Cellular Automata

モデル化とシミュレーション特論
2023年度前期
佐賀大学理工学研究科　只木進一

Discrete Modeling

- discrete: *eg.* integer values
  - *opp.* continuous
- Observations with
  - discrete time steps
  - discrete space positions
  - discrete space positions as an average
- Discrete internal states

https://github.com/modeling-and-simulation-mc-saga/CA

Pros and Cons of Discrete Modeling

- Motions which can not be described by differential equations
  - describing with evolution rules
  - need validation
- Simulations
  - easy to implement
  - integer operations are faster than floating point ones.
  - no numerical errors

発展方程式
規則

Cellular Automata

- Divide space into cells
- Evolution with discrete time steps
- Evolution rules
  - Next state of a cell decided by states of neighbors
- *automata*
  - plural of *automaton*
  - a machine that moves without human control

Example 2.1: One Dimensional CA

$$s_i(t+1) = (s_{i-1}(t) + s_i(t) + s_{i+1}(t)) \bmod 2$$

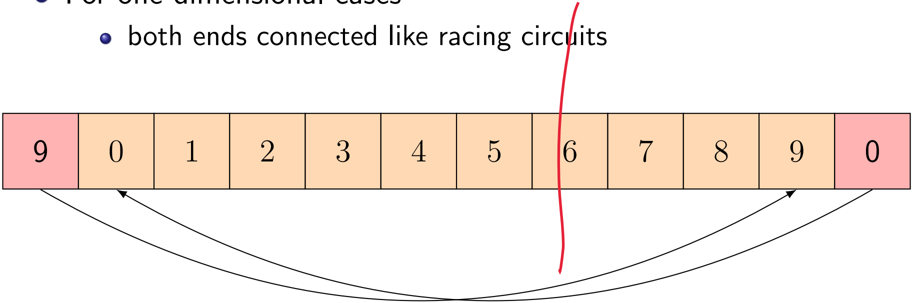
0	0	1	1	1	0	1	1	0	0	t

0	1	0	1	0	0	0	0	1	0	$t+1$

periodic boundaries

Periodic boundary conditions

- For one dimensional cases
    - both ends connected like racing circuits



Example 2.2: Periodic boundaries for N cells

- $s_i : \ 0 \le i < N$
- $s_{-1} = s_{N-1}$
    - $(0 - 1 + N) \bmod N = N - 1$
- $s_N = s_0$
    - $(N - 1 + 1) \bmod N = 0$
- General expression
    - right of i: $(i + 1) \bmod N$
    - left of i: $(i - 1 + N) \bmod N$

i = 0

i = N− 1

if i==0

else if i==N−1

else

One Dimensional CA in General

- Next state depending on states of $2r+1$ neighbors

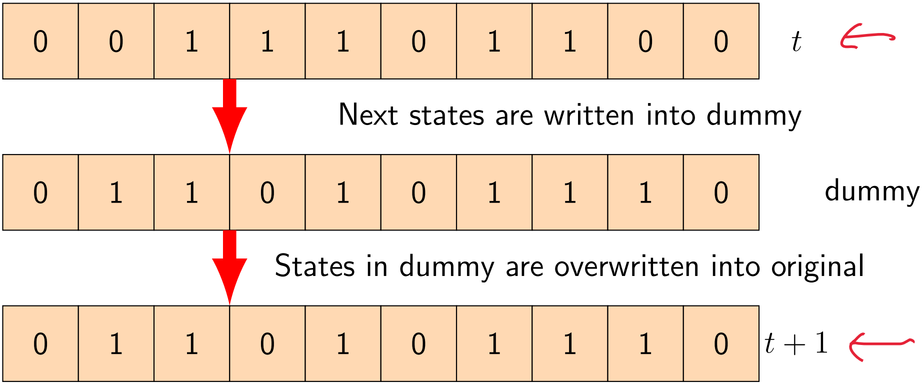
$$s_i(t+1) = F\left(s_{i-r}(t), s_{i-r+1}(t), \cdots s_i(t), \cdots s_{i+r}(t)\right) \tag{3.1}$$

- Apply the same rule F to all cells
- Update states of all cells simultaneously
  - Computers can update cell states sequentially.
  - How to simulate simultaneous (parallel) updates?

General parallel updates

- Prepare dummy cells to store states for the next time steps $t+1$
  - Do not modify the states for time step t
- Overwrite the values from the dummy cells into the original cells.

Example 3.1: $s_i(t) = (s_{i-1}(t) + s_{i+1}(t)) \bmod 2$



Periodic boundaries

Elementary One Dimensional CA

- $s_i = \{0, 1\}$
- $r = 1$

$$s_i(t+1) = F\left(s_{i-1}(t), s_i(t), s_{i+1}(t)\right) \tag{4.1}$$

- The number of input patterns is 3 bit $= 8$
- F is defined as a rule for assigning 0 or 1 for these 8 inputs.
  - $2^8 = 256$ patterns
- Wolfram's elementary CA:
  - Left-right symmetric
  - Stephen Wolfram (1959 –)

Examples

- 0b10111000 $= 184$

input	111	110	101	100	011	010	001	000
output	1	0	1	1	1	0	0	0

- 0b01011010 $= 90$

input	111	110	101	100	011	010	001	000
output	0	1	0	1	1	0	1	0

$(S_{i-1} + S_{i+1}) \bmod 2$

Class design

- `AbstractCA` class
  - Storing values of cells
- `CA` class
  - Wolfram's elementary CA
  - Converting ruleNumber to `ruleMap`
  - `update()` method
- `CA5` class
  - $r = 2$ case

r=1

←

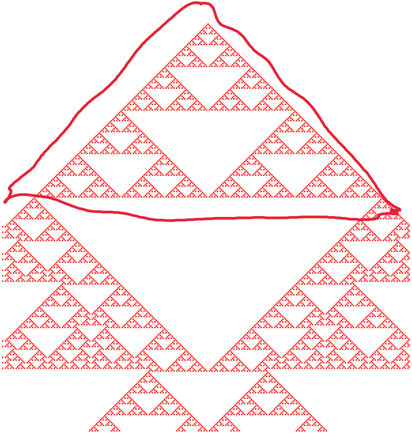
ruleMap array in CA class

- the size of `ruleMap` is 8
- `ruleMap` holds 0 or 1
- Example: rule 184

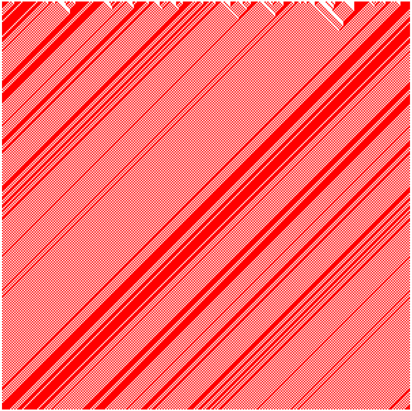
index	0	1	2	3	4	5	6	7
value	0	0	0	1	1	0	0	1

- `mkRuleMap()` method
  - Create `ruleMap` corresponding to the given integer.

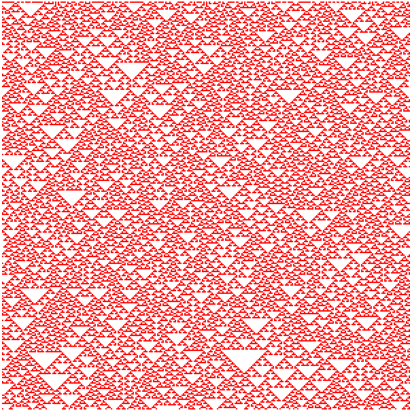
Rule-90



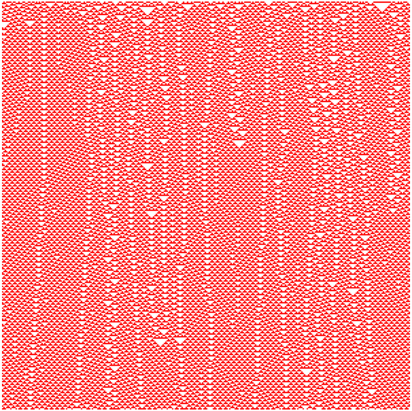
Rule-184



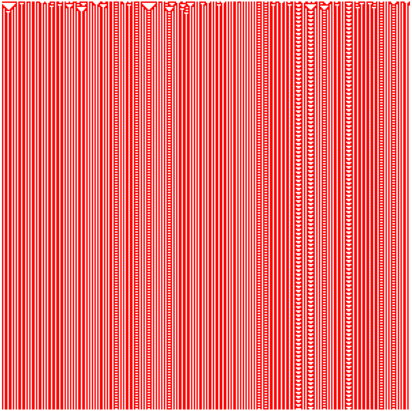
Rule-22



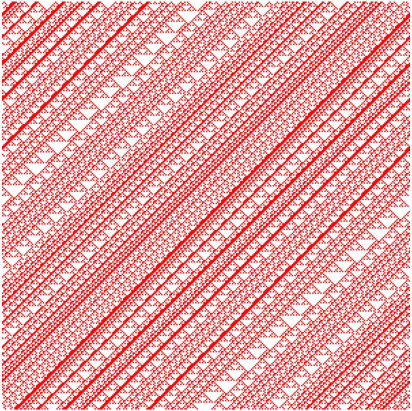
Rule-54



Rule-94

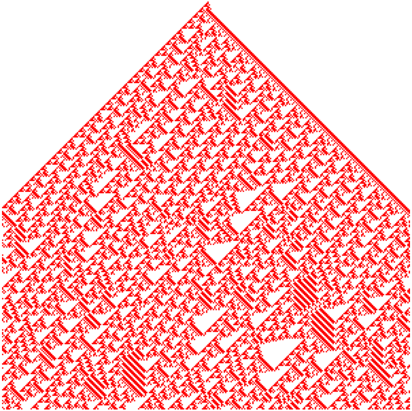


Rule-154

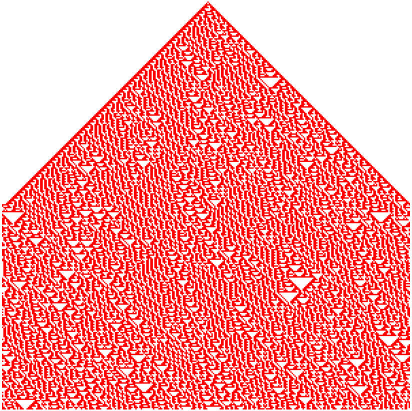


$r = 2$

$$s_i(t+1) = F\left(s_{i-2}(t), s_{i-1}(t), s_i(t), s_{i+1}(t), s_{i+2}(t)\right) \tag{5.1}$$



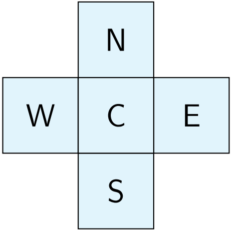
Rule-390097500



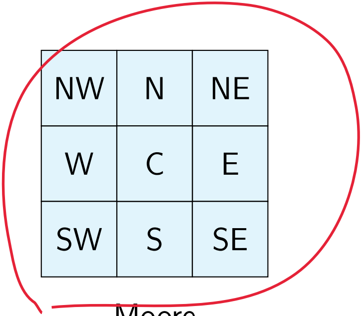
Rule-393410540

Game of Life

- John Horton Conway (1937 – 2020)
- Cell states: active or inactive
- Observe distribution of activities
- Apply Moore neighborhood



Nuemann



Moore

Time Evolution

- Active cells
  - become inactive if the number of active cells in neighborhood n is $n < 2$ or $n \geq 3$.
  - remain active otherwise
- Inactive cells
  - become active if $n = 3$