Overview – Lecture One

- Cellular automata
  - definition
  - modeling environment
- Applications (brief)
- Soil erosion model
  - Synchronous/asynchronous updating
- Prisoners' Dilemma model
  - Synchronous/asynchronous updating
- Summary and bridge to future lectures

John Von Neumann (1903-1957)

- 1920s-30s: math, quantum physics
  1930s-40s: game theory, economics
  1940s-50s: computing (stored program)
  1950s: automata, self-reproduction theory

- tried to prove self-reproduction possible

- kinematic model: a tangible automaton

- non-trivial self-reproducing CA

  - not a mindless robot

  - theory completed by A. Burks

`www.brunel.ac.uk/depts/AI/alife/al-vonau.htm`

Cellular Automata

- an infinite 2D grid of cells
- each cell in one of finite number of states
- system evolves in discrete time steps
- at each step, next state of cell (x, y) determined by:
  1. current state of cell (x, y)
  2. current state of neighboring cells
     - 4 cells: N, S, E, W
     - 8 cells: Moore neighborhood

Game of Life

- John Conway, 1970
- two cell states: {off, on}
- rules for state changes:
  1. if 3 neighbors on, next state is on
  2. if 2 neighbors on, next state is current
  3. otherwise, next state is off
- Interesting configurations:
  - 3-dot blinker
  - glider
  - eater

Other Applications of CA

- earthquakes
- forest fires
- snowflakes
- mollusk shells
- percolation
- fluid dynamics
- complex adaptive systems

Soil Erosion Model

- *Cellular Automata Machines*
  Toffoli and Margolus, MIT Press

- used to study land development and subsequent erosion

- deterministically develop all possible land

- avoid land collapse from erosion

The Landscape

- 2D $X \times Y$ cellular automaton
- each (x, y) cell represents plot of land
- two cell states: *undeveloped*, *developed*
  - undeveloped cell can become developed
  - developed cell cannot revert (no land reclamation)
- two processes can cause development
  1. intentional development by humans
  2. natural erosion from over-development

The Rules

- two deterministic rules model the processes

- stabilize rule: erosion from over-development

  an undeveloped cell is "stable" if there is an undeveloped cell in Moore neighborhood

    somewhere to north (N, NW, or NE),

    somewhere to south (S, SW, or SE),

    somewhere to east (E, NE, SE), **and**

    somewhere to west (W, NW, SW);

  otherwise, the undeveloped cell is "loose" and will erode (become developed)

The Rules (Cont.)

- develop rule: intentional development

  an undeveloped cell is “safe” to develop if

  undeveloped cell to immediate N,

  undeveloped cell to immediate S,

  undeveloped cell to immediate E, **and**

  undeveloped cell to immediate W;

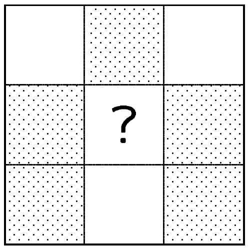
  otherwise, the undeveloped cell is “unsafe” and cannot be developed

- periodic boundary conditions:
  each cell is center of Moore neighborhood

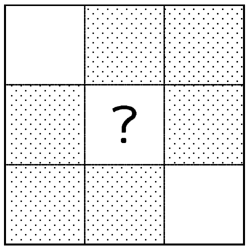
Visualizing the Rules

- white: undeveloped cell
  shaded: developed cell

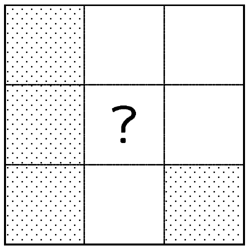
- the stabilize rule:



Stable

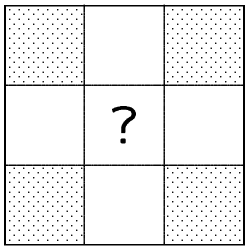


Stable

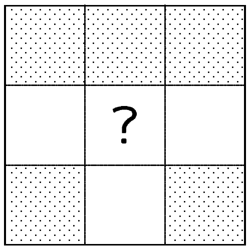


Loose

- the develop rule:



Safe



Unsafe

Evolving the Landscape

- initially, a small proportion p of random cells are developed

- landscape evolves via three-step process:

  1. repeatedly apply stabilize rule until no change in state of any cell
  2. apply develop rule to deterministically develop as many cells as possible
  3. again repeatedly apply stabilize rule until no change in state of any cell

- final stabilization causes landscape to reach
  - stable configuration ($p < 1.0$), **or**
  - complete erosion ($p = 1.0$)

Application of Develop Rule

- synchronous: apply to all cells in parallel
  - change in state not realized by other cells until next time step
  - two copies of landscape required
    1. one copy of current step values
    2. one copy of next step values (after application of develop rule)

- asynchronous: apply in sequence to (x, y) cells selected at random w/o replacement
  - only one copy of landscape required
  - change in state immediately recognized by subsequently selected cells

Algorithm: Synchronous Application

- for an $X \times Y$ landscape

```
while ( /* no change in any cell's state */ )
  for (x = 0; x < X; x++)
    for (y = 0; y < Y; y++)
      Stabilize(x, y);

for (x = 0; x < X; x++)
  for (y = 0; y < Y; y++)
    Develop(x, y);

while ( /* no change in any cell's state */ )
  for (x = 0; x < X; x++)
    for (y = 0; y < Y; y++)
      Stabilize(x, y);
```

Algorithm: Asynchronous Application

- for an $X \times Y$ landscape

```
while ( /* no change in any cell's state */ )
  for (x = 0; x < X; x++)
    for (y = 0; y < Y; y++)
      Stabilize(x, y);

while ( /* unselected cells remain */ )
  /* select (x,y) cell at random WOR */
  Develop(x, y);

while ( /* no change in any cell's state */ )
  for (x = 0; x < X; x++)
    for (y = 0; y < Y; y++)
      Stabilize(x, y);
```

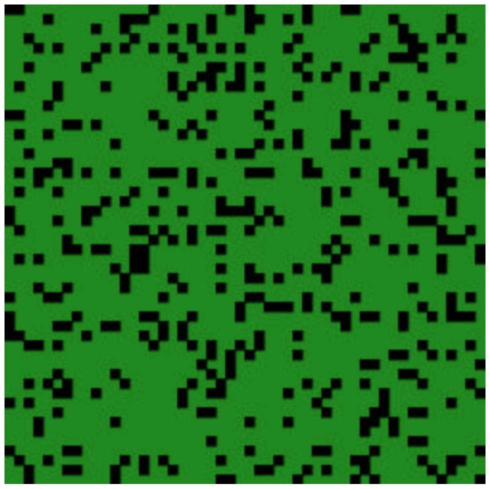
Differences in Output

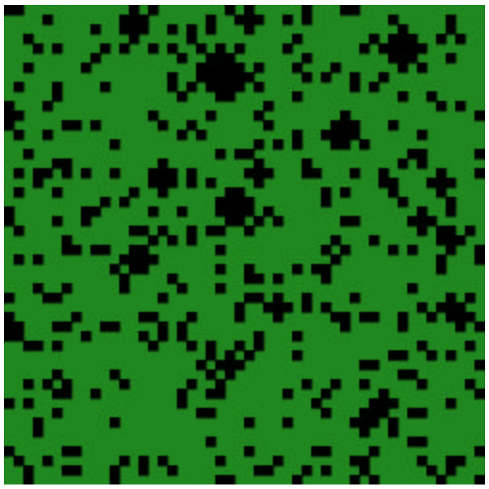
- very different results observed if develop rule applied synchronously/asynchronously

- consider $X \times Y = 50 \times 50$ landscape

(a) $p = 0.17$ cells initially developed

(b) $p = 0.19$ after initial stabilization

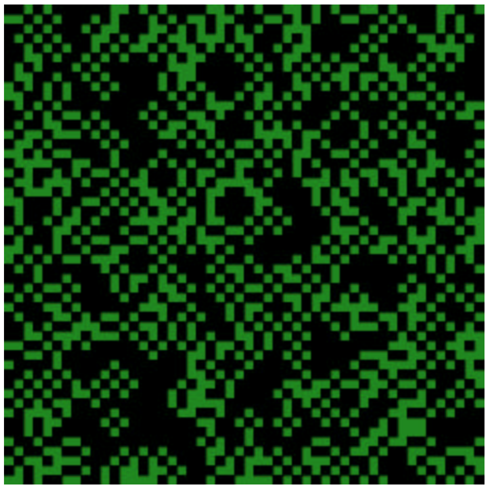

(a)


(b)

Differences in Output (Cont.)

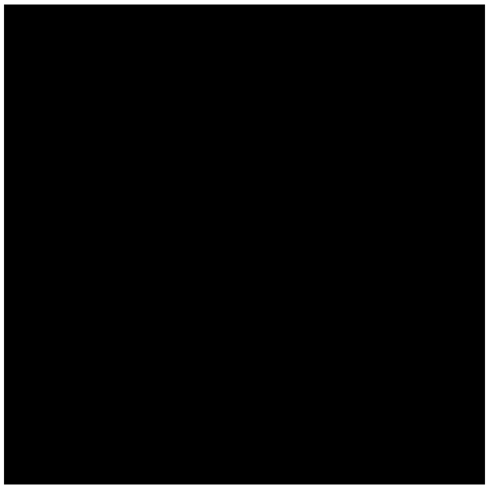
Synchronous

develop



(a) $p = 0.58$

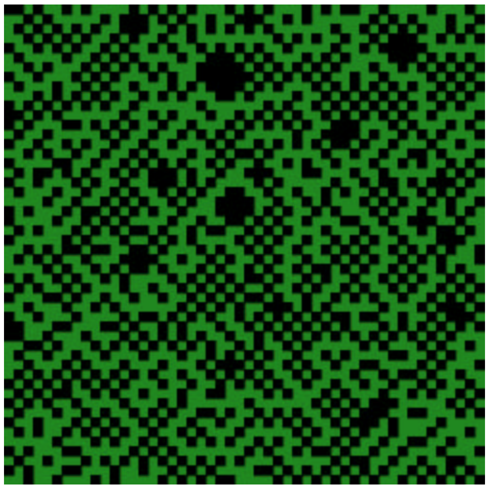
final stabilize



(b) $p = 1.00$

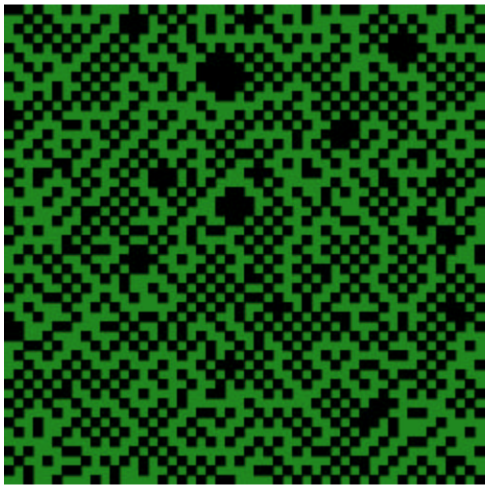
Asynchronous

develop



(c) $p = 0.39$

final stabilize



(d) $p = 0.39$

Soil Erosion Summary

- CA used to model land development
- each cell in one of finite number of states
- evolution controlled by deterministic rules
- different output can be realized by synchronous/asynchronous application of develop rule
- is this seen in other models?

Prisoners' Dilemma Model

- *Evolutionary Games and Spatial Chaos* Nowak and May, *Nature*, 10/1992

- used to study evolution of cooperative behavior (Axelrod, 1984)
  - two arrested for suspected joint crime
  - each interrogated separately
  - confess or deny?

- extend model to 2D landscape

The Landscape

- 2D $X \times Y$ cellular automaton
- two cell states: *cooperator*, *defector*
- The Dilemma:
  - an encounter between neighboring cells
  - payoffs awarded according to cells' states
- payoff p to (x_1, y_1) encountering (x_2, y_2):

state of (x_1, y_1)	state of (x_2, y_2)	p
defector	cooperator	T
cooperator	cooperator	R
defector	defector	P
cooperator	defector	S

where $S \leq P < R < T$

Synchronous Application of the Encounter

- synchronous: apply to all cells in parallel

  1. for each (x, y) cell, sum payoffs from encounters within Moore neighborhood (including self)

  2. state of each (x, y) cell becomes state of cell in neighborhood with largest *total* payoff

  3. this sequence of events (one time step) continues indefinitely or until no change in landscape

- change in state does not affect others until next time step

Algorithm: Synchronous Application

- for an $X \times Y$ landscape

```
while ( /* no change in any cell's state */ ) {

  for (x = 0; x < X; x++)
    for (y = 0; y < Y; y++)
      SumPayoffs(x, y);

  for (x = 0; x < X; x++)
    for (y = 0; y < Y; y++)
      EvaluateState(x, y);
}
```

Asynchronous Application of the Encounter

- asynchronous: apply in sequence to (x, y) cells selected at random w/o replacement

  1. compute initial payoffs for each cell
  2. select an (x, y) cell at random
  3. its state becomes state of cell in neighborhood with largest payoff
  4. any payoffs affected by this state change are recomputed
  5. goto step 2

- one time step is complete when all cells have been selected and updated

- change in state immediately recognized by subsequently selected cells

Algorithm: Asynchronous Application

- for an $X \times Y$ landscape

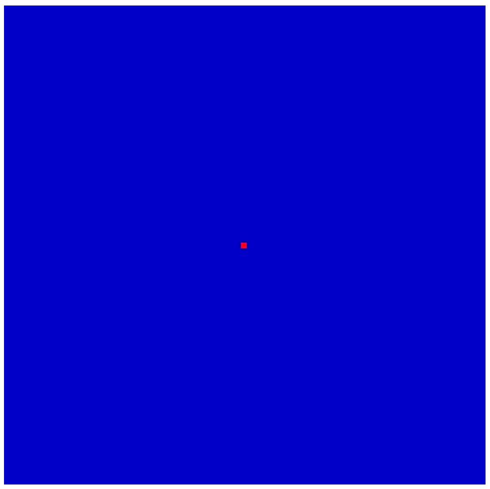
```
while (/* no change in cell's state */) {
  do {
    for (x = 0; x < X; x++)
      for (y = 0; y < Y; y++)
        if (/* any payoff needs recomputing */)
          SumPayoffs(x, y);

    /* select an (x,y) cell at random WOR */
    EvaluateState(x, y);
  } while (/* unselected cells remain */);
}
```

Initializing the Landscape

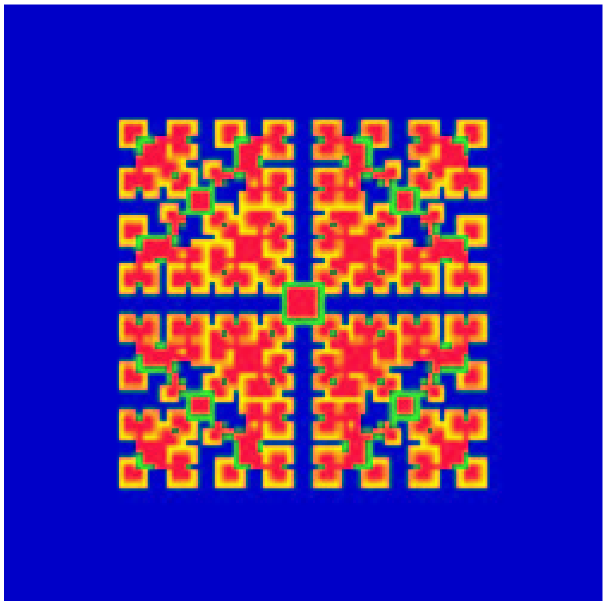
Initial configuration:

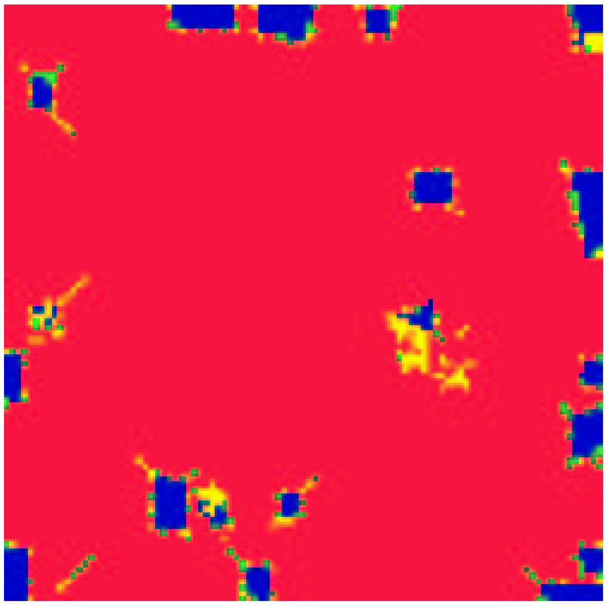
- $X \times Y = 99 \times 99$
- non-periodic boundary conditions
- $S = P = 0, R = 1$ and $T = 1.9$
- sole defector in middle, surrounded by all cooperators



Differences in Output

color	previous state	current state
blue	cooperator	cooperator
red	defector	defector
green	defector	cooperator
yellow	cooperator	defector


(a) sync, $t = 30$


(b) async, $t = 30$

- synchronous: kaleidescope patterns persist

- asynchronous: no obvious patterns

Summary

Soil Erosion and Prisoners' Dilemma models:

- CA used to model cooperative behavior
- each cell in one of finite number of states
- evolution controlled by deterministic rules
- different output for synchronous and asynchronous evolution

Future Lectures

- Extend CA (grid) to a more complex model
- Introduce agents (rational actors) to the landscape
- Define rules to control landscape and agents
- Try to infer global behavior from local rules
- Examine synchronous vs. asynchronous time evolution

Homework

- Implement synchronous Prisoners' Dilemma:

  $X \times Y = 99 \times 99$

  non-periodic boundary conditions

  $S = P = 0$, $R = 1$, and $T = 1.9$

- Provide:
  - number of defectors and landscape pattern for $t = 107$
  - number of defectors and landscape pattern for $t = 219$
  - to capture postscript version of pattern:

    `~bglaws/umsa/draw`