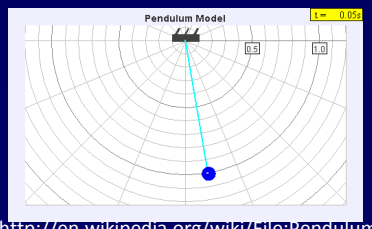
Computational universality, chaos, and computing with real numbers

Mark Braverman

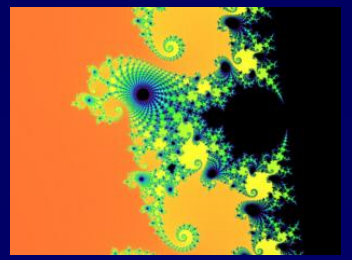
Princeton University

Part of the *Turing Centenary Research Project – Mind, Mechanism and Mathematics*, funded by the *John Templeton Foundation*.

June 23, 2012

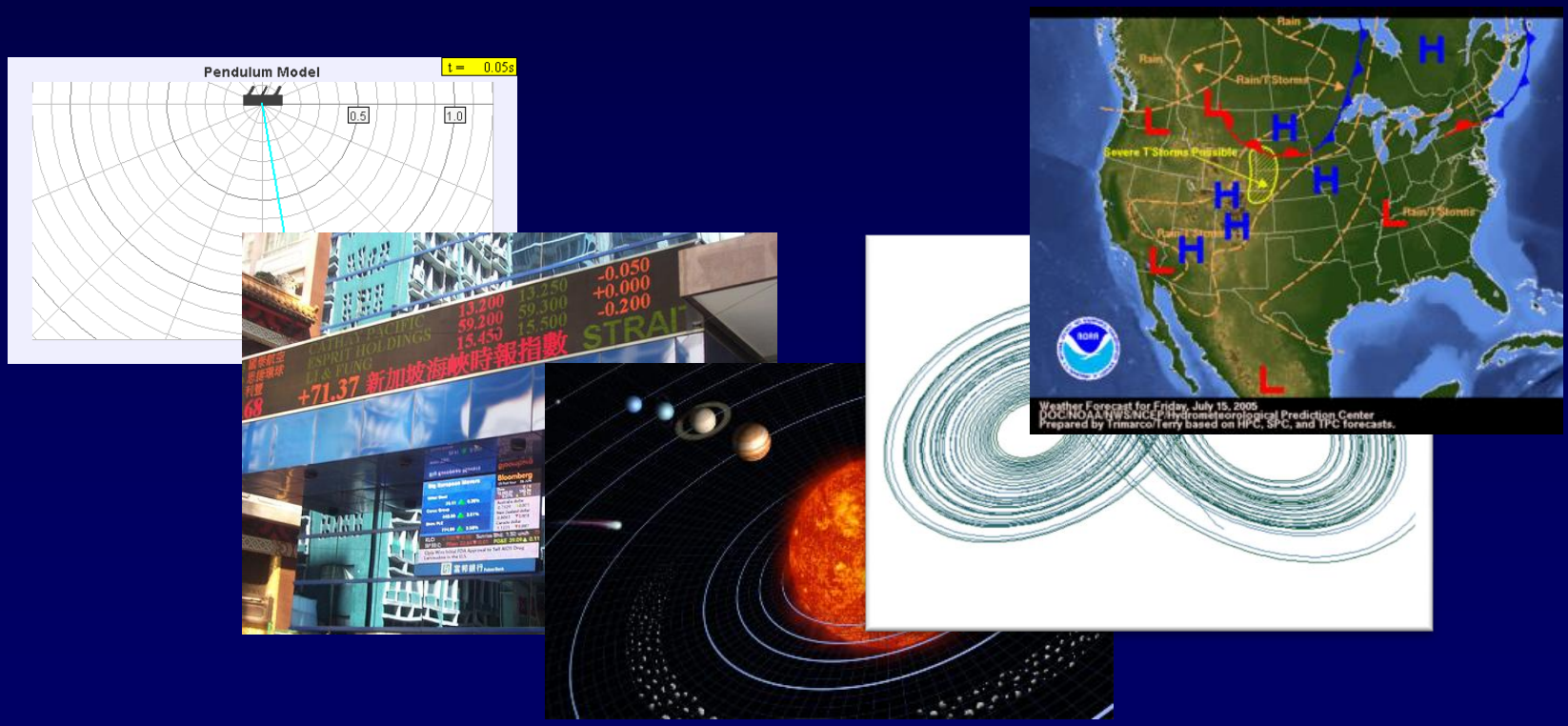


http://en.wikipedia.org/wiki/File:Pendulum2secondclock.gif



The big question

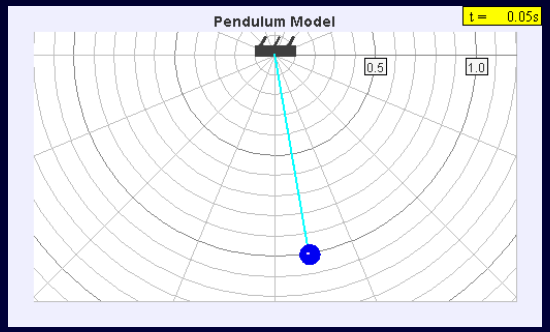
- Can all natural systems be predicted?



The mathematical abstraction of "everything": *dynamical systems*

- A *dynamical system* consists of a state space S and an evolution rule F.
- The initial state $X_0 \in S$ determines the behavior of the system $\{X_t\}_{t \in 0}$.
- Time may be discrete ($t=0,1,2,\ldots$) or continuous ($t \in [0,\infty)$).

Dynamical systems



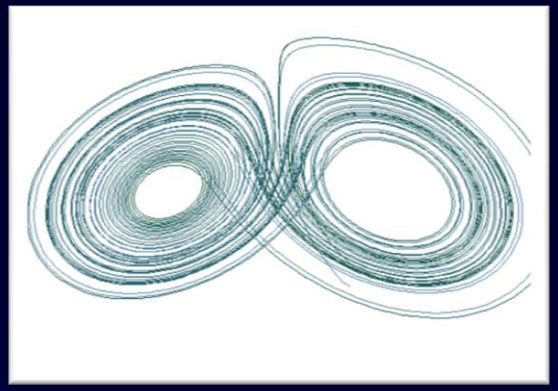
- Example: a simple oscillator.
- State space: $\mathfrak{S}=\mathbb{R}^2=\{(X,V)\}$

location *velocity*

- Evolution rule $\mathcal{F}$: $\begin{cases} X'(t) = V(t) \\ V'(t) = -\alpha \cdot X(t) \end{cases}$
- Solution: $\begin{cases} X(t) = \mathrm{A} \cdot \sin(\sqrt{\alpha}t + \omega) \\ V(t) = \mathrm{A}\sqrt{\alpha} \cdot \cos(\sqrt{\alpha}t + \omega) \end{cases}$

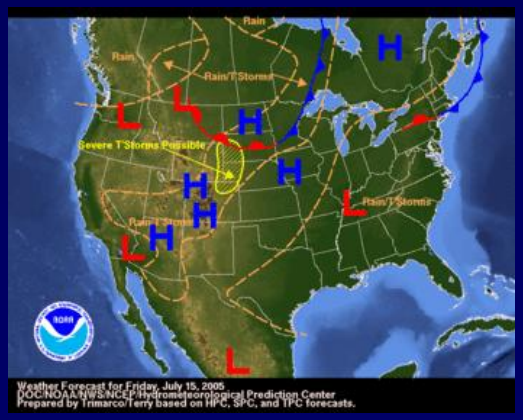
http://en.wikipedia.org/wiki/File:Pendulum2secondclock.gif

Dynamical systems

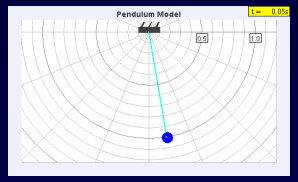


- Example: the Lorenz system.
- Used as a “toy” model for weather.
- State space $S=\mathbb{R}^3$.
- Evolution rule F:

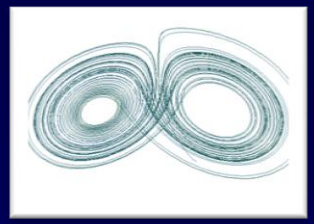
$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = x(\rho - z) - y$$
$$\frac{dz}{dt} = xy - \beta z$$



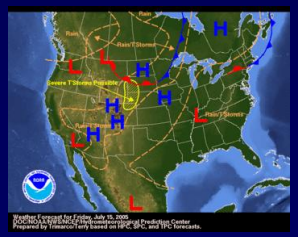
Examples of prediction questions



What will be the state of the pendulum at *t=20*?



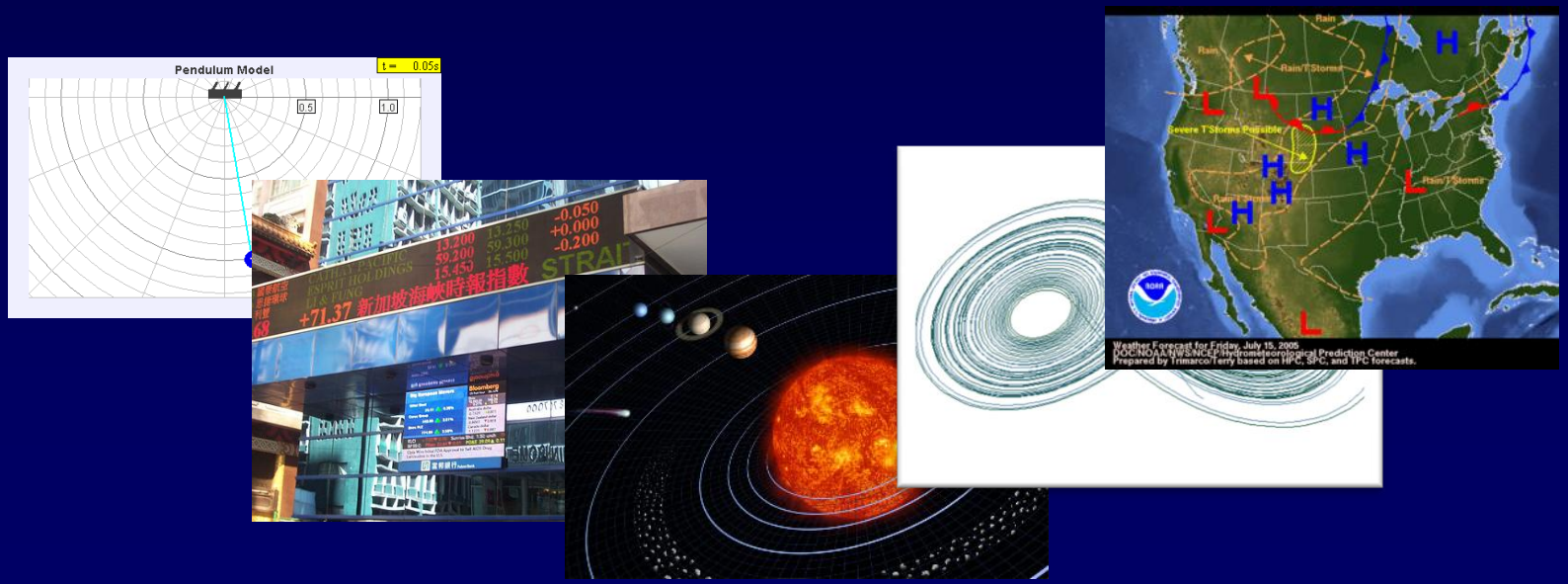
Describe a set of states to which the system will eventually converge.



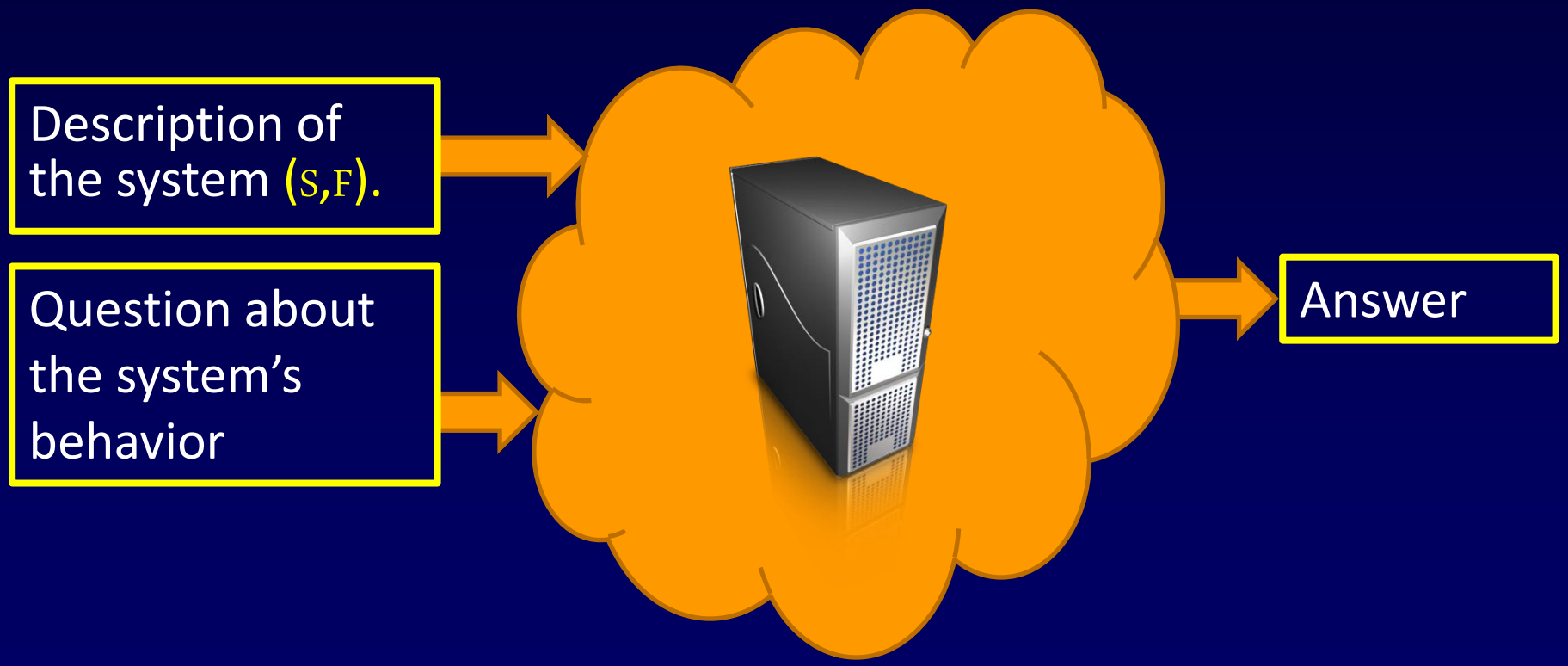
What initial conditions lead to tornadoes?

The big question – refined

- Can *all* questions about dynamical systems be answered *algorithmically*?



A “dream box” of Applied Mathematics



Obstacles to “predicting everything”

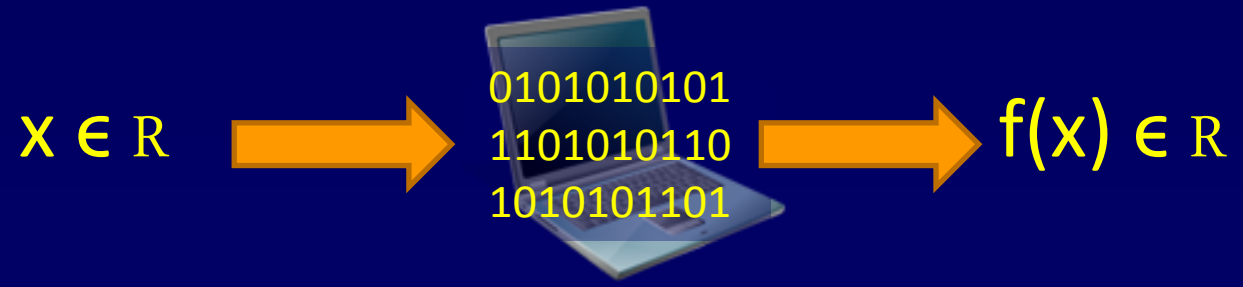
Several problems with the “dream box” picture:

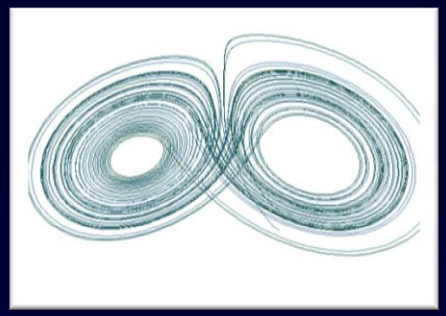
1. The systems are *continuous*; what do “input” and “output” mean?
2. *Chaotic behavior* of the system.
3. *Computational universality* of the system.

Obstacle #1: Computation and continuous systems

- Formalizing and reasoning about computation with continuous objects such as real numbers and functions over $\mathbb{R}$

  requires quite a bit of new theory.
- Dealt with in the fields of *Computable Analysis* and *Real Computation*.



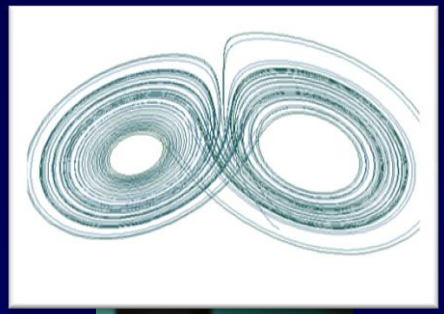


Obstacle #2: Chaotic behavior

- Microscopic perturbations to initial conditions lead to macroscopic effects down the road.
- Even a small amount of noise – which cannot be accounted for – significanlty affects the system in the long run.

Chaotic ≠ hard!

- Chaos only implies that predicting *individual trajectories* with confidence over time is hard.
- *General* and *statistical* properties of the system might be easy!
- Examples:
  - Average weather patterns.
  - The Lorenz attractor is computable.
  - A simple coin toss is easy to predict statistically.



Obstacle #3: Computational universality and undecidability

- The gravest of the concerns.
- The system itself might be too *computationally complex* to reason about!
- May apply *both* to individual trajectories and to global properties.
- Is a major obstacle in other settings.

Other shattered dreams

David Hilbert
(1862-1943)

Kurt Gödel
(1906-1978)

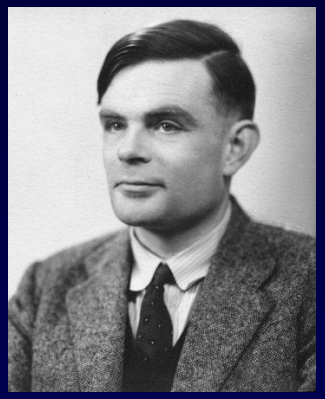
True statement in arithmetic

a set of arithmetic axioms

Proof

The Halting Problem

David Hilbert
(1862-1943)



Alan Turing
(1912-1954)

A computer program P()

math "solver"

Does P terminate?

Other examples

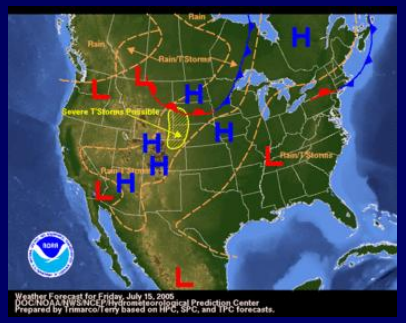
- Determining whether a Diophantine equation has a solution – Hilbert's 10th Problem [Davis, Matiyasevich, Putnam, Robinson].
- Generation problem in groups – a.k.a. the "the word problem for groups" [Novikov, Boone].

The computer as a dynamical system

- If we view the computer as a dynamical system C, with S = the possible memory configuration and F = the computation execution rule, then most long-term properties of $C=(S,F)$ are undecidable.
- For example, "will the system reach state *s*?" is equivalent to the *Halting Problem.*

Computational universality

- A dynamical system $T=(S,F)$ is *computationally universal* if it is capable of simulating a general computation.
- In other words, any computation can be "compiled" to "run" on $T=(S,F)$.



$\approx^{?}$

Computational Universality

A dynamical system $T=(S,F)$

A computer program P()

(effective) compiler

$s_0(P) \in S$

dynamics

$s_T \in S$

(effective) decoder

Output of P()

Computational undecidability vs. computational universality

"Universality => undecidability":

- If a dynamical system is *computationally universal*, i.e. powerful enough to simulate a computer...

  … then it is impossible to algorithmically compute most of its properties.

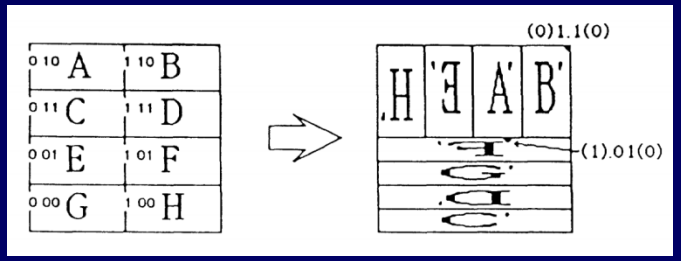
- Undecidability may occur even without universality.

Undecidable = hard!

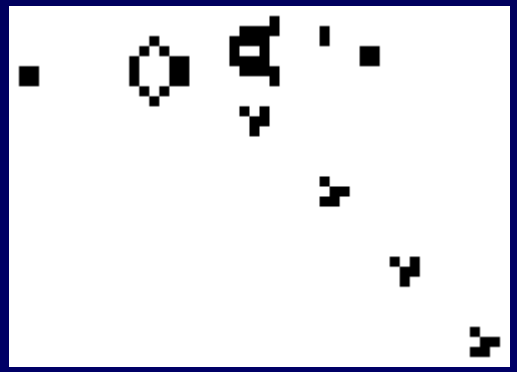
- Unlike difficulties due to chaotic behavior, undecidable results stick!
- If a problem is undecidable, it is actually computationally intractable.

Universality is there…

- Even very simple systems (S,F) are known to have universal behavior.
- Prominent examples:
  - Piecewise linear maps on [0,1] x [0,1];
  - Cellular automata – “the game of life”.



C. Moore. “Unpredictability and undecidability in dynamical systems”, PRL **64**(20), 1990.



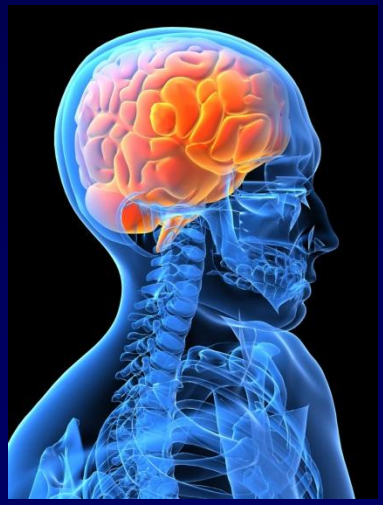
Universality is there… but is it relevant in practice?

- To be practically relevant, it is not enough for the system to be universal.
- Universality has to be robust.
- Robustness to:
  - Definitions;
  - Noise – both in initial conditions and evolution rule.

Robust universal systems do exist

- Implementations of the Turing Machine, e.g. this laptop.
- The human brain.

- But these systems are very complex.



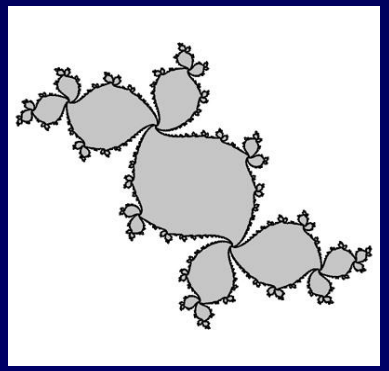
- What about simpler systems?

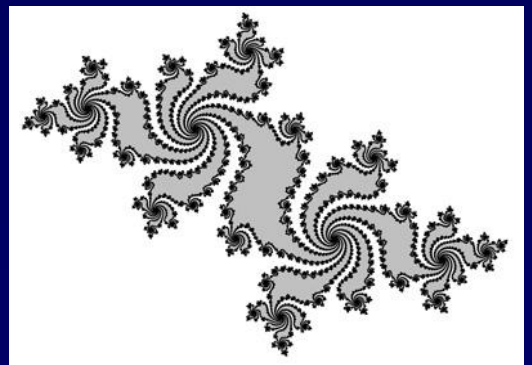
Universality in “simple” systems

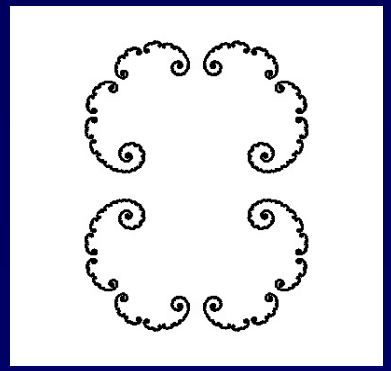
- There is evidence that in some “simple” systems universality is not robust.
- The system may be rich enough to support undecidability, but cannot do so if noise or perturbations are introduced into the system.

An example of a “simple” system

- (S,F) is the polynomial dynamics on the complex plane: $S = C$, and F is a polynomial, e.g. $F(z)=z^2+c$ for some parameter c in C.
- After 80+ years of study, a reasonably well understood system.

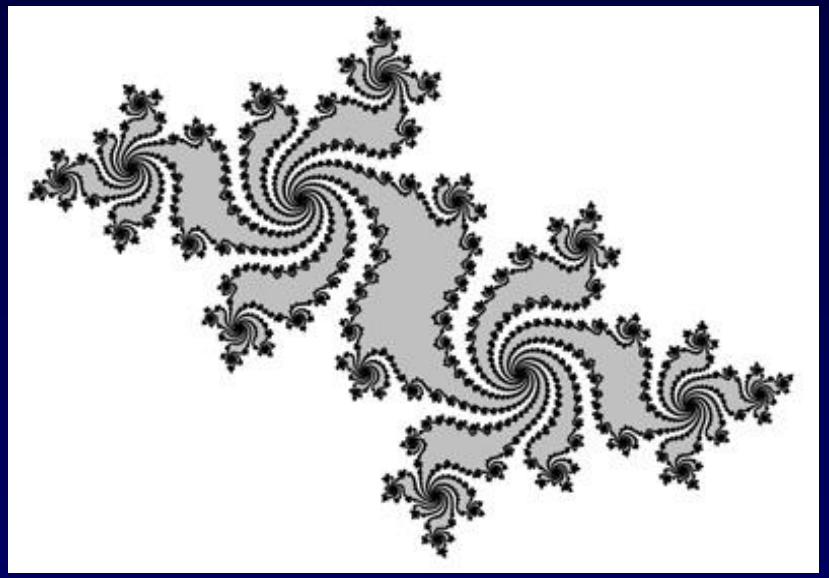






Julia sets

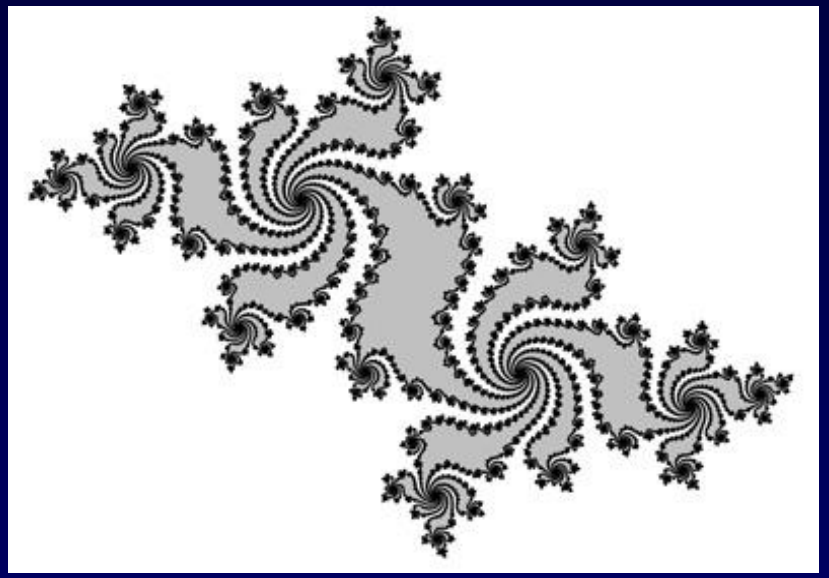
Gaston Julia
(1893-1978)

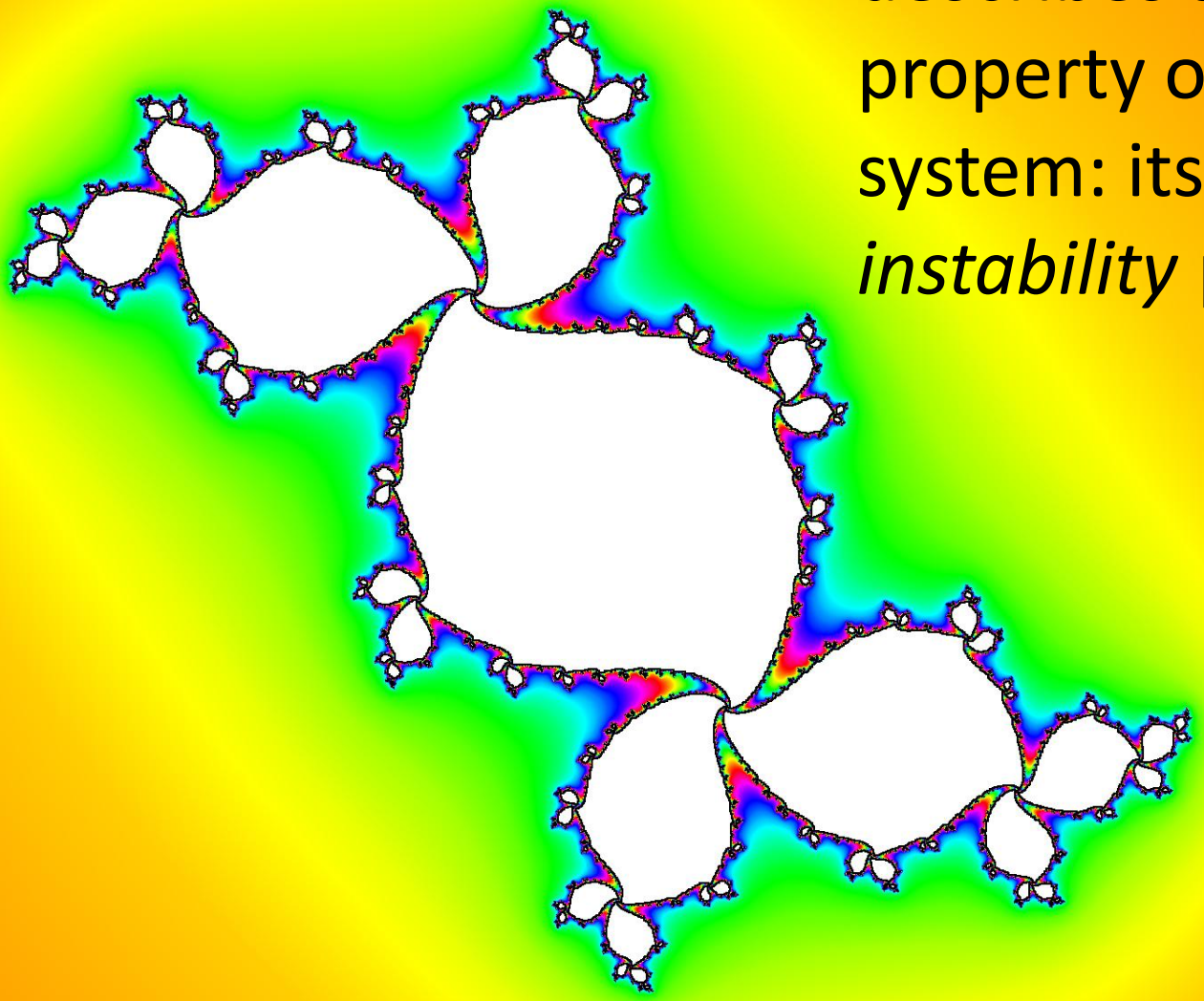


Julia sets

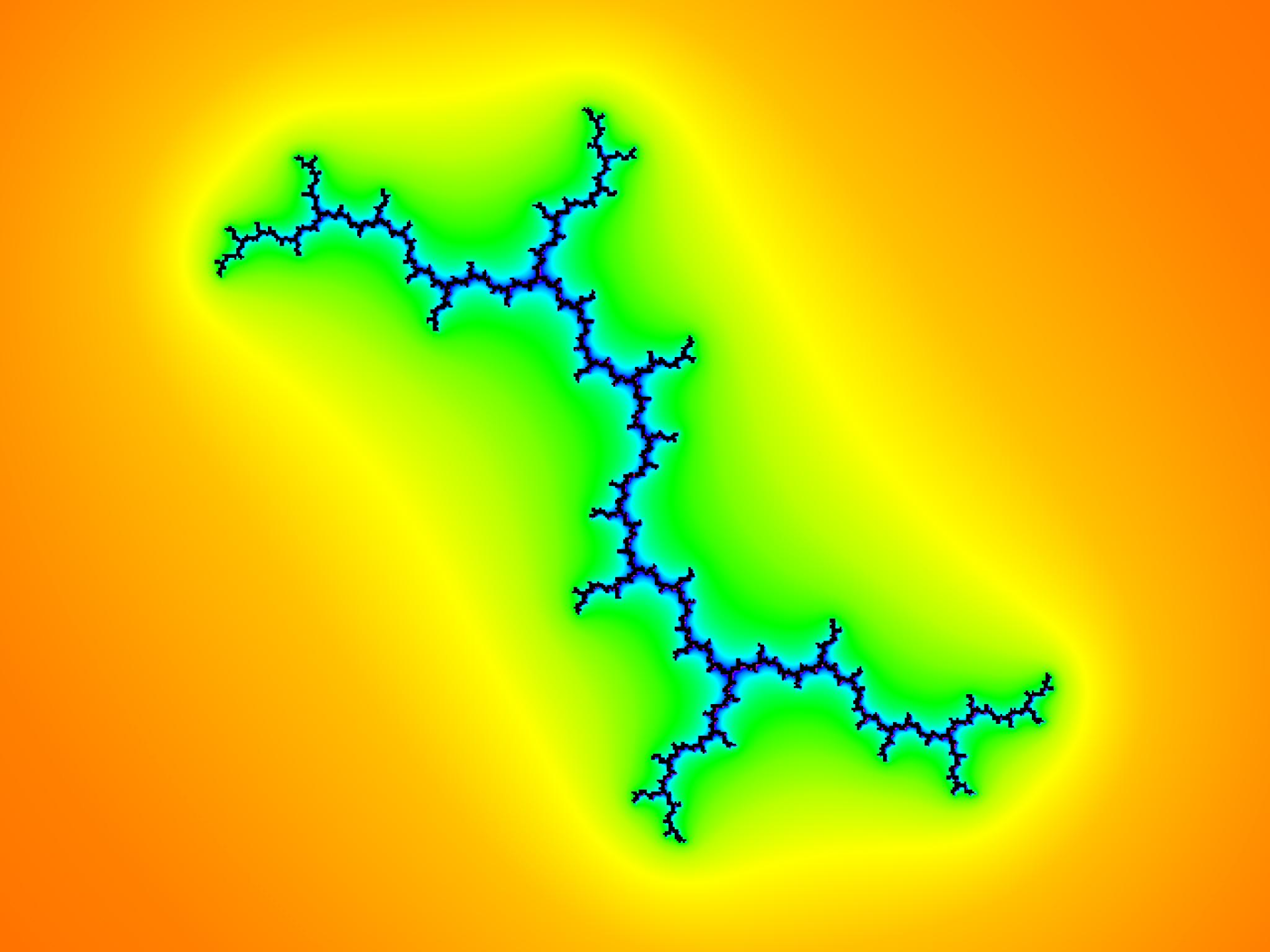
- Some orbits escape to ∞ and some do not.
- The filled Julia set K_c is the set of initial z's for which the orbit does not escape to ∞.
- The Julia set J_c is the boundary of K_c:

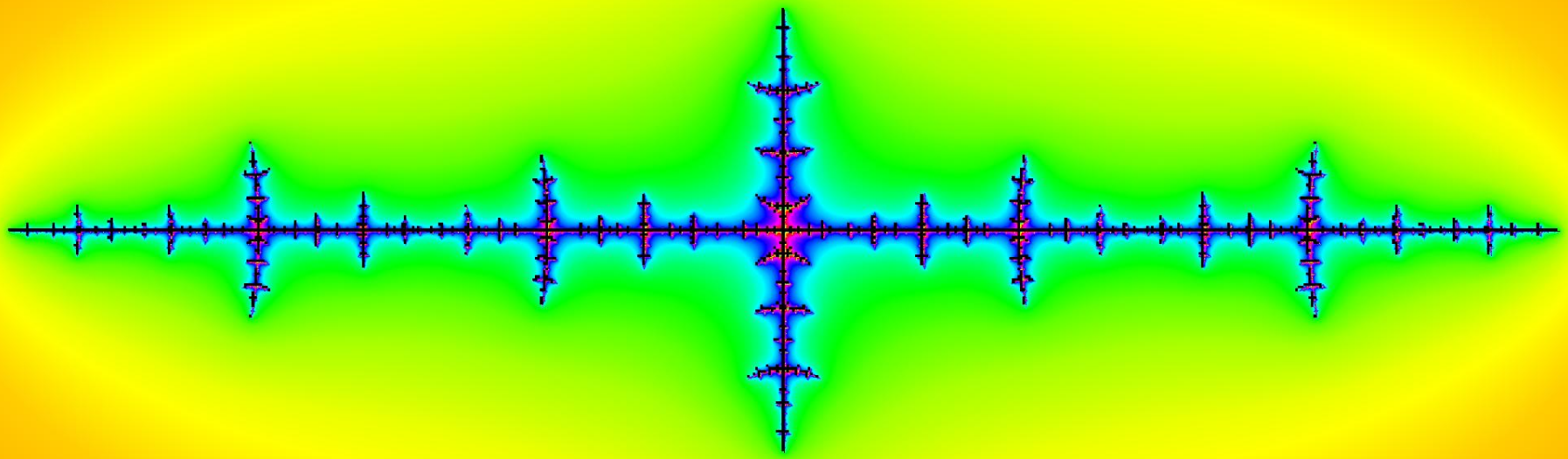
$$J_c = \partial K_c.$$





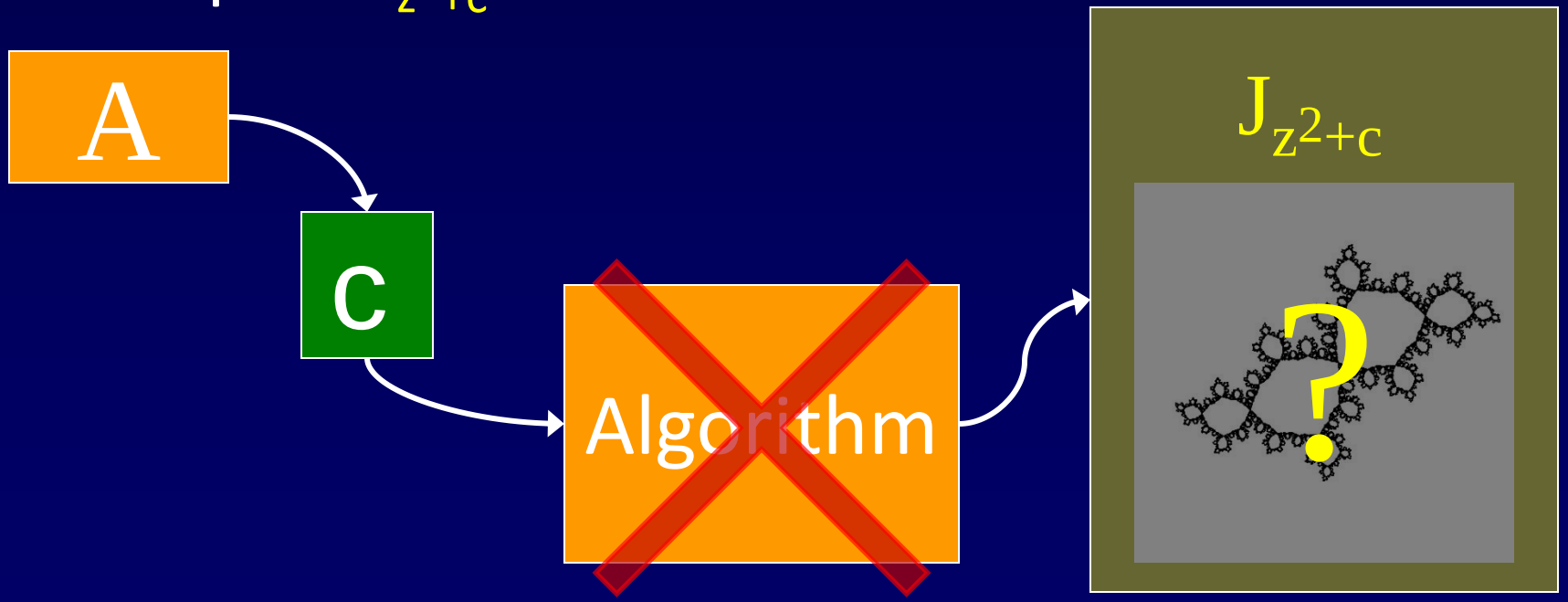
The Julia set J_{z^2+c} describes a global property of the system: its *instability* points.





There are non-computable Julia sets

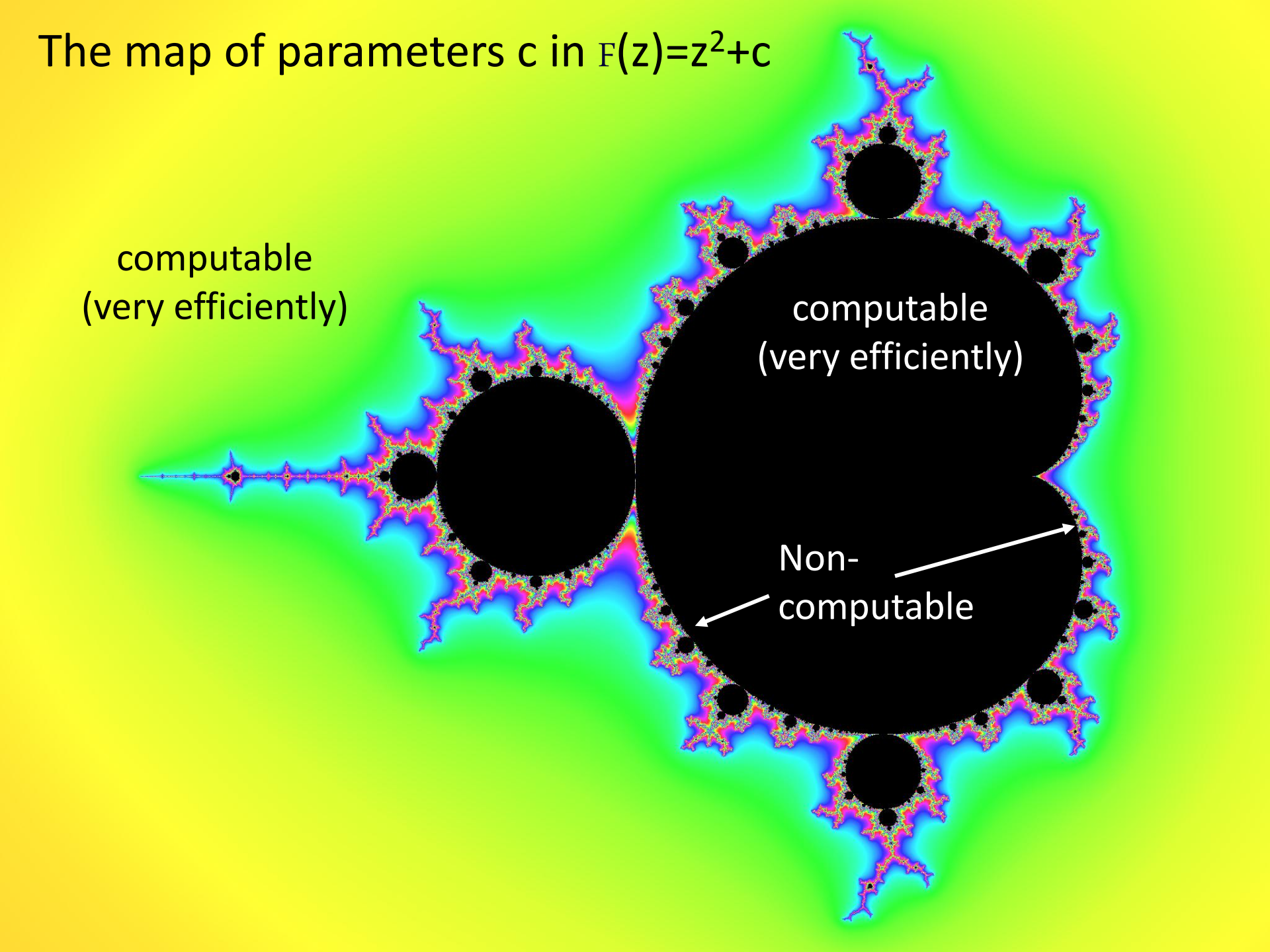
Theorem [B.-Yampolsky'07]: There is an algorithm A that computes a number c such that no machine with access to c can compute J_{z^2+c}.



But…

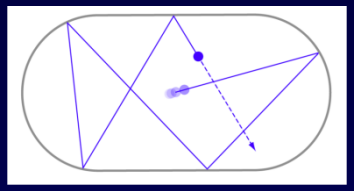
- Non-computability is rare.
- For almost all parameters c, the Julia set J_{z^2+c} is computable.
- Moreover, it is efficiently computable – in polynomial time!

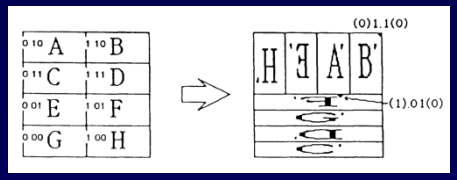
The map of parameters c in $F(z)=z^2+c$

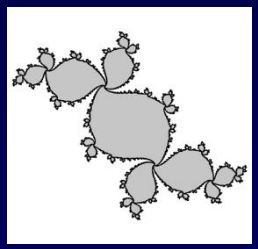


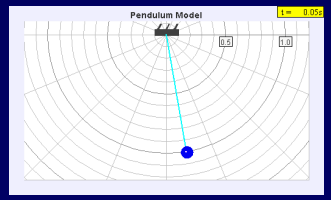
(Non)-Computability in dynamics

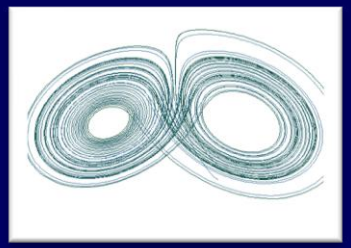
Non-computable









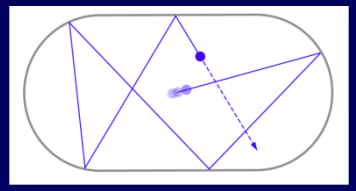


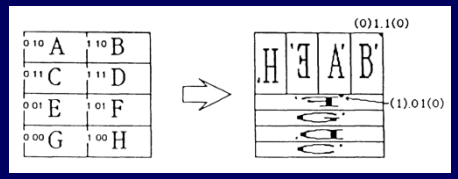
Computable

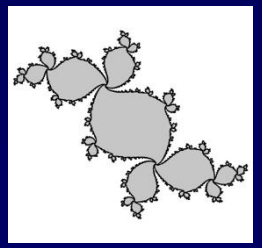
http://en.wikipedia.org/wiki/File:BunimovichStadium.png

(Non)-Computability in dynamics

Robustly Non-computable



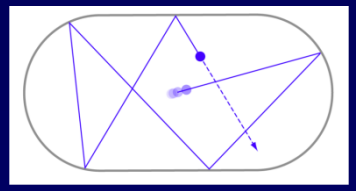


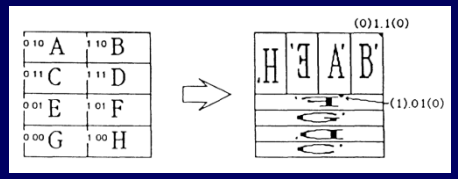


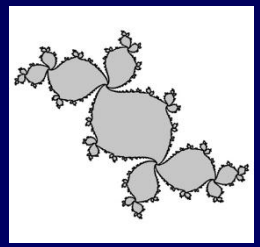
Become computable under noise

(Non)-Computability in dynamics

Robustly Non-computable



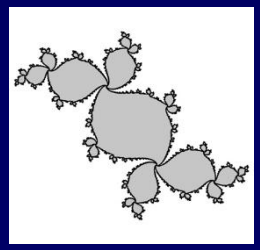


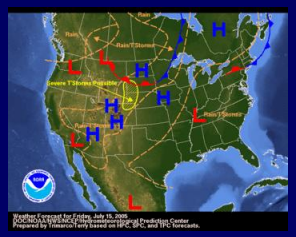


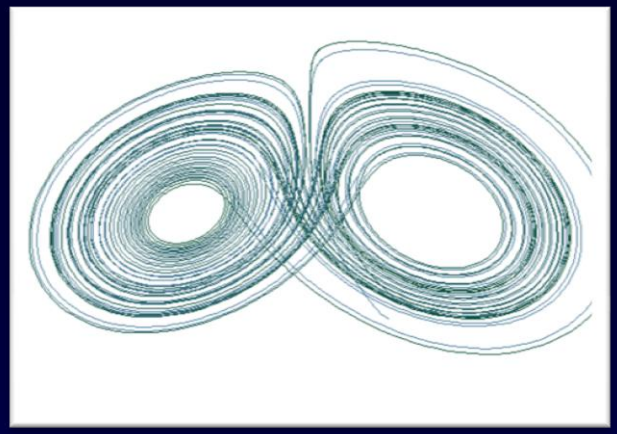
Become computable under noise

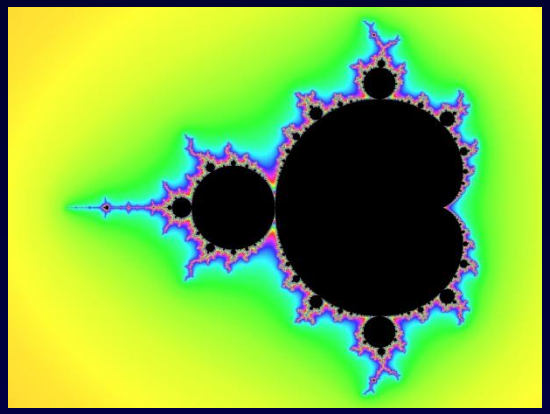
The road ahead

- When are questions about dynamical systems robustly undecidable?
- Can noise actually “destroy” non-computability?
- Is computational undecidability a “miracle” or “the rule”?









Thank You!