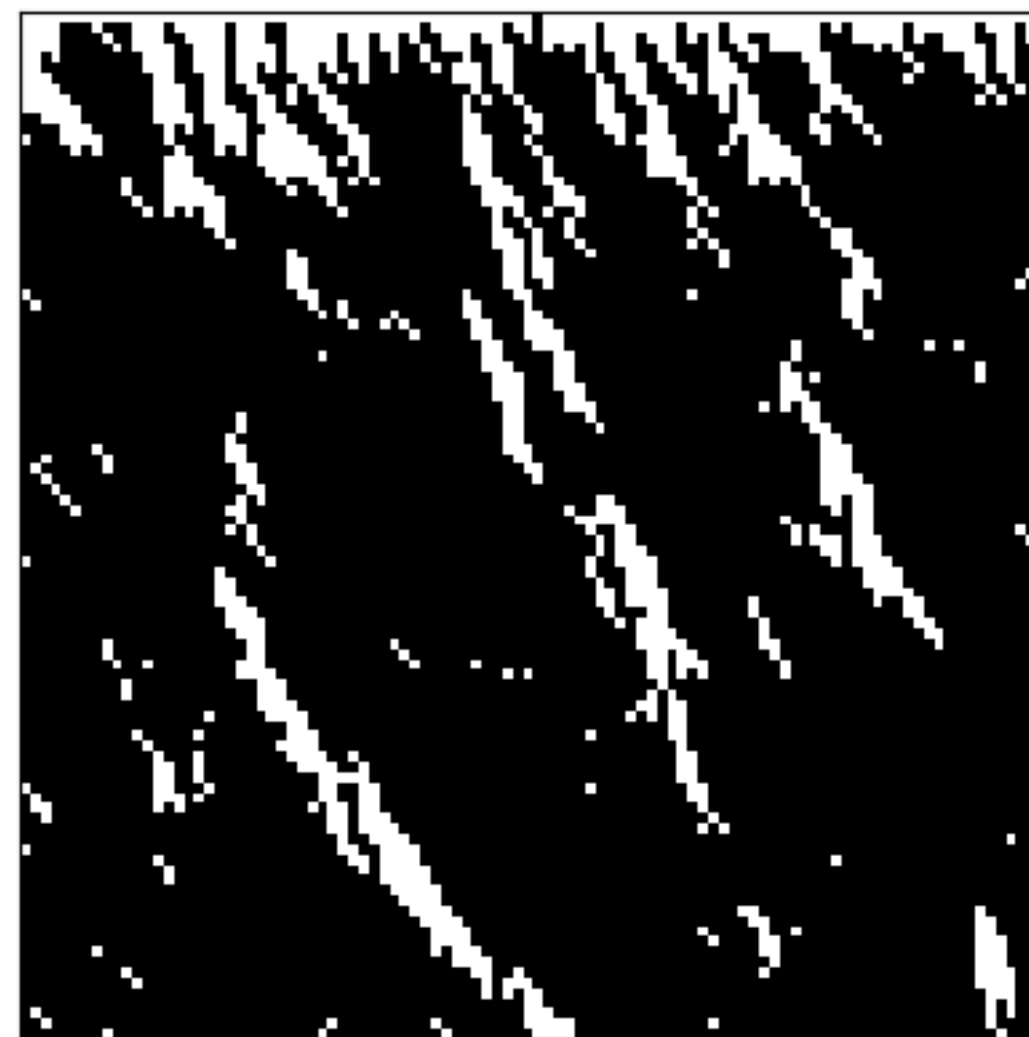
$p = 0.2$



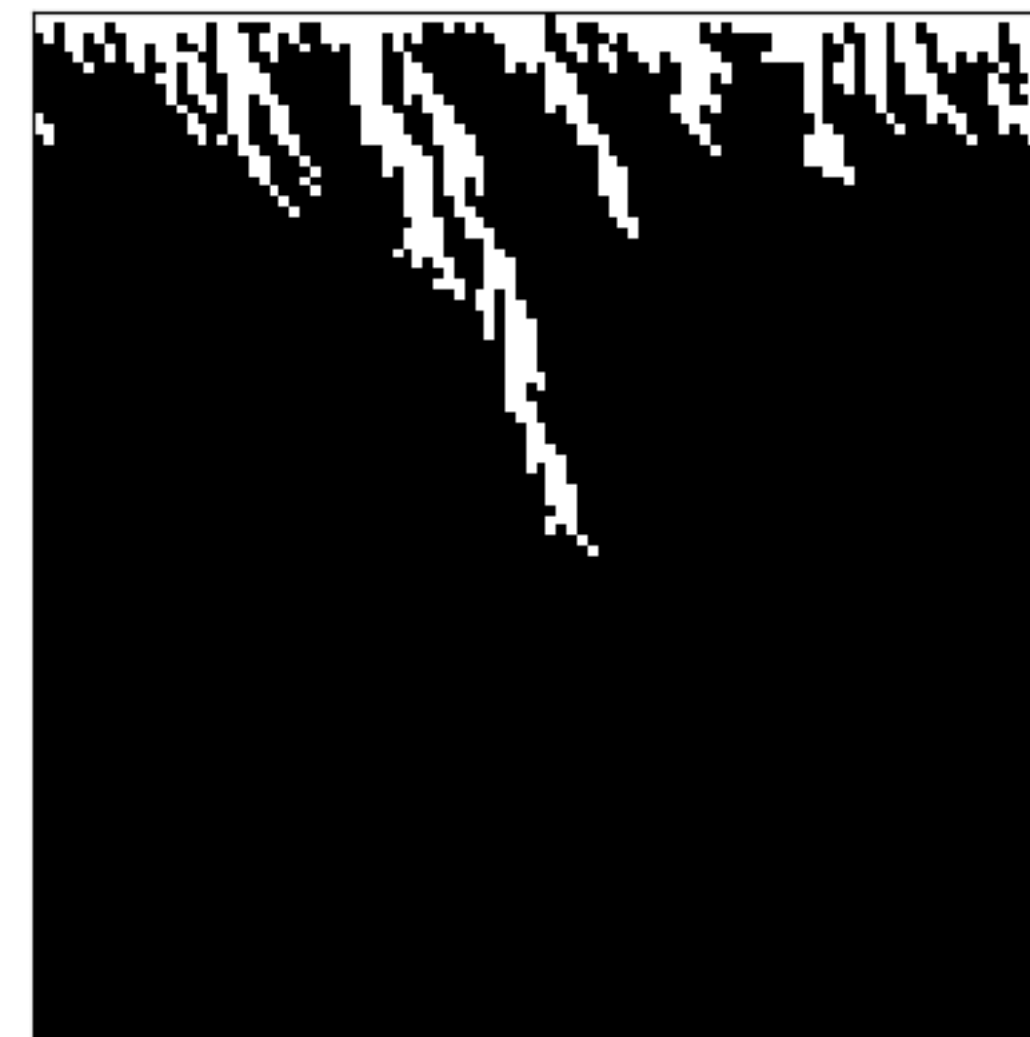
$p = 0.8$



$p = 0.9$



$p = 0.99$



$p = 1.0$

# Implementacja w Python

```

def sca_evolve(plut: np.ndarray, x: np.ndarray) -> np.ndarray:
    probs = plut[7 - (np.roll(x, 1) * 4 + x * 2 + np.roll(x, -1))]
    rnd = np.random.random(size=probs.shape)
    return (rnd < probs).astype(np.int8)

def sca_evolve_spacetime(plut: np.ndarray, initial_conf: np.ndarray, steps: int) -> np.ndarray:
    rows = [initial_conf]
    for _ in range(1, steps):
        rows.append(sca_evolve(plut, rows[-1]))
    return np.stack(rows)

def sca_space_time(plut: np.ndarray, initial_conf: np.ndarray, steps: int):
    spacetime = sca_evolve_spacetime(plut, initial_conf, steps)
    plt.figure(figsize=(5, 5))
    plt.imshow(spacetime, cmap='binary', interpolation='nearest')
    plt.xticks([])
    plt.yticks([])
    plt.show()

```

✓ 0.0s

Python

# Dekompozycja SCA



# Algorytm dekompozycji SCA

- Załóżmy, że dany jest SCA zapisany w postaci pLUT. Algorytm jest ogólny i działa dla dowolnego  $r \geq 0$ , dowolnej liczby stanów, dowolnego wymiaru itd.
- Niestety choć pomysł jest prosty, to wzory są straszne, więc zamiast je pisać zrobimy **przykład**! Zakładamy oczywiście jednowymiarowy automat binary, o promieniu sąsiedztwa  $r = 1$ .
- Niech pLUT będzie następujący:

$$\begin{bmatrix} 1,1,1 & 1,1,0 & 1,0,1 & 1,0,0 & 0,1,1 & 0,1,0 & 0,0,1 & 0,0,0 \\ 1 & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix}$$

# Krok 0: inaczej zapisujemy pLUT

- Nasz pLUT od teraz będziemy pisać tak:

$$\begin{bmatrix} 1 - p_7 & 1 - p_6 & 1 - p_5 & 1 - p_4 & 1 - p_3 & 1 - p_2 & 1 - p_1 & 1 - p_0 \\ p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0 \end{bmatrix}$$

- Czyli w naszym przykładzie to będzie tak:

$$\Pi_0 := \begin{bmatrix} 0 & 0.1 & 0.25 & 0.6 & 0.5 & 0 & 1 & 0.8 \\ 1 & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix}$$

# Krok 1: szukamy maksimumów w kolumnach

$$\Pi_0 := \begin{bmatrix} 0 & 0.1 & 0.25 & 0.6 & 0.5 & 0 & 1 & 0.8 \\ 1 & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix}$$

- W każdej z **kolumn** powyższej macierzy wybieramy element maksymalny. Jeśli maksimum nie jest jednoznacznie wyznaczone, to wybieramy pierwszy z elementów.
- To, który element wybieramy zaznaczamy sobie w macierzy  $L_0$  wstawiając w odpowiednie miejsce 1. Niewybrane miejsce oznaczamy przez 0. Dostajemy:

$$L_0 := \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

## Krok 2: wybieramy najmniejsze maksimum

$$\Pi_0 := \begin{bmatrix} 0 & 0.1 & 0.25 & 0.6 & 0.5 & 0 & 1 & 0.8 \\ 1 & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix}$$

- Z wybranych maksimumów z kolumn wybieramy **najmniejsze** maksimum i oznaczamy je przez  $\alpha_0$ .
- Oczywiście w przykładzie zachodzi:  $\alpha_0 = 0.5$ , bo:

$$\alpha_0 = \min\{1, 0.9, 0.75, 0.6, 0.5, 0.8\}$$

## Krok 3: odcinamy tyle ile się da :)

$$\Pi_0 := \begin{bmatrix} 0 & 0.1 & 0.25 & 0.6 & 0.5 & 0 & 1 & 0.8 \\ 1 & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix}$$

$$L_0 := \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \alpha_0 = 0.5$$

- Budujemy nową macierz  $\Pi_1$  zgodnie z wzorem:

$$\Pi_1 := \Pi_0 - \alpha_0 L_0$$

- Otrzymujemy (na żółto zaznaczono miejsca, w których coś się zmieniło względem  $\Pi_0$ ):

$$\Pi_1 := \begin{bmatrix} 0 & 0.1 & 0.25 & 0.1 & 0 & 0 & 0.5 & 0.3 \\ 0.5 & 0.4 & 0.25 & 0.4 & 0.5 & 0.5 & 0 & 0.2 \end{bmatrix}$$

## Krok 4: powtarzamy konstrukcję

$$\Pi_1 := \begin{bmatrix} 0 & 0.1 & 0.25 & 0.1 & 0 & 0 & 0.5 & 0.3 \\ 0.5 & 0.4 & 0.25 & 0.4 & 0.5 & 0.5 & 0 & 0.2 \end{bmatrix}$$

- Budujemy macierz  $L_1$  i znajdujemy liczbę  $\alpha_1$ , dokładnie tak jak poprzednio, startując od macierzy  $\Pi_1$  zamiast  $\Pi_0$ .
- Dostajemy:

$$L_1 := \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}, \alpha_1 = 0.25$$



## Krok 5: powtarzamy odejmowanie!

$$\Pi_1 := \begin{bmatrix} 0 & 0.1 & 0.25 & 0.1 & 0 & 0 & 0.5 & 0.3 \\ 0.5 & 0.4 & 0.25 & 0.4 & 0.5 & 0.5 & 0 & 0.2 \end{bmatrix}$$

$$L_1 := \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}, \alpha_1 = 0.25$$

- Stosujemy wzór:  $\Pi_2 = \Pi_1 - \alpha_1 L_1$  i dostajemy:

$$\Pi_2 := \begin{bmatrix} 0 & 0.1 & 0 & 0.1 & 0 & 0 & 0.25 & 0.05 \\ 0.25 & 0.15 & 0.25 & 0.15 & 0.25 & 0.25 & 0 & 0.2 \end{bmatrix}$$

## Krok 6: chyba już wszystko jasne?

$$\Pi_2 := \begin{bmatrix} 0 & 0.1 & 0 & 0.1 & 0 & 0 & 0.25 & 0.05 \\ 0.25 & 0.15 & 0.25 & 0.15 & 0.25 & 0.25 & 0 & 0.2 \end{bmatrix}$$

- No a stąd łatwo policzymy  $L_2$ ,  $\alpha_2$  oraz w końcu  $\Pi_3 = \Pi_2 - \alpha_2 L_2$ :

$$L_2 := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \alpha_2 = 0.15$$

$$\Pi_3 := \begin{bmatrix} 0 & 0.1 & 0 & 0.1 & 0 & 0 & 0.1 & 0.05 \\ 0.1 & 0 & 0.1 & 0 & 0.1 & 0.1 & 0 & 0.05 \end{bmatrix}$$

## Krok 7: chyba już wszystko jasne?!

$$\Pi_3 := \begin{bmatrix} 0 & 0.1 & 0 & 0.1 & 0 & 0 & 0.1 & 0.05 \\ 0.1 & 0 & 0.1 & 0 & 0.1 & 0.1 & 0 & 0.05 \end{bmatrix}$$

- No a stąd łatwo policzymy  $L_3$ ,  $\alpha_3$  oraz w końcu  $\Pi_4 = \Pi_3 - \alpha_3 L_3$ :

$$L_3 := \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \alpha_3 = 0.05$$

$$\Pi_4 := \begin{bmatrix} 0 & 0.05 & 0 & 0.05 & 0 & 0 & 0.05 & 0 \\ 0.05 & 0 & 0.05 & 0 & 0.05 & 0.05 & 0 & 0.05 \end{bmatrix}$$

## Krok 8: KONIEC! ... prawie ;)

$$\Pi_4 := \begin{bmatrix} 0 & 0.05 & 0 & 0.05 & 0 & 0 & 0.05 & 0 \\ 0.05 & 0 & 0.05 & 0 & 0.05 & 0.05 & 0 & 0.05 \end{bmatrix}$$

- No a stąd łatwo policzymy  $L_4$ ,  $\alpha_4$  oraz w końcu  $\Pi_5 = \Pi_4 - \alpha_4 L_4$ :

$$L_4 := \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}, \alpha_4 = 0.05$$

$$\Pi_5 := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\Pi_0 = \begin{bmatrix} 0 & 0.1 & 0.25 & 0.6 & 0.5 & 0 & 1 & 0.8 \\ 1 & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix} \quad \alpha_0 = 0.5 \quad L_0 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\Pi_1 = \begin{bmatrix} 0 & 0.1 & 0.25 & 0.1 & 0 & 0 & 0.5 & 0.3 \\ 0.5 & 0.4 & 0.25 & 0.4 & 0.5 & 0.5 & 0 & 0.2 \end{bmatrix} \quad \alpha_1 = 0.25 \quad L_1 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$\Pi_2 = \begin{bmatrix} 0 & 0.1 & 0 & 0.1 & 0 & 0 & 0.25 & 0.05 \\ 0.25 & 0.15 & 0.25 & 0.15 & 0.25 & 0.25 & 0 & 0.2 \end{bmatrix} \quad \alpha_2 = 0.15 \quad L_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

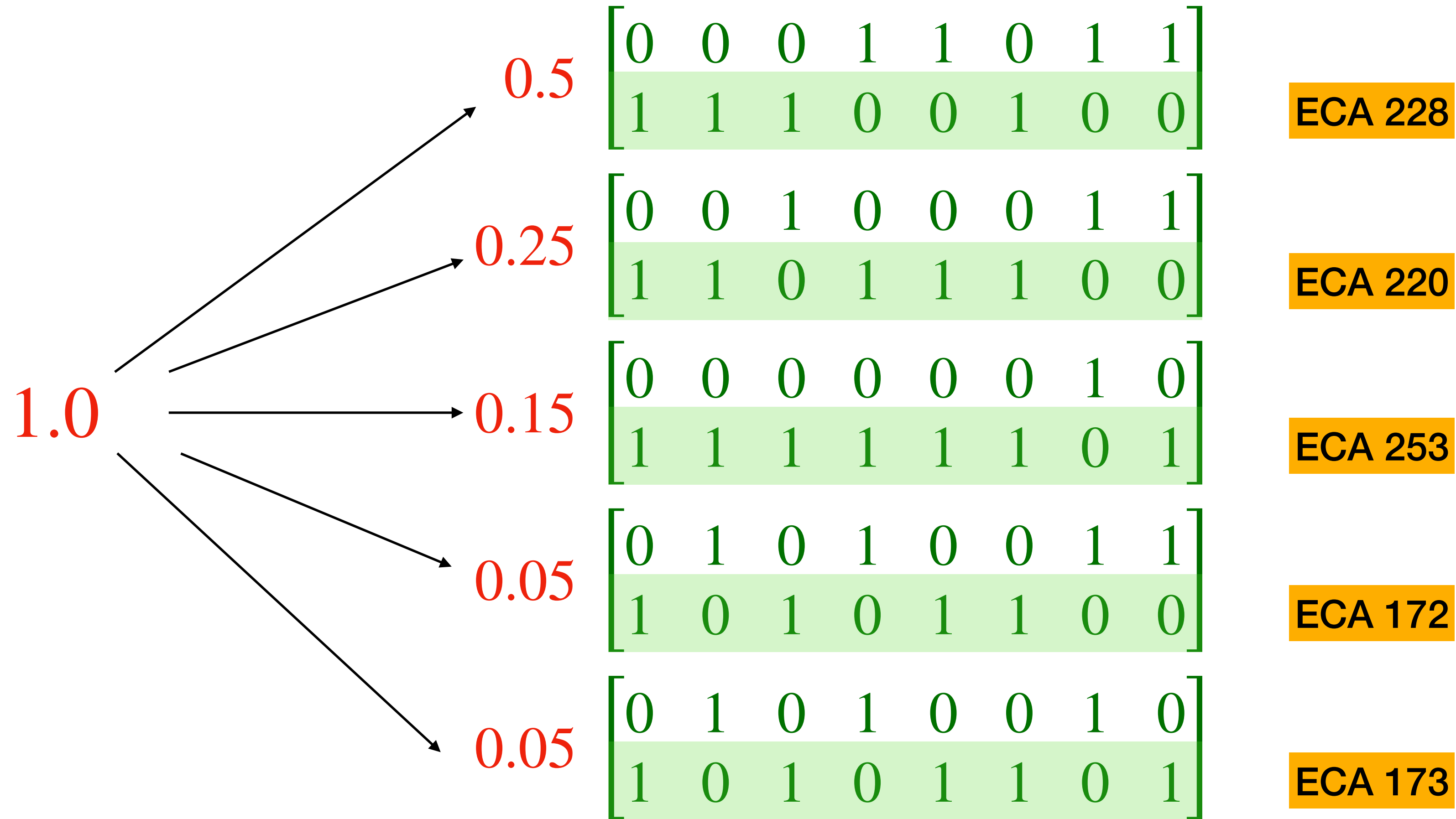
$$\Pi_3 = \begin{bmatrix} 0 & 0.1 & 0 & 0.1 & 0 & 0 & 0.1 & 0.05 \\ 0.1 & 0 & 0.1 & 0 & 0.1 & 0.1 & 0 & 0.05 \end{bmatrix} \quad \alpha_3 = 0.05 \quad L_3 = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$\Pi_4 = \begin{bmatrix} 0 & 0.05 & 0 & 0.05 & 0 & 0 & 0.05 & 0 \\ 0.05 & 0 & 0.05 & 0 & 0.05 & 0.05 & 0 & 0.05 \end{bmatrix} \quad L_4 = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad \alpha_4 = 0.05$$

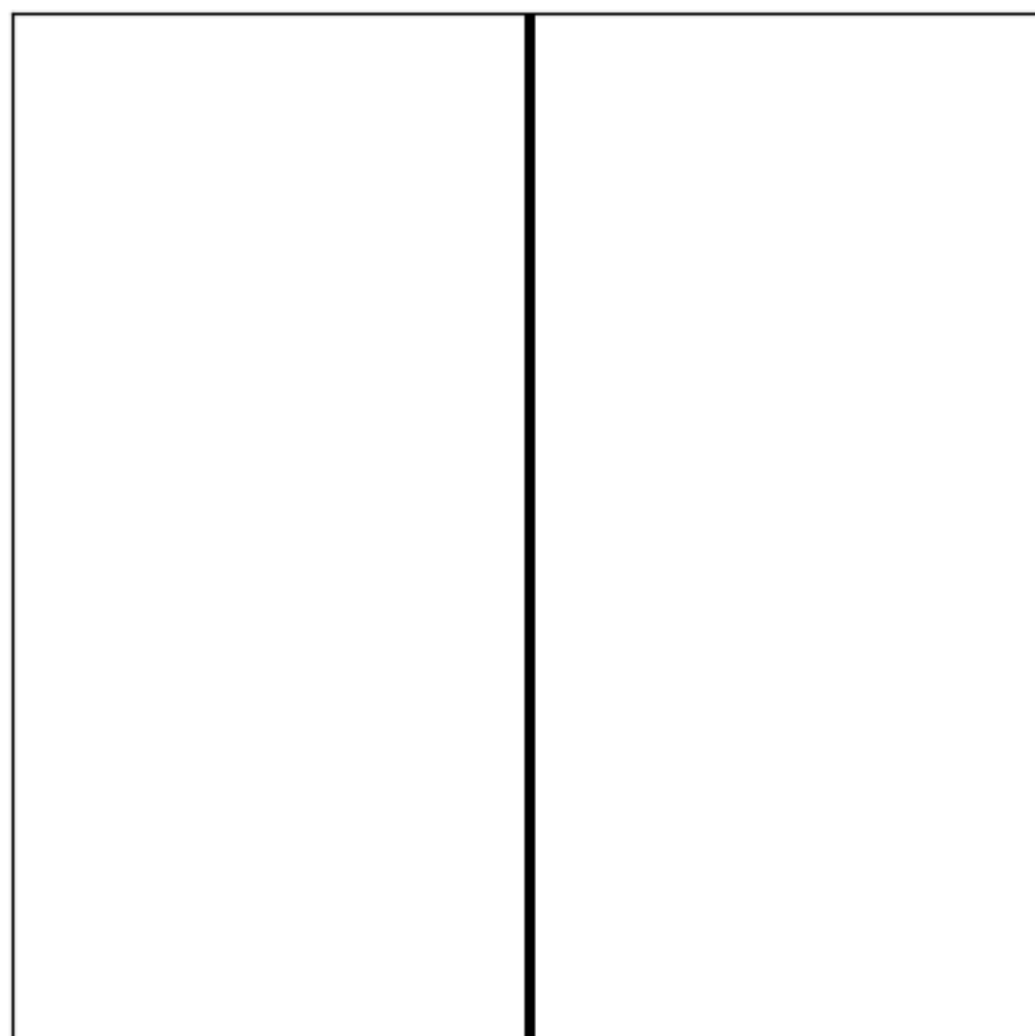
$$\Pi_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\Pi_0 = \alpha_0 L_0 + \alpha_1 L_1 + \alpha_2 L_2 + \alpha_3 L_3 + \alpha_4 L_4$$

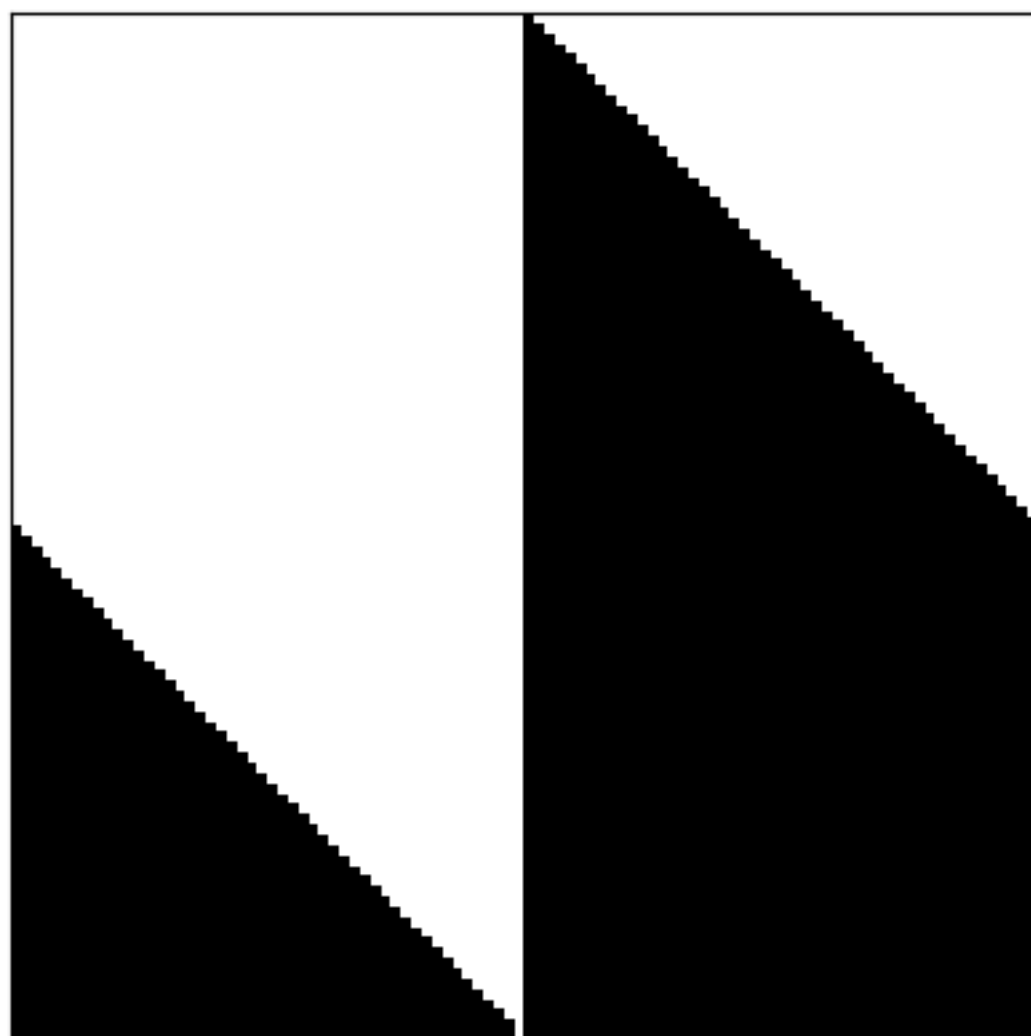
$$\Pi_0 = \begin{bmatrix} 0 & 0.1 & 0.25 & 0.6 & 0.5 & 0 & 1 & 0.8 \\ 1 & 0.9 & 0.75 & 0.4 & 0.5 & 1 & 0 & 0.2 \end{bmatrix}$$







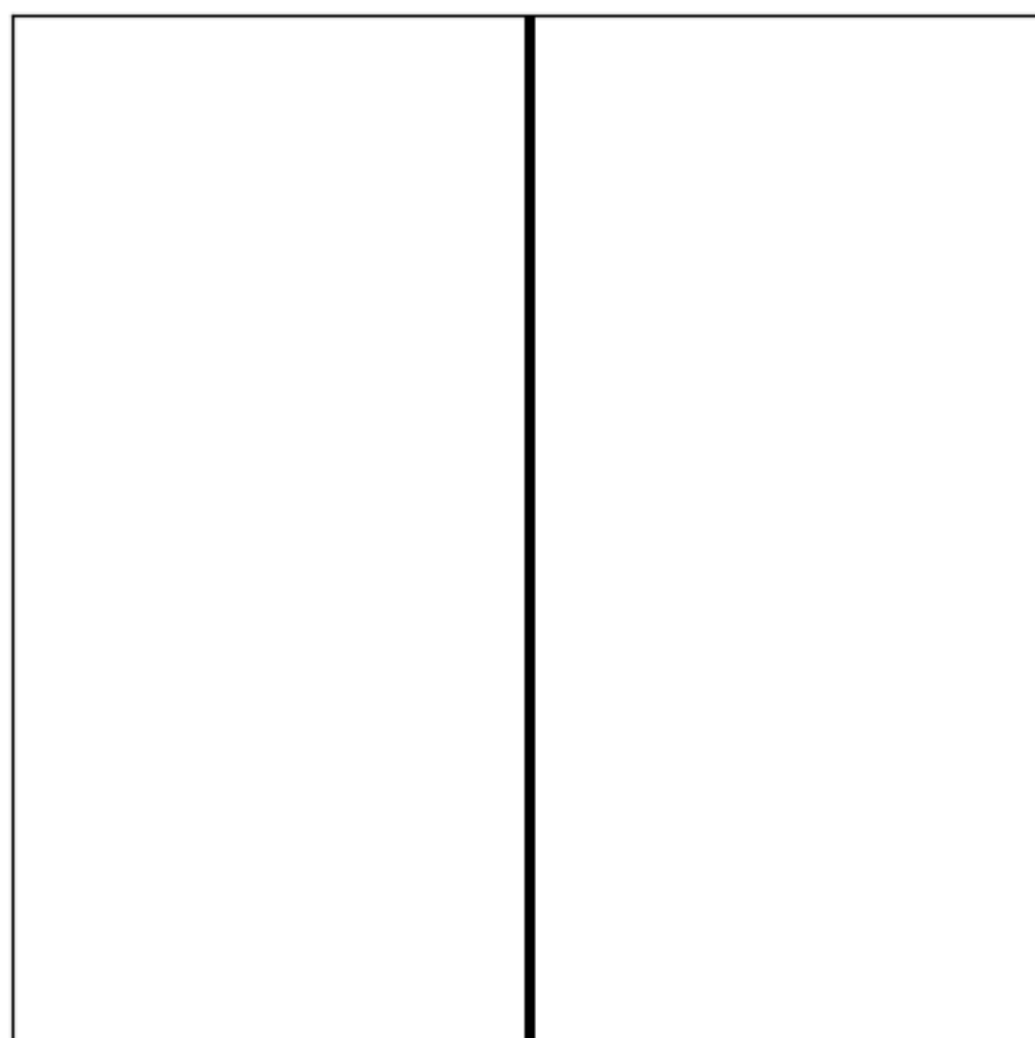
0.5 ECA 228



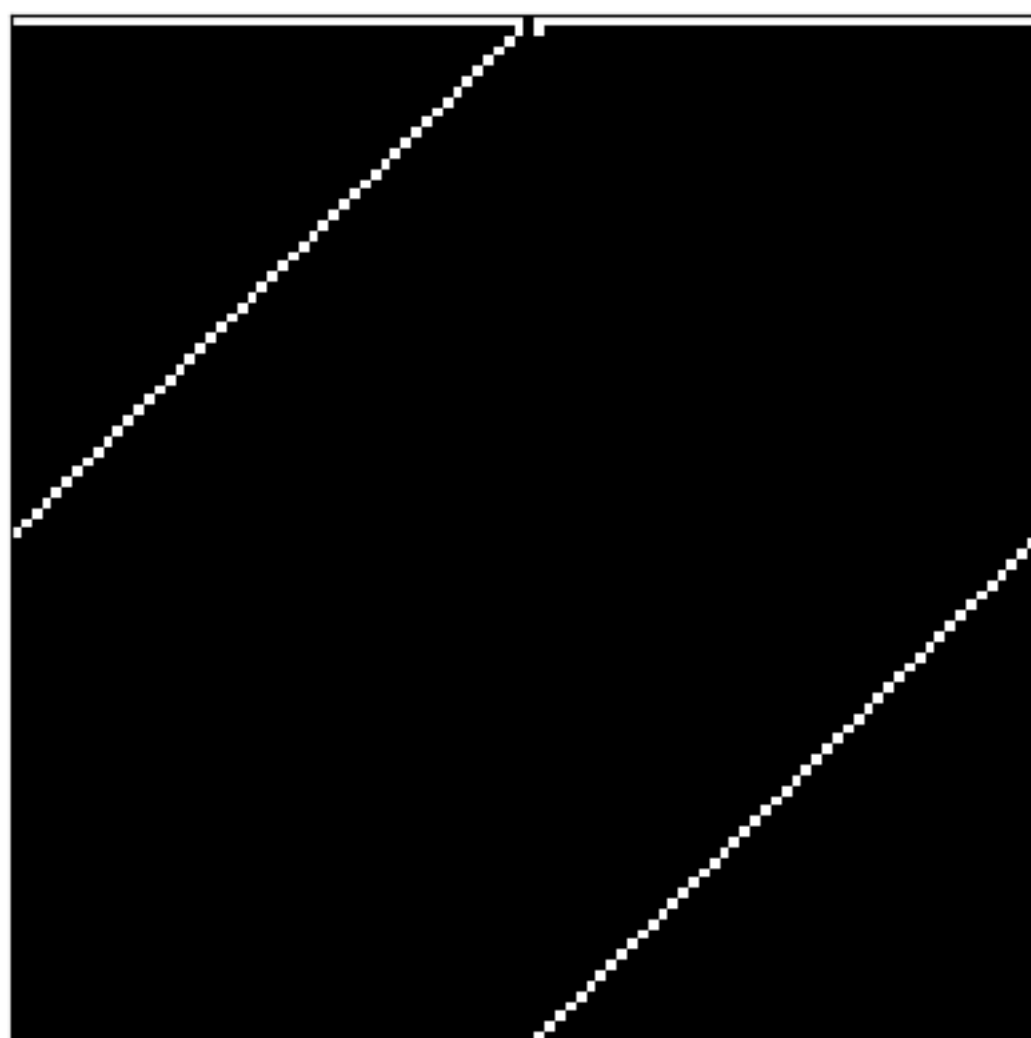
0.25 ECA 220



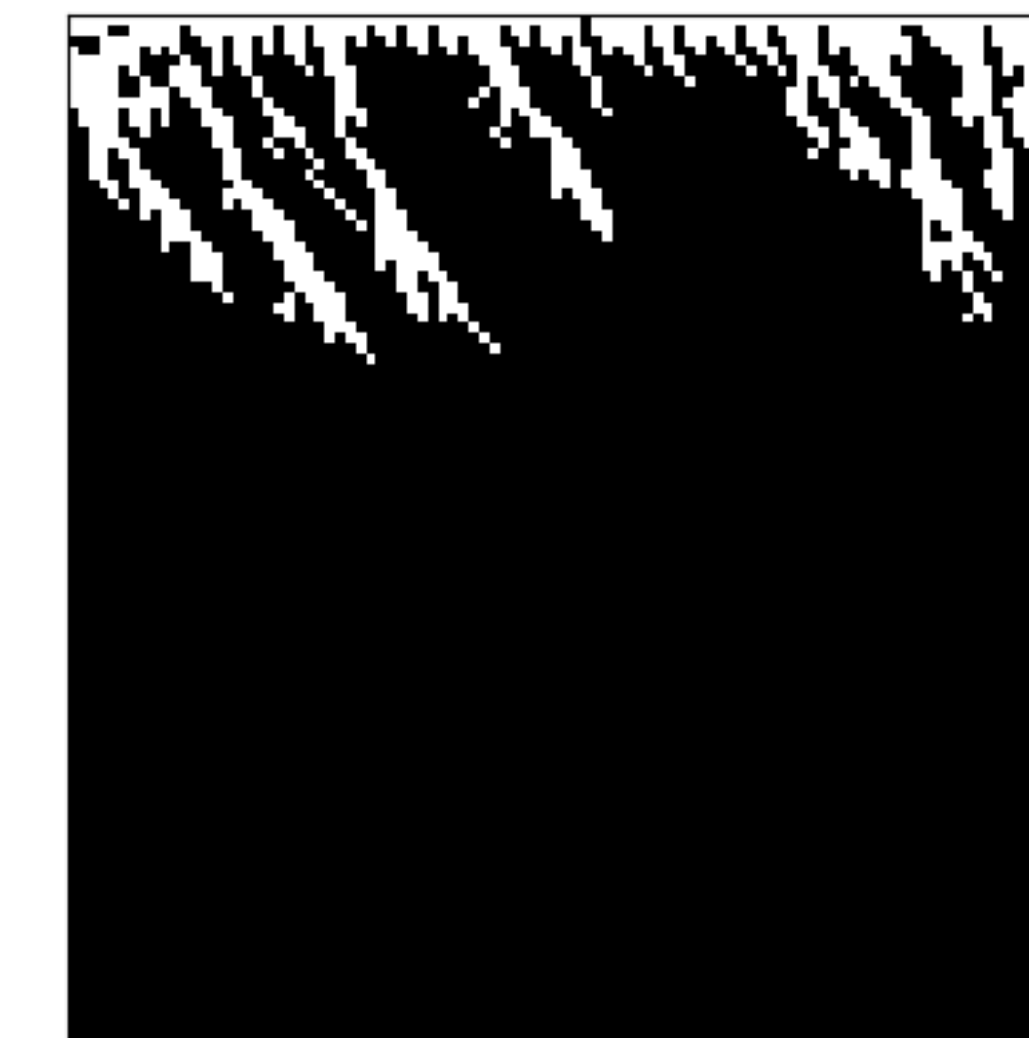
0.15 ECA 253



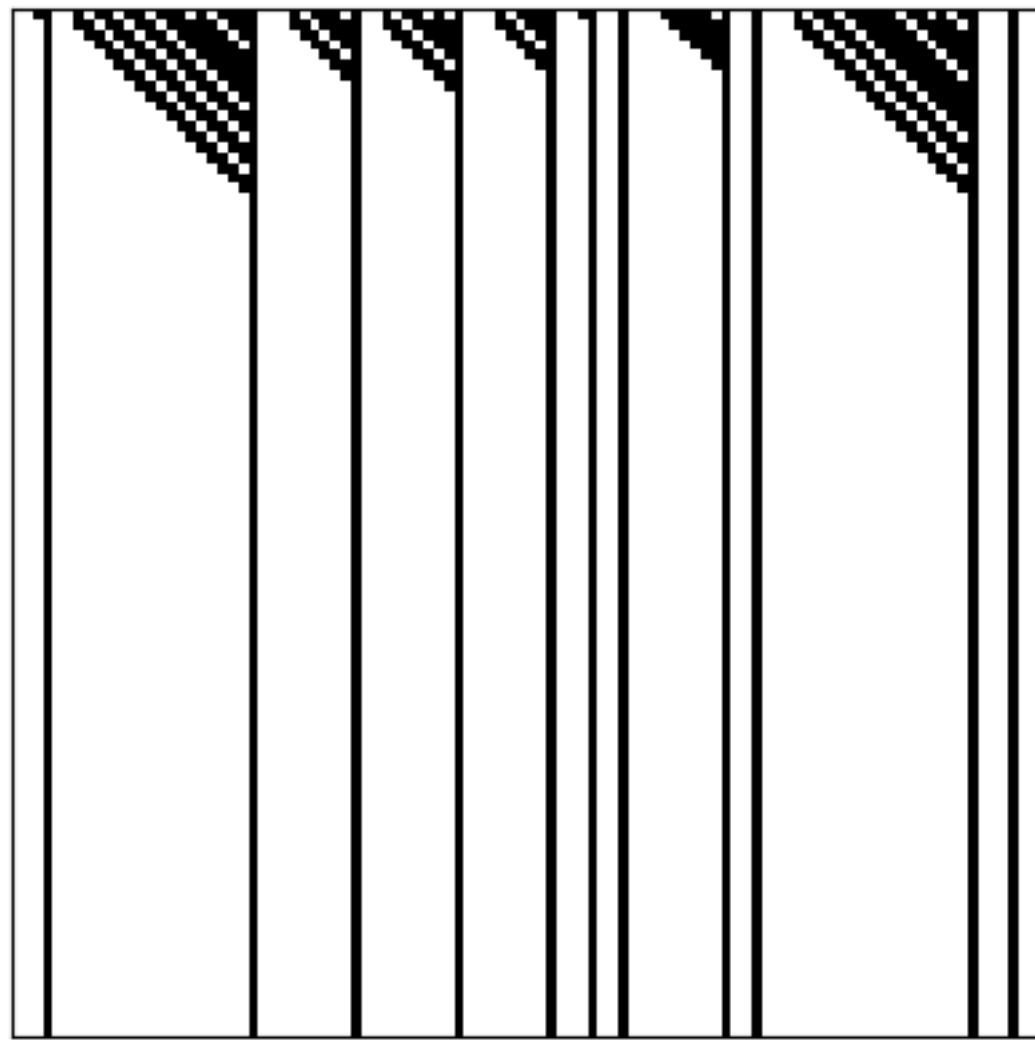
0.05 ECA 172



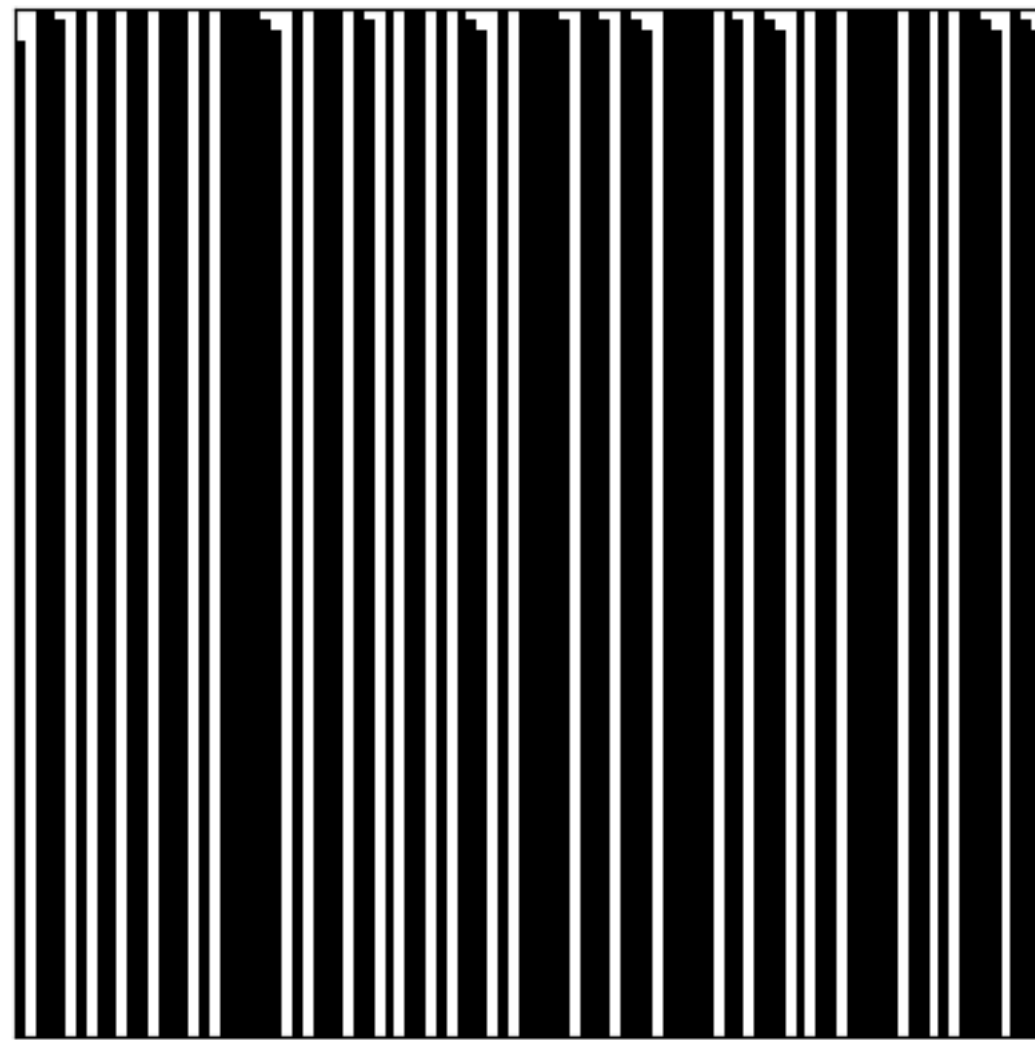
0.05 ECA 173



SCA



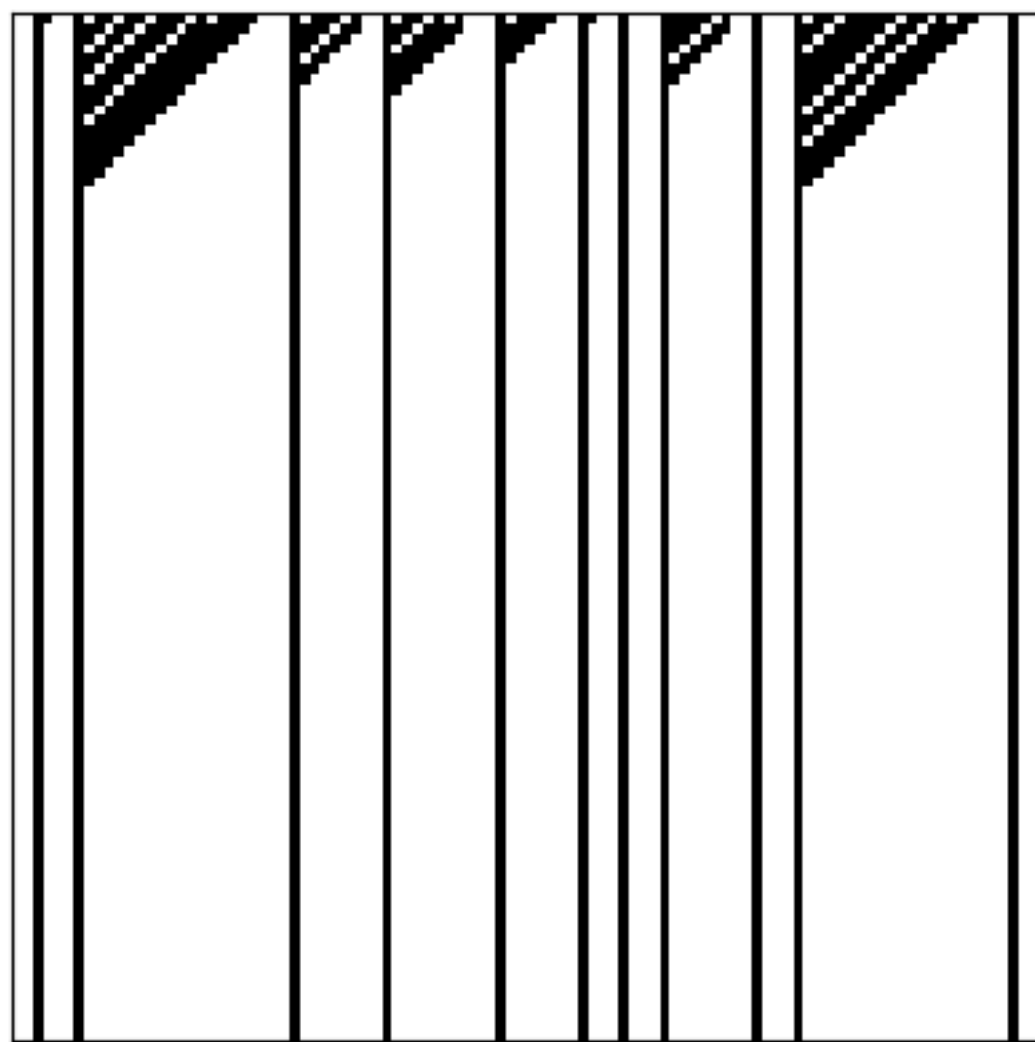
0.5 ECA 228



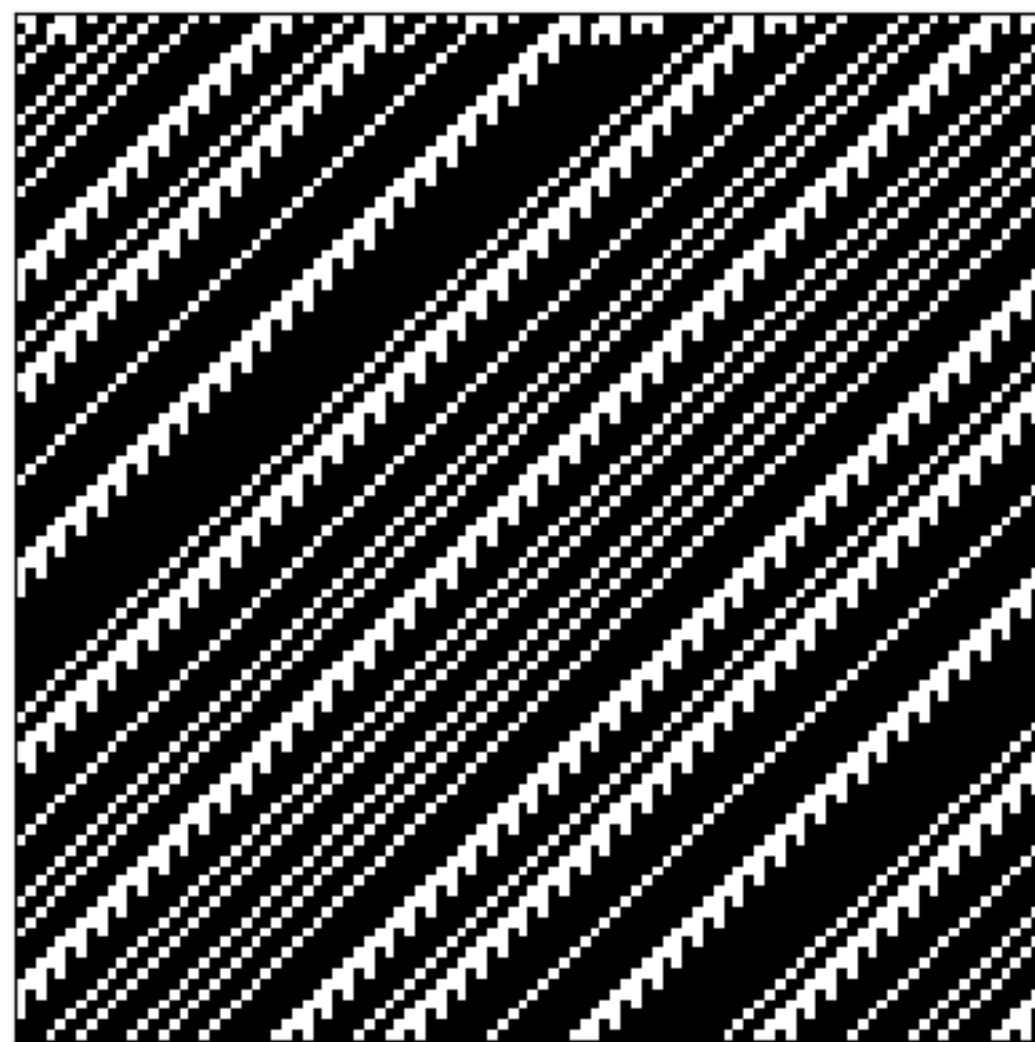
0.25 ECA 220



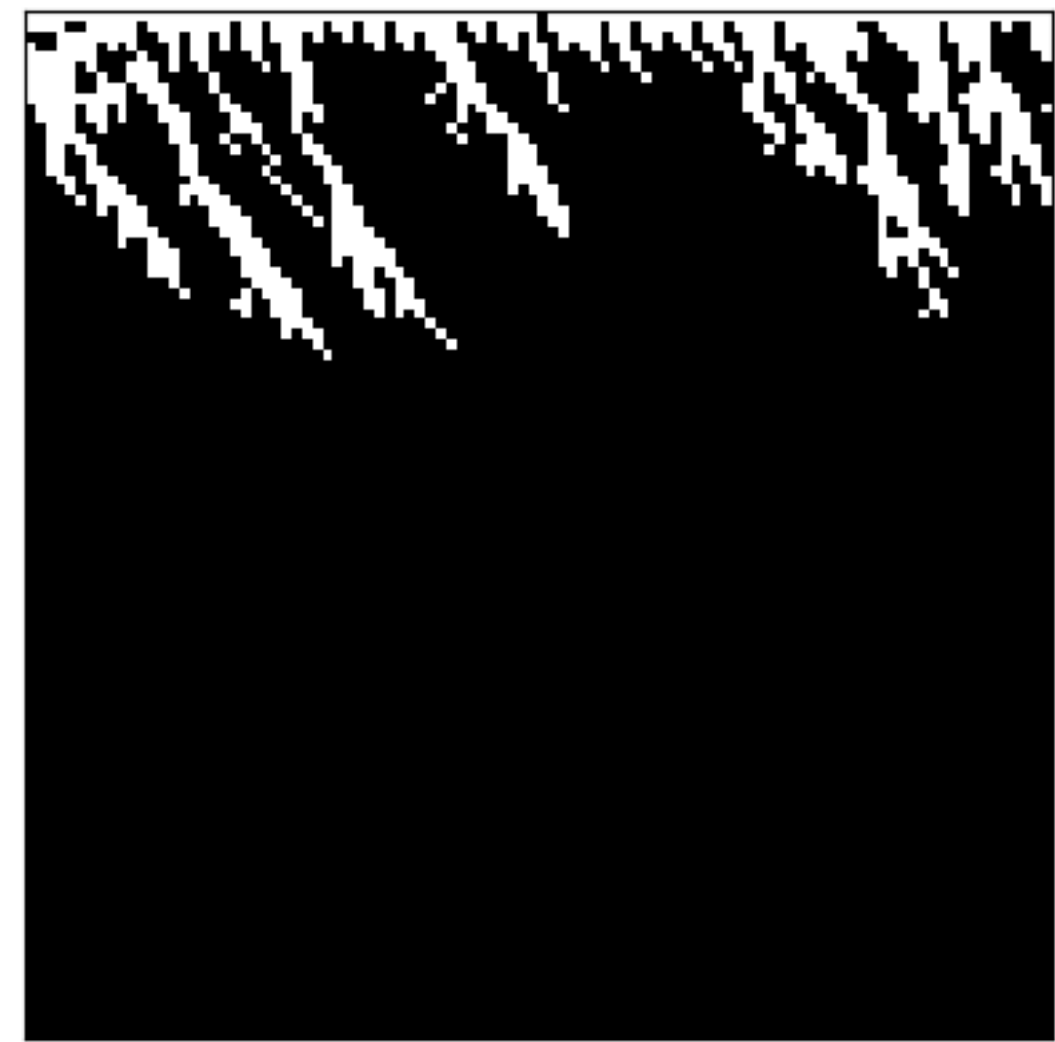
0.15 ECA 253



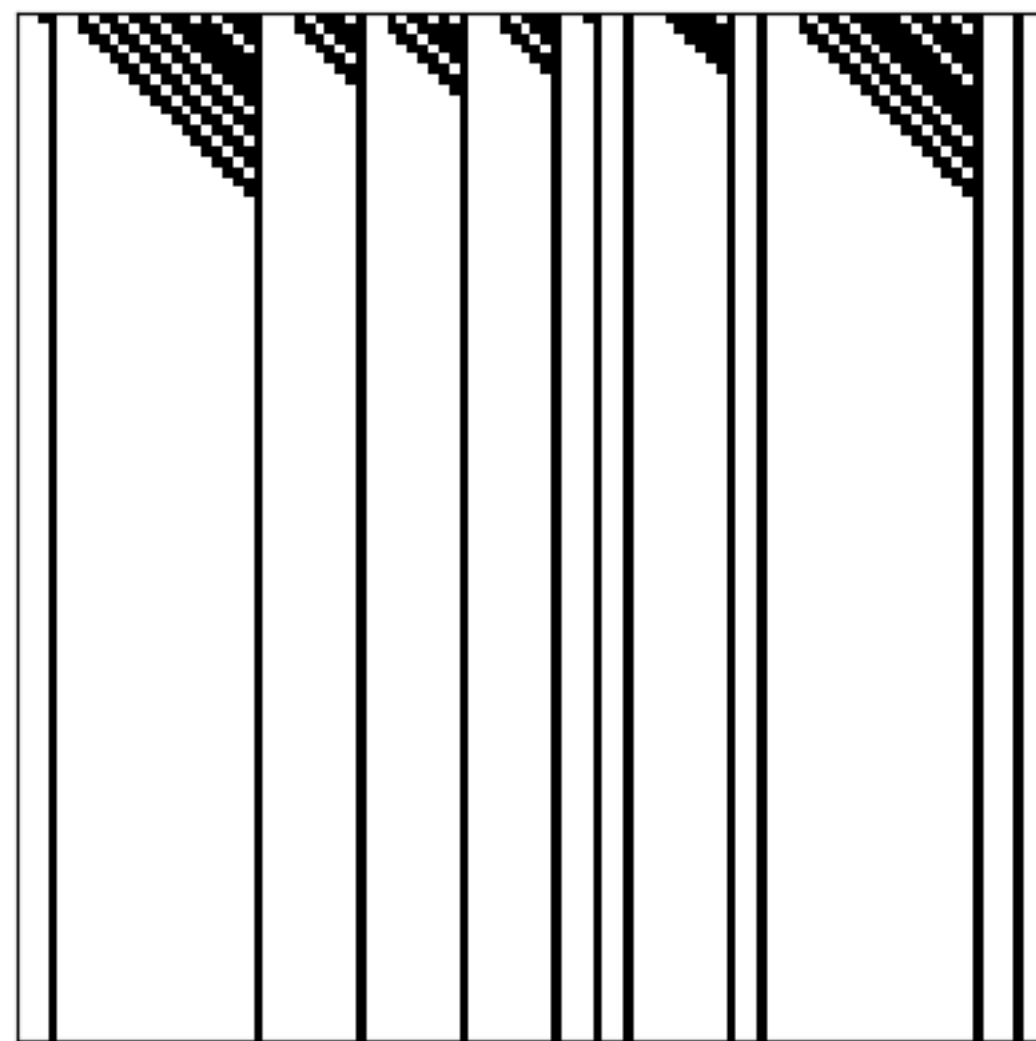
0.05 ECA 172



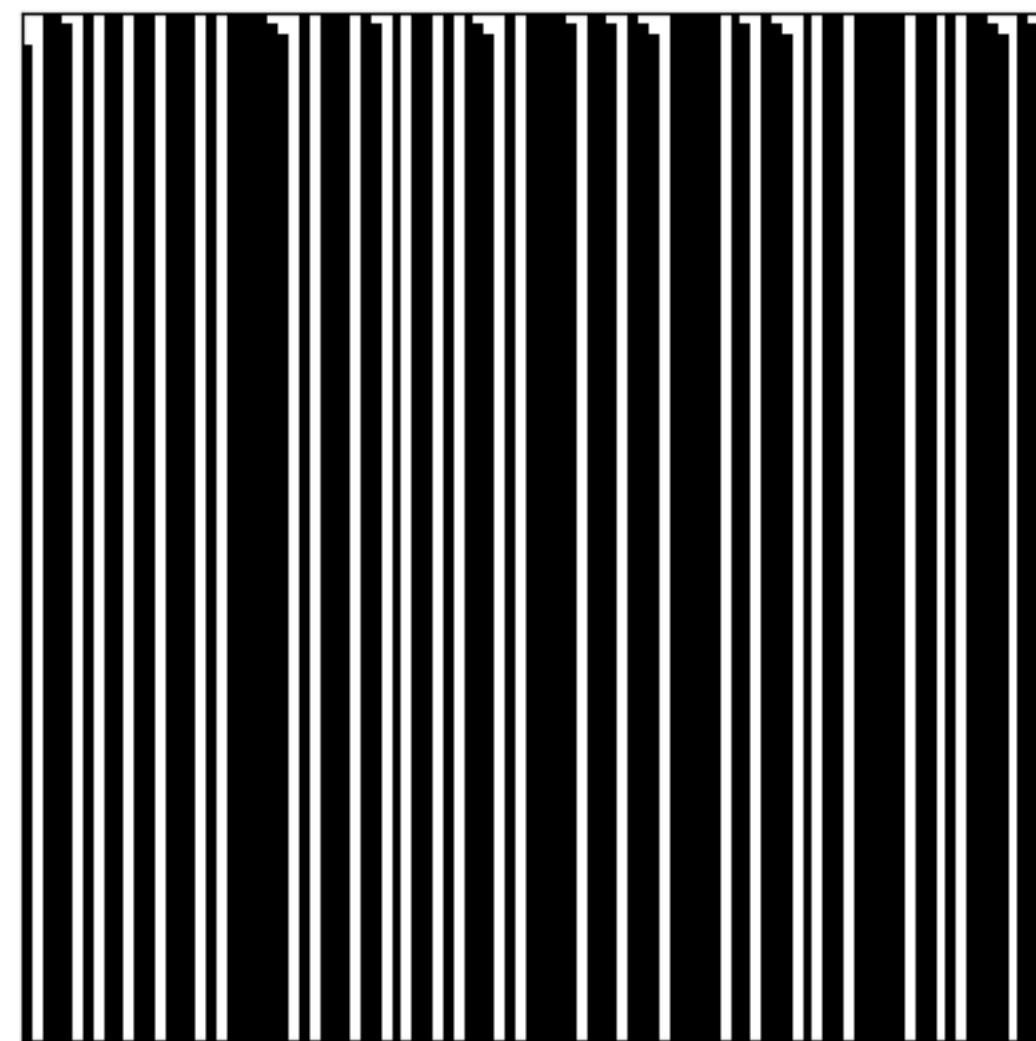
0.05 ECA 173



SCA



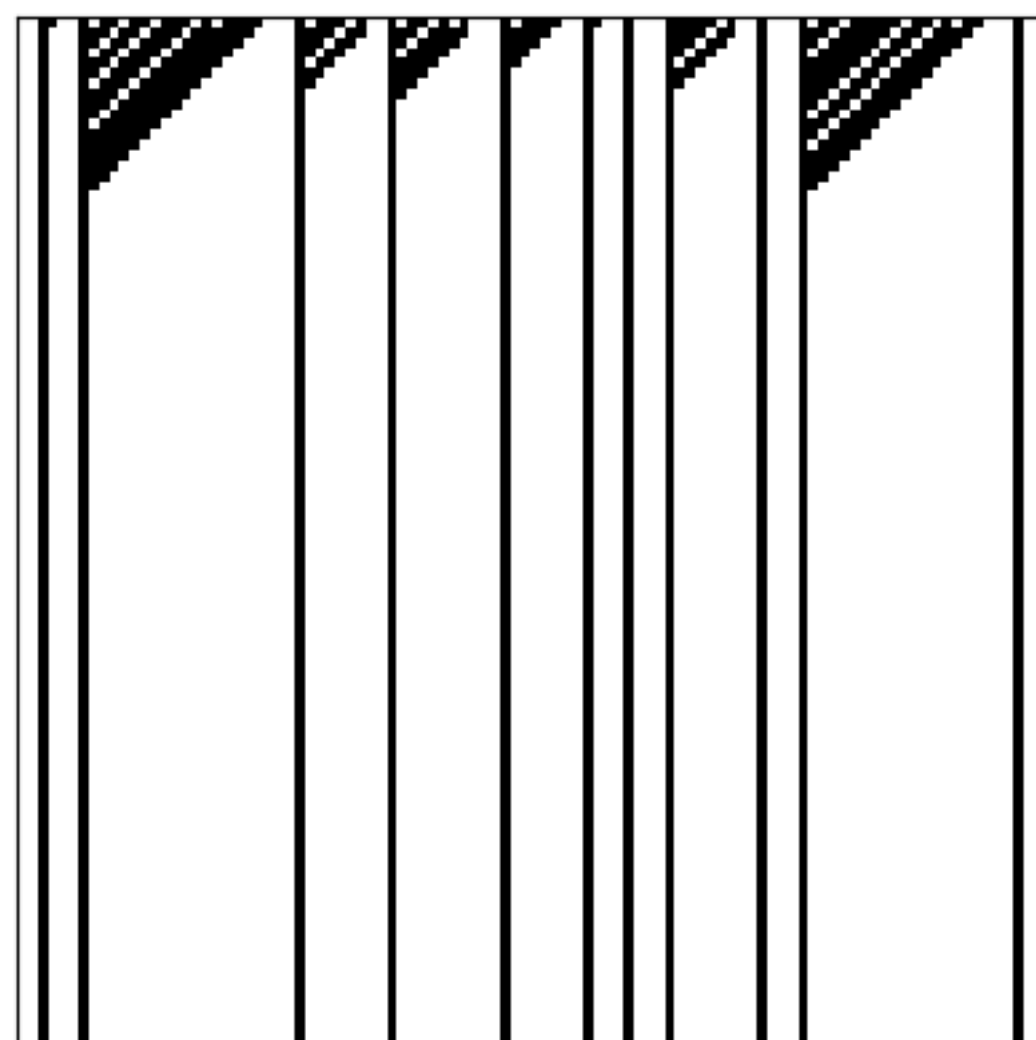
0.5  $\rightarrow$  0.2 ECA 228



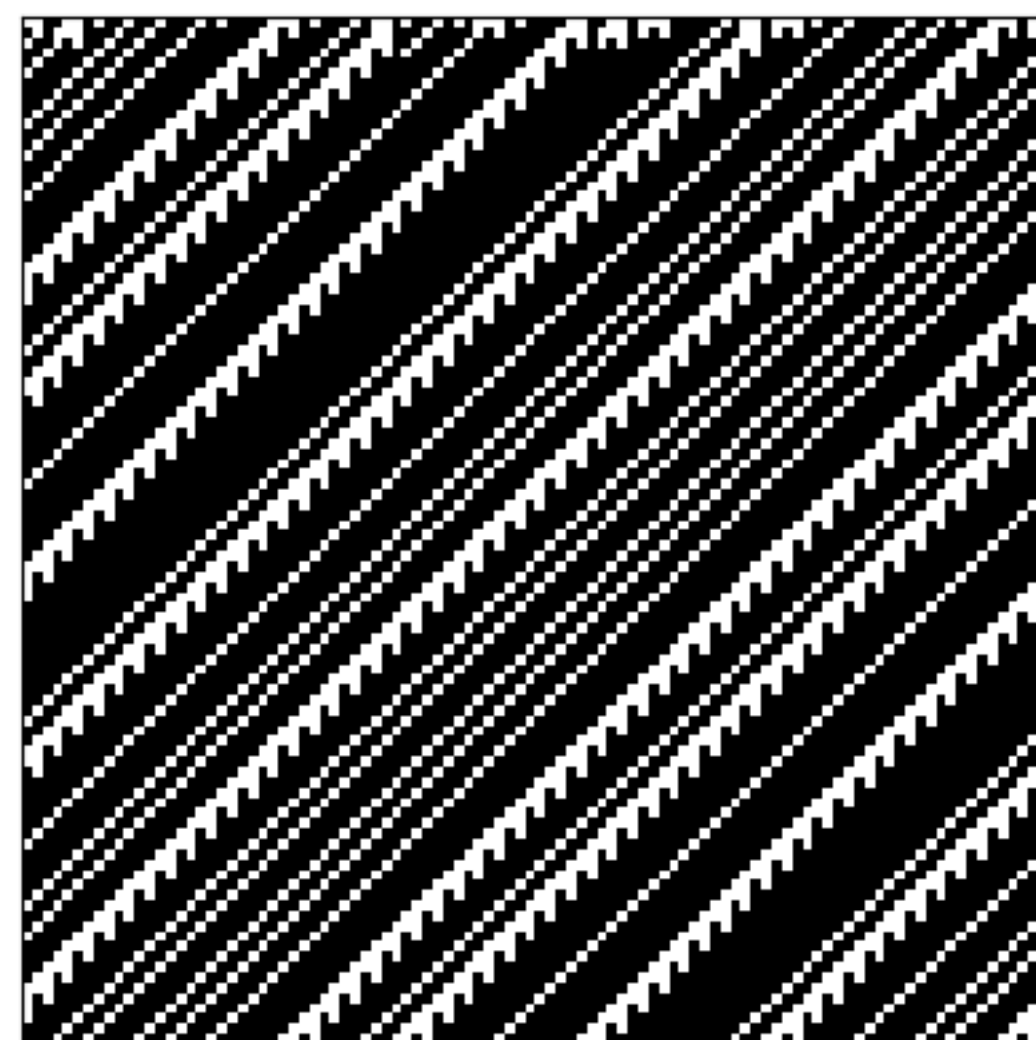
0.25  $\rightarrow$  0.05 ECA 220



0.15  $\rightarrow$  0.05 ECA 253



0.05  $\rightarrow$  0.25 ECA 172



0.05  $\rightarrow$  0.45 ECA 173



SCA

# Co z tego wynika?!

- Każdy pLUT może zostać rozłożony na kombinację skończenie wielu LUTów, w której współczynniki należą do  $[0,1]$  i sumują się do **jedności**.
- Innymi słowy dostajemy listę LUTów (deterministycznych CA) i rozkład prawdopodobieństwa.
- Takie przedstawienie SCAs nazywamy mieszanką stochastyczną (stochastic mixture).
- Każdy SCA można przedstawić jako mieszankę stochastyczną. Każda mieszanka stochastyczna definiuje pewien SCA. Zatem to jedno i to samo pojęcie! **Ale inna interpretacja.**
- W mieszance stochastycznej mamy zbiór deterministycznych reguł CA i w danej chwili czasu dla każdej komórki **niezależnie** losujemy jedną z tych reguł z określonym prawdopodobieństwem i ją stosujemy.

# $\alpha$ -Asynchroniczne Automaty Komórkowe ( $\alpha$ -ACAs)

Bardzo ważny przykład stochastycznej mieszanki

# Szczególna mikstura

- Niech dany będzie pewien ECA o kodzie Wolframa  $k$ , oraz prawdopodobieństwo  $\alpha \in (0,1)$ .
- Rozważmy SCA zdefiniowany jako mieszanka stochastyczna postaci:

$$\alpha \times \text{ECA } k + (1 - \alpha) \times \text{ECA } 204,$$

- To znaczy stosujemy regułę ECA  $k$  z prawdopodobieństwem  $\alpha$ , a z prawdopodobieństwem  $(1 - \alpha)$  stosujemy ECA 204... czyli **nic nie zmieniamy**.
- Innymi słowy, w danej chwili czasu każda komórka podejmuje “decyzję” czy **uaktualnić** swój stan zgodnie z regułą ECA  $k$  (dzieje się to z prawdopodobieństwem  $\alpha$ ) czy pozostawić go **bez zmian**.
- Takie SCA nazywamy  $\alpha$ -**asynchronicznym automatem komórkowym** ( $\alpha$ -ACA).
  - Przy czym zamiast ECA  $k$  możemy wybrać dowolny deterministyczny CA.



# A guided tour of asynchronous cellular automata

Nazim Fatès [nazim.fates@loria.fr](mailto:nazim.fates@loria.fr) Inria Nancy Grand-Est, LORIA UMR 7503  
F-54 600, Villers-lès-Nancy, France

August 27, 2014

## Abstract

Research on asynchronous cellular automata has received a great amount of attention these last years and has turned to a thriving field. We present a state of the art that covers the various approaches that deal with asynchronism in cellular automata and closely related models.

**Foreword:** This article is the preprint of an article that is to appear in the *Journal of cellular automata*. It is an extended version of the invited paper that appeared in the proceedings of **Automata'13**, 19th International Workshop on Cellular Automata and Discrete Complex Systems, LNCS 8155, 2013, p. 15-30.

<https://arxiv.org/pdf/1406.0792.pdf>



<https://members.loria.fr/nazim.fates/>



# Diploid CA

- Naturalnym rozszerzeniem definicji  $\alpha$ -ACAs są diploidy.
- Diploid to SCA, który można zapisać jako stochastyczna mikstura dokładnie **dwóch**, dowolnych CAs.

Diploid cellular automata: first experiments on  
the random mixtures of two elementary rules

Nazim Fatès  
Inria Nancy Grand Est, LORIA UMR 7503

March 22, 2017

## Abstract

We study a small part of the 8088 diploid cellular automata, that is, the rules that are obtained with a random mixture of two Elementary Cellular Automata (ECA). We try to carry out a first set of numerical simulations to spot the interesting phenomena that occur when one varies the rules that are mixed. We are particularly interested in studying phase transitions and various types of symmetry breaking. As the mathematical analysis of such systems is a difficult task, we use numerical simulations to get some insights into the dynamics of this class of stochastic cellular automata.

<https://inria.hal.science/hal-01493912v1/document>

A co dalej?

# Po co to wszystko?

- Zapoznaliśmy się nieco z SCAs tylko z dwóch powodów:
- Po pierwsze, wiem co nieco na ten temat - ale to nieważny powód :)
- Po drugie, zdecydowana większość modeli stosowanych w praktyce to automaty niedeterministyczne!
- W przyszłości (za kilka tygodni) poznamy kilka przykładów.
- A na dzisiaj to wszystko... jeśli chodzi o wykład.



**Dziękuję bardzo**  
**Witold.Bolt@ug.edu.pl**

