

Cellular Automaton Methods

One of the first *Cellular Automaton* models was John von Neumann's *self-reproducing automaton* which he constructed in 1948 in an attempt to explain biological reproduction. The model had many interesting features, including the fact that it was equivalent to a universal Turing Machine. The most famous cellular automaton is probably Conway's *Game of Life*.

Basic ingredients of cellular automaton models

Cellular automata model dynamical systems using discrete approximations including

1. Continuous space x, y, z is replaced by a finite number of *cells* fixed in space, usually in a regular array or lattice.
2. Continuous dynamical functions are also approximated by a discrete set of values at each cell site.
3. Continuous time t is made discrete.
4. The dynamical equation of motion is replaced by a *local rule*: at each time step, cell values are given new values which depend on the cell values in a small *local neighborhood*.
5. The cell values are updated *simultaneously* or *synchronously*

Cellular automata can be simulated very efficiently on parallel computing systems because:

- Since cell variables have a finite number of values, integer arithmetic can be used. (Also applies to serial programs of course!)
- Since update rules are local, domain decomposition of cells can be used with ghost-layer communication required at domain boundaries.

One-dimensional Cellular Automaton

A simple 1-d cellular automaton consists of cells arranged in a line. Each cell is assumed to have two values which can be labeled with the binary digits 0 and 1. This is therefore called a *Boolean* cellular automaton.

A simple choice for the neighborhood of a cell is the cell itself and its two nearest neighbors.

The next value of a cell depends on the values of its neighboring cells. Since each neighbor can take 2 values, the total number of values of the neighbors is $2 \times 2 \times 2 = 8$. An example of a local rule is

t:	111	110	101	100	011	010	001	000
t + 1:	0	1	0	1	1	0	1	0

The total number of such rules is $2^8 = 256$, which is the number of different 8-digit binary numbers, i.e., the number of different possibilities on the second line. The above rule is called rule 90 from the decimal representation of the second line

$$01011010 = 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 64 + 16 + 8 + 2 = 90 .$$

Wolfram studied analyzed cellular automaton models like this one and concluded that there are 4 possible types of behavior for such automata:

- **Limit point behavior:** The cell values tend to a unique fixed state independent of the initial state.
- **Limit cycle behavior:** Stable periodic structures emerge.
- **Chaotic behavior:** The time evolution is non-periodic.
- **Complex behavior:** Complex and localized propagating structures are formed.

Conway's Game of Life

The basic features of cellular automata are nicely illustrated by the *Game of Life* automaton, which was invented by the mathematician John Conway in 1970. This can be thought of as a model of an ecological system of creatures living on a 2-D substrate.

- Space is discrete: the cells are taken to form a 2-D square grid.
- Each cell can be in one of only two states:
 - dead (0), or
 - alive (1).

This makes it a *Boolean cellular automaton*.

- The local neighborhood of a cell is taken to be the *Moore neighborhood*:

X	X	X
X	0	X
X	X	X

which consists of the cell itself and 8 surrounding neighbors. Von Neumann's original automaton used the *von Neumann neighborhood*

	X	
X	0	X
	X	

- The cell value is updated by counting the number of live neighbors, which can take values 0, 1, ..., 8:

t	t + 1								
-	-----								
0 ->	0	0	0	1	0	0	0	0	0
	0	1	2	3	4	5	6	7	8
	<= No. of live neighbors								
1 ->	0	0	1	1	0	0	0	0	0

This automaton has many fascinating properties including a variety of *life forms* and the fact that it is capable of universal computation!

Lattice Gas cellular automata model of fluid flow

This suprisingly effective model of 2-D Navier-Stokes equations was published by U. Frisch, B. Hasslacher and Y. Pomeau in *Phys. Rev. Lett.* **56**, 1505 (1986). Their model has the following properties:

- They found that a lattice with *hexagonal symmetry* was required to include a sufficient degree of rotational symmetry necessary for the conservation of angular momentum in a continuous fluid:

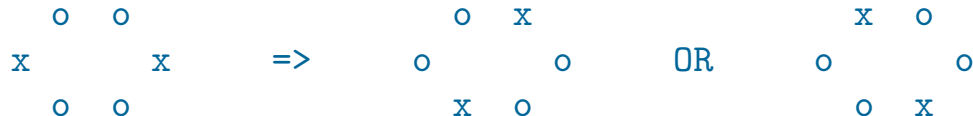
X	X	X	X	X	
	X	0	0	X	X
X	0	X	0	X	
	X	0	0	X	X
X	X	X	X	X	

- To simulate variable density, each cell can have up to six fluid particles of mass $m = 1$. Thus the fluid density ρ can take 7 different values.
- The *free-streaming* (Euler) properties of the model are implemented by the rules
 - Fluid particles have one of 6 possible velocities. The velocities are such that the particle moves to one of

six neighboring cells in one time step.

- The velocities of particles in a particular cell *must* be distinct.
- The *viscous* (Navier-Stokes) properties are implemented by a set of collision rules. If there are 2, 3 or 4 particles at a site after the free-streaming rule has been applied, the particles are made to collide and change their velocities according to rules which conserve momentum (an **x** represents a particle with velocity in that particular direction):

•



•



•



•

$$\begin{array}{ccc}
 \begin{array}{cc} \circ & x \\ x & \\ & x \end{array} & \Rightarrow & \begin{array}{cc} x & x \\ \circ & \circ \\ x & x \end{array} \quad \text{OR} \quad \begin{array}{cc} x & \circ \\ x & \\ \circ & x \end{array}
 \end{array}$$

In the first and fourth rules, there are two equivalent choices. One possibility is to select them at random: however this would make the automata probabilistic or stochastic. To keep the system deterministic the three choices in each case can be cycled in some order: choice of a particular order breaks *handedness* or *chiral* symmetry.