

# CDM

## Cellular Automata

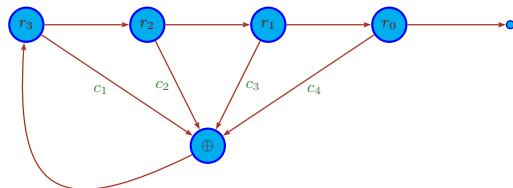
Klaus Sutner  
Carnegie Mellon University

Fall 2010

# Outline

- 1 Feedback-Shift Registers
- 2 Analysis of Cellular Automata
- 3 Discrete Dynamics

## Recall: Feedback Shift-Registers



A few bit-registers and a simple feedback function.

Iteration produces complicated and useful behavior: selecting the taps in a judicious manner, we can obtain bit-sequences with very long periods (near-Hamiltonian sequences).

## Pseudo-Random Number Generators

One application of such sequences is GPS.

Another, classical application is the generation of pseudo-random bits.

The general framework is to select an  $n$ -bit function

$$F : \mathbf{2}^n \rightarrow \mathbf{2}^n$$

that is iterated on some seed value  $\mathbf{x}_0$  to produce a sequence of vectors

$$\mathbf{x}_{n+1} = f(\mathbf{x}_n).$$

The actual pseudo-random bits are then typically obtained by some auxiliary, simple function (e.g. a projection  $g(\mathbf{x}) = x_i$ ).

## Can One Exploit This Phenomenon?

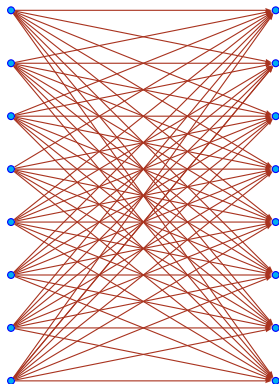
The message from these examples: simple Boolean functions can create rather complicated and useful behavior when iterated.

- Are there are other examples of this phenomenon?
- E.g., can we produce pseudo-random bits in some other way?
- Are there any other computational tasks that can be handled in a similar fashion?

We will consider one generalization with lots of interesting theoretical properties and a number of promising applications: **cellular automata**.

## Simplify

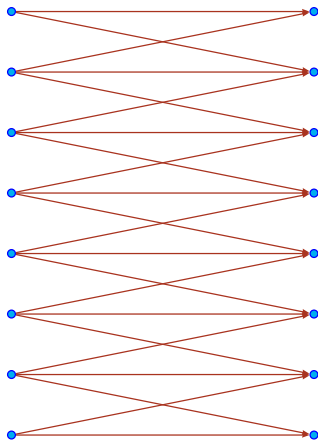
The input-output dependency of an  $n$ - $n$ -bit circuit looks like so:



Pretty, but too complicated.

## Simplify

Feedback-shift registers are based on simplifying this general layout (only one bit depends on all inputs, the others are shifted). Here is another simplification: localize the dependencies.



## Localize

More precisely, output bit  $x'_i$  depends only on its immediate neighbors, the input bits

$$x_{i-1}, x_i, x_{i+1} \rightsquigarrow x'_i$$

Another way to picture the flow of information is to think again of registers with certain interconnections:



Note that it takes 7 steps for information to propagate from register  $r_0$  to  $r_7$ : information cannot propagate instantaneously throughout the system.

## A Model

Think of the bits as a one-dimensional grid of registers. So we can visualize the state of the system:



It is customary in this context to refer to the registers as **cells**.

The picture shows a **configuration** of the system, an assignment of a bit to each cell.

The whole system updates **synchronously** (there is a global clock): each cell changes its state depending on its old state and the state of its immediate neighbors.

## Homogeneity

Locality is one important restriction, **homogeneity** is another one:

The same update rule map (a Boolean function  $\mathbf{2}^3 \rightarrow \mathbf{2}$ ) is used in each and every cell.

Thus, the local behavior of the system is the same everywhere.

Note that this looks a lot like classical physics: the laws of physics are also homogeneous (as far as anyone knows) and they are local, there is no “spooky action at a distance”.

Of course, quantum physics has made a few dents in this nice model. But for classical phenomena it's still perfectly accurate. E.g., fluid dynamics is local and homogeneous.

## Local versus Global Functions

Here is a more precise definition, with proper types.

### Definition

A **local configuration** is a function  $X : \{-1, 0, 1\} \rightarrow \mathbf{2}$ .

A **local map** is a function

$$\rho : ([-1, 1] \rightarrow \mathbf{2}) \longrightarrow \mathbf{2}$$

from local configurations to bits.

So, in essence, a local map is just a ternary boolean function  $\mathbf{2}^3 \longrightarrow \mathbf{2}$ , but our definition generalizes more easily.

Note that a local map can be specified by 8 bits, one for each local configuration.

## Global Maps

### Definition

Let  $\mathbf{2}^\infty = \mathbb{Z} \rightarrow \mathbf{2}$  denote the collection of all (bi-infinite) configurations.

We can extend any local map  $\rho$  to a global map

$$G_\rho : \mathbf{2}^\infty \longrightarrow \mathbf{2}^\infty$$

by setting

$$G_\rho(X)(i) = \rho(X(i-1), X(i), X(i+1))$$

Note that  $\mathbf{2}^\infty$  is an uncountable space.

## Finite Cellular Automata

Much of the recent work on CA is based on simulations of the global map. Of course, one cannot compute on bi-infinite configurations, so in practice one often makes do with configurations of finite length.

Indeed, a local map  $\rho$  also defines a global function

$$G_\rho : \mathbf{2}^n \longrightarrow \mathbf{2}^n$$

on configurations of length  $n$  by setting, as before:

$$G_\rho(X)(i) = \rho(X(i-1), X(i), X(i+1))$$

## Boundary Conditions

Alas, there is a small technical problem: we cannot apply the local map to update the first and last cell: there is only one neighbor.

There are two standard solutions (and variations thereof):

- **Fixed boundary conditions:** assume there is another cell in state 0.
- **Cyclic boundary conditions:** assume the grid wraps around (circle of cells).

We will use cyclic boundary conditions unless mentioned otherwise.

## Finite versus Infinite

The theory actually becomes quite a bit easier if one considers a bi-infinite grid of cells.

Simulations naturally take place on finite grids, but note that fixing the grid size to some arbitrary value  $n$  makes the system more complicated: we have to specify  $n$  in addition to the local rule.

As it turns out, the behavior of a finite CA may change drastically depending on  $n$ . No such complication exists in the infinite case.

## Spacially Periodic Configurations

Note that cyclic boundary conditions on a finite grid correspond exactly to spacially periodic configurations on the bi-infinite grid:

$$\begin{aligned}G_{\rho}(X) &= Y \\G_{\rho}(\dots XXXX \dots) &= \dots YYYYY \dots\end{aligned}$$

Unfortunately, for fixed boundary conditions this trick does not work.

### Exercise

*Explain exactly what problems arise when one tries to express fixed boundary conditions as a periodic bi-infinite configuration. Any solutions in sight?*

## Elementary Cellular Automata

### Definition

An **elementary cellular automaton (ECA)** is a local map  $\rho$ , a ternary boolean function. The underlying grid of cells is two-way infinite.

Note that there are 256 ECAs, so one can easily study all of them to get some idea of what kind of behavior a cellular automaton might exhibit.

One might suspect that the local functions of an ECA are too simple to produce anything of interest, but that reasonable suspicion would turn out to be quite wrong.

Of course, any simulation is necessarily based on a finite grid, so one has to be a bit careful about what insights can be gained from such rough approximations.

## Generalization: More States

It is clear how to get a larger class of cellular automata: fix some alphabet  $\Sigma$  and a finite interval  $I \subseteq \mathbb{Z}$ . Consider local configurations

$$X : I \rightarrow \Sigma \rightarrow \Sigma$$

A local map is again a function from all local configurations to the alphabet:

$$\rho : I \rightarrow \Sigma \rightarrow \Sigma$$

The standard choice is  $I = \{-r, \dots, 0, \dots, r\}$  in which case  $w = 2r + 1$  is the **width** of the CA and  $r$  its **radius**.

## Generalize Global Map

The corresponding global map

$$\rho : \Sigma^\infty \rightarrow \Sigma^\infty$$

is then obtained by

$$G_\rho(X)(i) = \rho(X_{i-r}, \dots, X_0, \dots, X_{i+r}).$$

More generally, to determine  $\rho(X)(i)$  we have to evaluate the global configuration  $X$  at places  $i + j$  for  $j \in I$ .

## Counting

### Proposition

Let  $|\Sigma| = k$ . Then there are  $k^{k^w}$  cellular automata of width  $w$  over alphabet  $\Sigma$ .

So even over a binary alphabet the number of CAs of width  $w$  gets out of hand very quickly: we cannot begin to look at all such binary CAs of width 7.

For larger alphabets even width 3 is already too much.

It is customary to use **digit alphabets** of the form

$$\Sigma_k = \{0, 1, \dots, k - 1\}.$$

## Constraints

One often imposes additional constraints on the local rules to get smaller rule spaces.

- **quiescent** rules:  $\rho(\mathbf{0}) = 0$ .
- **symmetric** rules:  $\rho(\mathbf{x}^r) = \rho(\mathbf{x})$ .
- **totalistic** rules:  $\rho(\mathbf{x}) = f(\sum x_i)$ .
- **outer totalistic** rules:  $\rho(\mathbf{x}) = f(x_0, \sum_{i \neq 0} x_i)$ .
- **additive** rules:  $\rho(\mathbf{x}) = \sum c_i x_i \bmod k$ .

## Exercise

*Determine how many rules of each kind there are (for all  $k$  and  $w$ ).*

## Generalization: Higher Dimensions

One can consider cellular automata operating on  $d$ -dimensional grids  $\mathbb{Z}^d$ , represented by local maps of the form

$$\rho : ([-r, r]^d \rightarrow \Sigma) \rightarrow \Sigma$$

A famous example is Conway's Game-of-Life, an outer-totalistic binary 2-dimensional cellular automaton with radius 1.

$$f(x_0, c) = \begin{cases} 1 & \text{if } x_0 = 0 \wedge c = 3, \\ 1 & \text{if } x_0 = 1 \wedge 2 \leq c \leq 3, \\ 0 & \text{otherwise.} \end{cases}$$

- Feedback-Shift Registers
- ② Analysis of Cellular Automata
  - Discrete Dynamics

## Two Problems

When one tries to understand cellular automata two very different types of problems appear:

### **Analysis**

Given a class of cellular automata, analyze and describe their behavior.

### **Synthesis**

Given certain specifications, construct a cellular automaton that conforms to these specifications. For example, construct a CA that expresses some process in physics (lattice-gas automata).

We'll ignore synthesis.

## Graphical Computing

So how does one analyze ECAs?

One important method is simulation: compute the orbit of some/all configurations on finite grids of different sizes.

Plot a 2-dimensional picture of these orbits and hope to find some patterns by visual inspection.

We can also look at phasespace, the directed graph obtained from the global map applied to all configurations of some rather limited size. For larger configurations we can only sample.

Of course, in the end pictures mean little: one has to be able to turn the pictures into theorems. Alas, it can be a long, long way from pretty pictures to a theorem.

## Wolfram Classes

In the early 1980's Wolfram suggested a classification along the following lines.

- All configurations die out after a while.
- All configurations become periodic after a while.
- Orbits are chaotic and seemingly random.
- Orbits produce complex persistent structures.

One can be more eloquent in describing these classes, but there is no accepted formal definition that captures these classes.

## Undecidability

Moreover, any formal classification seems to lead to undecidable problems. For example, consider the statement

On any finite grid, all configurations ultimately evolve to a fixed point.

It is undecidable whether a cellular automaton has even this relatively simple property.

The proof of this theorem is quite complicated and involves rather complicated cellular automata.

## Elementary CA

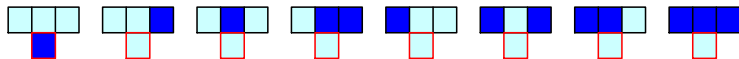
So how about elementary cellular automata?

Since there are only 256 and each one boils down to a ternary boolean function one might suspect that one can understand these simple systems completely.

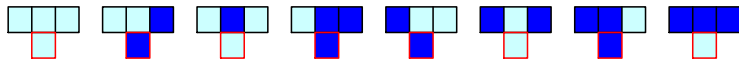
No such luck. Even elementary cellular automata are enormously difficult to analyze.

## Examples

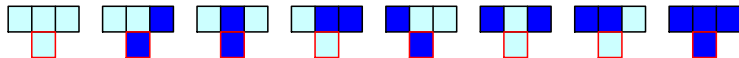
ECA number 1



ECA number 90



ECA number 150



## Iteration

Since even a finite configuration space  $C_n$  is quite large (except when  $n$  is very small) we cannot check out all configurations.

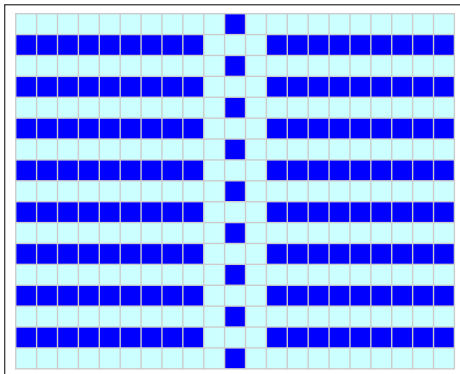
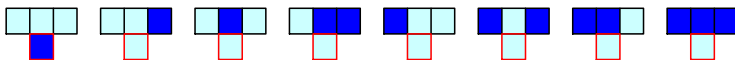
Instead, we pick a few special configurations and hope that their orbits give us some insight into the dynamics of the ECA.

Popular choices are:

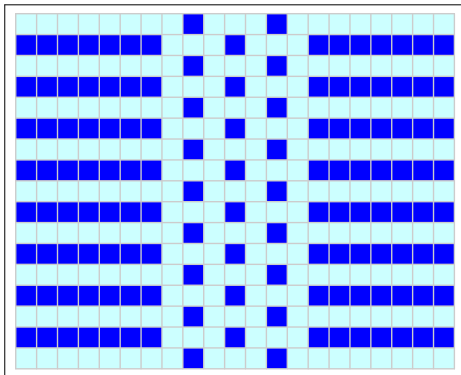
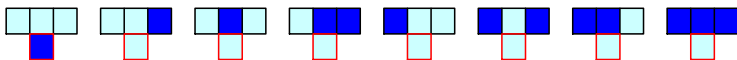
- Single cell in state 1, surrounded by 0's (one-point seed).
- Small block of non-zero cells, surrounded by 0's.
- Random vector of cell states.
- Carefully constructed patterns that produce special behavior.

Of course, the last type is contingent upon insight already gained from the other cases.

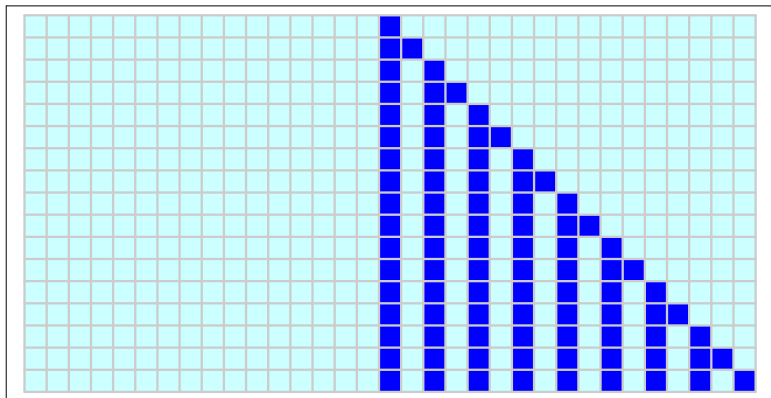
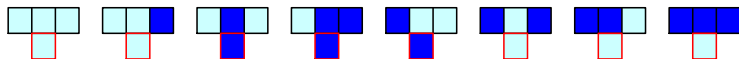
## ECA 1



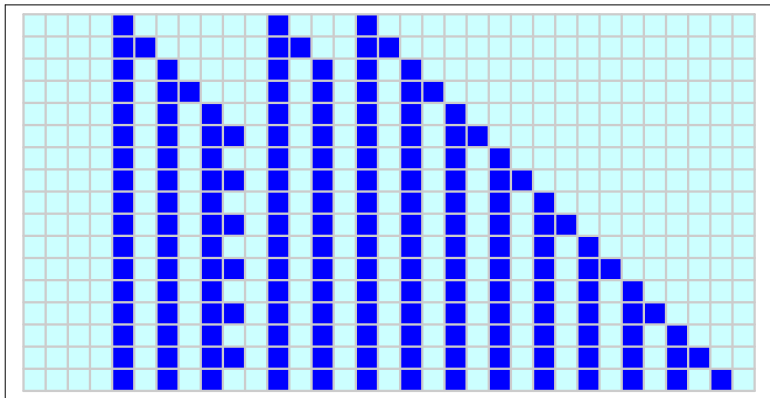
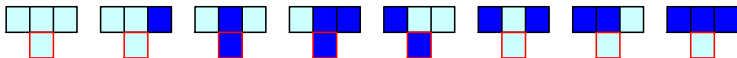
## ECA 1



## ECA 28



## ECA 28



## A Characterization

The pictures are pretty, but one might wonder if there is any deeper reason why CAs are a natural class of system. It turns out there is a very simple characterization of the global maps for infinite configuration spaces.

### Definition

The **shift operator**  $\sigma : \Sigma^\infty \rightarrow \Sigma^\infty$  is defined by  $\sigma(X)(i) = X_{i+1}$ .

### Proposition

*The global map of any cellular automaton commutes with the shift:*  
 $G_\rho \circ \sigma = \sigma \circ G_\rho$ .

This is really a consequence of homogeneity: shifting a configuration and applying  $\rho$  is the same as first applying  $\rho$  and then shifting the result.

### Exercise

*Prove the last proposition.*

## Convergence and Continuity

### Definition

Write  $X[m]$  for the finite block

$$(X(-m), \dots, X(0), \dots, X(m)) \in \Sigma^{2m+1}.$$

The sequence  $(X_n)$  **converges against**  $X \in \Sigma^\infty$ , written  $\lim_n X_n = X$ , if

$$\forall m \exists n_0 \forall n \geq n_0 (X[m] = X_n[m]).$$

A map  $f : \Sigma^\infty \rightarrow \Sigma^\infty$  is **continuous** if for all convergent sequences  $(X_n)$  we have

$$\lim_n f(X_n) = f(\lim_n X_n).$$

## Topology

This is the standard notion of continuity if we take the product topology on  $\Sigma^\infty$  where  $\Sigma$  is endowed with the discrete topology.

Continuity here means: to compute  $f(X)(i)$  it suffices to know a finite part of  $X$ .

Note that this topology is a bit strange: any sequence  $(X_n)$  whatsoever has a converging subsequence:

$$\lim_n X_{m_n} = X.$$

### Exercise

*Prove this assertion (exploit the fact that  $\Sigma$  is finite).*

## Hedlund's theorem

### Theorem

*A map  $f : \Sigma^\infty \rightarrow \Sigma^\infty$  commutes with the shift and is continuous if, and only if, it is the global map of a cellular automaton.*

*Proof.*

It is easy to see that any global map commutes with the shift and is continuous.

For the opposite direction suppose  $f$  commutes with the shift and is continuous. We need to construct a local map  $\rho$  such that  $f = G_\rho$ .

By continuity  $f(X)(0)$  is determined by a finite part of  $X$ .

More precisely, there must be some  $m$  and a partition  $U_a$  (empty blocks are allowed here) of  $\Sigma^{2m+1}$  such that

$$f(X)(0) = a \iff X[m] \in U_a.$$

## Proof, contd.

Now define a local map  $\rho : \Sigma^{2m+1} \rightarrow \Sigma$  by

$$\rho(\mathbf{x}) = a \iff \mathbf{x} \in U_a.$$

Since  $f$  commutes with the shift it follows that indeed  $G_\rho = f$ .

□

## Exercise

*Fill in all the gaps in the proof.*

## Exercise

*Construct some maps that commute with the shift but are not continuous.  
Construct some maps that are continuous but do not commute with the shift.*

## Theory Schmeory

It might be tempting to ignore a clean mathematical characterization like Hedlund's theorem and simply stare at the pretty pictures. Tempting, but a bad idea: one may waste a lot of time figuring out things that are quite obvious from the abstract point of view.

Here is an example: in 1972 D. Richardson wrote a paper titled "Tessellation with local transformation".

The paper studied the following interesting problem:

Suppose the global map  $G_\rho$  is injective. What can we say about the inverse map?

Hedlund had already shown the following result.

### Theorem

*If the global map  $G_\rho$  is injective it is also surjective.*

This can be shown by a rather elaborate counting argument.

## Inverting Global Map

So if  $G_\rho$  is injective it is actually a bijection.

In particular there is an inverse map

$$F : \Sigma^\infty \rightarrow \Sigma^\infty$$

such that  $G_\rho \circ F = F \circ G_\rho = I$ .

It is not hard to see that any such  $F$  commutes with the shift.

So the question arises:

Is  $F$  always the global map of some other cellular automaton?

## Richardson

Richardson showed that this is in fact the case, by a long and convoluted argument that seems to have at least one serious gap.

But by Hedlund's theorem all we need to show is that  $F$  is continuous.

That follows immediately from a general result in topology. Done.

If you know topology:

### Lemma

*A continuous map from any compact Hausdorff space to another is closed.*

- Feedback-Shift Registers
- Analysis of Cellular Automata
- ③ Discrete Dynamics

## Continuous Systems

What kind of questions are of interest when one tries to understand a cellular automaton?

It is helpful to step back a bit and consider classical predecessors: **dynamical systems**.

A dynamical system consists of a collection  $\Omega \subseteq \mathbb{R}^n$  of states (configurations, phase space) together with a “law” that determines, for any  $x_0 \in \Omega$  a trajectory  $(x_t)_{t \geq 0}$ .

The law is usually a differential equation:

$$\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} = F(\mathbf{x}, t)$$

One can think of the (non-negative) reals as acting on  $\Omega$ :

$$x_{s+t} = x_{s_t}$$

If the system is reversible then (all) the reals are acting on  $\Omega$ .

## Example: The Solar System

We can think of the solar system as a collection of point masses, one being much heavier than the others. Without loss of generality assume the sun is at rest at the origin of the coordinate system.

Each point mass of a planet is described by 6 real parameters: 3 for position and 3 for velocity. So we have a  $6 \times 9 = 54$  dimensional system if we consider only the nine planets (Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto).

The law for the system is given by Newton's law of gravity.

A question that has been studied carefully for about a century:

Is the solar system stable? I.e., given an arrangement of planets as we observe today, what is going to happen 1 billion years from now?

## Discrete Time

Even for classical, continuous dynamical systems it is often convenient to consider discrete time approximations.

More generally, we have a map

$$T : \Omega \rightarrow \Omega$$

and we are interested in the orbits of points  $x \in \Omega$ :

$$x_{n+1} = T(x_n)$$

So here the natural numbers are acting on  $\Omega$ .

Or the integers if the system is reversible.

## Example: The Logistic Map

Consider the map

$$T_p : [0, 1] \rightarrow [0, 1] \quad T_p(x) = px(1 - x)$$

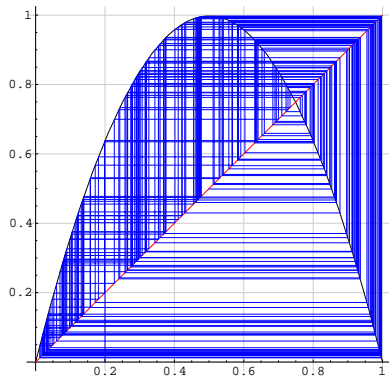
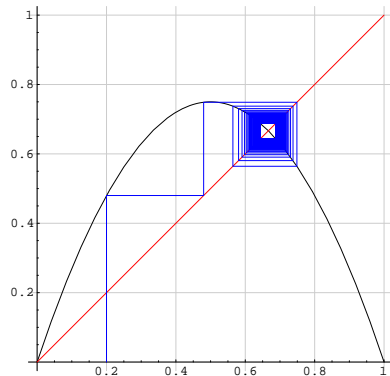
where  $p$  is a parameter,  $0 \leq p \leq 4$ .

So time is discrete in this case, but  $T_p$  is still a differentiable function.

As we have seen before, for some parameter values the orbits converge nicely but for others things become wildly complicated.

## Logistic Map Orbits

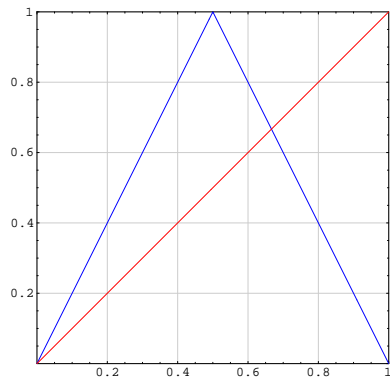
$p = 3$  and  $p = 3.99$ .



## Full Tent Map

Here is a simplification (no parameter) of the logistic map:

$$T : [0, 1] \rightarrow [0, 1]$$
$$T(x) = \begin{cases} 2x & \text{if } x \leq 1/2, \\ 2 - 2x & \text{otherwise.} \end{cases}$$

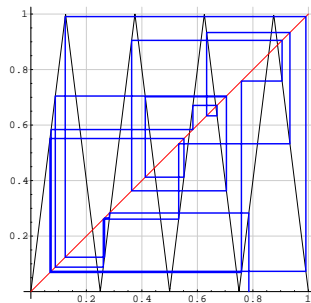
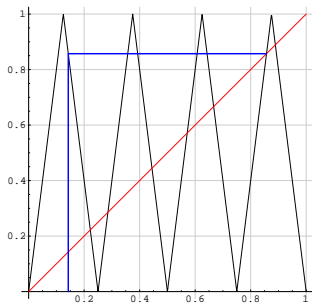


Too boring for words.

## Full Tent, Iterated

The full tent map iterated 3 times.

Initial values:  $x_0 = 1/7$  and  $x_0 = \pi/4$ .



## Symbolic Orbits

Here is a crucial idea: don't deal with real numbers (which are too complicated from a computational perspective), but project everything down into a discrete sequence.

More precisely, we can represent the orbit of a point in  $[0, 1]$  symbolically as follows:

$$\begin{aligned}\pi : [0, 1]^\omega &\rightarrow \mathbf{2}^\omega \\ \pi(\mathbf{x})(i) &= \begin{cases} 0 & \text{if } 0 \leq x_i \leq 1/2, \\ 1 & \text{if } 1/2 \leq x_i \leq 1. \end{cases}\end{aligned}$$

Note that we have partitioned the domain  $[0, 1]$  into two overlapping blocks  $P_0 = [0, 1/2]$  and  $P_1 = [1/2, 1]$ .

## Fudging It

Needless to say,  $\pi$  is not well-defined:  $\pi(1/2) = 0, 1$  according to the definition. But that is the only problem.

Moreover the orbit of  $1/2$  under  $T$  looks like  $(1/2, 1, 0, 0, \dots)$  so that the ambiguity occurs only in one bit.

So any orbit touching  $1/2$  has two images under  $\pi$ , all others have a unique image. Thus the “bad set” is countable and we can ignore it.

Other than that  $\pi$  has a number of good properties:

- $\pi$  is surjective
- $\pi$  is continuous
- $\pi$  is injective except on a countable set (where it is two-to-one).

### Exercise

*Prove all these properties.*

## Symbolic Dynamics

The last and crucial property of  $\pi$  is that it commutes with the operation in the two systems:

$$\begin{array}{ccc}
 [0, 1]^\omega & \xrightarrow{\pi} & \mathbf{2}^\omega \\
 \downarrow T & & \downarrow \sigma \\
 [0, 1]^\omega & \xrightarrow{\pi} & \mathbf{2}^\omega
 \end{array}$$

Here  $\sigma$  is the left shift:

$$\sigma(x_0x_1x_2\dots) = x_1x_2\dots$$

This is really amazing: instead of studying the behavior of  $[0, 1]$  under the full tent map  $T$  we may just as well consider the shift on all infinite binary sequences.

This translation from a continuous dynamical system to a purely discrete one is often possible.

## Shift-Commuting

The exact same diagram describes the shift-commuting property of the global map of a cellular automaton:

$$\begin{array}{ccc} \Sigma^\infty & \xrightarrow{G_\rho} & \Sigma^\infty \\ \sigma \downarrow & & \downarrow \sigma \\ \Sigma^\infty & \xrightarrow{G_\rho} & \Sigma^\infty \end{array}$$

Any continuous map for which this diagram commutes is already the global map of a cellular automaton.

## What Are The Questions?

In principle, we simply want to understand all (or at least: most) orbits.

- Which orbits are periodic or ultimately periodic?
- Which orbits have long transients?
- What is the long term behavior of the system?
- Does an orbit settle down in the long run?
- Are some regions of phase space attracting/repelling?
- Are two dynamical systems related (isomorphic in a sense)?
- Which dynamical systems are important in the sciences?
- How does one construct a dynamical system with certain given properties?
- Can one answer these questions automatically for certain classes of systems?

# Logic

## Finite Discrete Case

In the special case where  $\Omega$  is finite we can think of  $\langle \Omega, T \rangle$  as digraph: the functional digraph of  $T$  whose nodes are the points in  $\Omega$  and whose edges are given by

$$(x, T(x)).$$

If  $\Omega$  is rather small (say, a million points or 20 bits per point) then we can simply generate the whole graph and use standard algorithms to compute strongly and weakly connected components (in linear time).

### Exercise

*Suppose  $\Omega$  has cardinality  $2^{20}$  and you can easily compute  $T$ . How would actually compute the transients and periods for all points?*

## Gathering Evidence: Simulation

Of course, to understand even a binary CA we need to look at larger grids, not too large, not too small, say around 500 cells. Much smaller grids may obscure the dynamics: there is just not enough space for anything interesting to happen (think about ECA 110).

Now the central problem is that phase space is exponentially huge: we cannot look at all  $2^{500}$  configurations and the transients and periods may be enormously long.

So we sample in the usual places and keep our eyes open.

Also, it is always a good idea to do some brute force computation for the whole phase space for small  $n$  before getting involved with the general situation.

## Predictability

The size of phase space makes it really tempting to ask the following innocent question:

- Does one really have to compute the whole initial segment of an orbit in order to obtain  $G_\rho^t(X)$ ? Could there be a computational short-cut that simply produces the configuration at time  $t$  immediately?

This would be great for studies of the long term behavior: just look at the system for  $t$  a multiple of  $10^6$  or some such.

Ideally we would like to answer general questions about the orbit of  $X$  directly. For example, it is interesting to know if the orbit ultimately dies out: is there a time  $t$  such that

$$G_\rho^t(X) = \mathbf{0}?$$

Since  $t$  is only bounded by  $2^{500}$  a short-cut is really essential here.

## More Predictability

Note that from the point of view of abstract computability this is all trivial: constructing the full phase space is primitive recursive in  $n$ , done.

But in harsh reality exponential size is already enough to destroy practical computability.

We clearly can compute the function

$$F(X, t, i) = G_\rho^t(X)(i)$$

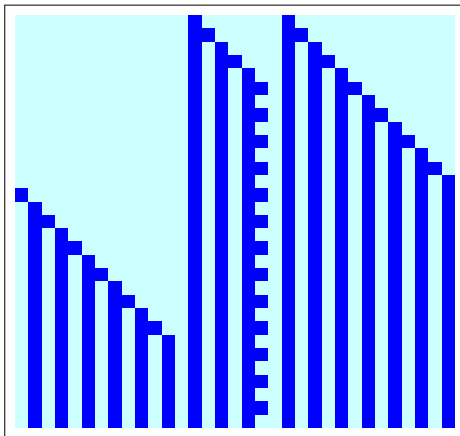
in  $O(nt)$  steps: simply generate the whole orbit of  $X$  up to time  $t$ . Here  $X \in \mathcal{C}_n$ ,  $t \geq 0$  and  $i \in [n]$ .

- Can we compute  $F(X, t, i)$  in significantly less time?

## Example: ECA 28

Let's take a look at a concrete example: ECA 28.

The picture of a two-point see orbit suggests things should be manageable here.

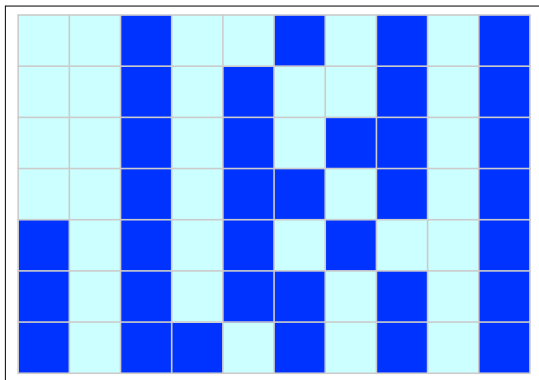


## Brute Force

For  $n = 10$  there are 3 fixed points and 30 cycles of length 2. The fixed points are easy:

(0000000000), (1010101010), (0101010101)

The two cycles are bit harder: here is a picture of all of them (eliminating rotations there are only 7).



## Transients

A table of all transient/period pairs, and the number of configurations that attain them.

$t$	$p$	count	$t$	$p$	count
1	1	21	5	1	80
1	2	160	5	2	40
2	1	40	6	1	10
2	2	115	7	1	30
3	1	150	8	1	10
3	2	235	9	1	10
4	1	40			
4	2	20			

Probably not a good idea to try to find a closed form for the actual counts, but the length of the transients should be OK. Also note that long transients seem to lead to fixed points.

## Line of Attack

Every configuration in  $\mathcal{C}_n$  other than  $\mathbf{0}$  can be written in the form

$$X = 10^{n_1} 10^{n_2} \dots 10^{n_{k-1}} 10^{n_k}$$

where  $n = \sum_{i=1}^k (n_i + 1)$  and  $1 \leq k$ .

Then the transient is essentially  $\max n_i$  and the period is 1 or 2 essentially depending on the parity of the  $n_i$ .

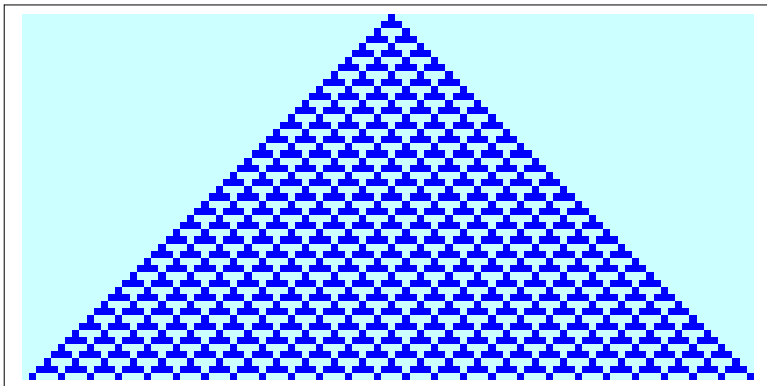
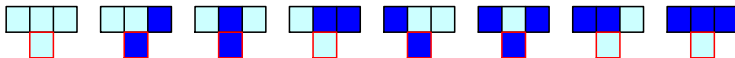
Note the qualifier “essentially”, things are bit more complicated than that (one has to deal with special cases for small values of  $n_i$ ).

### Exercise

*Carry out the details of this argument. Make sure to verify your results experimentally.*

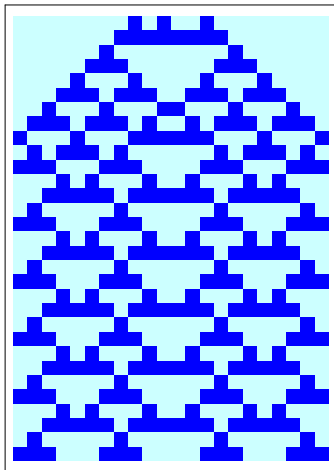
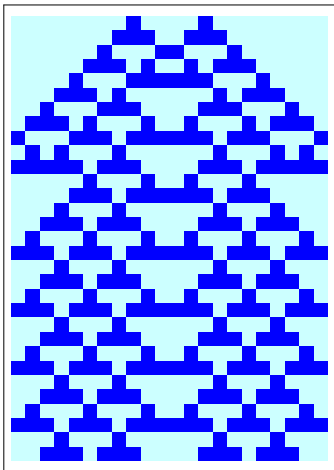
## ECA 54

But be warreded, things do not always work out so nicely. How about this one?

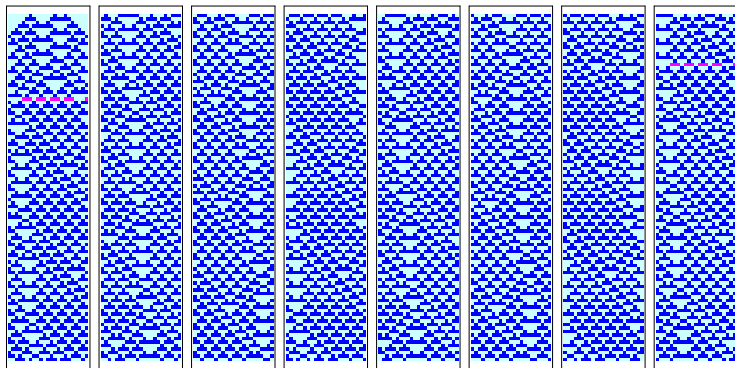


## Multiple Seeds

Apparently not bad. More complicated seeds don't seem to cause too much trouble, either.



## Disaster

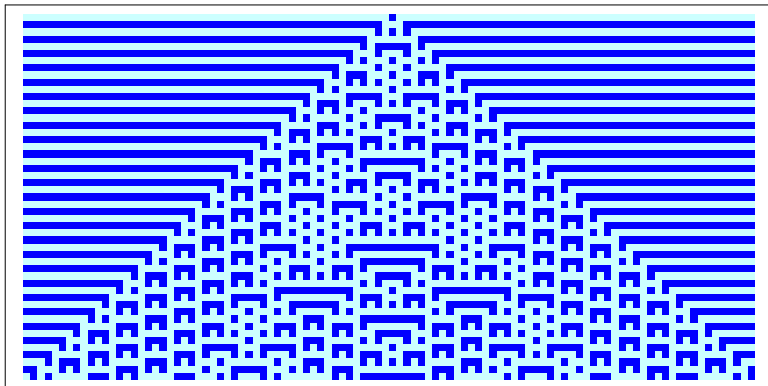
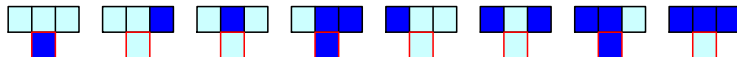


$$n = 23, t = 24, p = 690$$

The entry point to the limit cycle is marked in purple (first and last column).

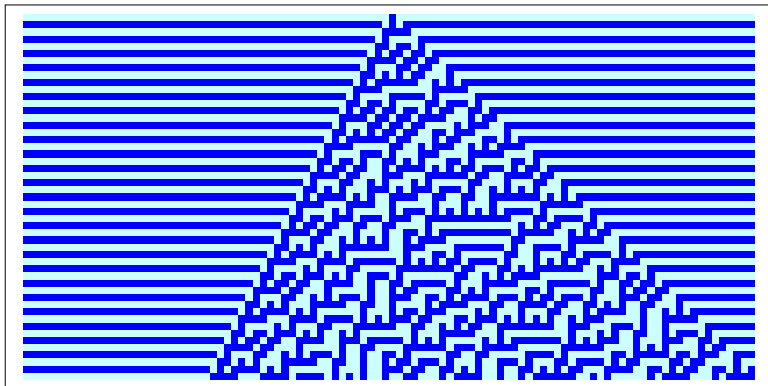
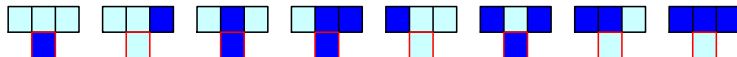
## ECA 73

Another apparently hopeless case.



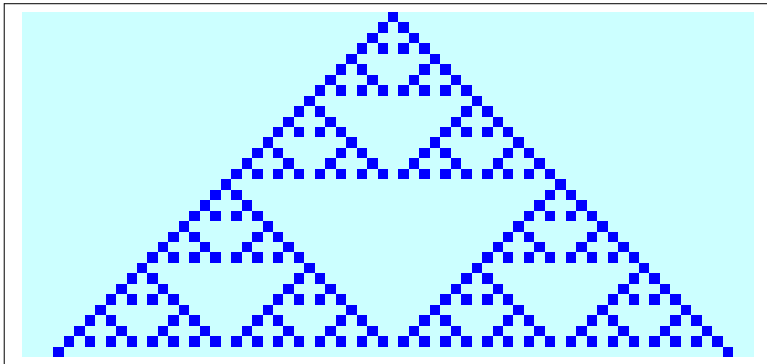
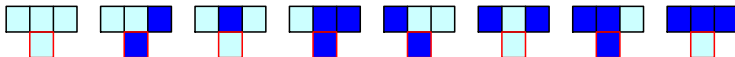
## ECA 45

This one seems even more intimidating.

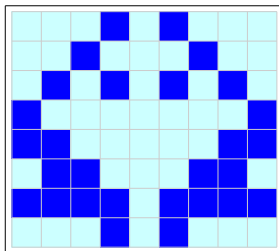
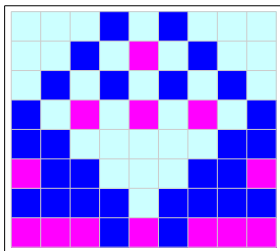
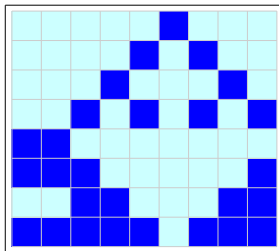
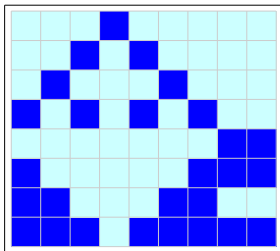


## ECA 90

But this one should be doable: The local rule is exclusive or (addition modulo 2) of the left and right neighbor.



## Additivity



## Additivity and Predictability

Additive cellular automata are “trivial” in the sense that

$$G_\rho(X \oplus Y) = G_\rho(X) \oplus G_\rho(Y).$$

Here  $\oplus$  stands for addition mod 2 (component-wise on the vectors).

Since for any configuration  $X$

$$X = \bigoplus_i X_i$$

where all the  $X_i$  are one-point configurations, it suffices to figure out the orbits of one-point seed configurations.

## Even Worse

The global map of ECA 90 is really a linear map: the ground field is  $\mathbb{F}_2$ , and the vector space is  $2^n$ .

But that means we can write it in terms of matrix multiplication:

$$G_\rho(X) = M \cdot X$$

where  $M$  is some  $n \times n$  boolean matrix. It follows that

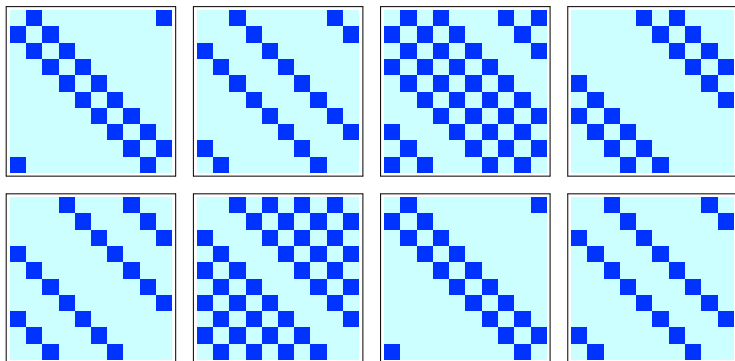
$$G_\rho^t(X) = M^t \cdot X$$

and the latter expression can be evaluated in  $O(n^3 \log t)$  operations.

So these systems are very strongly predictable with respect to time. Recall that  $t \leq 2^n$ , so the whole computation takes at most  $O(n^4)$  steps.

## Speedup

Actually, it's easy to do this much faster.



### Exercise

Give a fast algorithm to predict rule 90 on a grid of size  $n$ .

## Pseudo-Random Bits

So how about cellular automata and pseudo-randomness? Here we would want the system to be highly unpredictable: there should be no way to determine the million-th bit without running the whole simulation.

Is there a simple CA that, starting with a one-point seed configuration, will produce an orbit that can be used as a source fo pseudo-random bits?

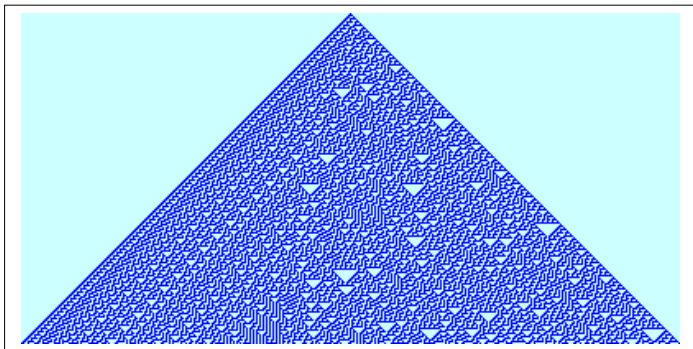
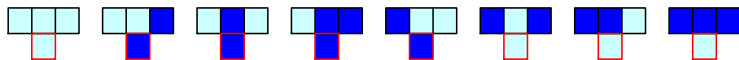
Preferably the CA should be binary and use a small neighborhood. An ECA would be perfect.

One might suspect that binary nearest neighbor CA are too restricted to produce anything resembling randomness.

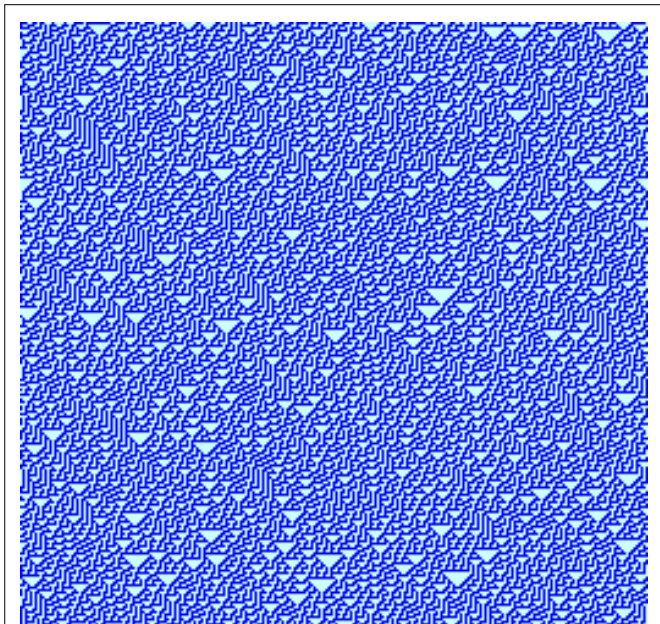
But actually there is a candidate: ECA number 30.

This CA was used in the Connection Machine as a PRN generator and is still used in Mathematica.

## ECA 30



## ECA 30



## CA Hardware

It is clear that a classical von Neumann single CPU computer is singularly ill suited to simulating a cellular automaton.

Given the amazing speed of modern processors one can still do interesting things, but to really get mileage out of this model one needs special hardware.

Alas ...

- <http://www.ai.mit.edu/projects/im/cam8/>
- <http://mission.base.com/tamiko/cm/cm-image.html>

## Summary

- Elementary cellular automata are another example of extremely simple computational devices: trivial in a single step, hopelessly complicated under iteration.
- One can get mileage out of hardness, though: random number generation, cryptography.
- Interesting model since it is much closer to physics than other standard models.
- Have been used successfully for simulations in physics.
- Probably important for future improvements in computational power: many identical, small units with simple interconnections.
- The theory of infinite CA is a bit easier than the finite case (can apply ideas from topology).