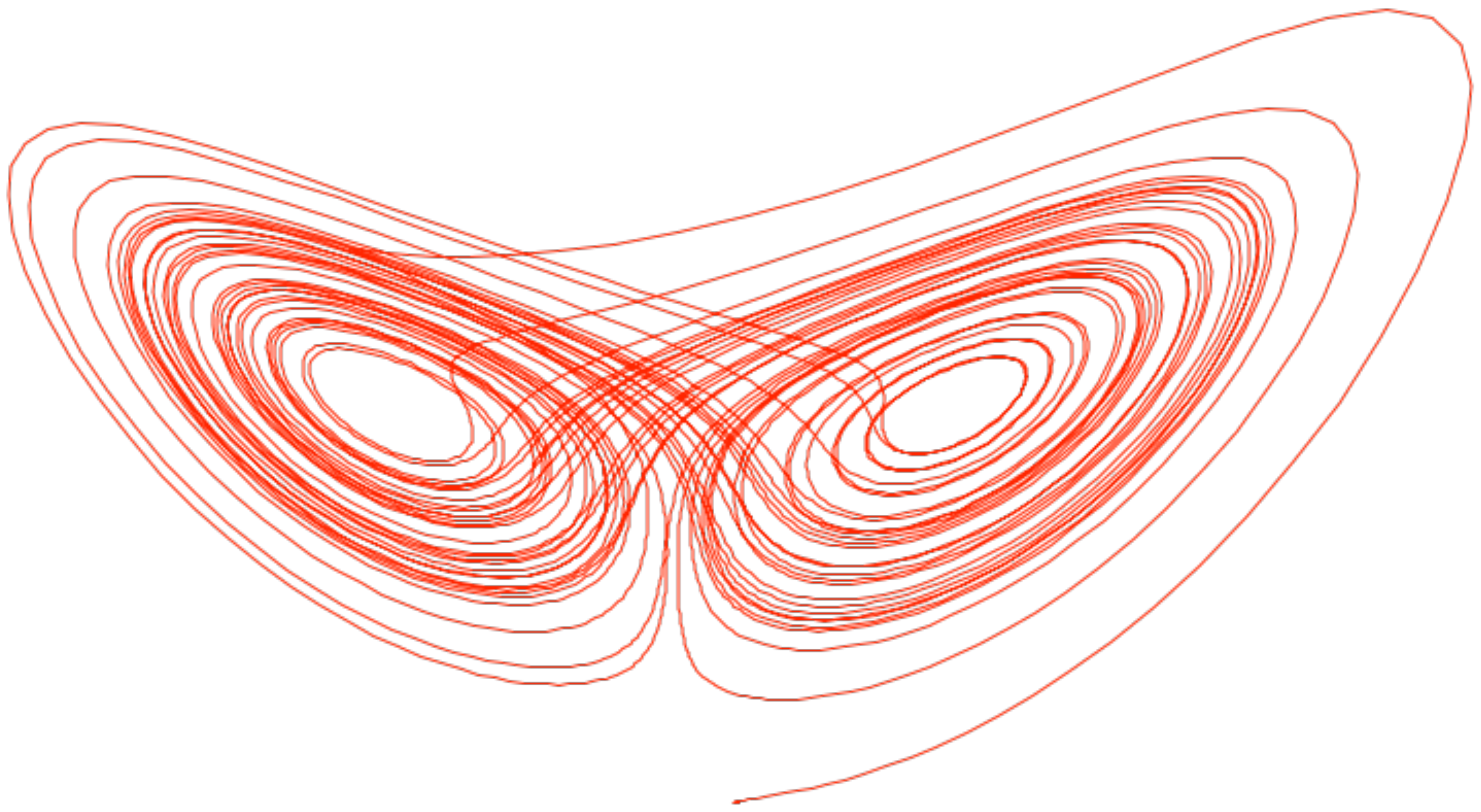
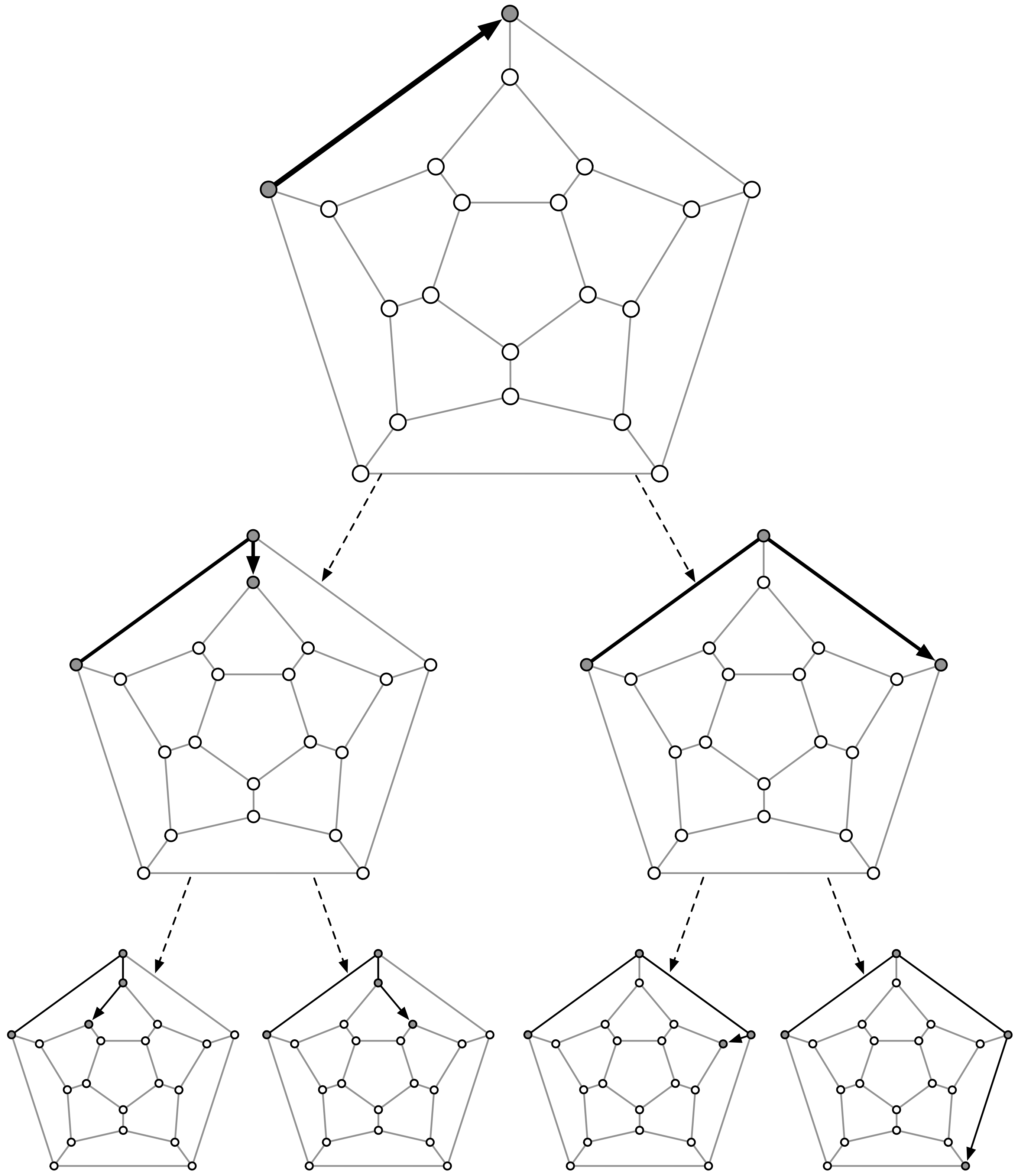
Universality, Hardness, Engineering, and Messiness

Cristopher Moore
Santa Fe Institute

Do we have to simulate?



Do we have to search?

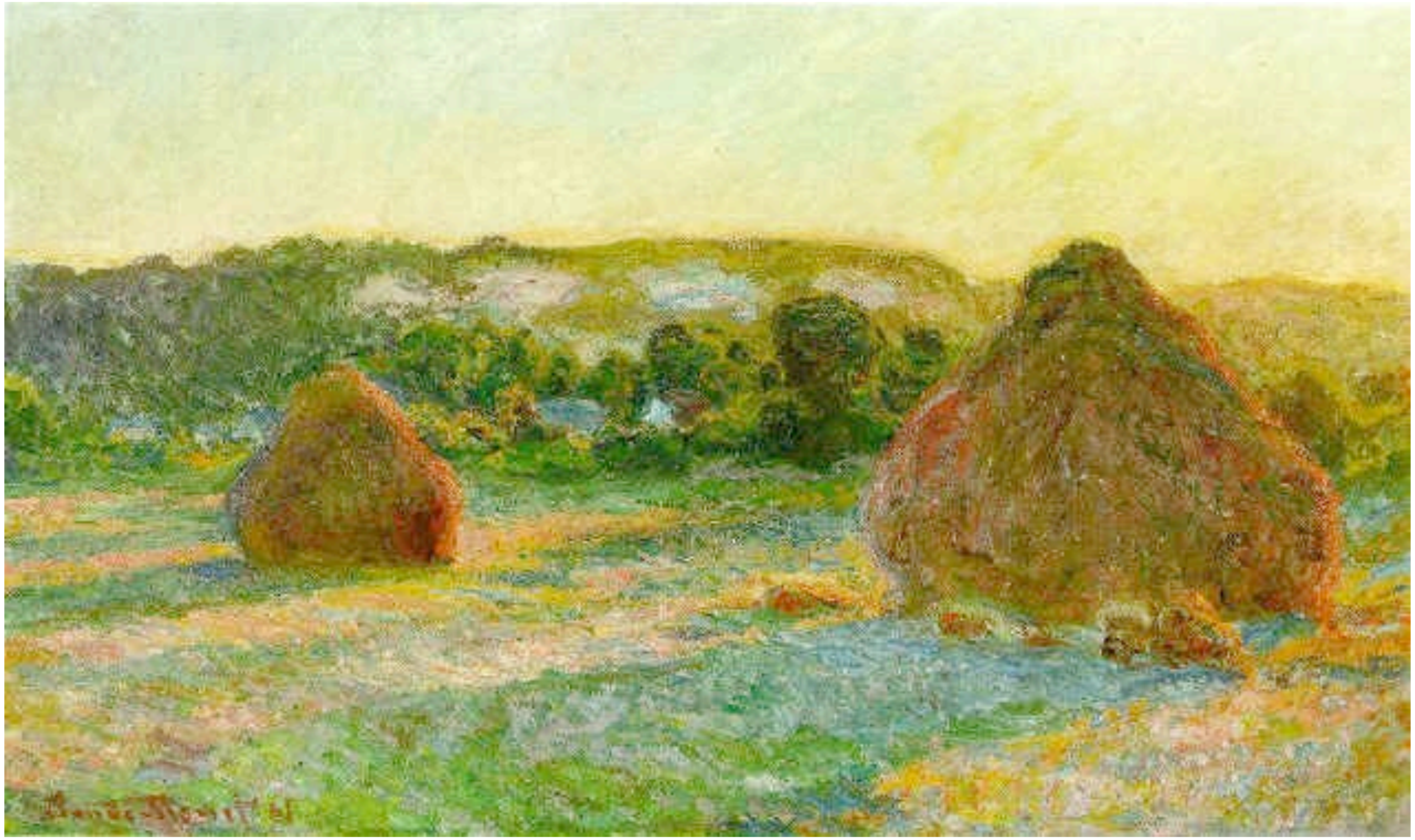


Story #1: Worst case vs. average case

Needles in haystacks

P: we can find a solution efficiently

NP: we can *check* a solution efficiently: tilings, tours, proofs...



NP-complete problems

Some problems are “universal” in that they capture all of NP: any problem in NP can be converted to them

3-SAT: Boolean variables $x_1, \ldots, x_n$

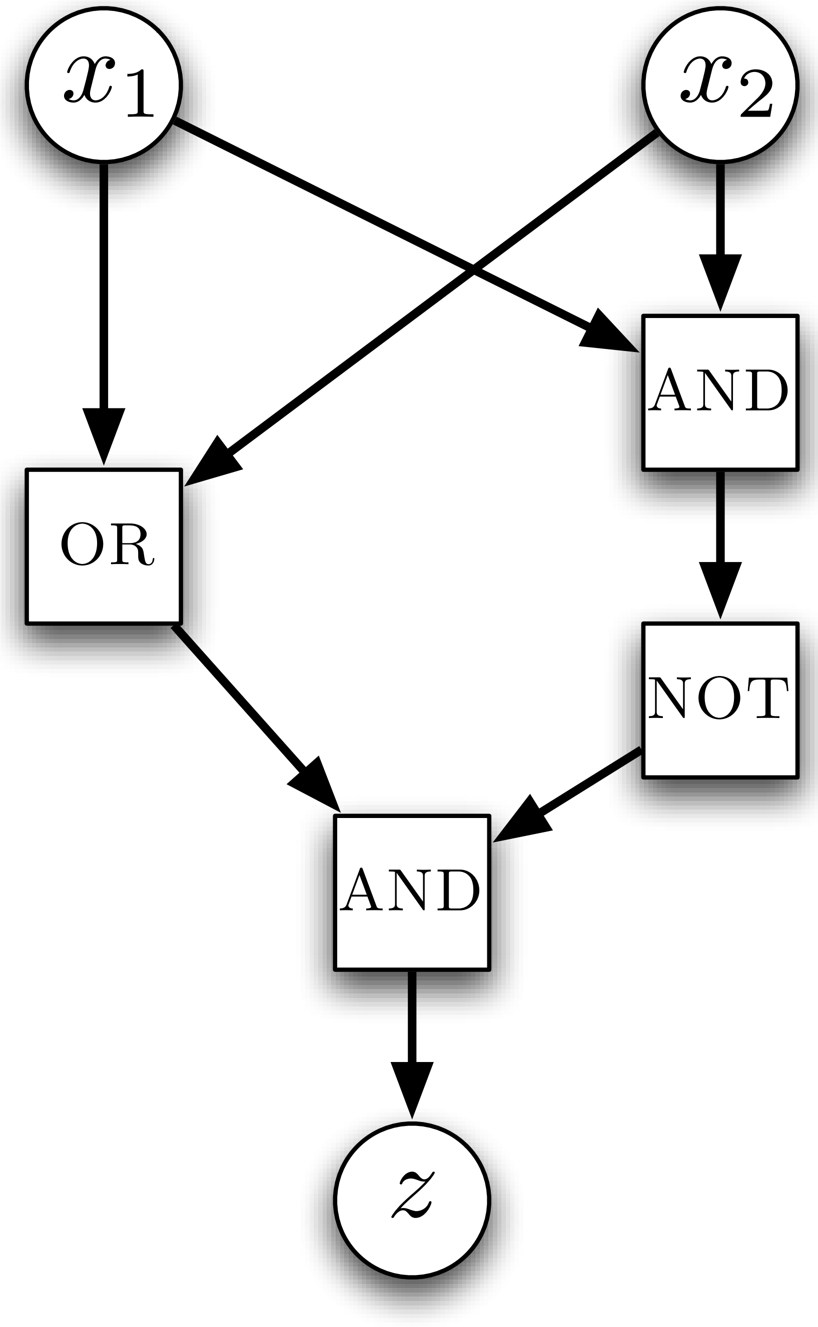
Constraints $(x_1 \vee \overline{x_3} \vee x_6) \wedge (x_3 \vee x_4 \vee \overline{x_{17}}) \wedge \cdots$

Is this formula satisfiable? That is, is there a truth assignment for $x_1, \ldots, x_n$ that satisfies all the constraints?

We can build a computer out of Boolean gates

Take any problem in NP, like Hamiltonian Path

Any program that tests proposed solutions can be “compiled” down to a Boolean circuit, that outputs “true” if the solution works



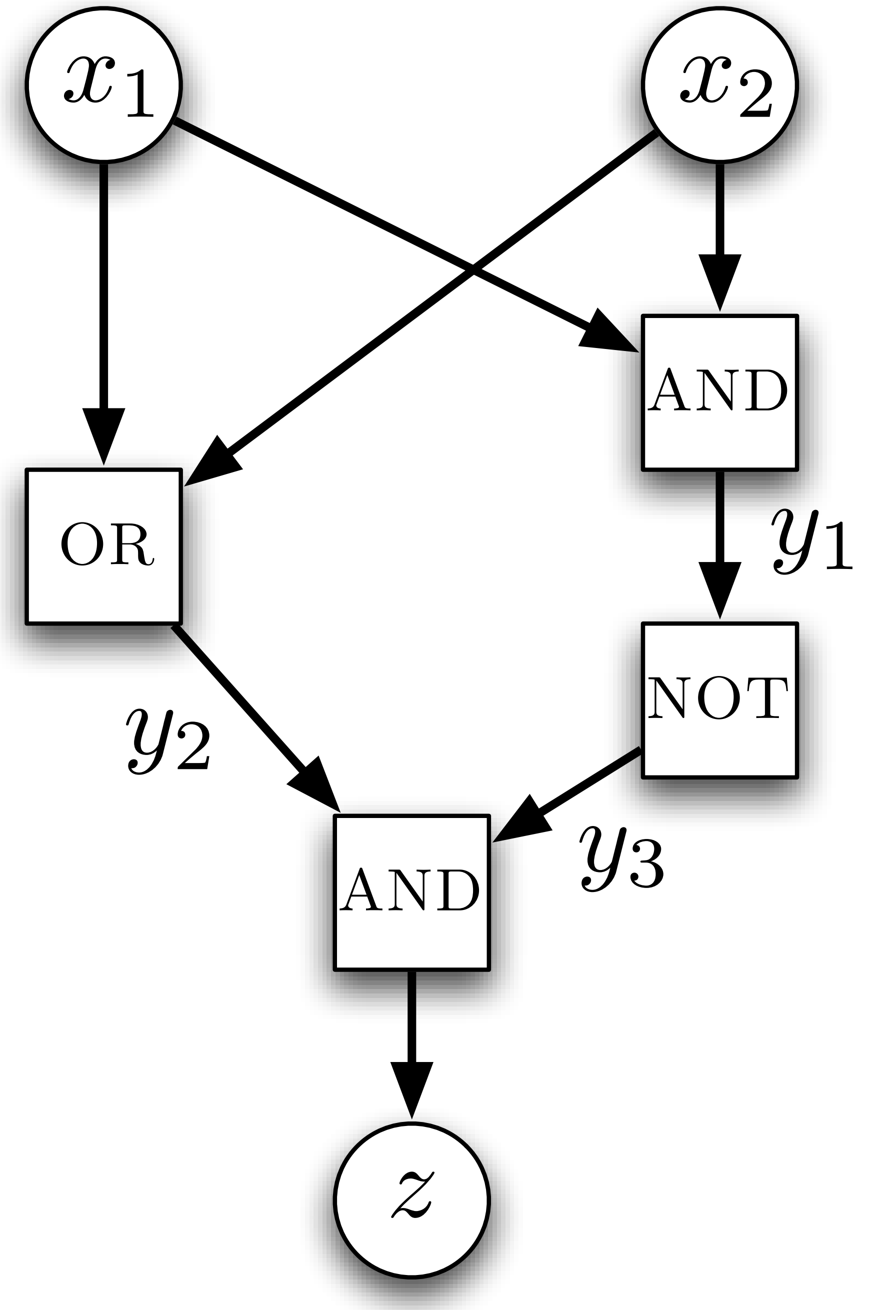
We can build a computer out of SAT clauses

Add variables representing the truth values of the wires

The condition that each AND or OR gate works, and the output is “true,” can be written as a Boolean formula:

$$(x_1 \vee \overline{y}_1) \wedge (x_2 \vee \overline{y}_1) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee y_1)$$
$$\wedge \cdots \wedge z \;\; .$$

This formula is satisfiable if and only if a solution to the original problem exists



Oh, cruel world!

NP-completeness is a worst-case notion...

We assume that instances are designed by a clever adversary to encode hard problems

A good assumption in cryptography, but not in most of nature

The scientist is always working to discover the order and organization of the universe, and is thus playing a game against the arch-enemy, disorganization. Is this devil Manichaean or Augustinian? Is it a contrary force opposed to order or is it the very absence of order itself?

— Norbert Wiener, *Cybernetics*

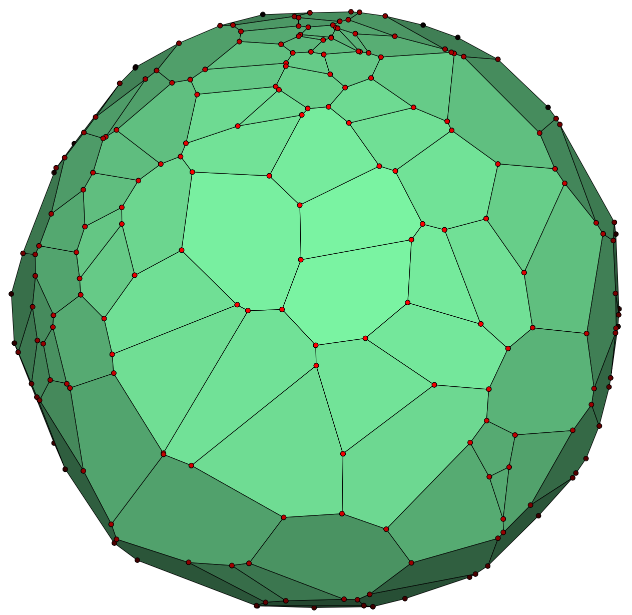
Alternatives

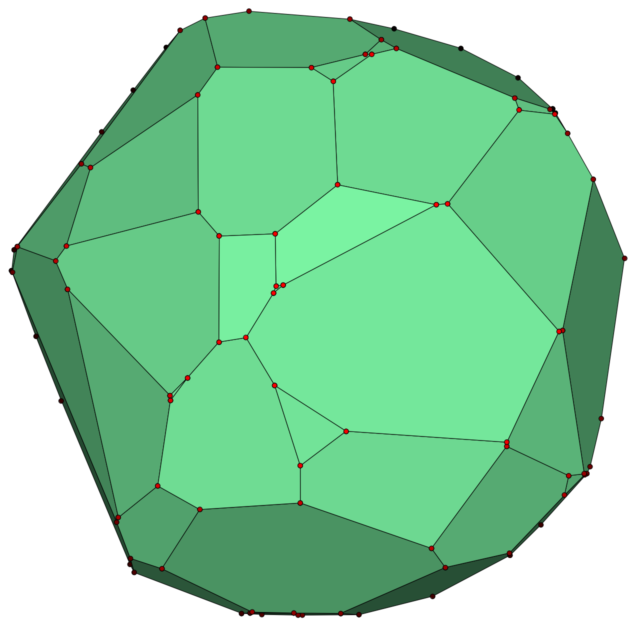
Probably Approximately Correct [Valiant]

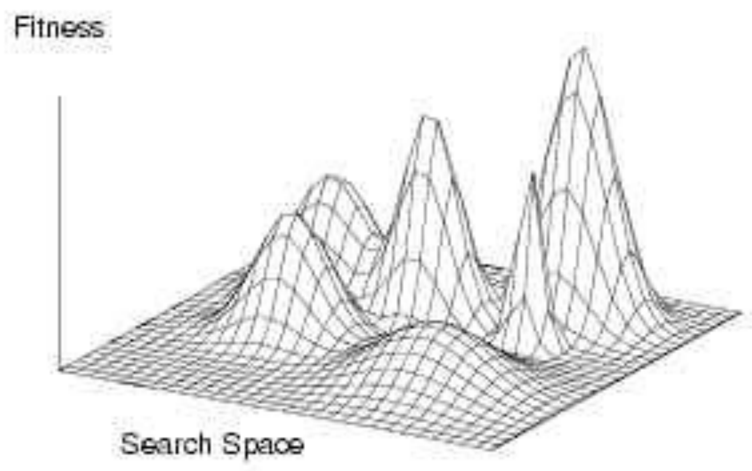
Noise can foil the adversary [Spielman and Teng, smoothed analysis]

Landscapes are not as bumpy as they could be: good solutions are close to the optimum [Balcan, Blum, and Gupta, clustering]

In nature, problems and algorithms coevolve (e.g. protein folding)







Random problems and phase transitions

What if the constraints are chosen randomly instead?

Inspired by spin glasses, random graphs

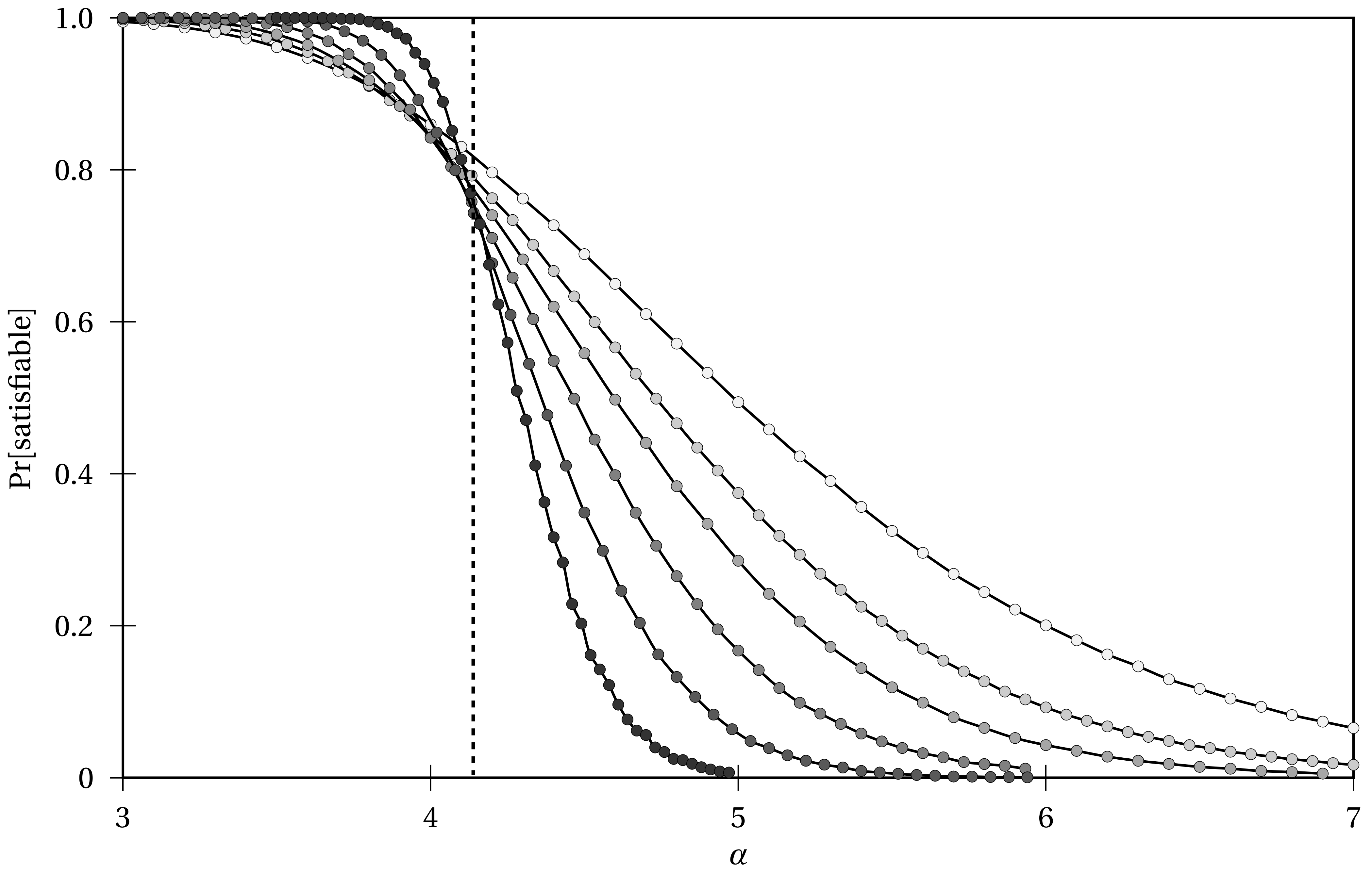
As we add more constraints, more contradictions arise

When the density α=(# clauses)/(# variables) crosses a critical threshold, a sudden drop from satisfiability to unsatisfiability

Like water freezing or iron magnetizing: analogies with statistical physics

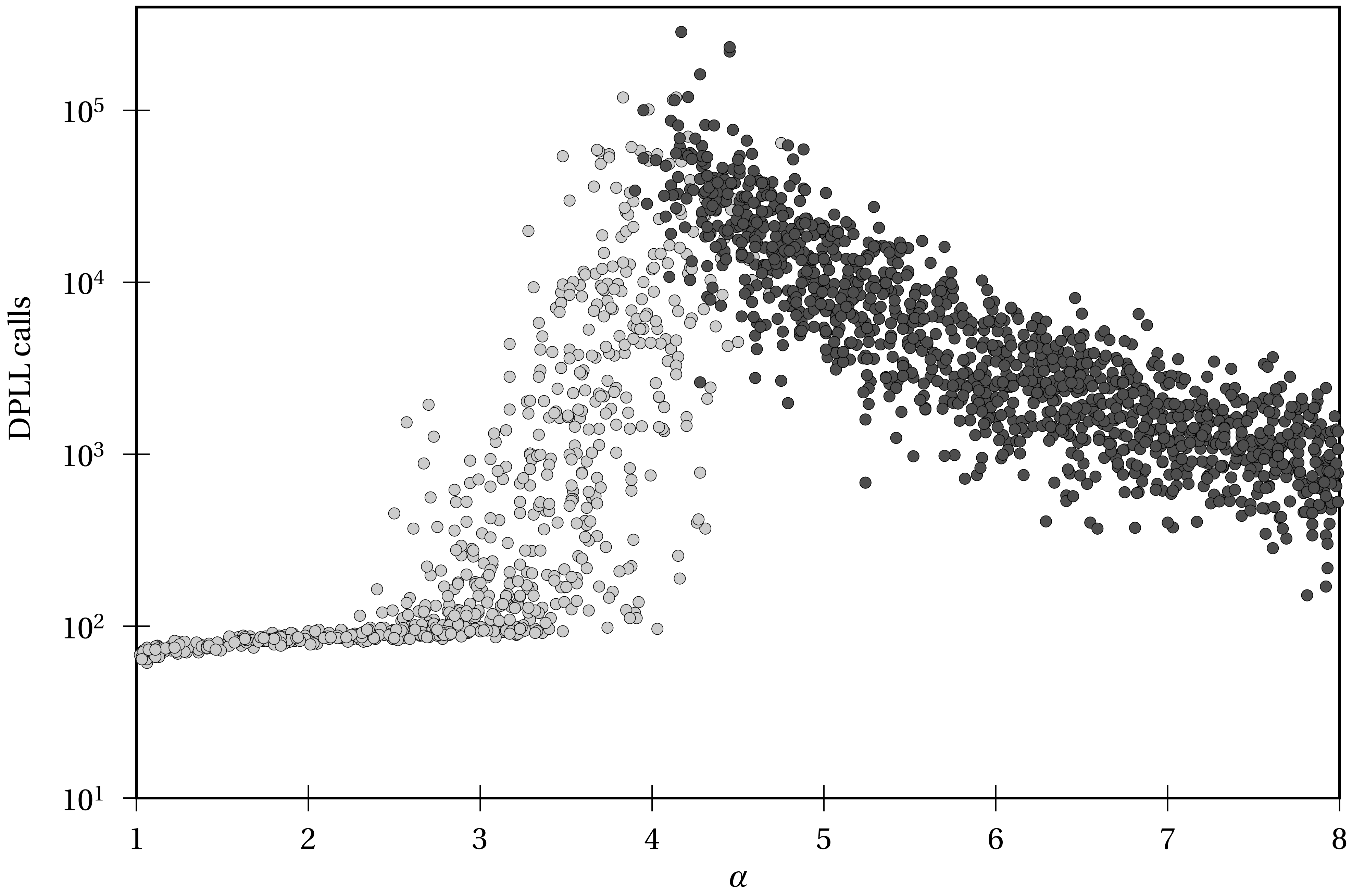
Random problems and phase transitions

The probability of satisfiability as a function of density

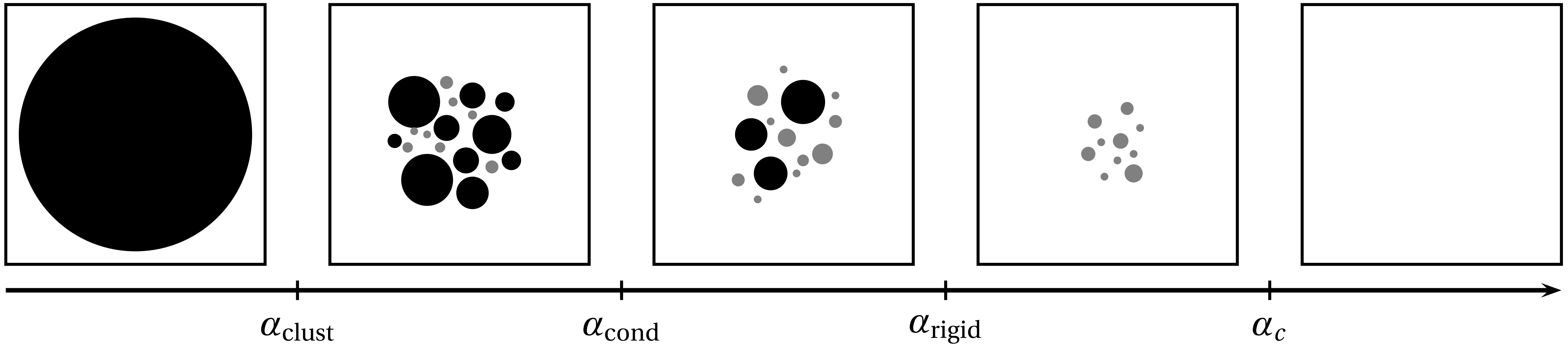


Where the hard problems are

Search times are highest at the transition



What makes a problem hard?



At a certain density, solutions break up into clusters

These clusters become “rigid” or “frozen” — many variables take a fixed value

If a search algorithm sets any of these variables wrong, it’s doomed, but it takes an exponentially long time to realize it

Between α_{rigid} and α_c, there are solutions, but (we believe) they are hard to find

[Achlioptas, Coja-Oghlan, Krzakala, Mezard, Molloy, Monasson, Montanari, Moore, Ricci-Tersenghi, Zdeborová, Zecchina...]

But physics isn't everything...

The statistical properties of a problem don't determine its complexity

Use XOR (addition mod 2) instead of OR:

$$x_1 \oplus x_2 \oplus x_3 = 1$$
$$x_1 \oplus x_2 \oplus x_4 = 0$$
$$x_2 \oplus x_3 \oplus x_4 = 1$$

Random instances have many of the same properties as 3-SAT:
clustering, freezing, and a phase transition to unsatisfiability

But XORSAT is easy! Just linear equations, can solve with Gaussian elimination:

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Story #2: Hard vs. messy

Problems in the gap

To prove that a problem *A* is hard, we build a computer out of the variables and constraints of *A*, showing that *A* is computationally universal

This is almost the only technique we have...

There are problems that are hard but not universal:
undecidable, but easier than the Halting Problem [Friedberg-Muchnik]
outside P, but not NP-complete [Ladner]

These proofs are nonconstructive; they don't produce "natural" problems

But there are natural candidates: Factoring, Graph Isomorphism

NP-hard and Turing-universal problems are **easy...** not to solve, but to engineer

Do we have to simulate?

P: can solve in polynomial time — can predict a sandpile by running it

NC: can solve in poly(log *n*) time with poly(*n*) processors

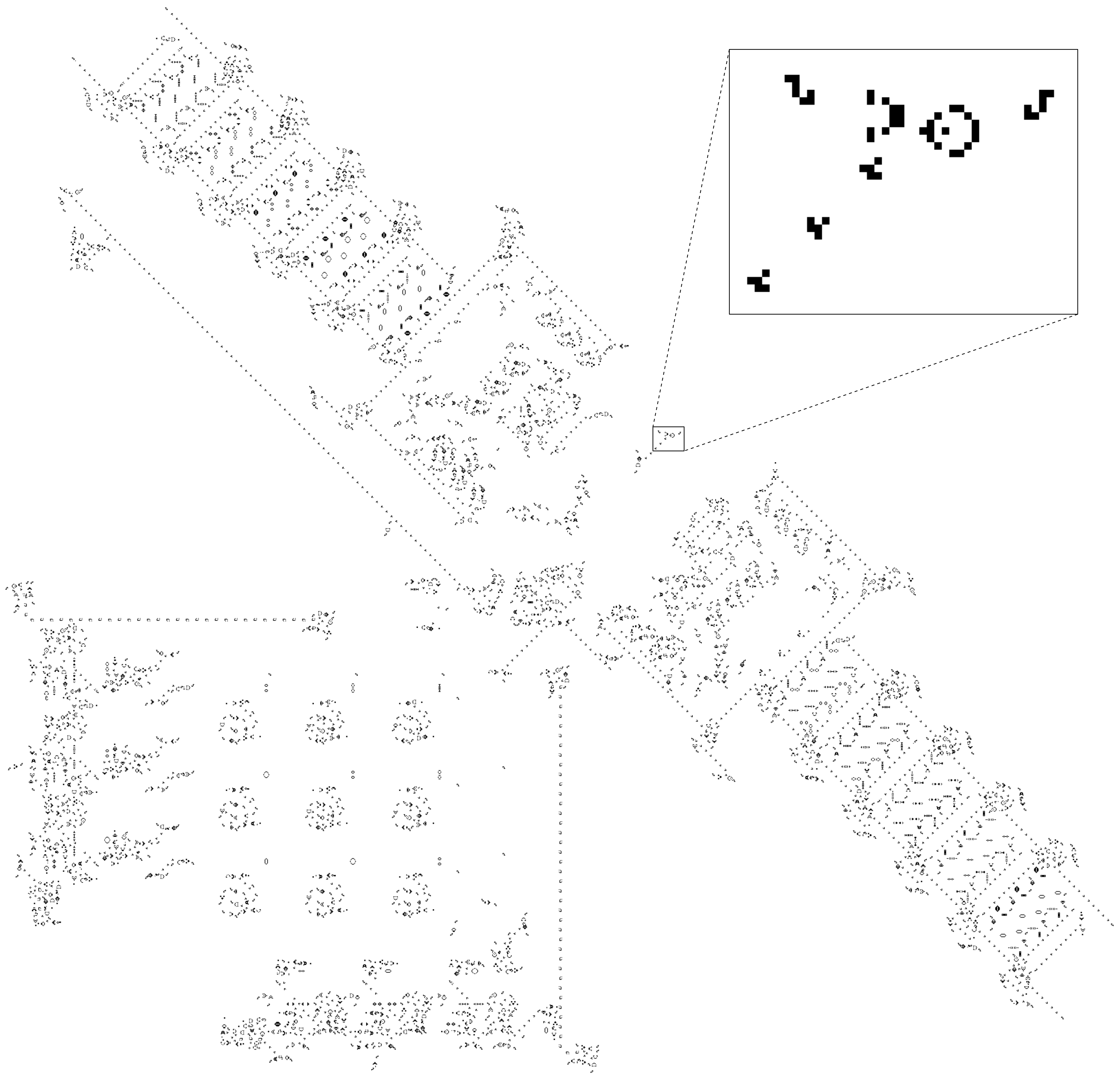
NC vs. P: can every polynomial-time algorithm be efficiently parallelized?
Or are there problems that we have to solve step-by-step?

Depth=time, width=# of processors

Can every circuit of polynomial size and depth be compressed to
poly(log *n*) depth and polynomial width?

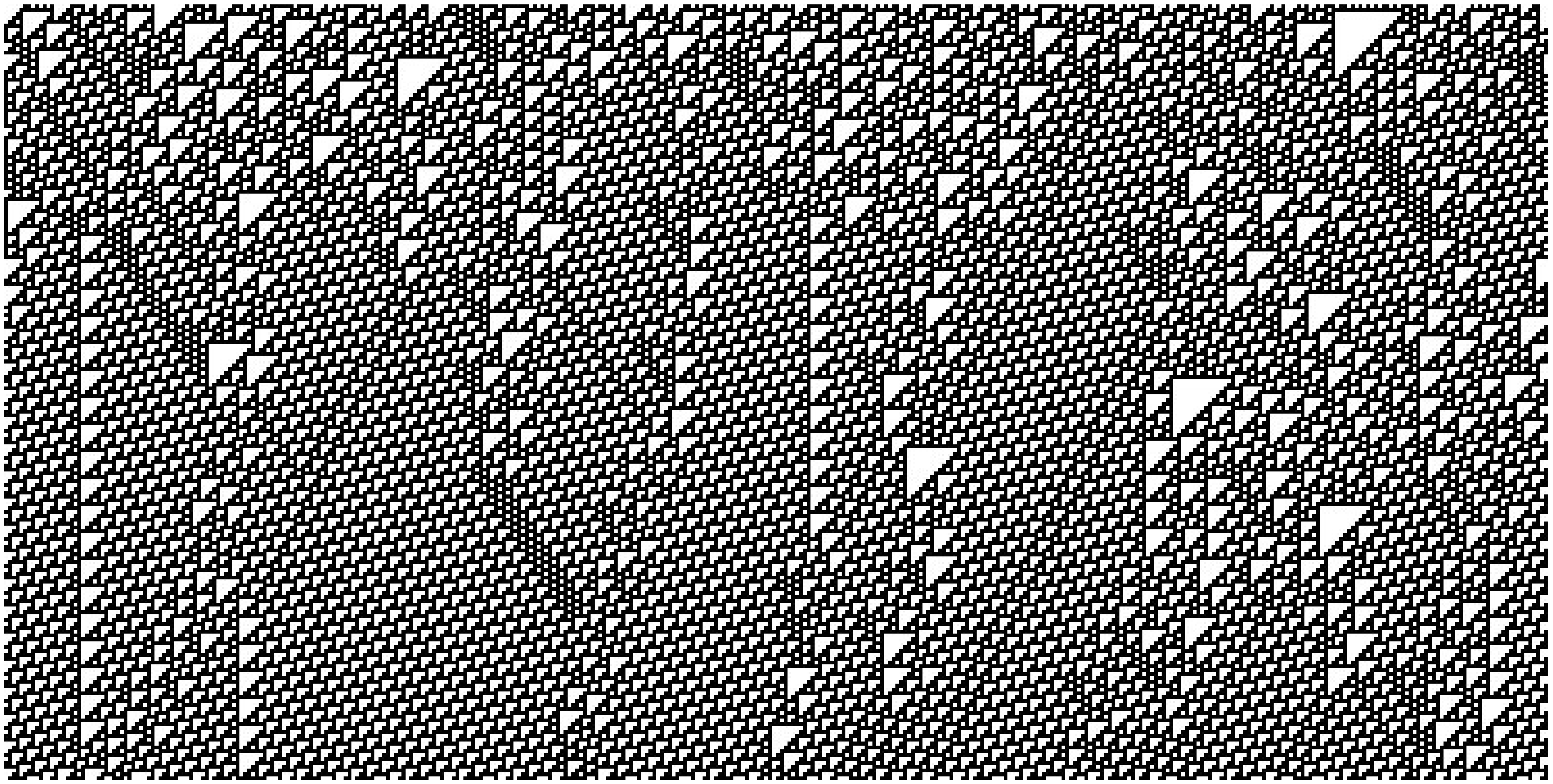
If predicting a system is P-complete, this is evidence that it has to be
simulated explicitly—we can’t skip over its history

Universal cellular automata



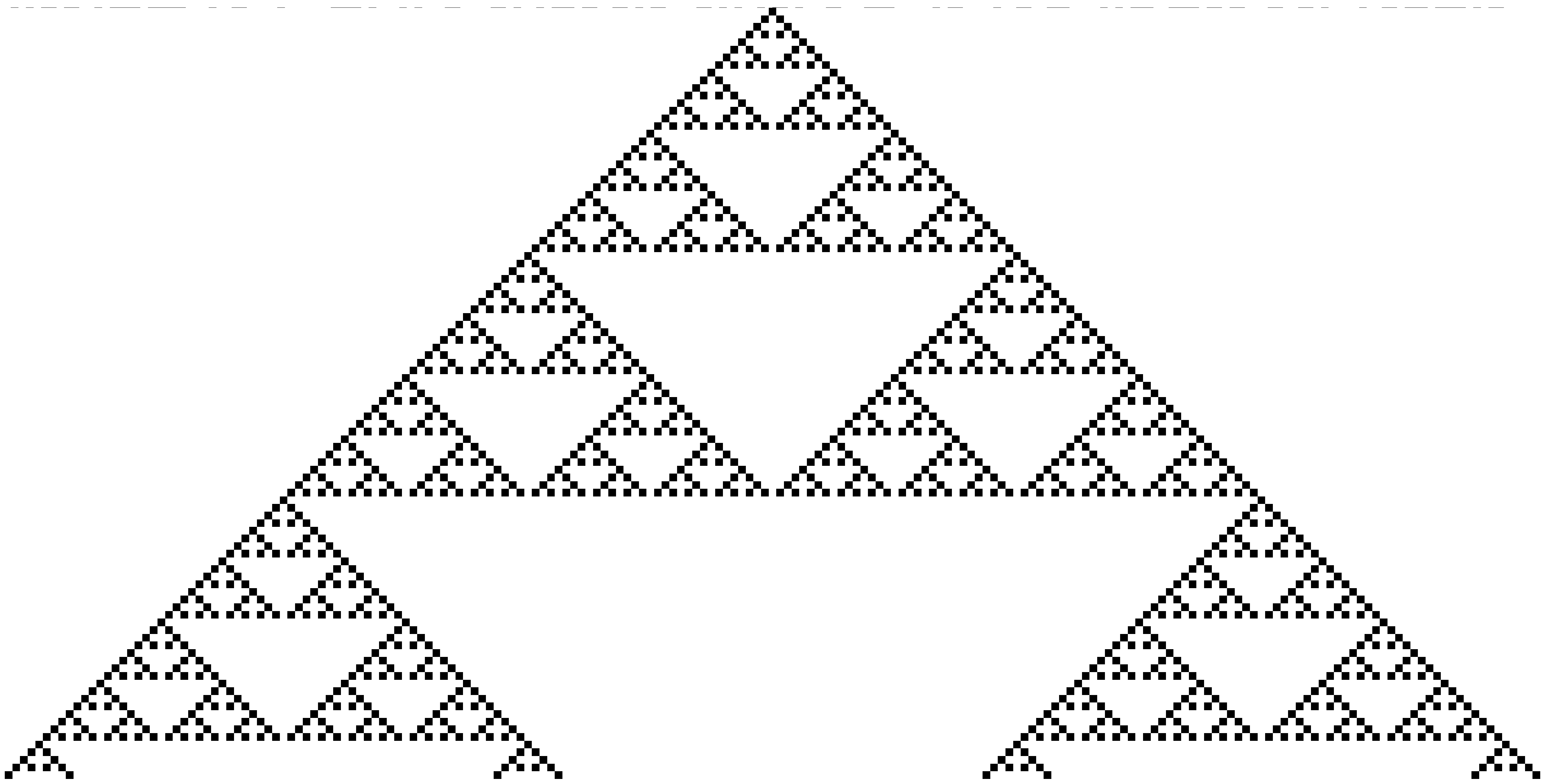
[Rendell]

Universal cellular automata

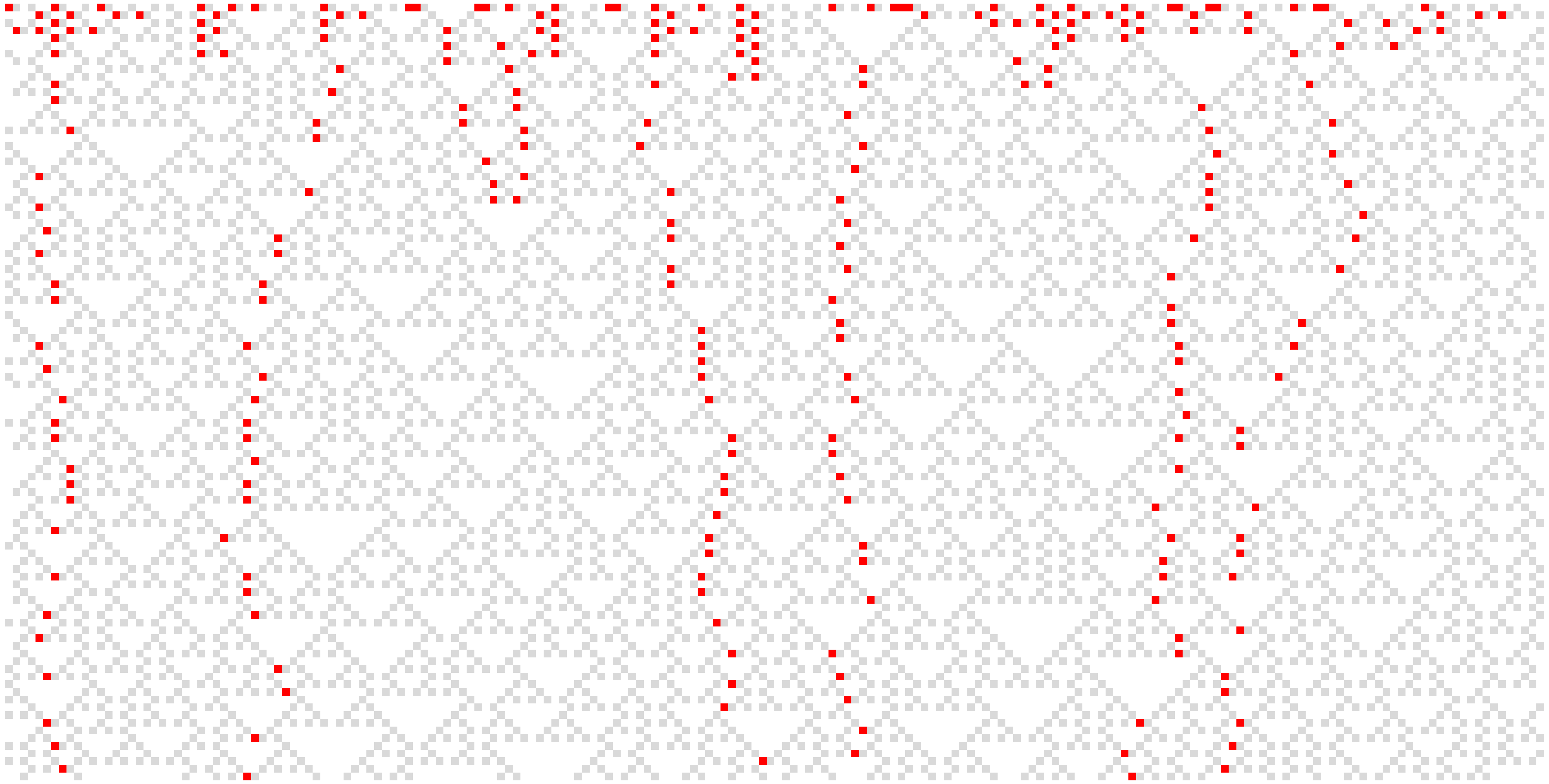


[Cook & Wolfram, Neary & Woods]

Easy cellular automata



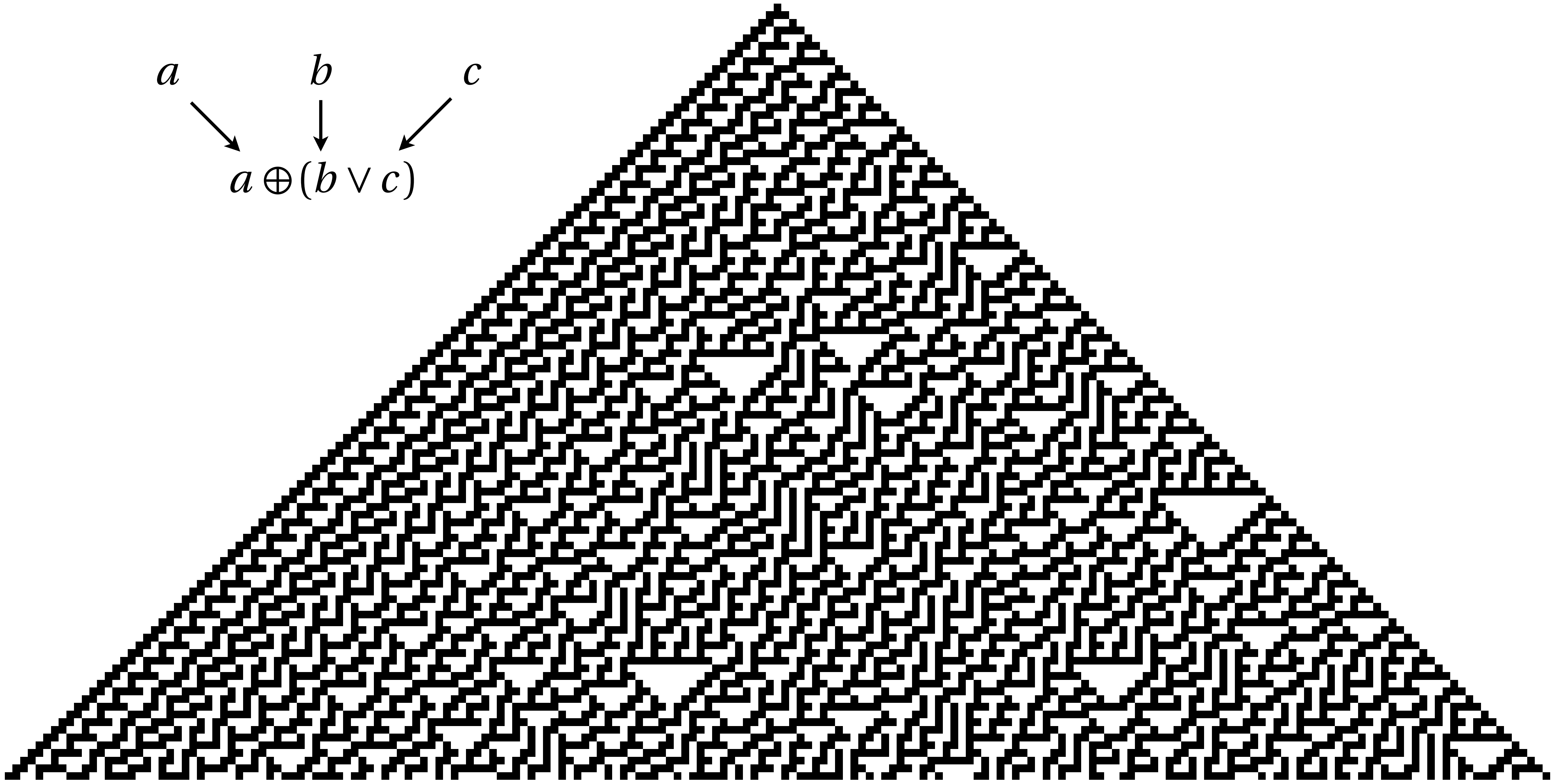
Messy cellular automata



[Jen]

Messy cellular automata

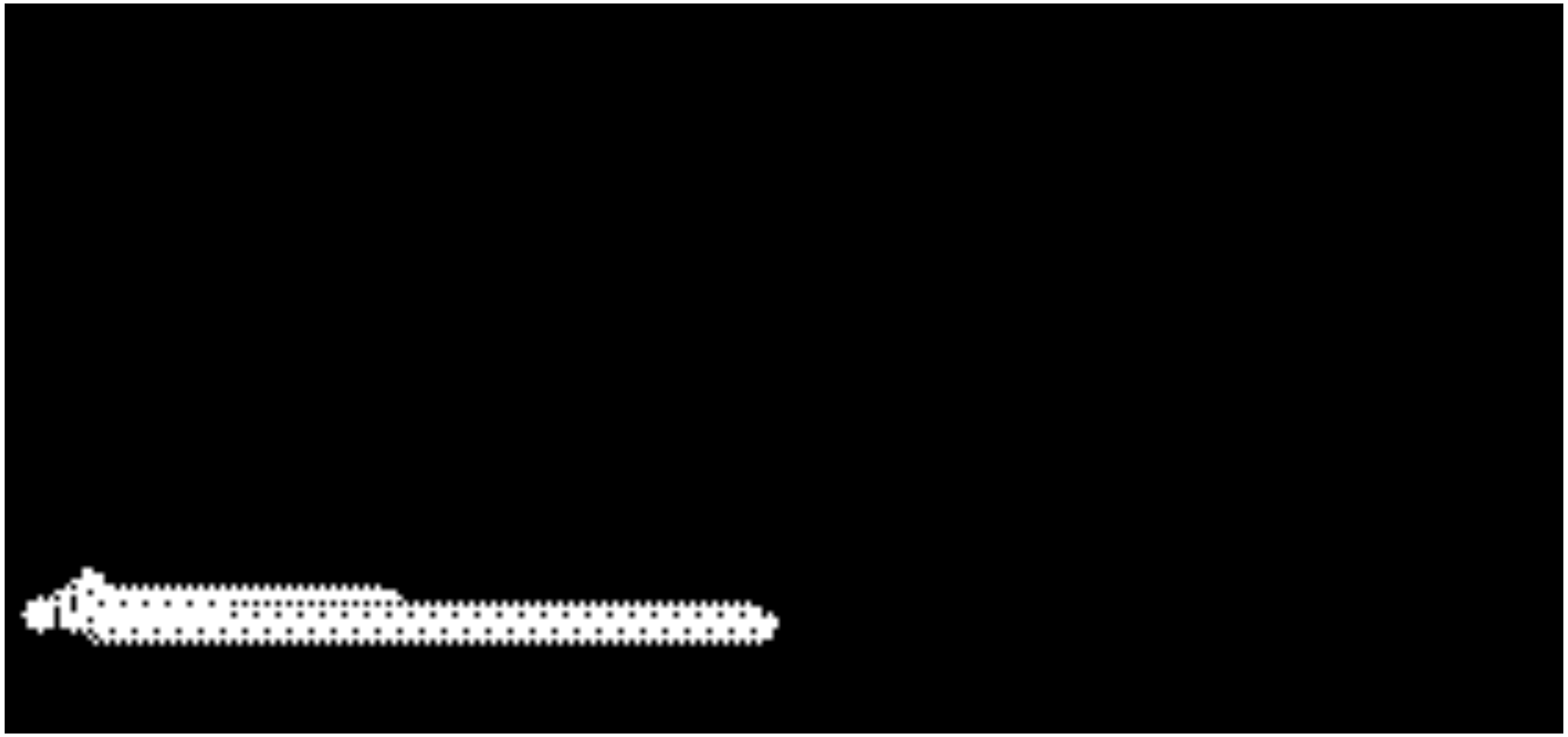
$$\begin{tikzcd} a \arrow[dr] & b \arrow[d] & c \arrow[dl] \\ & a \oplus (b \vee c) & \end{tikzcd}$$



Partly messy

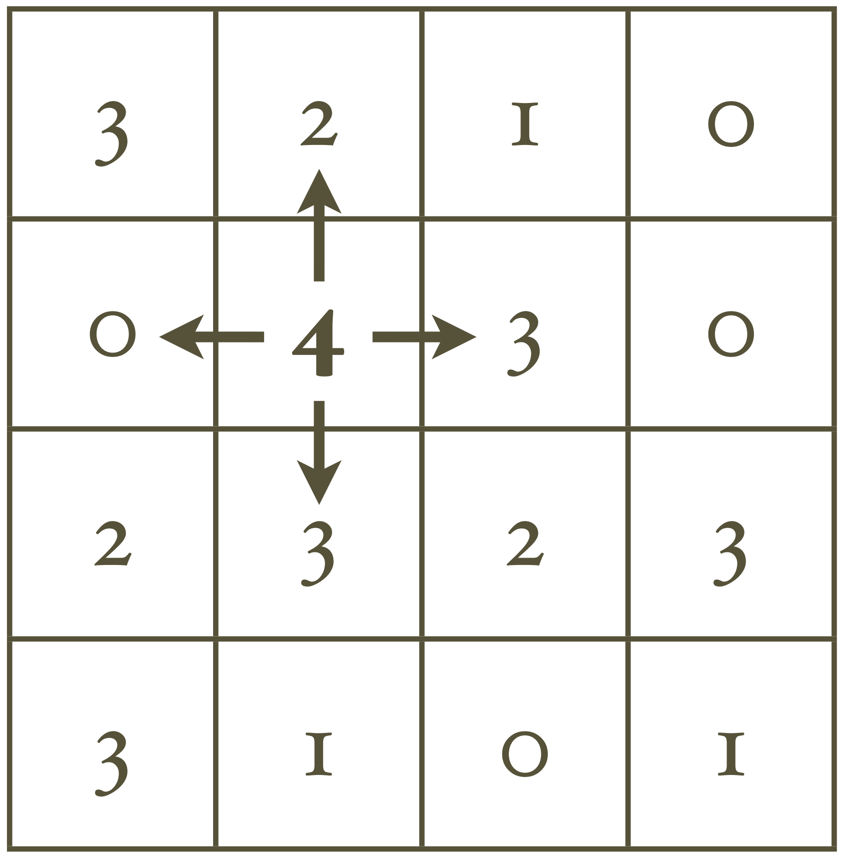
[Griffeath, Moore]

Partly messy

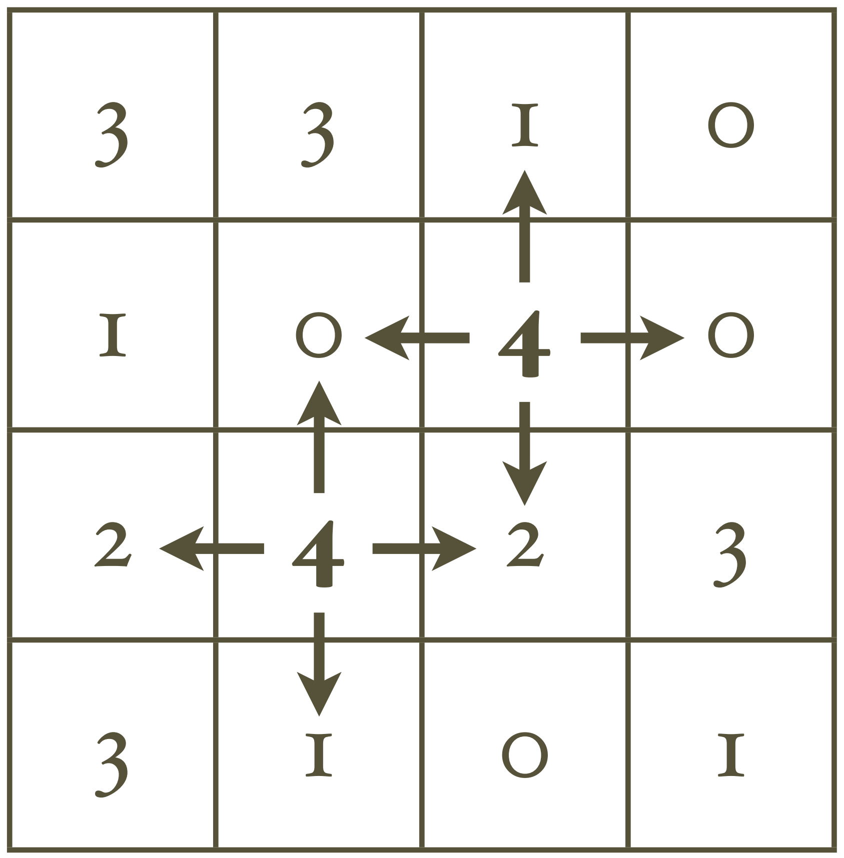


[Griffeath & Moore]

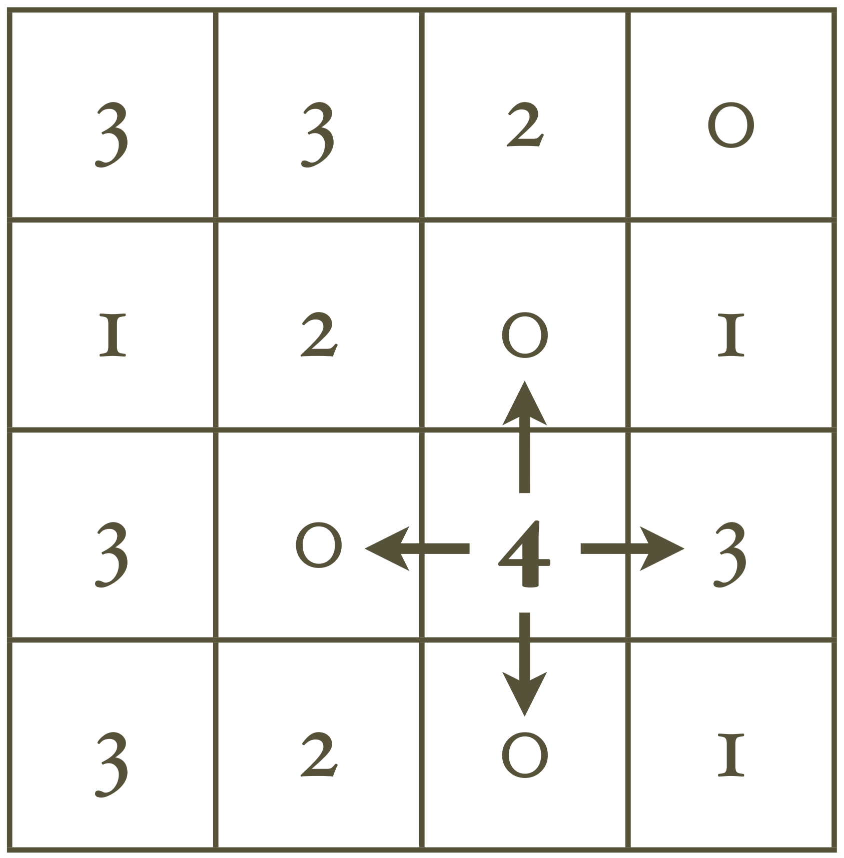
Sandpiles



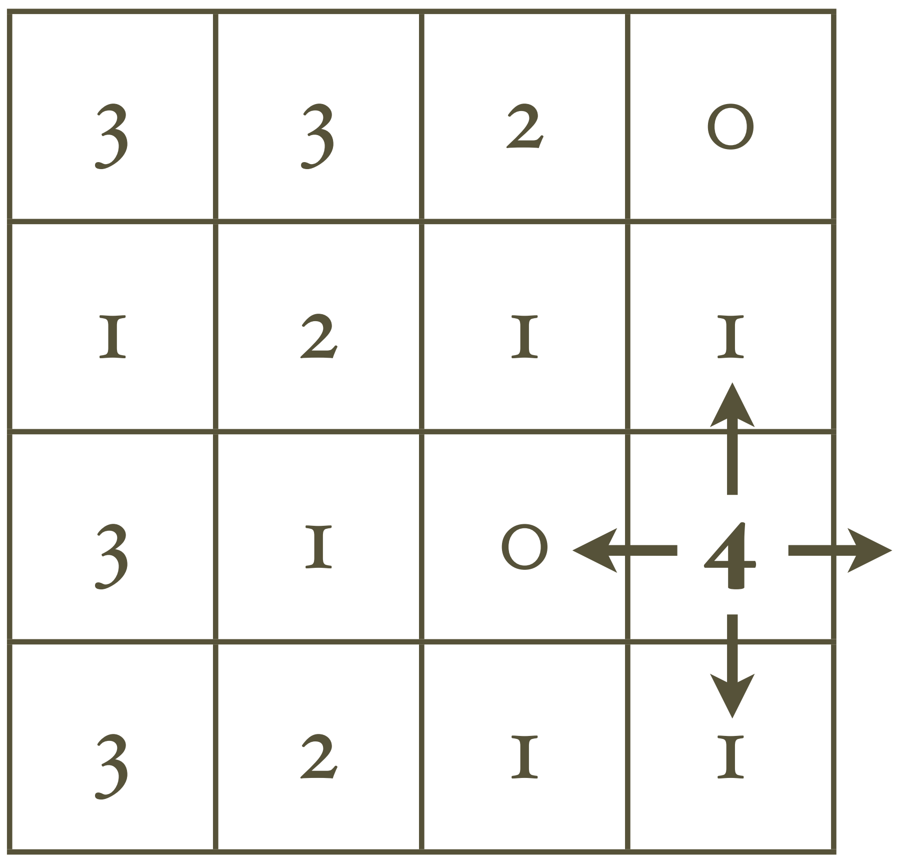
Sandpiles



Sandpiles



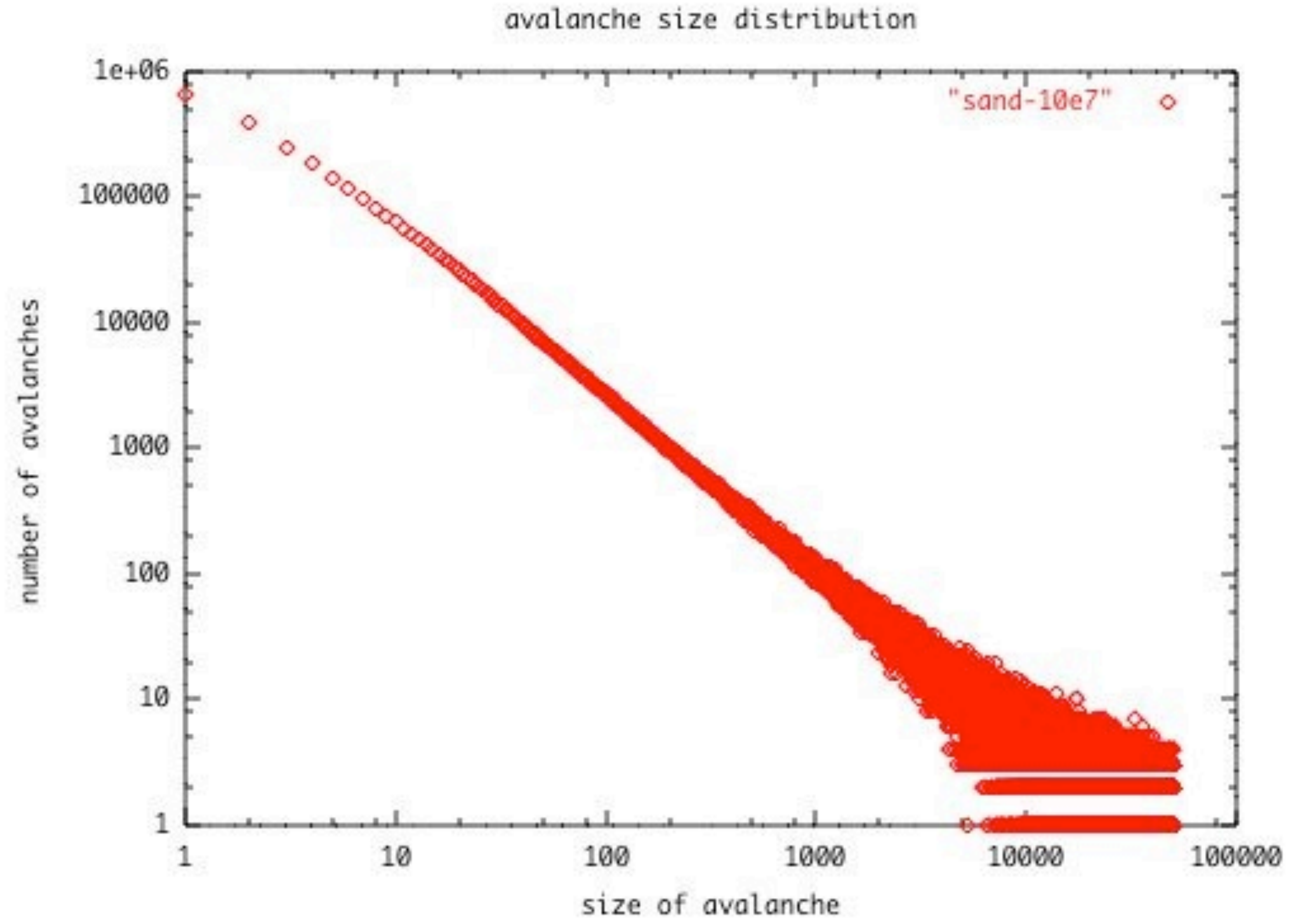
Sandpiles



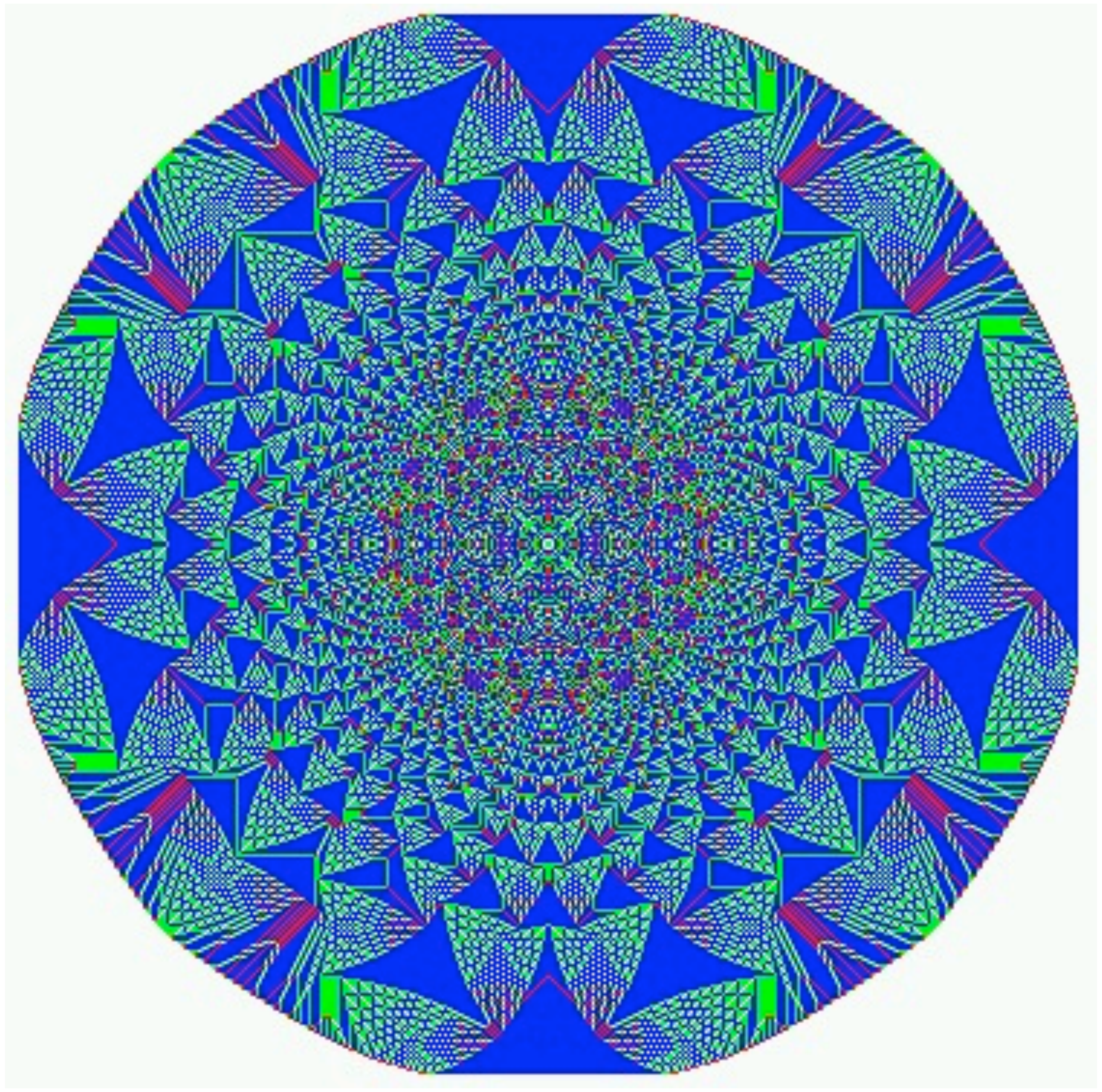
Sandpiles

3	3	2	0
1	2	1	2
3	1	1	0
3	2	1	2

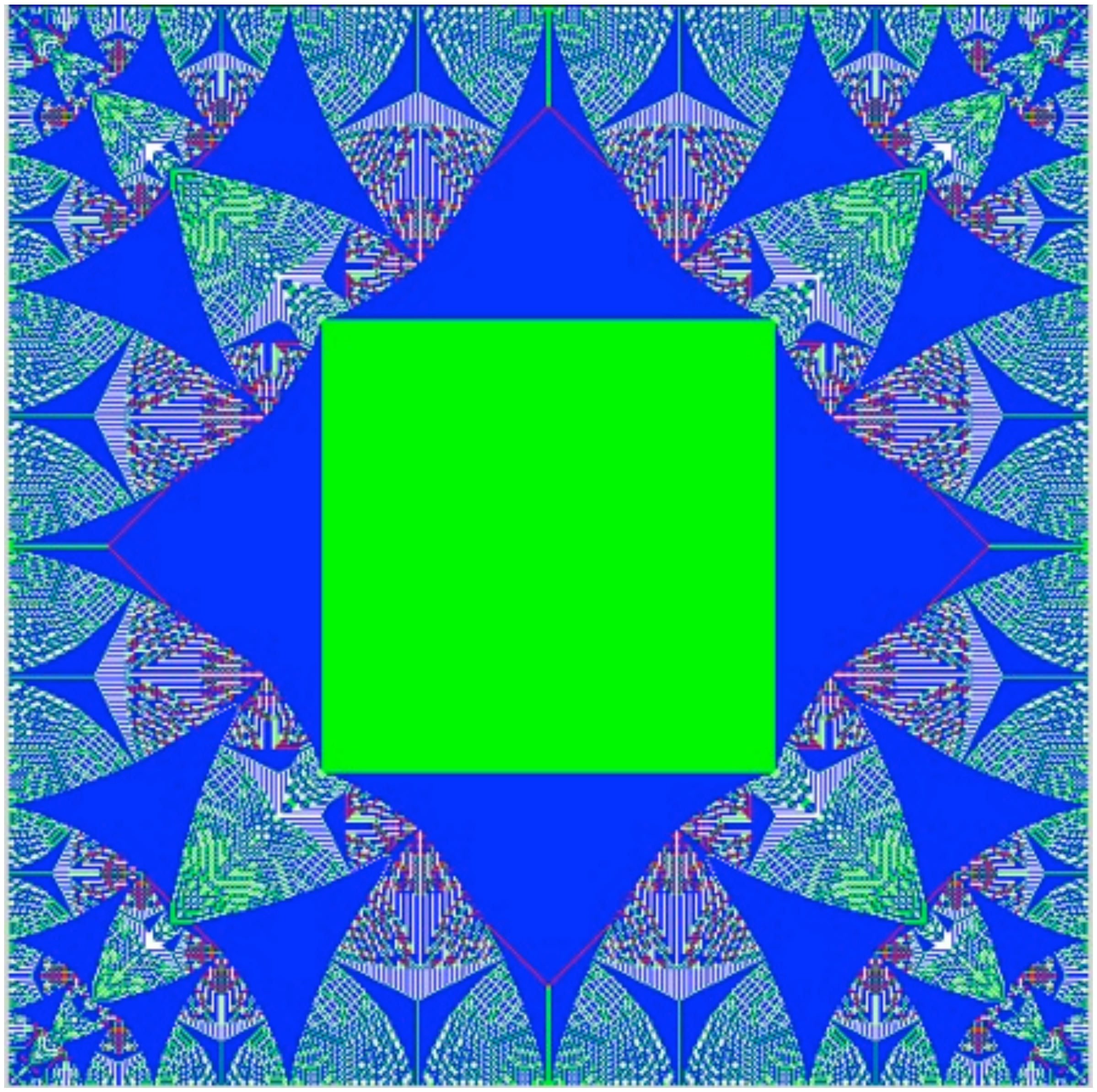
Avalanche size distribution



The mysterious mandala

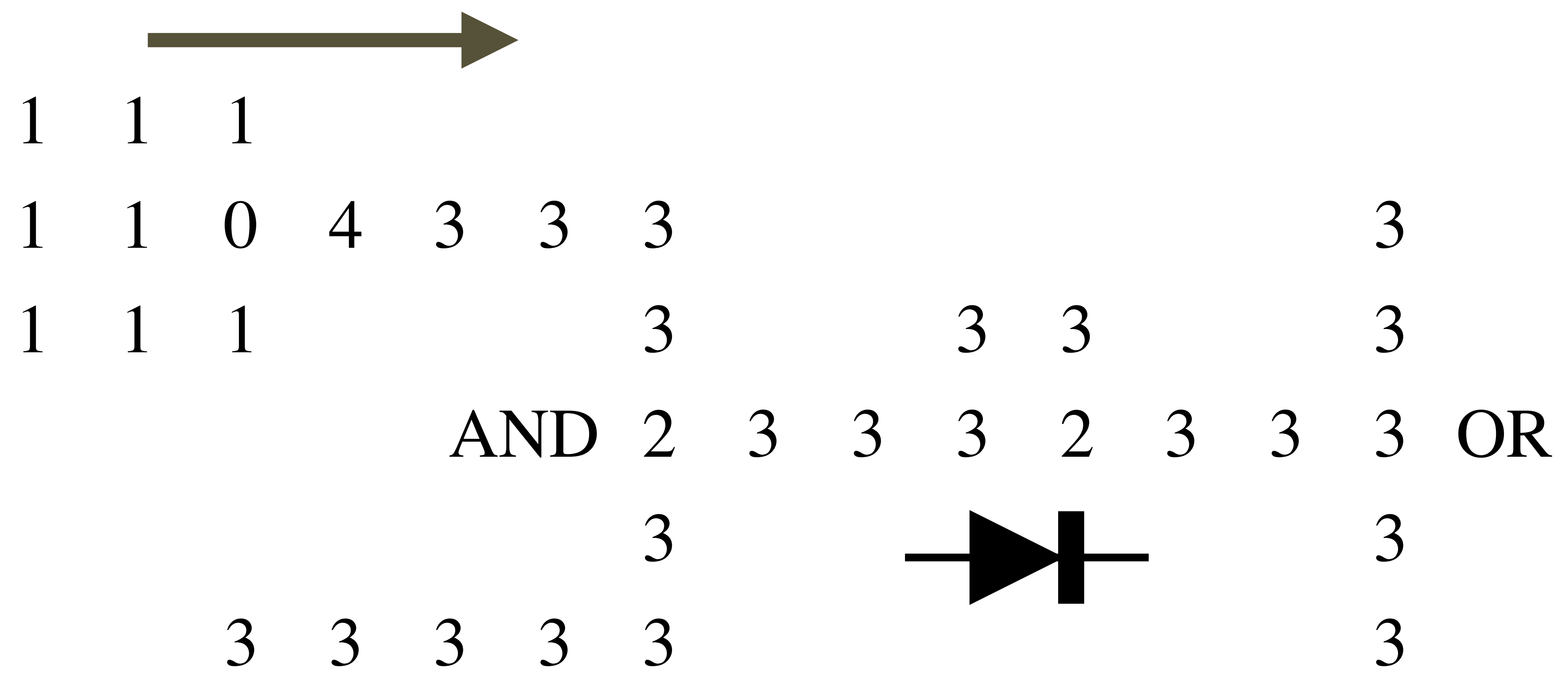


The amazing identity



Circuits made of sand

Wires (or “fuses”) connected by AND and OR junctions



[Moore & Nilsson]

The complexity of sandpiles

These circuits are monotone — AND and OR but no NOT

Evaluating monotone Boolean circuits is P-complete...

...but not in the planar case!

If wires intersect, then signals propagate messily

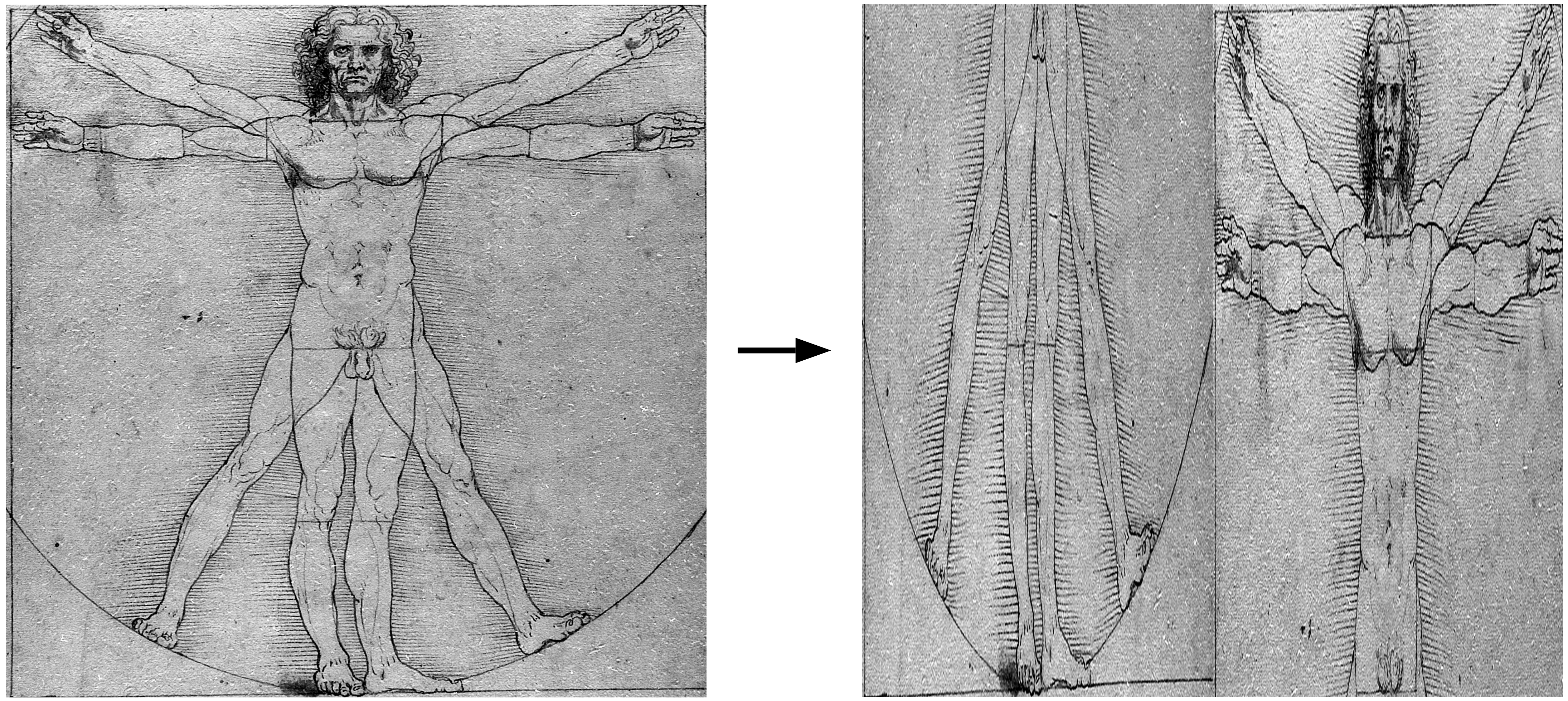
In d≥3, predicting sandpiles is P-complete

In d=1, it's in NC

In d=2, unknown; perhaps in the gap between NC and P-complete

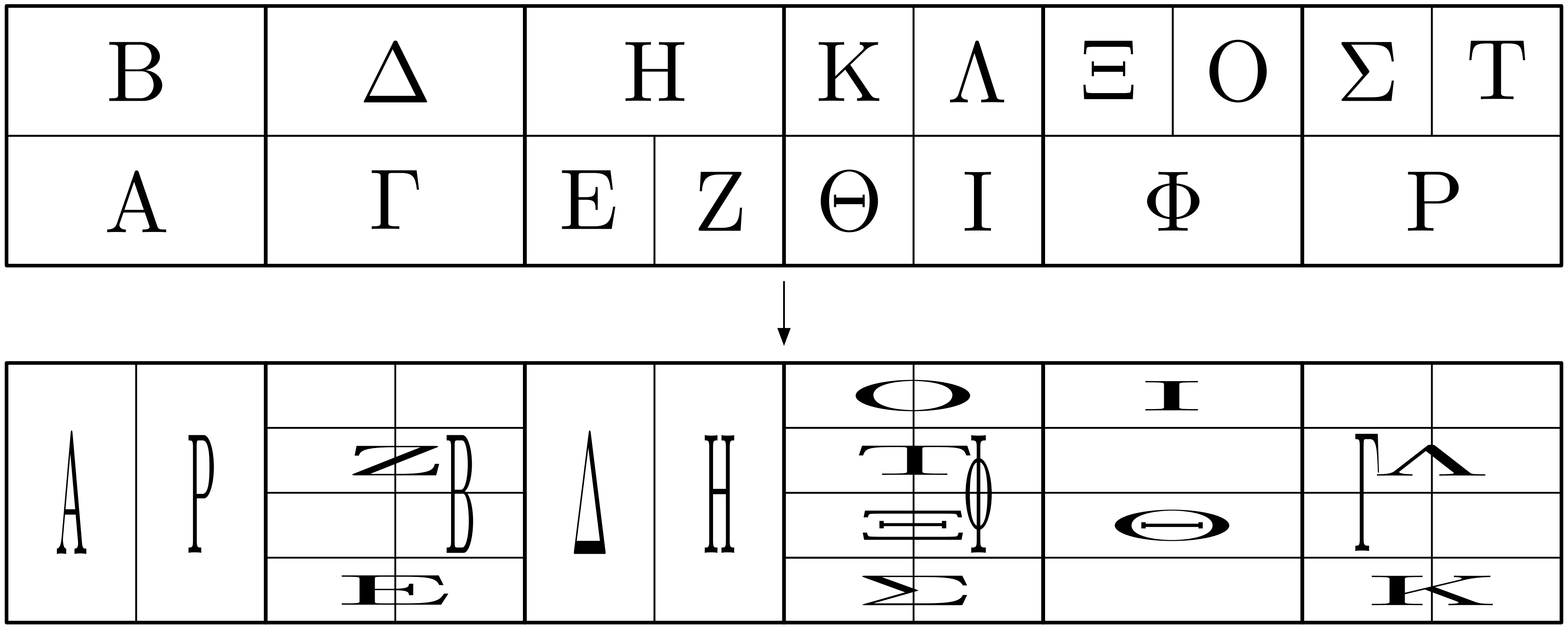
[Moore & Nilsson]

Building a computer out of low-dimensional dynamical systems



$$\ldots y_3 y_2 y_1 \,.\, x_1 x_2 x_3 \ldots$$
$$\Downarrow$$
$$\ldots y_3 y_2 \,.\, y_1 x_1 x_2 x_3 \ldots$$

Building a computer out of low-dimensional dynamical systems



F	s_1	s_2	s_3	s_4	s_5	s_6
0	$0, s_1, L$	$0, s_6, L$	$0, s_2, R$	$1, s_5, R$	$1, s_4, L$	$1, s_1, L$
1	$1, s_2, L$	$0, s_3, L$	$1, s_3, L$	$0, s_6, R$	$1, s_4, R$	$0, s_4, R$

Virtually any question about long-term dynamics is undecidable

[Moore, Siegelmann & Sontag, Reif...]

Fragile vs. robust analog computation

This construction packs an arbitrary number of bits into just two real numbers

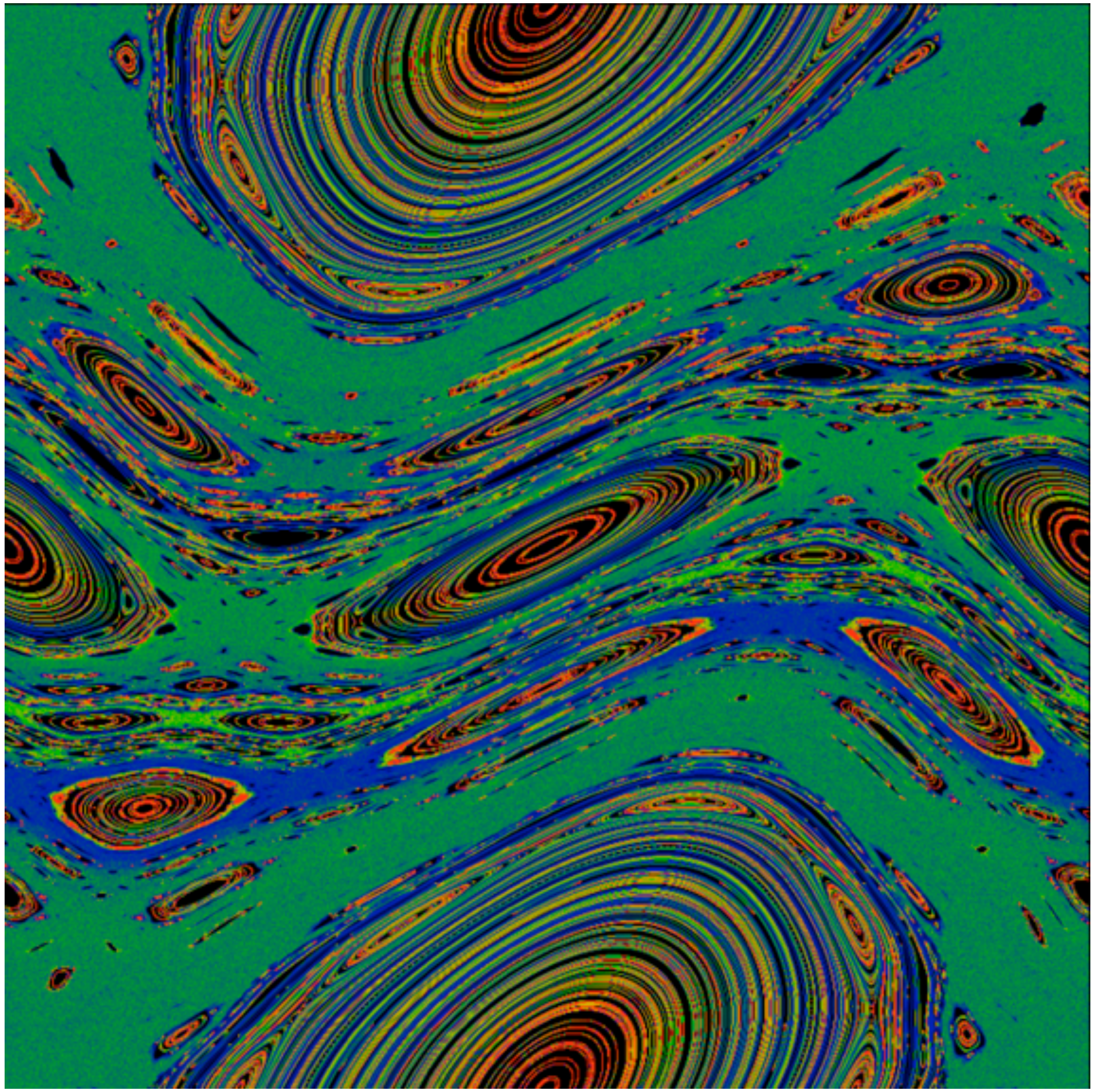
Small perturbations seem to destroy all but a few of these bits

Real analog computation encodes digital information in more robust ways: many particles, many voltages, many genes...

But the computational complexity of many low-dimensional dynamical systems remains open

Chaos vs. computation

$$p_{t+1} = p_t + K \sin \theta_t$$
$$\theta_{t+1} = \theta_t + p_{t+1}$$



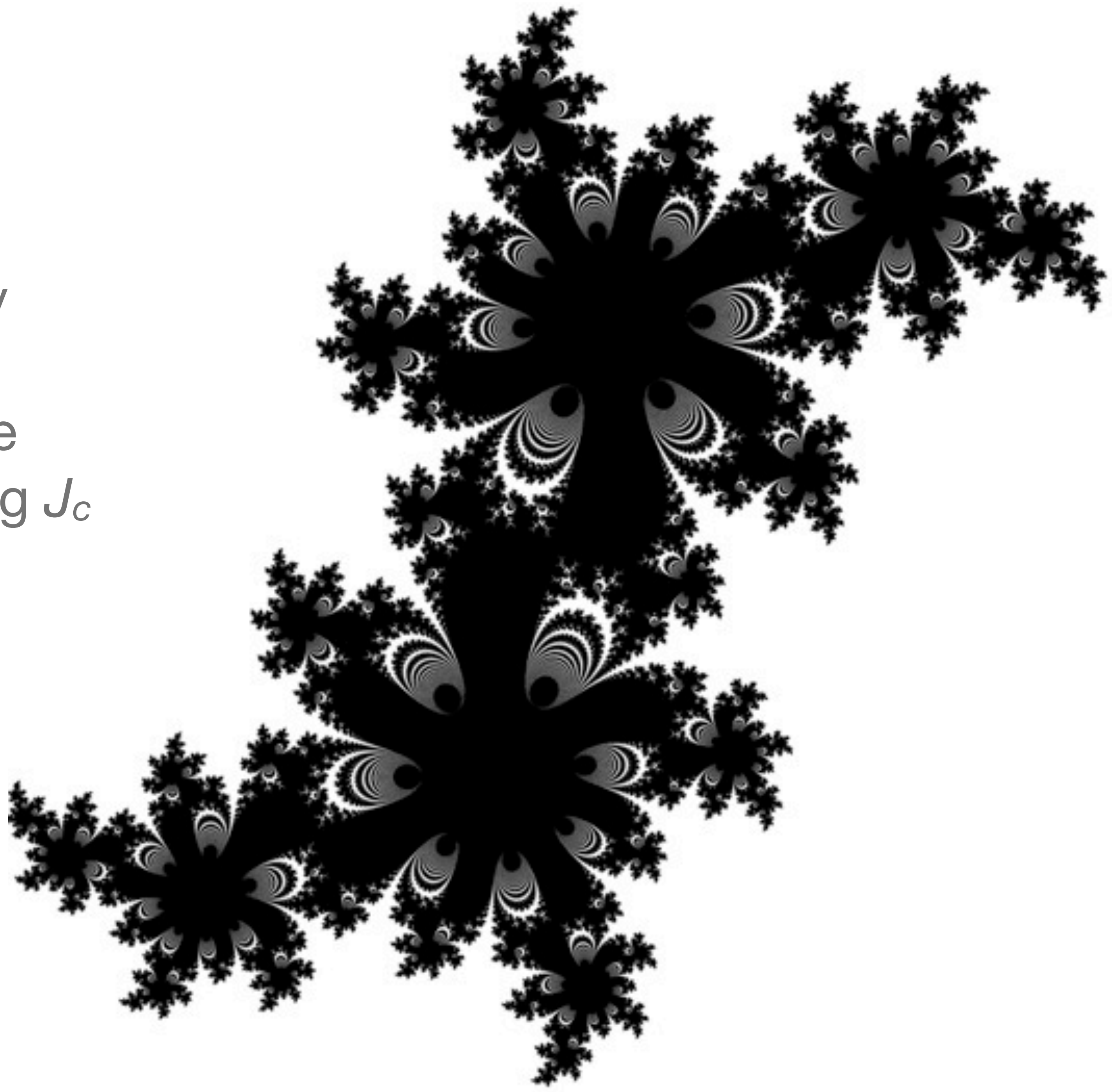
Julia sets

For a given *c*, J_c is the set of complex numbers *z* such that iterating

$$f(z) = z^2 + c$$

doesn't cause *z* to fly off to infinity

Braverman & Yampolsky: there are computable *c* such that computing J_c is as hard as the Halting Problem!



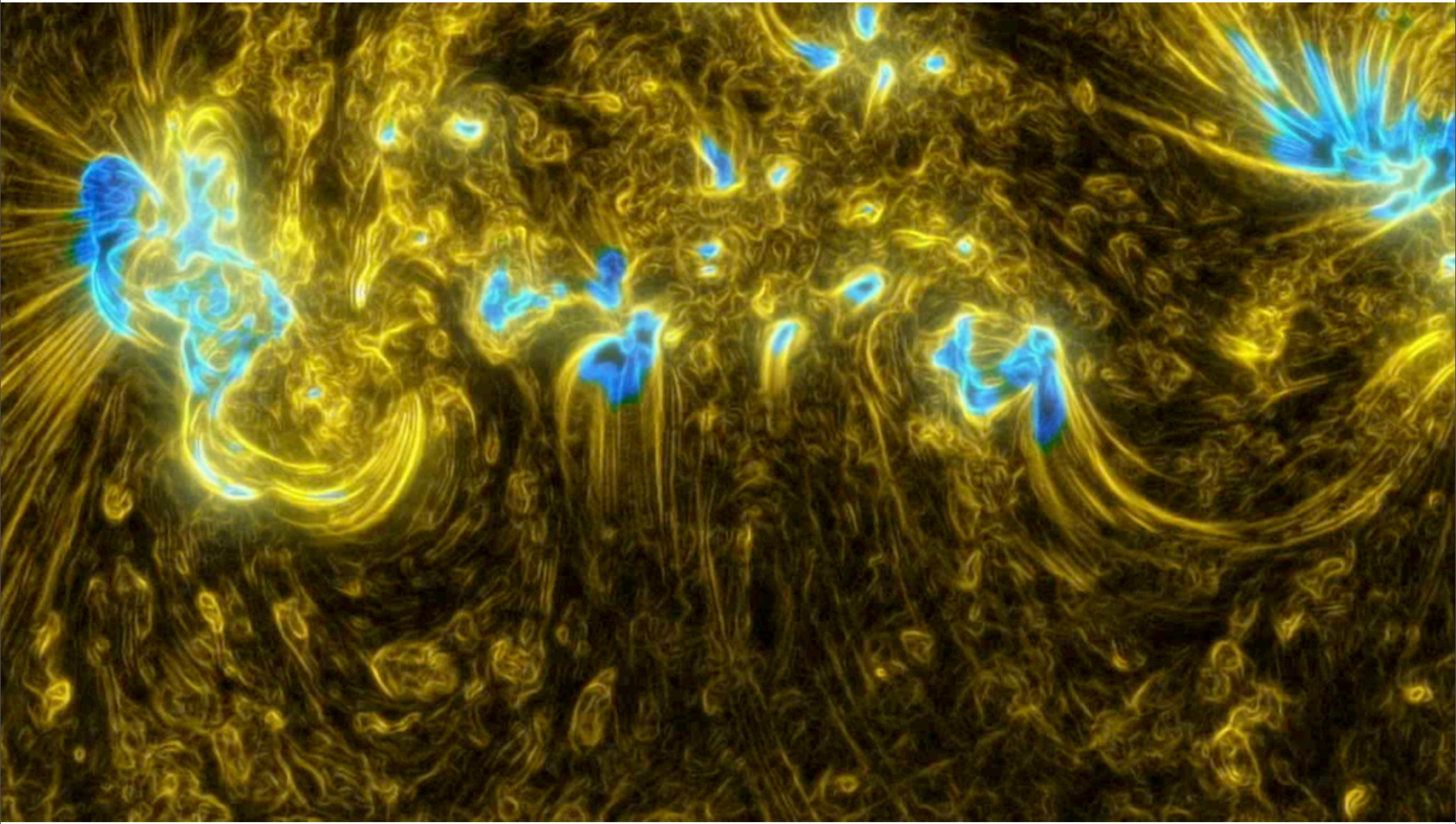
How generic is computation?

We can build computers out of gears and cams, or semiconductors, or DNA and enzymes, or pipes and water

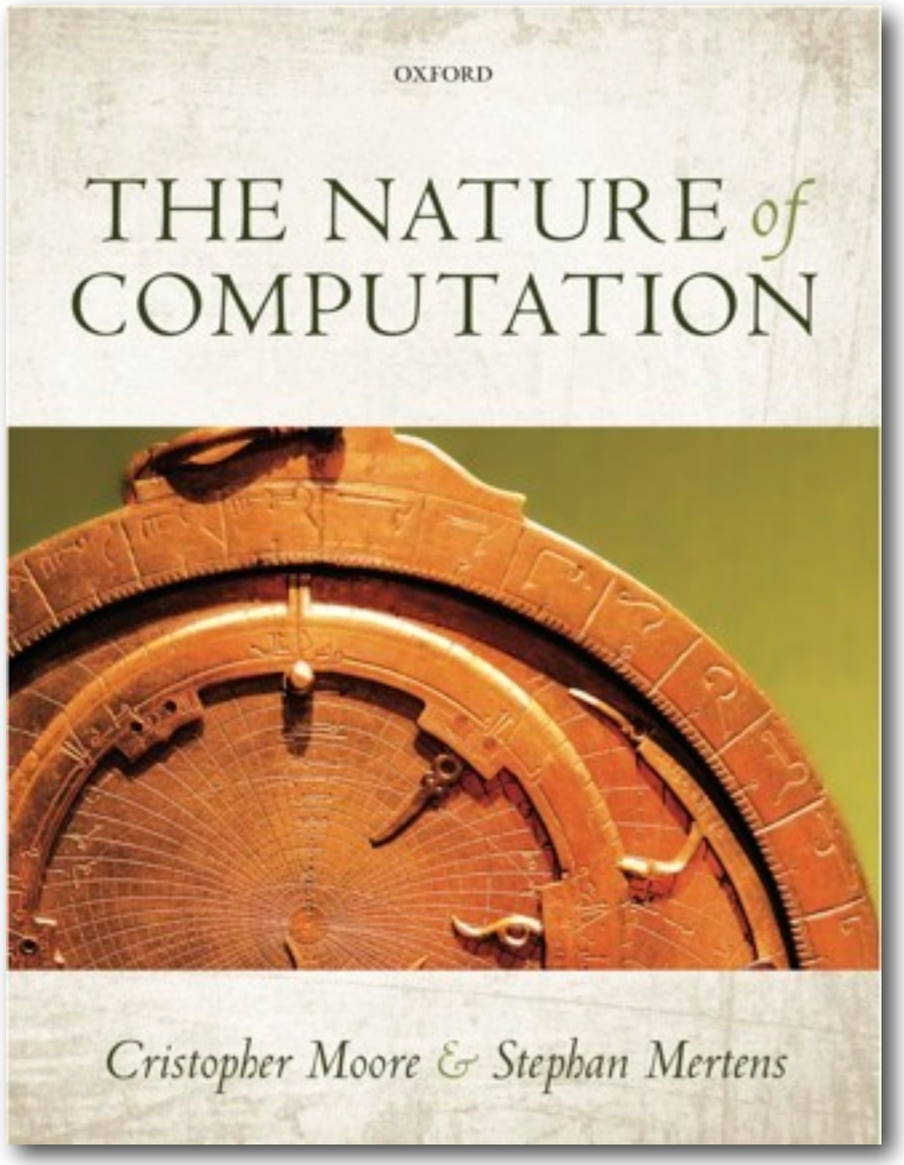
What about just water? Or just gravity? Or just plasma?

Can we prove that messy systems are hard, even if we can’t build computers out of them?

Wild problems



Shameless Plug



www.nature-of-computation.org

To put it bluntly: this book rocks! It somehow manages to combine the fun of a popular book with the intellectual heft of a textbook.

Scott Aaronson, MIT

A creative, insightful, and accessible introduction to the theory of computing, written with a keen eye toward the frontiers of the field and a vivid enthusiasm for the subject matter.

Jon Kleinberg, Cornell

A treasure trove of ideas, concepts and information on algorithms and complexity theory. Serious material presented in the most delightful manner!

Vijay Vazirani, Georgia Tech

A fantastic and unique book, a must-have guide to the theory of computation, for physicists and everyone else.

Riccardo Zecchina, Politecnico de Torino

This is the best-written book on the theory of computation I have ever read; and one of the best-written mathematical books I have ever read, period.

Cosma Shalizi, Carnegie Mellon

Acknowledgments

and NSF, DARPA/AFOSR, ARO, and NIST