

Cellular Automata

Cellular Automata (CA) is a general class of models of dynamical systems:

- a cellular automaton describes a one- or two-dimensional lattice of “sites”.
 - sites will be labeled using indices i, j .

- each “site” can be at a number k of different “states”.

– states will be labeled using integers x_i (if 1D) and x_{ij} (if 2D). Example of 1D CA with $k = 2$:

$$x_i = \begin{cases} 0 & \text{if site } i \text{ is empty,} \\ 1 & \text{if site } i \text{ is occupied.} \end{cases}$$

Example of 2D CA with $k = 3$:

$$x_{i,j} = \begin{cases} 0 & \text{if site } (i, j) \text{ is "susceptible",} \\ 1 & \text{if site } (i, j) \text{ is "infected",} \\ 2 & \text{if site } (i, j) \text{ is "immune".} \end{cases}$$

- states are updated simultaneously at discrete time moments $t = 1, 2, \dots$
- for each site, its state at time $t+1$ is a *deterministic* function of the states of sites in its local “neighborhood” at time t . Examples of “neighborhoods”:
 - 1D: radius $r = 1$ (two nearest neighbors plus itself) and $r = 2$ (four nearest neighbors plus itself):

- 2D: von Newman neighborhood with radius $r = 1$

- 2D: Moore neighborhood with radius $r = 1$

- update rule is homogeneous: all sites use the same rule.

- boundaries are “periodic”, i.e., the lattice is on a circle (if 1D) or a torus (if 2D).

Relation to models already considered in the course

- CA are deterministic (stochastic CA will be considered later in the course)
- Mathematical description of an 1D update rule with $r = 1$:

$$x_i(t + 1) = F(x_{i-1}(t), x_i(t), x_{i+1}(t))$$

- Mathematical description of an update rule with radius $r = 0$:

$$x_i(t + 1) = F(x_i(t))$$

1D CA with $k = 2$ and $r = 1$

Example of an update rule:

Neighborhood state at t	111	110	101	100	011	010	001	000
Site state at $t + 1$	0	1	1	0	1	1	1	0

This rule is completely specified by a sequence of 8 binary numbers 01101110.

In base 10, this sequence correspond to number

$$0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 110$$

Thus, this is Rule 110.

There are $2^8 = 256$ different rules with $k = 2$ and $r = 1$

Question: how many rules are there with arbitrary k and r ?

Some additional notions

- a rule is *legal* if a neighborhood of all 0's maps to state 0 and there is spatial isotropy (for example, 100 and 001 map to the same state). Of the 256 possible rules (with $k = 2, r = 1$), only 32 are legal.

- a rule is *totalistic* if $x_i(t + 1)$ depends only on the sum of $x_i(t)$ of the sites in the neighborhood. For example,

$$x_i(t + 1) = f(x_{i-1}(t) + x_i(t) + x_{i+1}(t))$$

Of the 32 legal rules, only 8 are totalistic.

- a rule is *peripheral* if it does not depend on the state of the site i . For example,

$$x_i(t + 1) = f(x_{i-1}(t), x_{i+1}(t))$$

General strategy of modeling CA

- set parameters
- set initial conditions
- set update rule
- set graphics

1D CA in Matlab

- Open a new M-file *ca_1D*
- Specify parameters *RULE* (the rule number), *cells* (the number of sites), and *T* (the number of iterations). Initially, set *cells* to be a small number, e.g., 3 or 4.
- Specify initial state *init* as a $(1 \times cells)$ -matrix. Set *init* to be a sequence of 0's with a single 1 in the middle.
- Transform *RULE* into a 1×8 binary array *rule* using command *rule=bitget(RULE,8:-1:1)*. Check how it works.
- The system state at time *t* will be held in the *t*-th row of a $(T \times cells)$ -matrix *X*. Copy *init* to the first row of *X*.
- Let *Y* be the *t*-th row of *X*. *Y* describes the state of the system at time *t*. To find the state of the system at time *t* + 1 do the following:
 - Construct a $(3 \times cells)$ matrix *d* with rows:
$$d(1, :) = [Y(end) \quad Y(1 : (end - 1))];$$
$$d(2, :) = Y;$$
$$d(3, :) = [Y(2 : end) \quad Y(1)];$$
 - Convince yourself that the *i*-th column of *d* describes the states of the sites in the neighborhood of site *i*.
 - Convince yourself that command *rule(8-vec*d)* where *vec*=[4 2 1] produces the states at time *t* + 1. For this, check the outcomes of commands *vec*d* and *8-vec*d* first.

- Place the procedure for finding $X(t+1, :)$ inside a *for t=1:T ... end* loop.

- To view the outcome do:

```
z = zeros(size(X));  
image(cat(3,X,z,z));  
axis equal  
axis tight  
xlabel('space')  
ylabel('time')  
title(['Rule ', num2str(RULE)])
```

- Add *clear all* and *close all* at the beginning.
- To run the finished version, use $T = 100$ and $cells = 100$. Check out different *RULES* and notice the resulting patterns.

Four classes of CA (Wolfram 1984)

- Class I: evolve to a fixed homogeneous state
- Class II: evolve to fixed inhomogeneous states or cycles
- Class III: evolve to chaotic or aperiodic behavior
- Class IV: evolve to complex localized structures

Major lesson: very complex behavior can emerge from very simple rules

Q: What are the relative frequencies of different classes? Experiment with different rules to fill the table. Write down the most interesting rules.

Class	number of rules
I	
II	
III	
IV	

What are CA good for?

- provide an alternative simple method for studying connections between microscopic and macroscopic world,
- examine, by virtue of speedy computations, huge numbers of parameter ranges,
- provide a simple testing tool for qualitative predictions such as: “a local mechanism X can generate phenomenon Y”

However, CA models are almost never able to make precise quantitative predictions. They are suitable for modeling processes in which basic laws are hard to identify.

2D CA: Conway's Game of Life

- Each site can be “alive” (1) or “dead” (0).
- There are 8 other sites in the neighborhood (i.e. Moore neighborhood).
- A dead site with exactly 3 alive neighbors becomes alive.
- A living site with 2 or 3 living neighbors remains alive.
- A living site with less than 2 or more than 3 living neighbors dies.

Run "calife.m" to explore the Game of Life.

go to <http://www.tiem.utk.edu/~gavrila/ca1.html>

Check the following 3 web sites and run different CA applets:

<http://psoup.math.wisc.edu/mcell/>

<http://cell-auto.com/links/>

<http://users.libero.it/acnard/ant.html>

Deterministic epidemic CA

Assume each site can be susceptible (S), infectious (I), or immune (R) to a stomach virus. The infection lasts a days, and immunity only lasts b days before the individual becomes susceptible again. Assume an individual is at each grid point. In the simulation, the state of each site can be one of the following:

- $d = 0$ (susceptible individual, S)
- $d = 1, \dots, a$ (infectious individual, I)
- $d = a + 1, \dots, a + b$ (immune individual, R).

Initially, a small proportion p of individuals are infected or immune at different randomly assigned stages. All other individuals are susceptible.

The following update rules apply, where the term "neighbor" applies to the sites to the north, east, south, or west (von Newman neighborhood):

- a susceptible individual becomes infected if at least one neighbor is infected,
- an individual who was infected for a days becomes immune,
- an individual who was immune for b days becomes susceptible.

Matlab code for Deterministic epidemic CA

The matlab code `sir.m` simulates the spreading of the disease. At the end (after you hit Quit), a figure describing the dynamics of S , I and R is produced.

1. Try to understand the code (disregard the GUI part at the beginning before PARAMETERS).

2. Run the simulations at least a couple of times for each of 27 combinations of $a = 2, 4, 8$, $b = 2, 4, 8$ and $p = 0.01, 0.05, 0.10$. Notice the dynamic patterns as observed in the 2D figure and in the figure for S , I and R . Try to give an interpretation/explanation of these patterns.