Lecture 17:
Cellular Automata 2 / Discrete-Time Dynamical Systems 5

Teacher:
Gianni A. Di Caro
Being multidimensional iterated maps, CAs are very complex entities, therefore, to study them, let’s make a few reasonably simplifying assumptions:

- **Homogeneous CAs:**
  - Lattice is a regular grid, in 1D or 2D
  - All cell functions $a$ have the same (relatively simple) neighborhood mapping $\mathcal{N}(a)$ → they all have the same number of neighbors defined according to the lattice
  - All cell functions have the same state transition function, $F(a, \mathcal{N}(a))$
  - States are encoded in a few bits, typically, 2 or 3
  - **Synchronous updating**
1D CA

- **Simplest case**: State variables / Cells are **Boolean units**, $S = \{0, 1\}$
- The neighborhood of a cell $a_i$, $\mathcal{N}(a_i)$ corresponds to the *one or two closest neighbors in both left and right directions*
- → Transition function $F$ is a **Boolean function** of $n = 3$ or $n = 5$ arguments

  $$F(a_i, a_{i-1}, a_{i+1}, a_{i-2}, a_{i+2}) = \begin{cases} 
1 & \text{if } a_i + a_{i-1} + a_{i+1} + a_{i-2} + a_{i+2} > 2 \\
0 & \text{otherwise}
\end{cases}$$

- → A 1D Boolean CA with $n$ cells is an $n$-dimensional binary vector $\mathbf{a}(t)$, the **state vector of the CA**, that evolves over time by the iterated application of the map $F : \mathbf{a}(t + 1) = F(\mathbf{a}(t))$
- **State space of the CA**: All possible *configurations* of the vector $\mathbf{a}$
1D Boolean CA, Some Numbers

- \( k = |S| = \text{number of (cell) states} \)
  - \( S = \{0,1\} \rightarrow k = 2 \)

- \( M = \text{number of cells} \rightarrow 2^M \) possible configurations of CA’s state vector,
  - \( M = 100, k = 2 \rightarrow 2^{100} \approx 10^{30} \) !!!!

- \( r = \text{range} = \lfloor |\mathcal{N}(a)|/2 \rfloor \) (assuming a symmetric neighborhood)

- \( k^{2r+1} = k^{|\mathcal{N}|+1} \) possible configurations of neighbor set
  - If \( r = 1, k = 2 \rightarrow 8 \) possible neighbor configurations
  - If \( r = 2, k = 2 \rightarrow 32 \) possible neighbor configurations

- \( k^{k^{2r+1}} = k^{k^{|\mathcal{N}|+1}} = \text{possible evolution functions for the CA} \)
  - If \( r = 1, k = 2 \rightarrow 256 \) possible Boolean evolution functions
  - If \( r = 2, k = 2 \rightarrow 4 \cdot 10^9 \) possible Boolean functions!
- $S = \{0, 1\}, \ r = 1 \rightarrow k = 2, |\mathcal{N}| + 1 = 8, \ 256$ possible Boolean functions

<table>
<thead>
<tr>
<th>111</th>
<th>110</th>
<th>101</th>
<th>100</th>
<th>011</th>
<th>010</th>
<th>001</th>
<th>000</th>
</tr>
</thead>
</table>

Transition function $F$ (rule of the CA)

<table>
<thead>
<tr>
<th>111</th>
<th>110</th>
<th>101</th>
<th>100</th>
<th>011</th>
<th>010</th>
<th>001</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_{111}</td>
<td>a_{110}</td>
<td>a_{101}</td>
<td>a_{100}</td>
<td>a_{011}</td>
<td>a_{010}</td>
<td>a_{001}</td>
<td>a_{000}</td>
</tr>
</tbody>
</table>

Example:
Rule 30: $(00011110) \rightarrow 30$

This is a bit string $\rightarrow$ Decimal number

**Wolfram code**
SOME RULES ...
**Direct problem (Prediction):** Given the function, what’s the behavior?

- Time evolution of the cell vector: *attractor points, oscillatory behaviors, emergence of spatial regularities, dependence from initial conditions, dependence from perturbations, …*

- **Fixed point:** \( \bar{a}^* = F(\bar{a}^*) \). From a time evolution point of view, a fixed point exist if, given that \( \bar{a}(0) = \bar{a}^*, \bar{a}(t) = \bar{a}^*, \forall t \)

- A fixed point is **asymptotically stable** if the above relation holds for all initial conditions in a specified neighborhood of \( \bar{a}(0) \)

- For two close initial conditions: \( \bar{a}(0) \) and \( \bar{a}(0) + \varepsilon \), after \( k \) iterations, the configurations \( F^k(\bar{a}(0)) \) and \( F^k(\bar{a}(0) + \varepsilon) \) can be different.

  **Lyapunov exponent** \( \lambda \): \( \varepsilon e^{k\lambda} = |F^k(\bar{a}(0) + \varepsilon) - F^k(\bar{a}(0))| \)

- For \( k \to \infty, \varepsilon \to 0 \), \( e^{\lambda(\bar{a}(0))} = \text{divergence speed between the two initial conditions.} \lambda > 0 \text{ dependence on initial conditions, } \lambda < 0 \text{ implies convergence.} \)
Linear CA, \( \{0, 1\} \) states, \( F(a_{i-1}, a_i, a_{i+1}) \) is a Boolean function of 3 bits \( (n = 3) \):

<table>
<thead>
<tr>
<th>Neighborhood state:</th>
<th>111</th>
<th>110</th>
<th>101</th>
<th>100</th>
<th>011</th>
<th>010</th>
<th>001</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>New cell state:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>00000000010000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000000111100000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000000110111100000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000110011111011110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00110111100111111110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11000111100111111110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Class 3 cellular automata: overall the evolution presents *regularities*, however, the state sequence generated by the central cell is used as *random generator* in *Mathematica*! (randomness deriving from a purely deterministic process with no external ‘noisy’ inputs)
<table>
<thead>
<tr>
<th>0</th>
<th>32</th>
<th>72</th>
<th>104</th>
<th>128</th>
<th>160</th>
<th>200</th>
<th>232</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>36</th>
<th>76</th>
<th>108</th>
<th>132</th>
<th>164</th>
<th>204</th>
<th>236</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>18</th>
<th>50</th>
<th>90</th>
<th>122</th>
<th>146</th>
<th>178</th>
<th>218</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
<td><img src="image21.png" alt="Image" /></td>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>22</th>
<th>54</th>
<th>94</th>
<th>126</th>
<th>150</th>
<th>182</th>
<th>222</th>
<th>254</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image25.png" alt="Image" /></td>
<td><img src="image26.png" alt="Image" /></td>
<td><img src="image27.png" alt="Image" /></td>
<td><img src="image28.png" alt="Image" /></td>
<td><img src="image29.png" alt="Image" /></td>
<td><img src="image30.png" alt="Image" /></td>
<td><img src="image31.png" alt="Image" /></td>
<td><img src="image32.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Class 1: After a few steps the system reaches a homogeneous configuration independent from initial conditions.

Class 1 - Constant

- Degenerate, single color, homogenous
- Nothing really interesting or surprising, all cells are either on/off

Rule 168
Rule 250
Class 2: After a few steps these CAs show simple spatial-temporal configurations made of separate regions which are either constant or periodic ($\ll 2^M$). The general structure of the arising configurations is relatively independent from initial conditions.

Class 2 – Repeats, Local Structures

- Periodic structures, nested patterns, e.g. rules 90, 108, 170

The direction and location of the lines depend on the initial conditions, but the structural fact that we will have lines in a certain direction is independent from initial conditions.
Class 3: For certain subsets of initial conditions they show a chaotic behavior (no periodic structures). In many cases for all but one cell set to one, a self-similar behavior arises.

Class 3 - Pseudo Random

- Statistical analysis show randomness
Class 4: Strong dependence from initial conditions, highly complex, irregular, and moving structures.

Class 4 - Complex

- Beyond “randomness” (Wolfram)
- Neither regular nor completely random

Universal computation!

All have two Lyapounov exponents measuring the propagation of the information on the initial conditions in both directions. \( \lambda = 0 \) for 1 and 2, \( \lambda > 0 \) for 3, and \( \lambda > 0, \ \lambda \rightarrow 0 \) for 4.
Rule 110, starting from a 1 in the center

Neither completely random nor completely periodic $\Rightarrow$ class IV CA
RULE 110: SPACE-TIME SCALES
DEPENDENCE ON THE INITIAL STATE

Dependence ~ Elaboration of initial conditions

No dependence
Trivial elaboration

Structure does not depend but lines do
→ Identification of parameter of structure

Strong dependence
→ Chaotic behaviors

Complex elaboration,
→ Hard to predict
Dependence on initial state

- **Class 1:** Small changes eventually die out, the final state is not affected.
- **Class 2:** Small changes may persist, but effect remains local.
- **Class 3:** Small changes spread out, and eventually regions arbitrary far away are affected.
- **Class 4:** Small changes may or may not spread out via complicated, but sometimes highly regular dynamics.
Two Lyapunov exponents: measuring information propagation on initial conditions along the two directions

Positive exponents, initial information travels far away

Both 0 exponents, information doesn’t travel

Positive exponents, going to zero
The Rule 110: A Universal Computer

- Turing machine
- On/off-states
- Halting state
- Read/Write head
- Bi-directional Movement
- Unlimited tape, finite input string, blank symbol towards left and right infinity
Claimed by Wolfram and proven by Cook:

The rule 110, as a function of its input, has a dynamic equivalent to a universal Turing machine.

This is an important result, because it shows that the definition of a computation via Turing machines is a concept that is deeper than the Turing machine itself. In other words: Two different views of computation, either via a TM or via a (potentially) infinite number of FSA leads to the same set of computable problems!
Each individual has local non-linear interactions with its neighbors ... but in practice every cell depends from the state of all the other cells → the CA has to be considered as a single complex system

- **Neighbors** \( \approx \) individuals that exert an influence, individuals that we trust, close-by people, ...  
- **“Individuals”** can be biological cells, people, molecules or aggregate of molecules, birds, ...  
- **Individuals can have different dynamics, that is, they can belong to different species/classes**  
- CAs have been applied to the study of many different systems composed of multiple individuals/units bound by some spatial relationship: social (e.g., voting, driving), biological (e.g., immune system, cell reproduction), physical (e.g., crystal growth, bubble formation), artificial (e.g., pattern detection, solution of differential equations), ...  

Since (Jen, 1988) several works have pointed out that in order to make it possible for a CA to achieve certain desired results (e.g., in pattern recognition) a minimal size of the neighborhood exists for the problem at hand. **The larger the neighborhood the better, in principle**
Given the behavior, find the rule: this is called the **inverse problem**

- It’s “useful” to let the CA carrying out computations we are interested in (e.g., analogous to setting/learning the weights of the connections in a neural network)

- A widely studied example is the **density/majority/parity problem**: Find a transition rule that, given an initial state of a CA with an odd number of cells, and a finite number $T$ of max iterations to run, will result in an “all zero” state ($\bar{a}(T) = \bar{0}$) if $\bar{a}(0)$ contains a majority of cells at state $0$, or in an “all one” state otherwise.

- The CA becomes a parallel computer that detects the relative density of 0 or 1 symbols in a configuration

- The simplest rule (switch to the same state of the majority of my neighbors) does not always work! Many different rules have been studied. For 2D automata with 149 cells the best recognition rate for large test sets of random initial configurations is $\approx 83\%$.

In general sense, CAs can be used to **detect patterns** in the input (configuration). In this case the CA should converge to the detected pattern. The challenge is to find the right filtering rules and the dimension of the minimal neighborhood that allows to accomplish the task
No single CA can solve the parity problem, but applying a sequence of elementary CAs can do it, for instance the following operator applied to a lattice of length $L$:

$$G = F_{254}^{[L/2]} F_{76}(F_{132}^{[L/2]} F_{222}^{[L/2]})^{[L/2]}$$