Introduction to Complexity and Applied Complexity, Spring 2021

Module 4 — Cellular Automata, Self-Organization, and Pattern Formation

Notes by Sav Sidorov

Readings

- Stephen Wolfram — A New Kind of Science (chapter 2)

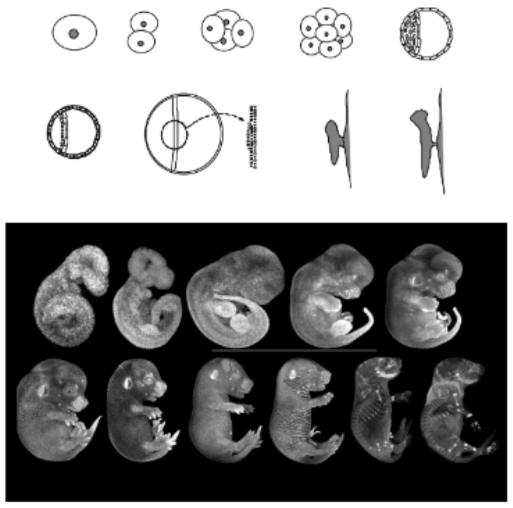
Motivation
Last module, we looked at dynamical systems, and even looked at an example of chaotic motion with the logistic map. We were looking at one-dimensional systems. This module, we're going to look at coupled systems.

Specifically, we'll go through a class of systems called **cellular automata** — as popularized by Stephen Wolfram in *A New Kind of Science*. Our goal with these systems is to show the coupling between components.

Another theme we'll cover in this module is **self-organization**, meaning that there's no central controller in the system. The behavior is emergent, not dictated by a central authority.

Sources of Order

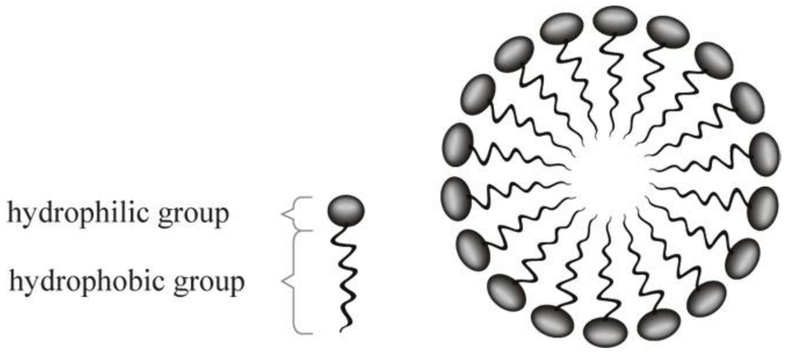
Take a look at the juxtaposition of these two images:



Here we see two different types of construction. On the right, we see construction of a house using a blueprint — a clear example of the central controller approach. The design is planned out beforehand, and the building is assembled strictly according to the design. On the left, we see embryonic development, from a cell to an organism. The DNA of an organism is not a blueprint. Rather, it's a set of instructions — a dynamical law of the system — according to which each cell behaves. Each cell takes in signals from the outside and information from its internal state. Then, it decides what to do next, whether it be synthesizing proteins or adjusting the network by which proteins become synthesized. Each cell is sensing what's going on in its local environment, based on which it "decides" how to act.

This notion of self-organization comes from distributed agents engaging in local interactions and behaviors, leading to global order. When we say local, we don't mean that long-range connections cannot exist. We simply mean the cell (or any other agent) receives information directly; the cell knows only about itself.

As an example of spontaneous order, consider the micelle. A micelle is a structure that emerges when — say, in water — a set of molecules, each having a hydrophilic (water loving) and hydrophobic (water hating) group, get lumped together. Due to the properties of each molecule, they form a spherical structure in water, with the hydrophilic side oriented outwards and the hydrophobic oriented inwards.



Reading: A New Kind of Science
A quote from this reading:

> *"We have no choice but to develop a whole new kind of intuition."*

Like we said from the outset, one of the things we want to do is develop an intuition when it comes to thinking about complex dynamics, not just collect chunks of knowledge.

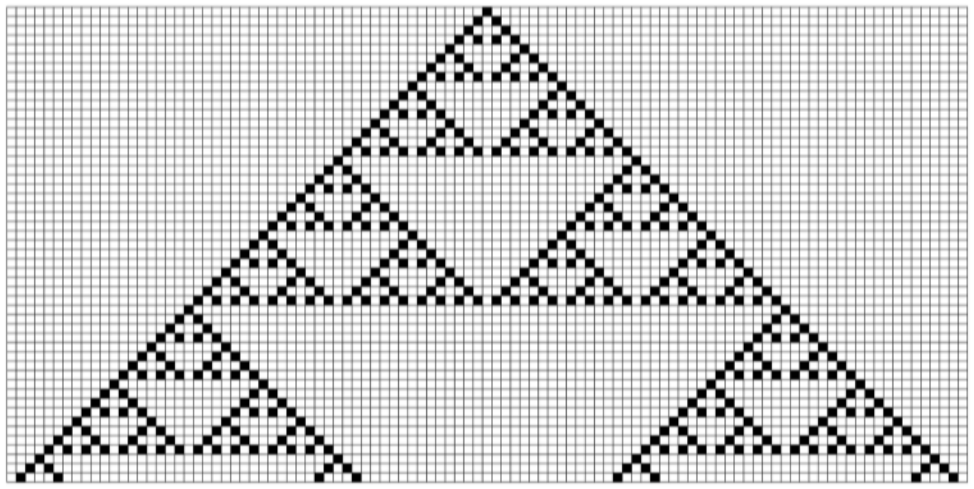
One thing to note about Wolfram's work: it's incredibly powerful to tinker with models on a computer — changing the rule sets, the details, the assumptions and watching how the macroscopic behavior changes. It's almost always counterintuitive — you think you know what some change will do, but in reality it does something completely different.

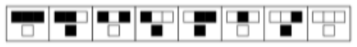
The dynamics of a complex system are anywhere from difficult to impossible to capture with "solved equations" or any kind of closed-form solution. It turns out that running the model on a computer is one of the most straightforward ways of figuring out how a system will behave.

Elementary Cellular Automata

The class of models that Wolfram emphasizes in *A New Kind of Science* are called **elementary cellular automata**. They're as simple as you can get while still having rich dynamics emerge — if you go any simpler, then the behavior isn't interesting.

Put another way, they're **coupled automata with maximally simple internal structure and behavior**. Here, for example, is a cellular automaton that produces a pyramidal structure (rule 90, as Wolfram calls it):





A cellular automaton that produces an intricate nested pattern. The rule in this case is that a cell should be black whenever one or the other, but not both, of its neighbors were black on the step before. Even though the rule is very simple, the picture shows that the overall pattern obtained over the course of 50 steps starting from a single black cell is not so simple. The particular rule used here can be described by the formula $a_i' = Mod[a_{i-1} + a_{i+1}, 2]$. In the numbering scheme of Chapter 3, it is cellular automaton rule 90.

How do cellular automata work?

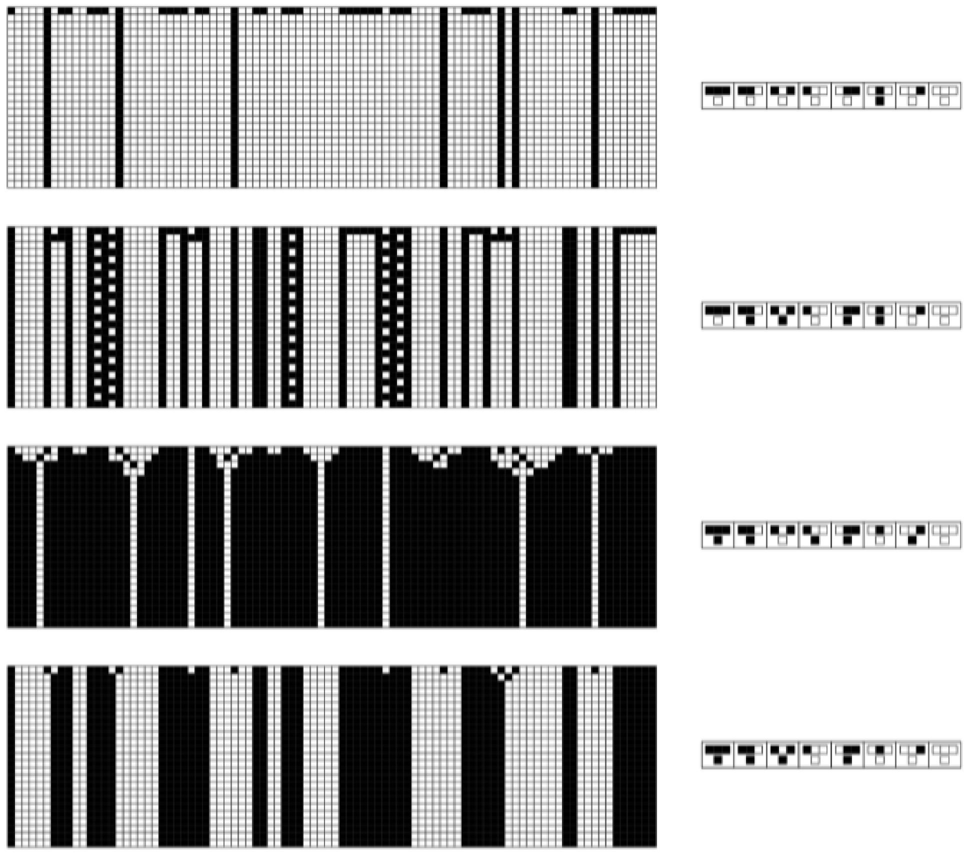
- Each cell can take on two states: 0 or 1 (represented as black and white).
- These models are considered one-dimensional. However, this notion is different from the one we looked at previously, where one variable was changing with time. Rather, it's one-dimensional in a spacial sense: the cells are arranged

horizontally in a line, each having two neighbors. As we go down the grid of a cellular automaton, we can see the evolution of these cells through time. So those two-dimensional representations of cellular automata are actually showing us space (horizontally) and time (vertically).

- Each cell interacts with its neighbors to decide what state to be in next, according to a set of rules (you can see, for example,the rule set for the pyramidal automaton listed below the picture).

Let’s look at some specific rules.

Here is a class of rules that produce order from an initial random arrangement. The type of order, however, depends on the initial conditions. Feel free to take grid paper and try evolving these rulesets yourself:

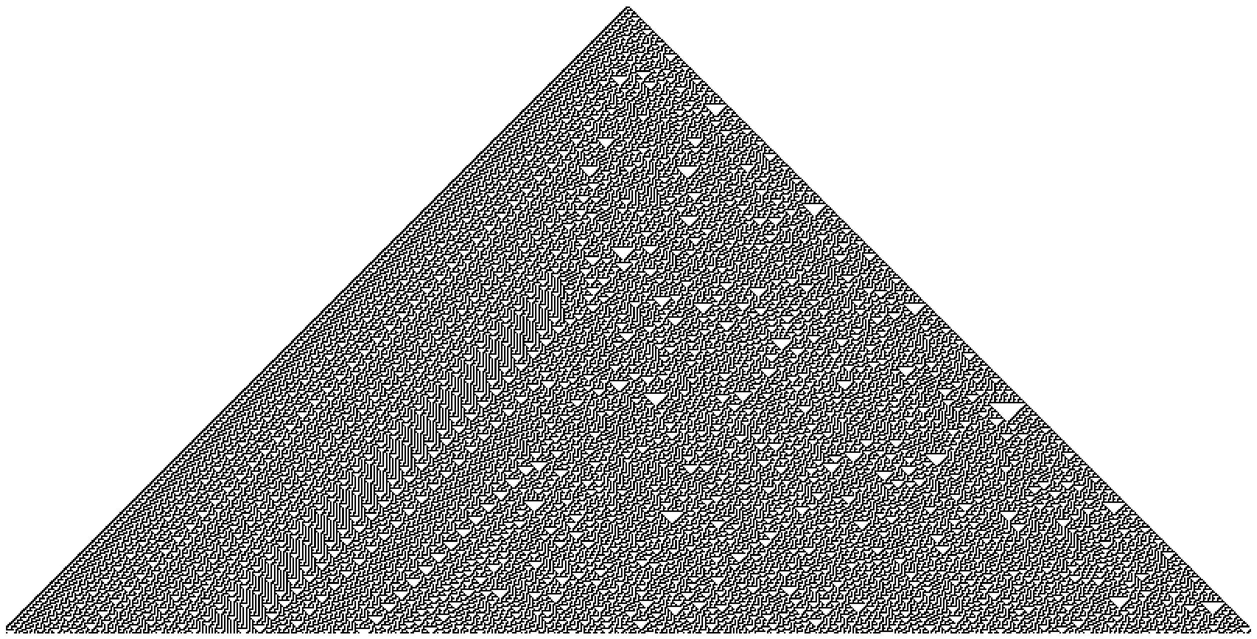


Examples of cellular automata that evolve from random initial conditions to produce a definite set of simple structures. For any particular rule, the form of these structures is always the same. But their positions depend on the details of the initial conditions given, and in many cases the final arrangement of structures can be thought of as a kind of filtered version of the initial conditions. Thus for example in the first rule shown here a structure consisting of a black cell occurs wherever there was an isolated black cell in the initial conditions. The rules shown are numbers 4, 108, 218 and 232.

Cellular automata are also another example of recursive models. The current state of the system (the current horizontal line of cells) is taken as input, transformed by the rule set into a new state, and this process is carried out repeatedly.

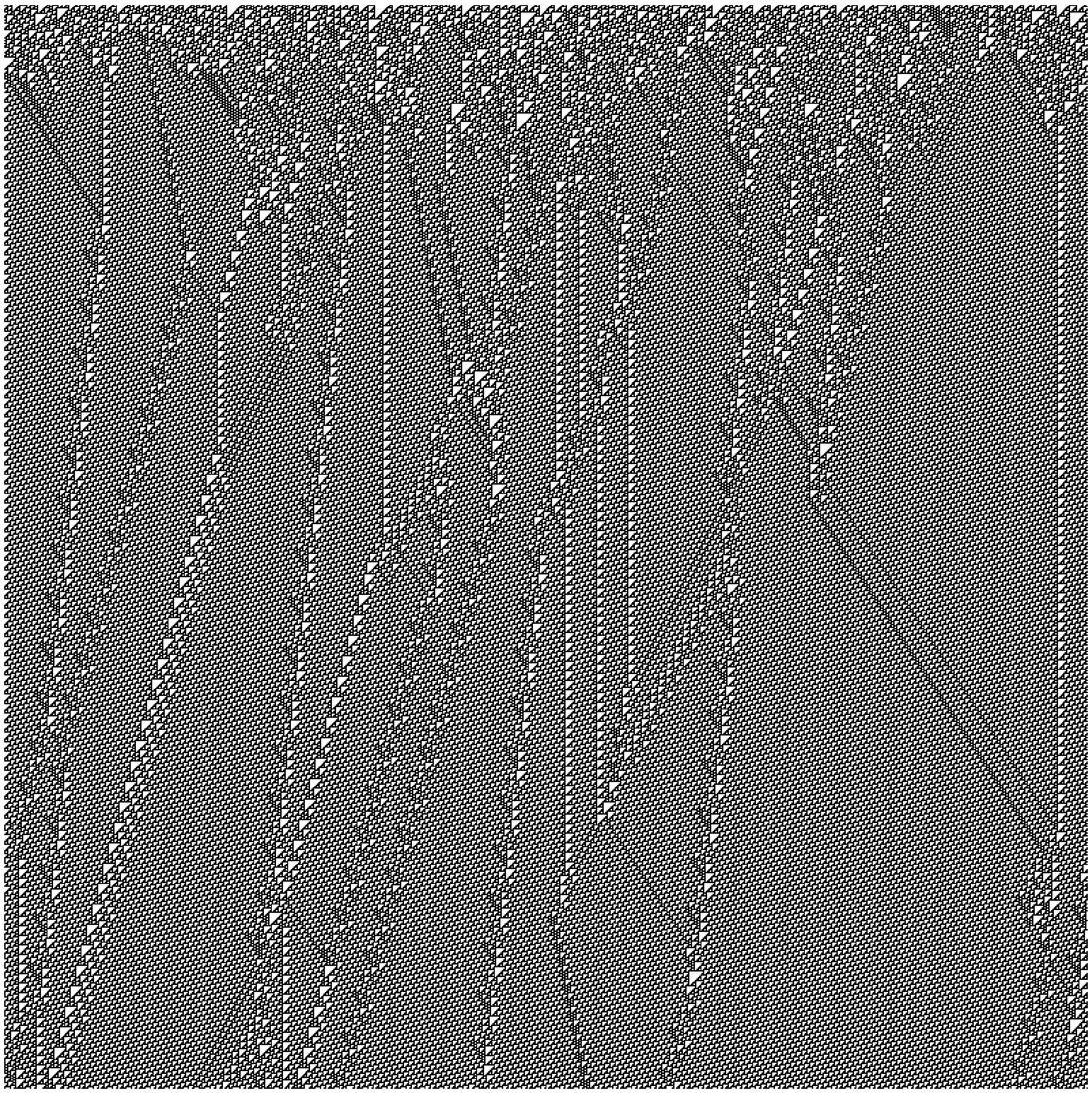
They are also discrete models in several ways. The space is discrete, since you can be in one cell or another, not in between. Time is also discrete — rows evolve step by step, you can't be in the middle of two rows. Finally, the state space is discrete — a cell can be black or white, nothing in between.

Looking at rules 4, 108, 218 and 232 shows us the regularity that can emerge out of recursing over a rule set. Yet not all rules are like this — there are 256 possible rules (in a stretch of 3 cells, there are 2^3=8 possible combinations, that then transform the center cell into either a black cell or a white cell, so in effect we get 2^8=256), and some behave unpredictably. The only way to determine how the system behaves is to watch it play out — something Wolfram calls **computational irreducibility**. You can't, for example, come up with some formula or some other shortcut that will let you “skip” the step-by-step computation. Take rule 30 for instance:



One important point to note about these kinds of systems that exhibit chaotic dynamics: if you apply statistics over the chaotic (i.e strange attractor) state space, you’ll notice stationary probability distributions. For example, if you measured the sizes of the triangles and plotted out their frequency distribution, you’d see that the values you get are quite stable.

Let’s look at another rule that plays out in an unpredictable, chaotic way. The famous rule 110:



Rule 110 is a unique case — it has local patches of order, yet globally, it's chaotic. As you can see above, these locally-consistent patterns that persist for a long time collide with each other to form new kinds of patterns. People have actually done work to classify these locally-consistent patterns as kinds of particles, and have invoked the mathematics of particle dynamics when studying them, hence dealing with the system on a completely different scale than what's encoded in the rule set.

It turns out that, given the intricate interactions within it, rule 110 creates a rich enough system that it can do computation. Rule 110 is what we call **Turing complete**, or a **universal Turing machine**. The Turing machine is a conceptual model that underlies all of the computing we do today. What makes a Turing machine universal is its ability to simulate all other Turing machines. Rule 110 has such a property. Of course, you'd never use rule 110 in a practical setting, but it's interesting to note that it's capable of universal computation.

And of course, like rule 30, rule 110 is computationally irreducible. It is also **structurally irreducible**. We can't know what's going to happen with any given cell without knowing first what its neighbors will do. Additionally, over time, because the effects of one cell travel through and affect the whole system, the ultimate effects also impinge on the state of each cell. It would be absurd to try to break down rule 110 into its individual cells and study the cells. Every cell is deeply dependent on its interactions — the marker of a complex system.

Representation and Perceptibility

A question worth raising: if rule 110 can act like a Turing machine, meaning you can program it to run any computer program (i.e. you would translate the computer program into the initial conditions of rule 110 and then recurse on the initial conditions), why don't we use it for computation?

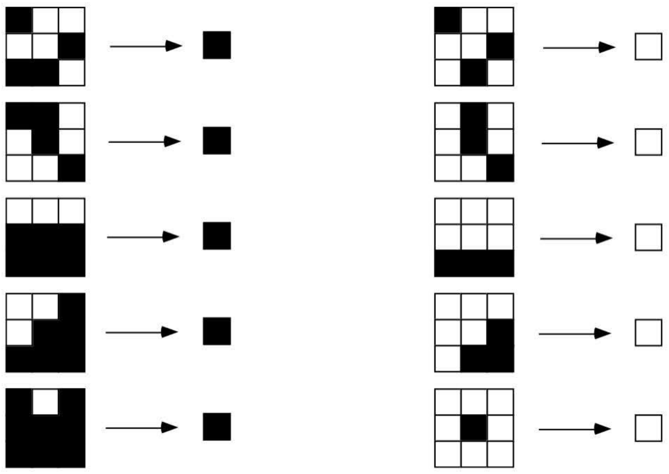
Simply put, because it's inconvenient. It's not at all obvious what the mapping would be between a computer program in the way that we currently think of it, and the initial conditions of rule 110.

What we see here is that there's a *formal equivalence* between our computer programs and rule 110 — at the mathematical level — yet these things are very much unlike one another in terms of the way they interface with you, the user. Interface, not just formal equivalence, is important when it comes to how we think about building or intervening in systems. Often, you see the **IYI types** turn to formalisms, leaving questions of interface in the background. One should ask: are two systems truly equivalent, or is there some important difference that we're not thinking about because we've only scoped in on a fraction of the system, only looking at its internal logic and not at how it interfaces with other systems that interact with it?

To sum up, you would never program in rule 110, even though you could.

The “Panic” Rule

Let’s look at an example of a two-dimensional cellular automata. To give you an idea of the ruleset, here are some example mappings:



The next state of the center cell is determined by its surroundings. It’s almost the majority rule, but not quite. It’s biased — if there are 4 or more black cells, the center cell becomes black.

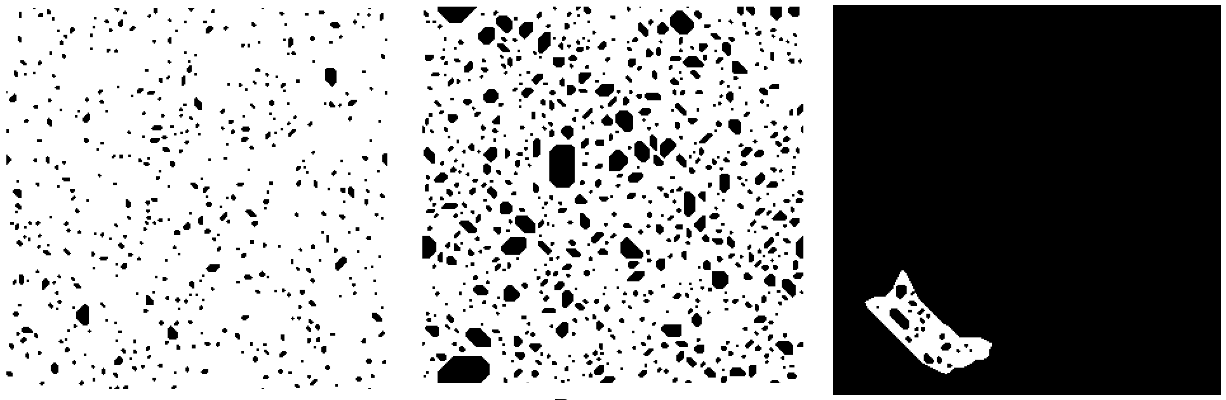
This ruleset is sometimes called *the panic rule*, because you can use it to model panic. Imagine that the cells are seats in a theatre, for example. Someone yells “fire”, some people start to panic, others stay calm. As more people panic, that leads to even more people panicking. You can see how this system evolves like a contagion. In fact, it turns out that social behavior like panic, rioting, etc *are contagious*.

If we designate the cells as +1 and -1, we can actually write out an equation that describes this behavior:

$$x_i = Sign\left(\sum_{neighbors} x_j + 2\right)$$

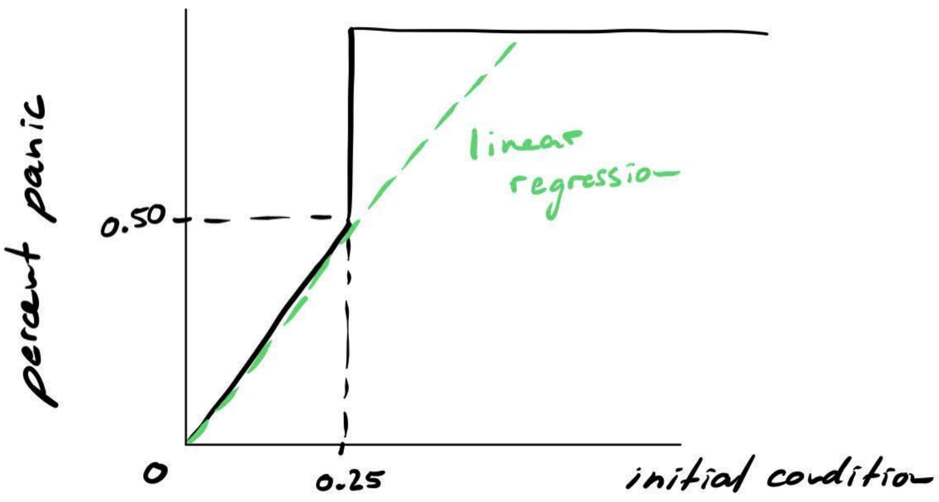
The sign of cell *x* at index *i* is a function of the sum of its nearest neighbors (x_j's) plus 2, which acts as our bias.

You can play around with an **interactive version of the panic rule**, producing results such as:

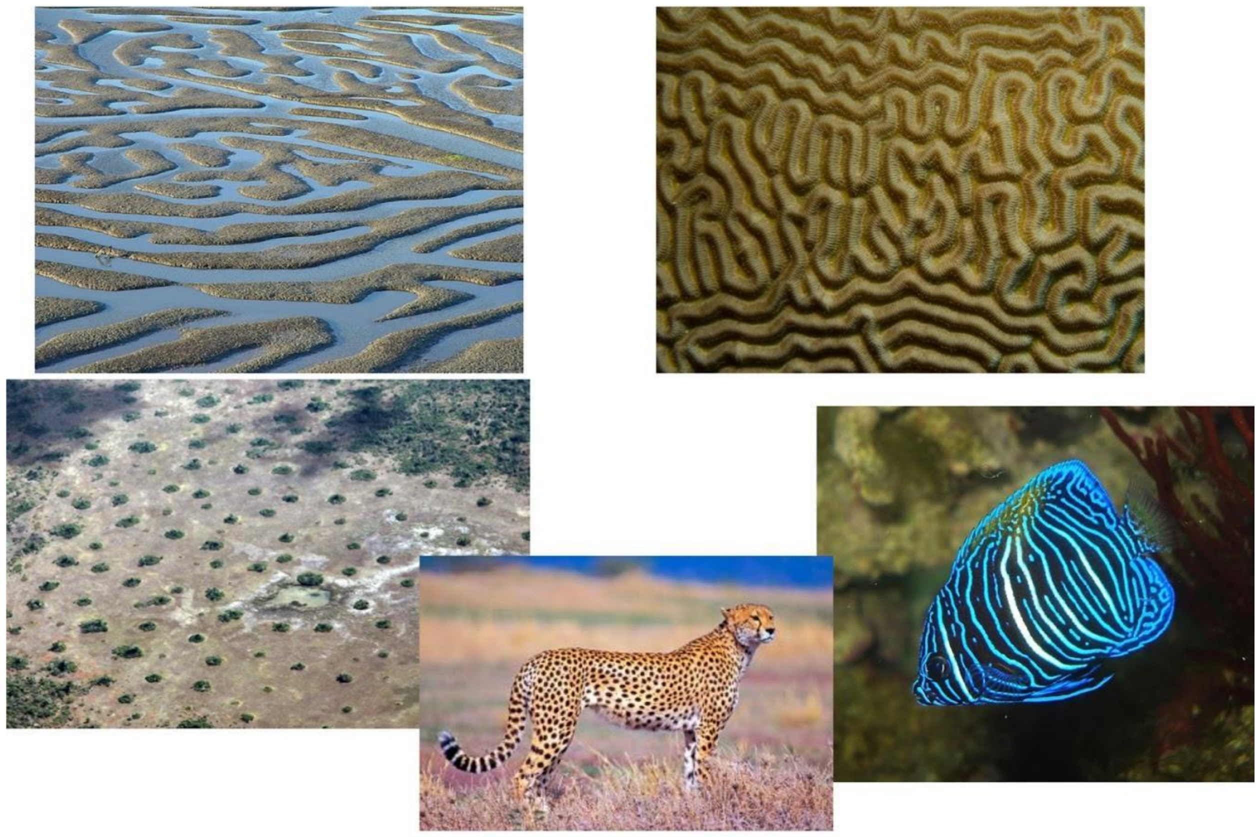

P = 0.15 P = 0.24 P = 0.25

Notice: as we increase the initial percentage of black "panic" cells, we get different results. Below 25%, the system stabilizes to patches of panickers in a background of non-panic. As we raise that percentage above 25, we can see something different start to happen. Something potentially concerning depending on what you might really be modelling here — panic, disease, etc. With only a small change in initial conditions, we go from continuously reaching a stable state where at least some of the system was in a non-panic mode, to global, homogenous panic. This is like a phase transition — a small step in parameter space produces a nonlinear jump in solution space.

You can imagine many scenarios where something like this is going on and people are busy doing linear regression on the outcomes. Of course, if you did something like that you'd totally miss that there's a jump in system behavior. Imagine somebody studying behavior at P = 0 through P = 0.20 and drawing their linear regression (in green). That person would think that the situation is better than it actually is, and fail to properly model the behavior of the system (in black).



Let’s now look at some more intricate patterns found in nature:



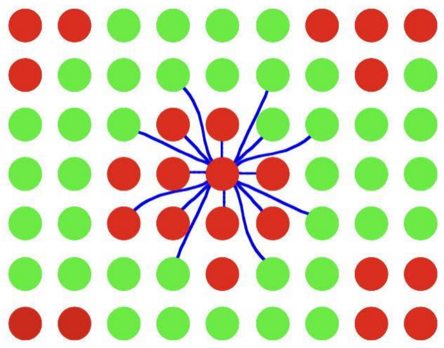
All of these systems leverage the same mechanism, despite the fact that they’re materially quite different from one another. A key feature in all of them is that they’re thermodynamically open. To get something like that pattern with sand and water, for example, you need tides flowing in and out. When the water flows out, it tends to pool up into channels. It digs that channel deeper, which makes the banks next to that channel higher. The pattern forms due to this process happening over and over.

Local Activation, Long-Range Inhibition & Morphogenesis

Interestingly, this is the second time that Alan Turing is going to come up in our discussion. He's known for two things that seem to be very different: 1) the development of the **Turing machine**, thereby laying the groundwork for modern computing, and 2) coming up with a model for pattern formation in distributed systems — the **reaction–diffusion theory of morphogenesis**, or **Turing patterns**.

> **Morphogenesis** is a fancy way of saying "organism development". One thing to note with Turing's work is that it only accounts for patterns on patches of tissue, not the overall development of the organism. There's something still missing from our understanding of morphogenesis — we still don't have a unified theory that accounts for it in its totality.

Of course, Turing was dealing with continuous state-space models, but for the sake of simplicity, let's look at a discretized equivalent. The mechanism involves local activation and long-range inhibition. Basically, every cell wants to be like its closer neighbors, and unlike its further neighbors.



Again, if we take one of the colors to be negative and the other positive, we can write up the behavior in an equation that determines the sign of our cell in question.

$$x_i = sign\left(J_1 \sum_{|i-j|<r_1} x_j + J_2 \sum_{r_1<|i-j|<r_2} x_j + h\right)$$

In this case, we have two distances that account for near neighbors and far neighbors.

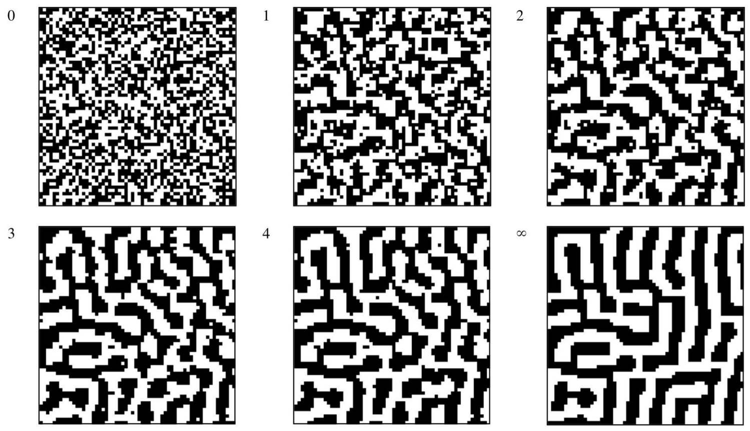
In the case of near neighbors, if the distance between cell i and cell j on the grid — |i-j| — is less than some radius r_1, add up the x_js and scale them by a (typically positive) coefficient J_1.

In the case of far neighbors, if the distance between cells i and j on the grid is between some radii r_1 and r_2, add up the x_js and scale them by a (typically negative) coefficient J_2.

We also have a third parameter, h. This is just like having a 2 as a bias in our panic rule equation, except more general.

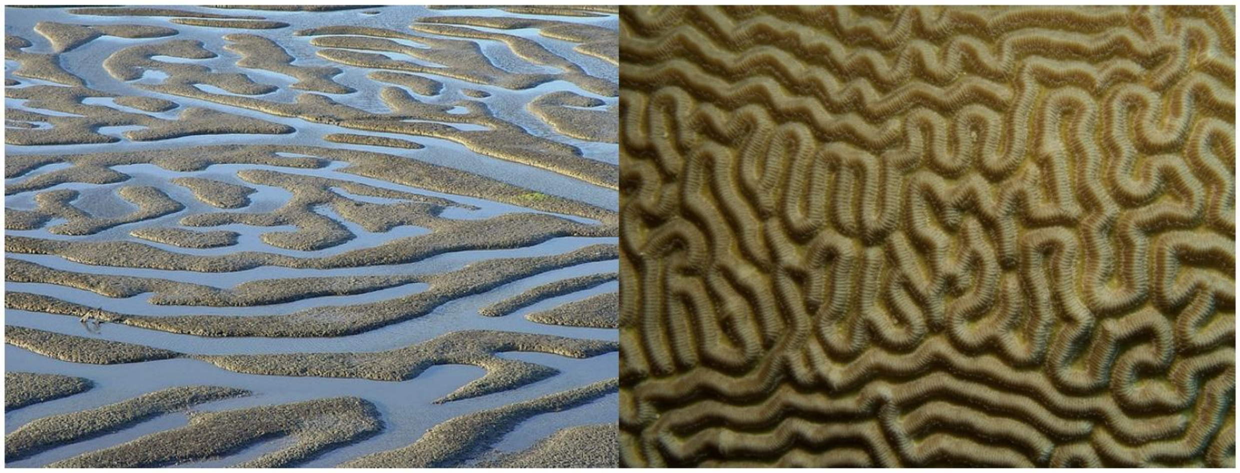
Adding up these three components gives us the sign of cell x_i.

Here's an example run of our local activation, long-range inhibition model. We start with some random initial condition (step 0), and let the model run (steps 1 through ∞). Play with it yourself (the parameter you control here is *h*).



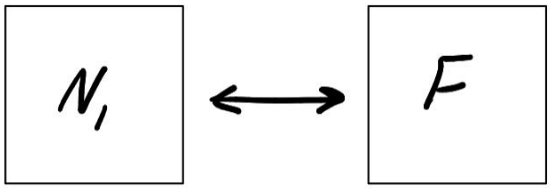
Universality & Formal Analogies

Let's explore the notion of a **universality class**. Just like we said that multiple systems can map to a Turning machine, here we see multiple systems mapping to the same kind of pattern despite being materially different.

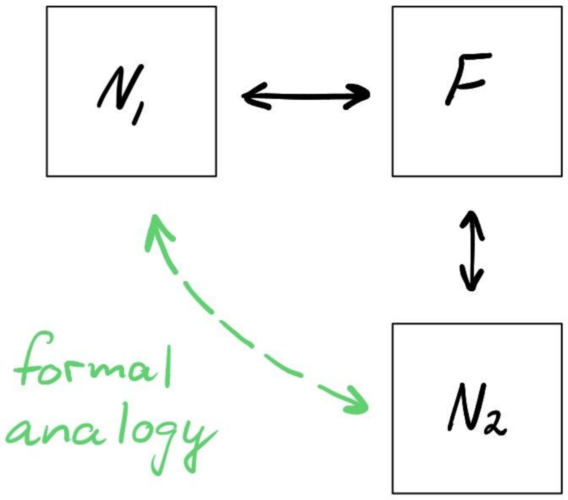


When physicists talk about universality classes, this is what they mean; totally different systems share some common mechanistic dynamic that leads to the same kinds of patterns and behaviors.

Rosen talks about universality in terms of "formal analogies". Let's say we have some **formal system** F (a formal system being a mathematical or computational system that evolves strictly according to some ruleset). Instead of causality, formal systems have something called **inference**. Given a current state of the system and a ruleset by which it evolves, we can infer some future state. Inference in a formal system is equivalent to causality in what we might call natural systems. Typically, when we model a natural system mechanistically (say, N_1), what we're doing is establishing some mapping between it and F — the causality of N_1 gets mapped to the inferential structure of F.



If we now take some different natural system, *N_2*, and map it to the same formal system *F*, then what we've developed is what Rosen calls a **formal analogy** between these systems.

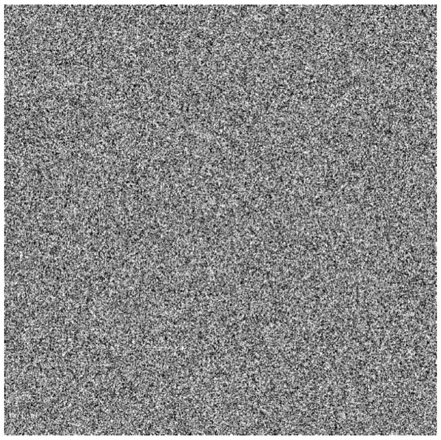


In a way, *N_1* and *N_2* belong to the same universal class. However, it's also worth noting that there might be important differences between the systems — just because some part of them can be described by the same formalism, it doesn't imply that they're equivalent in every way.

Symmetry Breaking

One other concept to touch on here is one mentioned in the Anderson paper, *'More is Different'* — the notion of **symmetry breaking**. When you encounter it in the physics literature, it’s often presented in a way that’s hard to grasp, but it actually turns out to be quite simple.

Let’s say we have a system of black and white cells with this initial condition:



Here, the distribution of black and white cells is approximately 50:50. With a majority rule and a perfect balance of black to white cells, the system won’t evolve at all. In other words, it will preserve symmetry. However, given even a slight imbalance, the system will tip towards either the black or the white, breaking the symmetry.

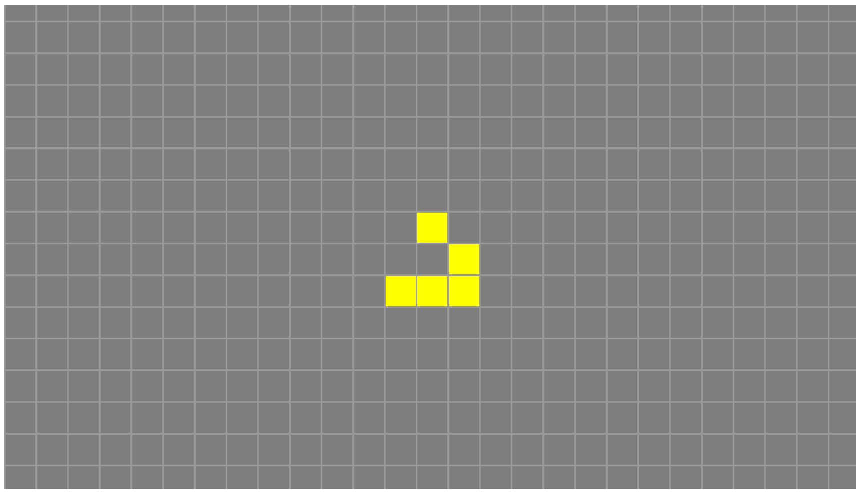
It’s kind of like the example of a ball on a hill. The 50:50 distribution is an unstable equilibrium, where a slight nudge will completely change the state of the system.

The Game of Life

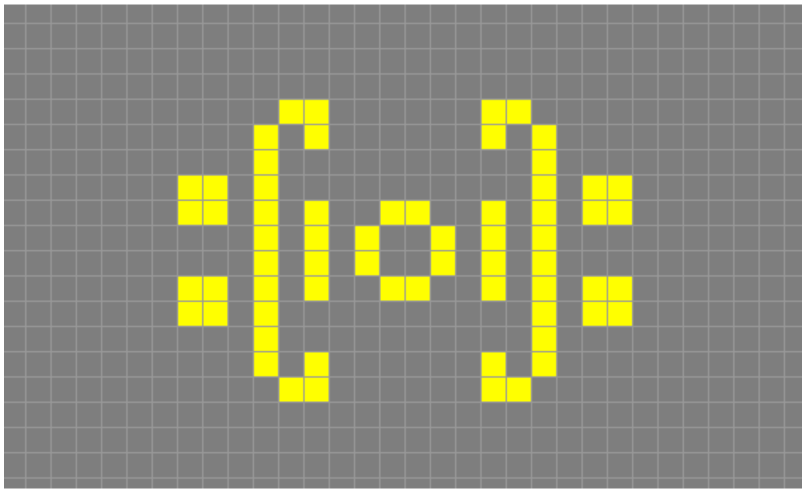
Another type of cellular automata — a very famous one — is the Game of Life, created by mathematician John Conway. The rules are very simple, and evolve on a 2D grid:

1. Any live cell with fewer than two live neighbours dies, as if by underpopulation.
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies, as if by overpopulation.
4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Play with it yourself!



Let’s look at an example configuration. The above configuration will slide across the screen, with its structure preserved. Notice that this is somewhat analogous to locomotion, yet nowhere in the ruleset is locomotion specified. This behavior emerges out of the organization of the system.



Here's another organization of cells — this one happens to be stationary. As we saw before: same rules + different organization → different behavior.

The Game of Life is also another example of a Turing-complete system. Given that a Turing-complete system can perform any computation, you can actually run the Game of Life as a metalayer inside the Game of Life! Watch the video for yourself — showing first the micro-dynamics, then zooming out to show the emergent Game of Life dynamics coming out of the micro-dynamics.

Organization

The kind of self-organization that we looked at in this module is the norm in the world we live in — most order in the world is not dictated by some central agent.

We saw that complex behavior can emerge from simple rules. It's also worth mentioning that SIMPLE behavior can emerge from complex agents (e.g. panic). People, for example, are quite complex in and of themselves, but when you get them together and they start having interactions with each other, they might behave quite simply. We often highlight the fact that complexity can emerge out of simple rules, but it's worth noting that it can work the other way around as well.