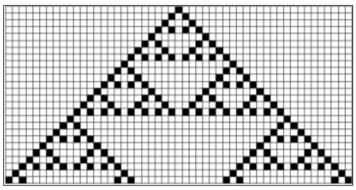
Modeling with Cellular Automata:
1 Dimensional CAs

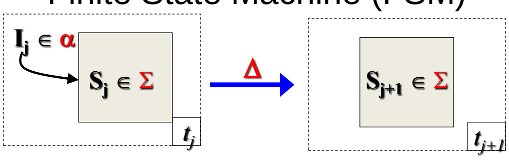


Modeling the world

Model/Variable	**State**	**Space**	**Time**
PDE's	C	C	C
Integro-difference Equations	C	C	D
Coupled ODEs	C	D	C
Interacting Particle Systems	D	D	C
Coupled map lattices, systems of difference equations, LBE models	C	D	D
Cellular Automata and Lattice Gas Automata	D	D	D

Table I: Mathematical and numerical modeling approaches to spatio-temporal processes. PDE: Partial Differential Equation; ODE: Ordinary Differential Equation; LBE: Lattice Boltzmann Equation. For more details see **Sloot & Hoekstra CRC Handbook**

Finite State Machine (FSM)



Define sets α, Σ:

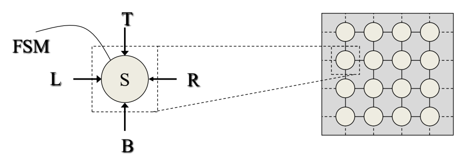
$\alpha = \{a_1, \ldots, a_n\}$ = finite input alphabet

$\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ = finite set of states

And a function:

Δ: $(\alpha, \Sigma) \rightarrow \Sigma$ = transition function

Cellular Automata



- D-dimensional lattice of FSMs.
- Each cell has N neighbors ($\alpha = \Sigma^N$).
- Transition function identical at every cell:

$$\Delta : \underbrace{\Sigma \times \Sigma \times \ldots \times \Sigma}_{N} \rightarrow \Sigma$$

Why study Cellular Automata

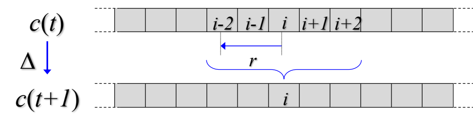
1. Emergence of complex, systemic behavior out of simple, local dynamics
   - Classification of behavior of complex dynamical systems
   - Study the 'microscopic origin' of emergence; how exactly do simple rules lead to complex behavior?
2. As original models of fundamental physics
   - Explore possibility that nature locally and digitally processes information
   - ...Universe is a CA? (https://en.wikipedia.org/wiki/Digital_physics)
3. Discrete Dynamical System Simulators
   - Simple finite dynamical implementations of local conservations laws can reproduce continuum system macroscale behavior
   - Time-reversible rules can simulate physical processes; sometimes combined with conservation laws (E.g. https://en.wikipedia.org/wiki/Lattice_gas_automaton, click video)
   - Glaciers, brains, forest fires, cells, cancer, food chains, ecology, …, universe, ...
4. As powerful computation engines
   - Highly parallel computational implementations of lattice models in physics
   - Could implement a devoted hardware

Applications

- Simulation of Biological Processes
- Simulation of Cancer cells growth
- Predator – Prey Models
- Art (http://wolfrule.com/)
- Simulation of Forest Fires
- Simulations of Egress
- Opinion Spread
- Car traffic
- …many more.. It's a very active area of research.

One-dimensional CAs

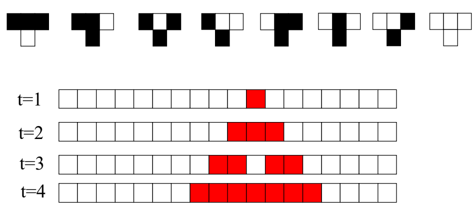
One-dimensional CA



- $c_i(t) \in \Sigma$: value of i^{th} cell at time *t.*
- *r :* range (on left and right)
- Δ : transition (update) function.
- CA states evolve as:
  - $c_i(t) = \Delta(c_{i-r}(t - 1), c_{i-r+1}(t\text{-}1), \dots , c_{i+r-1}(t\text{-}1), c_{i+r}(t\text{-}1))$

Example rule Δ for r = 1

Two states: on | off



Term summary

- $|\Sigma|$: number of possible states of automata – *alphabet*
- Define
  - $k = |\Sigma|$
- Can define neighborhood size
  - N = 2r+1 (for 1D)
- Size of **input** alphabet to an automata:
  - $|\alpha| = k^{2r+1} = k^N$
- Define an **output** of an automata:
  - $\Delta: \Sigma^{2r+1} \text{->} \Sigma$

Defining Δ

- To fully define Δ
  - Define a single output state for all possible input states
- How many possible transition functions for 1D CA with alphabet size ***k*** and range ***r***?
- Rule space for k=2, r =1?

Rule-space for 1D CA

$c_{i-r}(t\text{-}1)$	$c_{i-r+1}(t\text{-}1)$	…	$c_i(t\text{-}1)$	…	$c_{i+r}(t\text{-}1)$	$c_i(t)$
0	0		0		0	$\Delta(0,0,\dots,0)$
0	0		0		1	$\Delta(0,0,\dots,1)$
:	:		:		:	:
k	*k*	…	*k*	…	*k*	$\Delta(k,k,\dots,k)$

- Each cell takes on $k = |\Sigma|$ possible states.
- Δ assigns any of *k* values to each of the k^{2r+1} possible tuples.
- $\Delta_{k,r}$ set of all rules for the CA.
- Total of $|\Delta_{k,r}| = k\text{^}(k\text{^}(2r+1))$ possible rules.
  - For nearest neighbors (*r*=1) and *k* = 2, $|\Delta_{k,r}| = 2^8 = 256$

Wolfram Codes 1D CAs
r = 1, k = 2

- Stephen Wolfram defined a naming system for 1D CAs
  1. List all the Σ^{2r+1} possible state configurations of the neighbourhood of a given cell.
  2. Interpreting each input configuration as a binary number, sort them in descending numerical order.
  3. For each input, list the state which the given cell will have on the next iteration.
  4. Interpret the resulting list of output states again as a binary number, and convert this number to decimal.

Rule Codes for r = 1, k = 2

$\Sigma = \{0, 1\}$

$\alpha = \{111, 110, 101, 100, 011, 010, 001, 000\}$

111	110	101	100	011	010	001	000
↓	↓	↓	↓	↓	↓	↓	↓
$\Delta_{1,1,1}$	$\Delta_{1,1,0}$	$\Delta_{1,0,1}$	$\Delta_{1,0,0}$	$\Delta_{0,1,1}$	$\Delta_{0,1,0}$	$\Delta_{0,0,1}$	$\Delta_{0,0,0}$

- Define the rule code **R**[Δ] as:

R[Δ] = $2^7 \cdot \Delta_{1,1,1} + 2^6 \cdot \Delta_{1,1,0} + 2^5 \cdot \Delta_{1,0,1} + 2^4 \cdot \Delta_{1,0,0} + 2^3 \cdot \Delta_{0,1,1} + 2^2 \cdot \Delta_{0,1,0} + 2^1 \cdot \Delta_{0,0,1} + 2^0 \cdot \Delta_{0,0,0}$

Binary Numbers (Revision)

- Base 10 Numbers:

356 = 3 * 10^2 + 5 * 10^1 + 6 * 10^0

1023 = 1 * 10^3 + 0 * 10^2 + 2 * 10^1 + 3 * 10^0

- Base 2 Numbers:

101 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0
  - (= 5 in decimal)

10110 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0
  - (= 22 in decimal)

Binary Fractions

- Base 10 Numbers:

356.82 = 3 * 10^2 + 5 * 10^1 + 6 * 10^0 + 8 * 10^{-1} + 2 * 10^{-2}

- Base 2 Numbers:
  - 101.11 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 1 * 2^{-1} + 1 * 2^{-2}

1/2 1/4

Rule Codes for r = 1, k = 2

- Define the rule code **R**[Δ] as:

R[Δ] = $2^7 \cdot \Delta_{1,1,1} + 2^6 \cdot \Delta_{1,1,0} + 2^5 \cdot \Delta_{1,0,1} + 2^4 \cdot \Delta_{1,0,0} + 2^3 \cdot \Delta_{0,1,1} + 2^2 \cdot \Delta_{0,1,0} + 2^1 \cdot \Delta_{0,0,1} + 2^0 \cdot \Delta_{0,0,0}$

- i.e., Define the rule code as binary decimal:
  - (Specific standard order of neighbours)

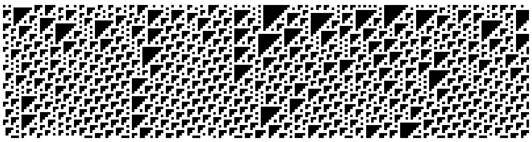
$\Delta_{1,1,1}$	$\Delta_{1,1,0}$	$\Delta_{1,0,1}$	$\Delta_{1,0,0}$	$\Delta_{0,1,1}$	$\Delta_{0,1,0}$	$\Delta_{0,0,1}$	$\Delta_{0,0,0}$	
0	0	0	1	1	1	1	0	**= 30**

Productie Regel 110

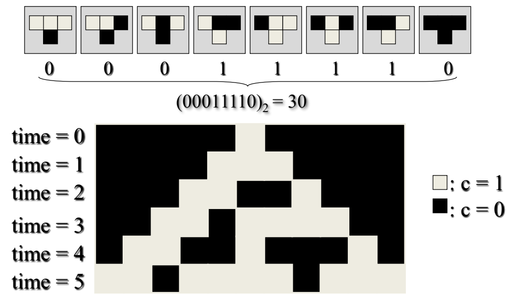
111	110	101	100	011	010	001	000
0	1	1	0	1	1	1	0

t=0	1	0	1	0	0	1	1	0	0
t=1	1	1	1	0	1	1	1	0	1

Time Evolution of 1D Cellular Automata 110

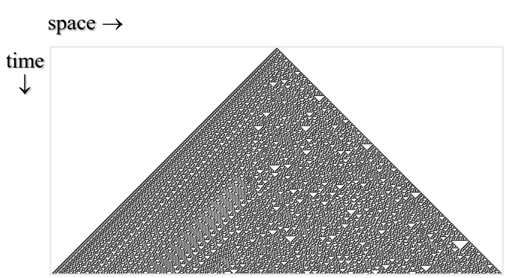


Example: R30



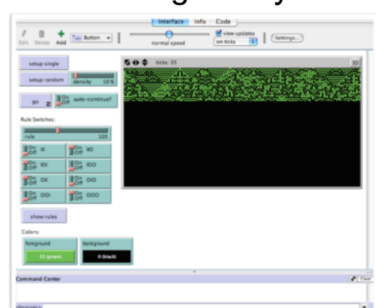
5 generations

Space-time pattern for R30

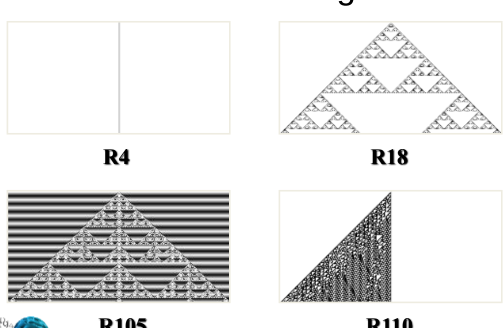


300 generations

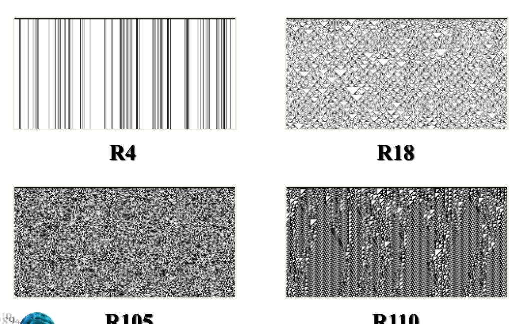
Netlogo… try



Patterns from a single seed



Patterns from a random seed



Complexity

WOLFRAM CLASSIFICATION

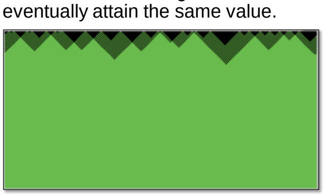
Wolfram's Classes of CA

- Each CA can be classified into one of four types depending on how ***interesting*** its behaviour is.
- Some lead to homogeneous static state, some lead to periodic stable states.
- Others lead to completely chaotic patterns.
- One last class will lead to ***complex*** periodic patterns – very interesting.

http://www.stephenwolfram.com/publications/articles/ca/84-universality/6/text.html

Behavioral Classes of CA

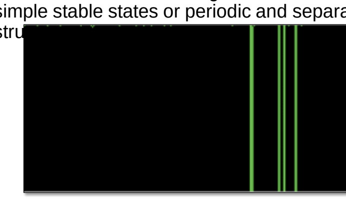
- **Class 1:**
  - Evolution leads to a homogeneous state, in which all cells eventually attain the same value.



  - Examples are rules 0, 32, 160 and 250.

Behavioral Classes of CA

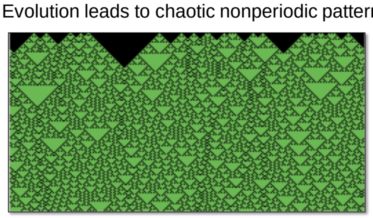
- **Class 2**:
  - Evolution leads to inhomogeneous state: either simple stable states or periodic and separated stru



  - Examples are rules 4, 108, 218 and 232.

Behavioral Classes of CA

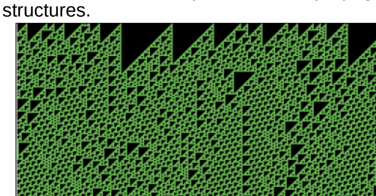
- **Class 3**:
  - Evolution leads to chaotic nonperiodic patterns.



  - Examples are rules 22, 30, 126, 150, 182.

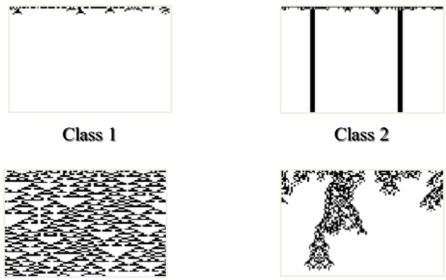
Behavioral Classes of CA

- **Class 4**:
  - Evolution leads to complex, localized propagating structures.



  - Rule 110

Examples



Class 1 Class 2

Class 3 Class 4

Informal definition of classes

Most initial states evolve to…

1. Homogeneous state (all 0s or all 1s)
2. Small cycles and small transients (not forever growing with N)
3. Large cycles. Also *deterministic chaos*. (Growing with N.)
4. Large transients.

Transients and cycles

Evolving rule 102

"Transient path"

001
↓
011
↓
101
↓
110
↓
011
↓
101
↓

Use periodic boundary conditions and remember cell at **t+1** is sum (mod 2) of cell and right neighbour at **t**

Repeats, in a "cycle" every 3 iterations

We say rule 102 has cycle length 3, for system size 3, (for initial state 001)

Try other system sizes…

- How about size 4?
  - 0001 -> 0011 -> 0101 -> 1111 -> 0000 -> 0000
  - Cycle of zero (degenerate)
- Size 5, 6?

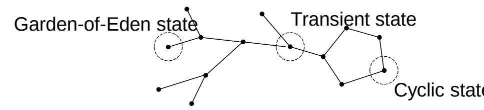
Cycle lengths for Rule 102

System size	Cycle length
1	1
2	1
3	3
4	1
5	15
6	6
7	7
8	1
9	63
10	30

Cycle lengths?

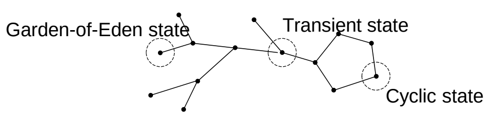
- What does it mean?
- It is possible to prove upper and lower bounds of cycle lengths (not covered here)
- Some rules have no cycle (never repeat), some have short cycle lengths and some long.
  - The length tells us something about the regularity or complexity of the rule.

Properties on a Finite Lattice



- **Garden-of-Eden states**
  - A pattern which has no father pattern and therefore can occur only at generation 0
- **Transient states**
  - States taken on before reaching periodic cycle
- **Cyclic States**

Properties on a Finite Lattice



- There is always one arrow for every state (dot)
  - Thus, if one state has two incoming arrows, then some other state must have zero incoming arrows
  - Then necessarily a transient path exists.
- Class 4 tend to have long transients
- Class 3 tend to have long cycles

Computation and the edge of Chaos

LANGTON PARAMETER

Each class is useful…

- Class 1,2: deterministic computation on input and then 'halts'
- Class 3: pseudo-random numbers, cryptography
- Class 4: emergent, interacting structures; universal computation; digital physics; life?
- …But how to 'find' them? (Especially for larger *k, r*)
  - Can you look at a state transition table and predict which class of behavior it will exhibit? Or design one for a class?

Langton's λ-parameter

- A single parameter to differentiate behavior of CAs
- λ used to specify the rule set Δ of the CA. '*How random is the rule set*'
- PDF available on blackboard – good to read.

Chris G. Langton. 1990. **Computation at the edge of chaos: phase transitions and emergent computation**. In Proceedings of the ninth annual international conference of the Center for Nonlinear Studies on Self-organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks on Emergent computation (CNLS '89).

Langton's λ-parameter

- Pick an arbitrary state $s \in \Sigma$, and call it the quiescent state s_q
- Count the number of rules in Δ that produce this particular quiescent state, and call it *n*
- The other k^N-n transitions must produce the non-quiescent states of Σ - s_q, but may otherwise be chosen at random.

$$\lambda(\Delta) = \frac{k^N - n}{k^N}$$

Langton's λ-parameter

- Define λ(Δ) as the fraction of entries in Δ that map to a non-zero value, i.e., $s_q = 0$

$$\lambda(\Delta) = 1 - \frac{N_0(\Delta)}{k^N}$$

$N_0(\Delta)$ = number neighbour configurations = 0

Langton's λ-parameter

- If **n = k^N**, all rules lead to s_q, λ = 0
- If **n = 0**, no rules lead to s_q, λ = 1

$$\lambda(\Delta) = \frac{k^N - n}{k^N}$$

- All states represented equally:
  n = k^N/k, λ = 1-(1/k)

$(k^N - k^N/k) / k^N = (1 - 1/k) / 1 = 1 - 1/k$

Building Δ from λ

- Two methods to build rule table for a particular value of λ :
  - **Random table** : λ is a bias on the random selection of states from Σ
  - **Table walkthrough** : start with table entirely set to s_q and change some to random according to λ

Random Table for λ

- For each rule r_i in all possible rules k^N
  1. Generate uniform random number *g* in [0, 1]
  2. *if* g>λ set output for r_i to be s_q
  3. *else* set output for r_i set to some random state $s_p \in S$, $p \neq q$

Table-walk-through for λ

- Initialize all k^N rules with output s_q

To increase to λ' from λ

1. select (λ' - λ) . k^N rules ($\Delta_{\lambda'}$) at random and set output all rules $r_i \in \Delta_{\lambda'}$ to be $s_p \in S$, $p \neq q$, p randomly chosen

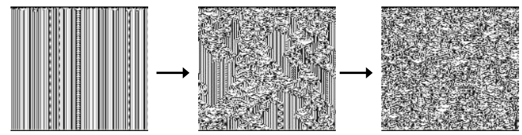
To decrease to λ' from λ

1. select (λ-λ') . k^N rules ($\Delta_{\lambda'}$) at random and set output of rules $r_i \in \Delta_{\lambda'}$ to be s_q

Langton's λ-parameter

- Other parameterizations of CA rule space exist, but the simplicity and single-dimensionality of λ make it attractive
- λ discriminates well between dynamical regimes for "large" values of K and N, but not for small dimensional spaces. For example, λ is only roughly correlated with dynamical behavior for 1-D CAs with K=2 and N=3
- Langton sticks to CAs with K ≥ 4 and N ≥ 5 (r = 2), which results in transition tables of size $4^5 = 1024$ or larger, a total rules space of 4^4^5 = 1.125 * 10^{15}

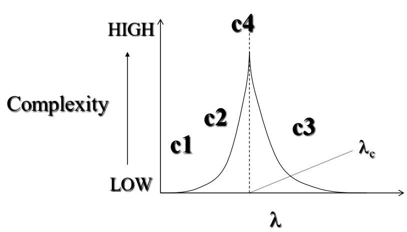
Variation in λ



λ = 10/32 λ = 12/32 λ = 14/32

- Dynamical behavior a function of increasing λ:
  fixed-point → periodic → "complex" → chaotic
- Analogous to Wolfram's classes:
  c1 → c2 → c4 → c3

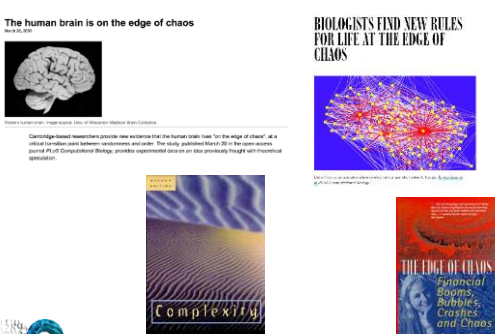
"Edge of chaos"



What is complex?

MEASURING COMPLEXITY

"Edge of chaos"



Measures of complexity

- In Langton's paper he looked at a number of measures – one was transient length:

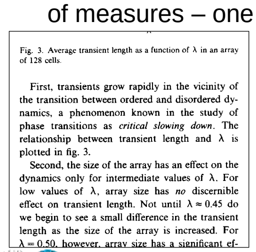


Fig. 3. Average transient length as a function of λ in an array of 128 cells.

First, transients grow rapidly in the vicinity of the transition between ordered and disordered dynamics, a phenomenon known in the study of phase transitions as *critical slowing down*. The relationship between transient length and λ is plotted in fig. 3.

Second, the size of the array has an effect on the dynamics only for intermediate values of λ. For low values of λ, array size has *no* discernible effect on transient length. Not until λ ≈ 0.45 do we begin to see a small difference in the transient length as the size of the array is increased. For λ = 0.50, however, array size has a significant ef-

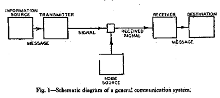
We don't cover this in lecture, but do read:

Lindgren, Kristian, and Mats G. Nordahl. "Complexity measures and cellular automata." Complex Systems 2.4 (1988): 409-440.

https://www.complex-systems.com/pdf/02-4-2.pdf

Shannon entropy

- Information Theory:
  - (Efficiently) **store** information
  - (Efficiently) **communicate** information
- C.E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, July, October, 1948



- how to convert efficiently a message into a signal (transmitter)
- how to decipher efficiently the signal back into a message (receiver)
- how to cope with noisy environments which alter the signal.

Shannon entropy

- Example: N coin flips

x N

- Represent a binary string… 1=H, 0=T

$b_1, b_2, \ldots, b_N$

- We want to communicate the outcome (binary string length N) to someone. What is the minimum of bits we can transmit so that we can reconstruct the message?

Shannon entropy

- Say: N = 1000
- Write down 01000100…011 representing result as **bits***
- If **N =** $10^{9?}$ Is there a more efficient way? Perhaps not…
  - Each event is independent
  - Two equally likely outcomes
  - So really need to provide information for every event
- If you're lucky and get all heads…
  - can say we got 10^9 heads

***bit** = binary digit, coined down in 1948 by Shannon (originally Tukey in '37)

Shannon entropy

- What if coin is biassed? $p_0 = 1/3$ (Tail) and $p_1 = 2/3$ (Head)
  - Most likely that our string has twice as many 1's as 0's
- Consider 2 subsequent tosses (could look a 8 x 3-tuples also):

$p(b_i b_{i+1} = 00) = \frac{1}{3} \times \frac{1}{3} = \frac{1}{9}$ **a=00**

$p(b_i b_{i+1} = 01) = \frac{1}{3} \times \frac{2}{3} = \frac{2}{9}$ **b=01**

$p(b_i b_{i+1} = 10) = \frac{2}{3} \times \frac{1}{3} = \frac{2}{9}$ **c=10**

$p(b_i b_{i+1} = 11) = \frac{2}{3} \times \frac{2}{3} = \frac{4}{9}$ **d=11**

- Rewrite in a,b,c,d form...
- **110010110111 ➔ dacdbd**
- No real gain… 500 x 2 bits

Shannon entropy

- Intuition: give likely outcomes **short** codes, unlikely ones **long** codes
  - d= 0, c =10, b=110, a=111 (What is this encoding?)
  - **1100101101111111 ➔ dacdbddd ➔ 0111100110100**
  - 16 bits ➔ 13 bits
- On average we need:

$$\frac{N}{2} \times (1p_d + 2 \times p_c + 3 \times p_b + 3 \times p_a) = \frac{N}{2} \times \left(\frac{4}{9} + \frac{4}{9} + \frac{6}{9} + \frac{3}{9}\right) = \frac{17}{18} N$$

Don't forget this is for just pairs, we could try encoding triplets, etc.

Shannon entropy

- For a random variable *X* taking values in a finite set *X* with probability *p*, we call the entropy of *X*,

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

N i.i.d. random variables each with entropy H(X) can be compressed into more than NH(X) bits with negligible risk of information loss, as N tends to infinity; but conversely, if they are compressed into fewer than NH(X) bits it is virtually certain that information will be lost.

- $H(b) = -(p_1 \log_2 p_1 + p_0 \log_2 p_0) = -(2/3 \log_2 (2/3) + 1/3 \log_2 (1/3))$
  ~= 0.918296
  vs. 17/18 = 0.9444

Shannon entropy

- The amount of information being transmitted through the evolution of a CA could be a good measure of complexity.
  - Or measure information at state t, t+1, t+2, t+3 … somehow
- i.e., given the state *s(t)*, count neighborhood configurations

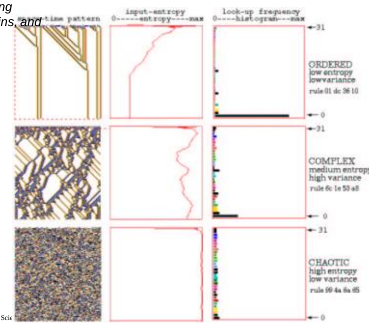
Neighborhood	Count	p_i	$p_i \log_2 p_i$
111	1	1/12	-0.298746875
110	3	3/12	-0.5
101	1	1/12	-0.298746875
100	2	2/12	-0.430827083
011	3	3/12	-0.5
010	0	0/12	NA
001	2	2/12	-0.430827083
000	0	0/12	NA

$$H(X) = -\sum_{i=1}^{k} p_i \log p_i = 2.459$$ (compression rate of s(t))

Shannon entropy

Classifying Cellular Automata Automatically;
Finding gliders, filtering, and relating space-time patterns, attractor basins, and the Z parameter.

Andrew Wuensche



Shannon entropy

- In Langton's paper he looked at the average single cell entropy over 10000 CA runs.

$$H(X) = -\sum_{i=1}^{k} p_i \log p_i$$

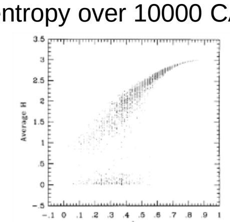


Fig. 6. Average single cell entropy $\bar{H}$ over λ space for approximately 10000 CA runs. Each point represents a different transition function.

Measures of complexity

- In Langton's paper he looked at a number of measures – one was transient length:

Fig. 3. Average transient length as a function of λ in an array of 128 cells.

First, transients grow rapidly in the vicinity of the transition between ordered and disordered dynamics, a phenomenon known in the study of phase transitions as *critical slowing down*. The relationship between transient length and λ is plotted in fig. 3.

Second, the size of the array has an effect on the dynamics only for intermediate values of λ. For low values of λ, array size has *no* discernible effect on transient length. Not until λ ≈ 0.45 do we begin to see a small difference in the transient length as the size of the array is increased. For λ = 0.50, however, array size has a significant ef-

We don't cover this in lecture, but do read:

Lindgren, Kristian, and Mats G. Nordahl. "Complexity measures and cellular automata." Complex Systems 2.4 (1988): 409-440.

https://www.complex-systems.com/pdf/02-4-2.pdf

Shannon entropy

- Information Theory:
  - (Efficiently) **store** information
  - (Efficiently) **communicate** information
- C.E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, July, October, 1948

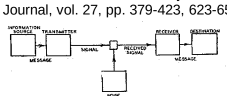


Fig. 1—Schematic diagram of a general communication system.

- how to convert efficiently a message into a signal (transmitter)
- how to decipher efficiently the signal back into a message (receiver)
- how to cope with noisy environments which alter the signal.

A Little CA History

- CA's and complexity are relatively new sciences.
  - von Neumann and Ulam created CA concept in 1940's.
    - Were trying to build self-replicating patterns and hence self-reproducing robots – hey, they worked at Los Alamos.
  - In 1960's Zuse proposed that universe is a cellular automa
    - New branch of physics.
  - Conway invented "game of life" in late 60's.
    - Became popular pastime for math/computer folks.
  - Many others contributed
- But didn't take off until computers made CA simulations simple.
  - Studied in detail and championed by Wolfram in 1980's.
    - In 2002 wrote pop-science tome *A new Kind Of Science*to mixed reviews.
  - Now CA studied by many in many different disciplines.
    - research related to glaciers, bird behavior, riots, etc.