Complex Systems

from a Computer Science point of view

Nicolas SCHABANEL
CNRS - Université Paris Diderot - IXXI

Complex systems & Computer science

« Computer science is no more about computers than astronomy is about telescopes »

Edsger W. Dijkstra (1930-2002)

- Computer science is the *science of computation* not computers.

- **Birth date of C.S.: 1930, when one was able to express the time needed to solve a problem in term of the size of the problem**, *something that no other field was able to do before.*

A brief history:
Facing complexity

Before complexity emerged...

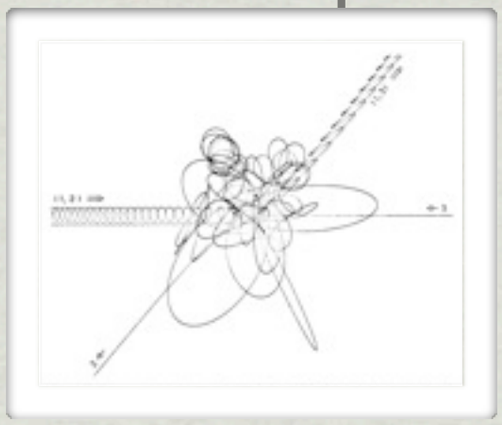
- ... there was an hope that:

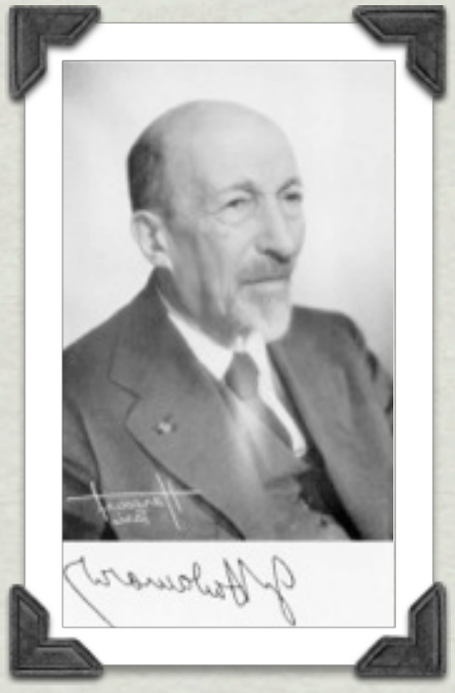
  Once the (necessarily simple) rules governing nature will be found, then we will be able to predict and control everything

- A quite persistent myth: *e.g., the hope initially aroused by DNA decoding in the 1990s*

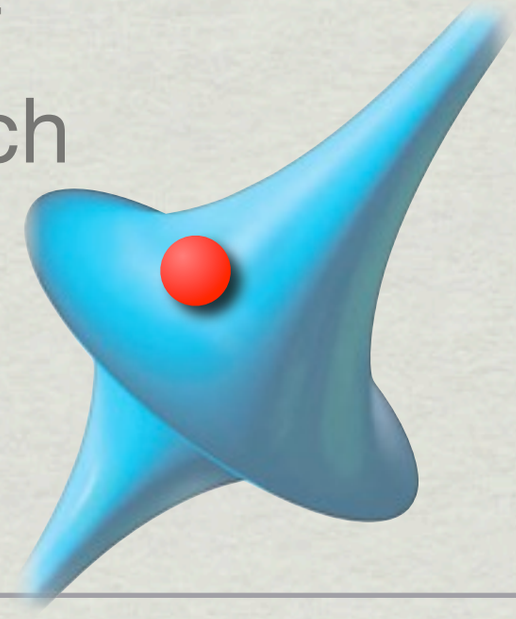
Strike 1: Birth of chaos

- **Poincaré (1890) 3-body problem:** infinite precision is required for some initial conditions



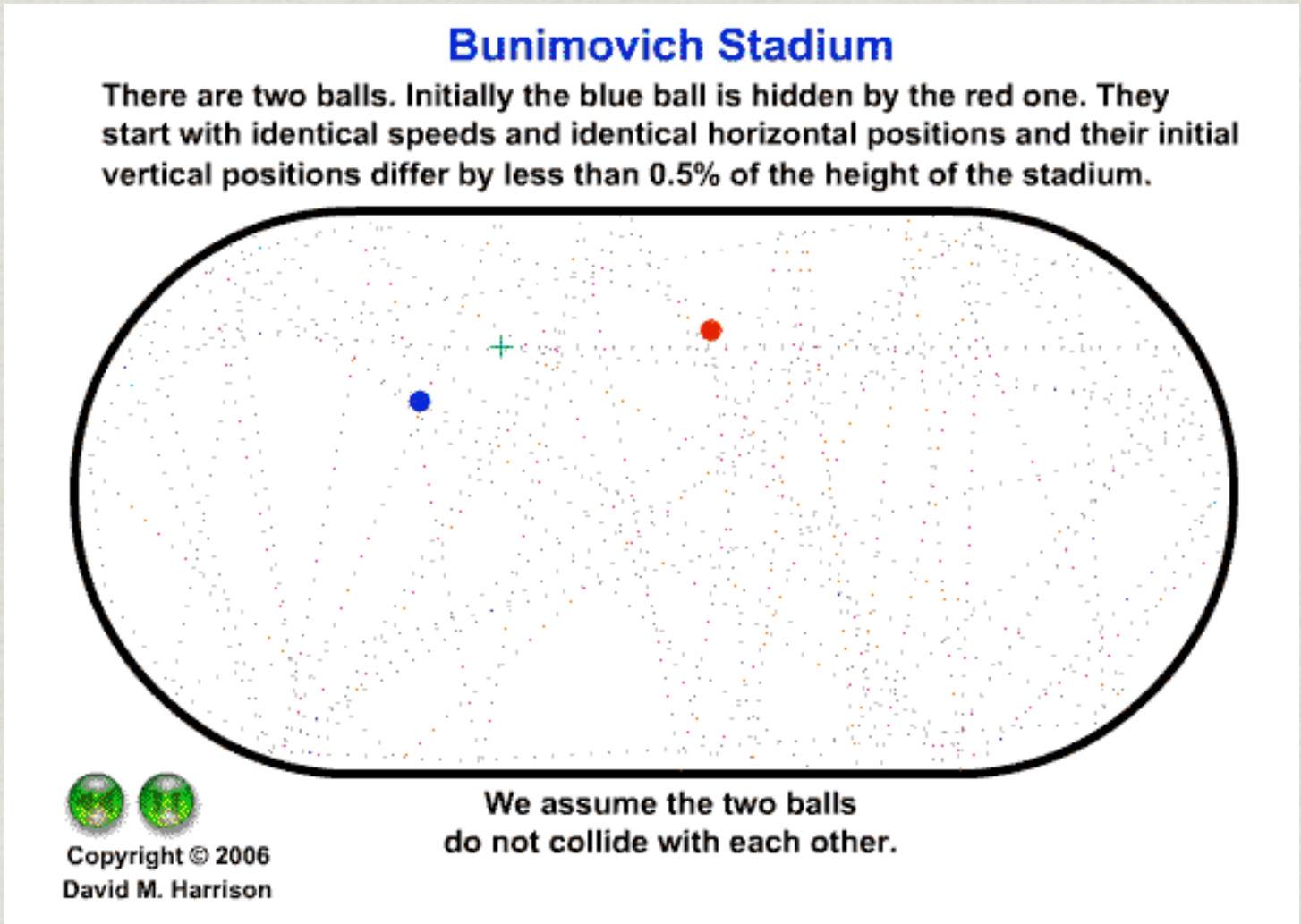


- **Hadamard (1898):** *any* pair of trajectories get away from each other exponentially for some negative curvature surfaces



Other example of chaotic behavior: billiards

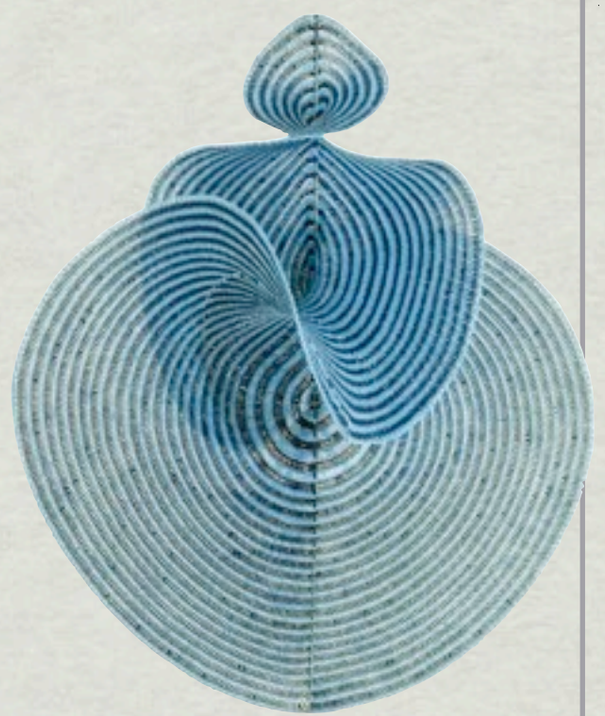
Other example of chaotic behavior: billiards



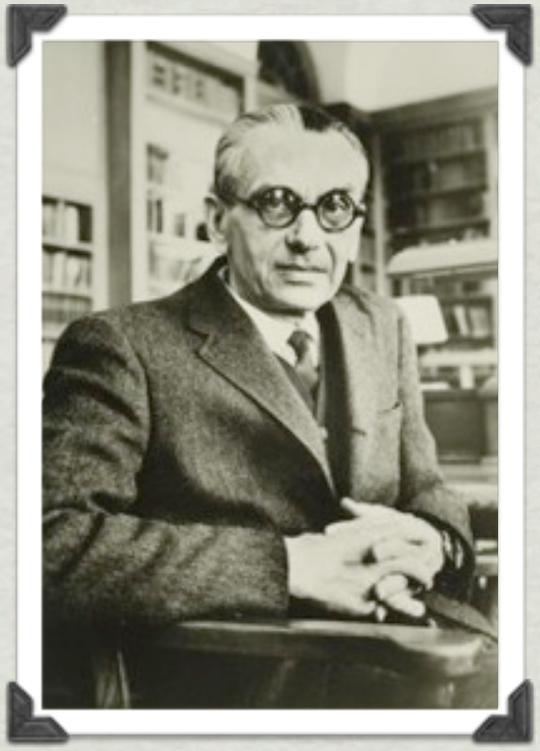
Strike 1bis: Chaos is everywhere

- **Lorenz (1960-):** Larger-scale computer simulations reveal that chaos is in a lot of simple-looking systems

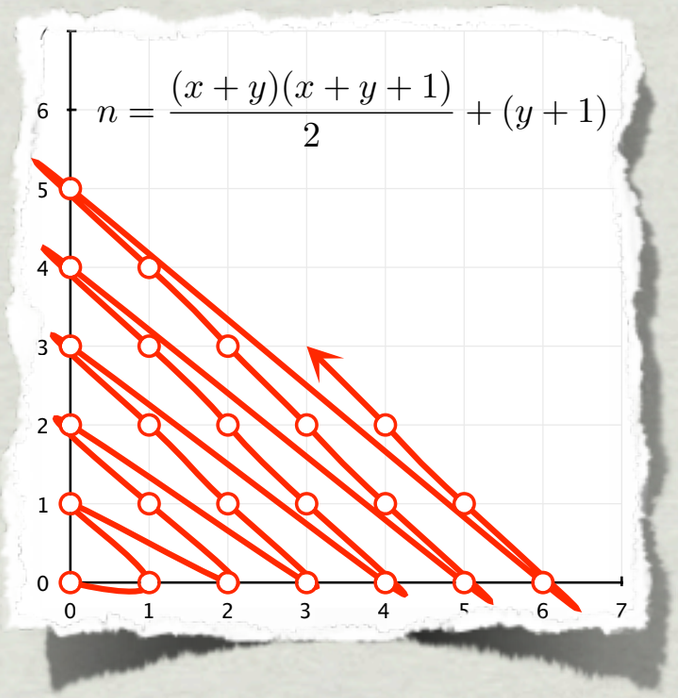
- **Poincaré (1890):** *« a very little cause that we cannot see has great consequences, so we say it is random »*



Strike 2: Gödel (1930s)



- **Gödel (1931):** Encode & decode any computation *arithmetically* into one integer



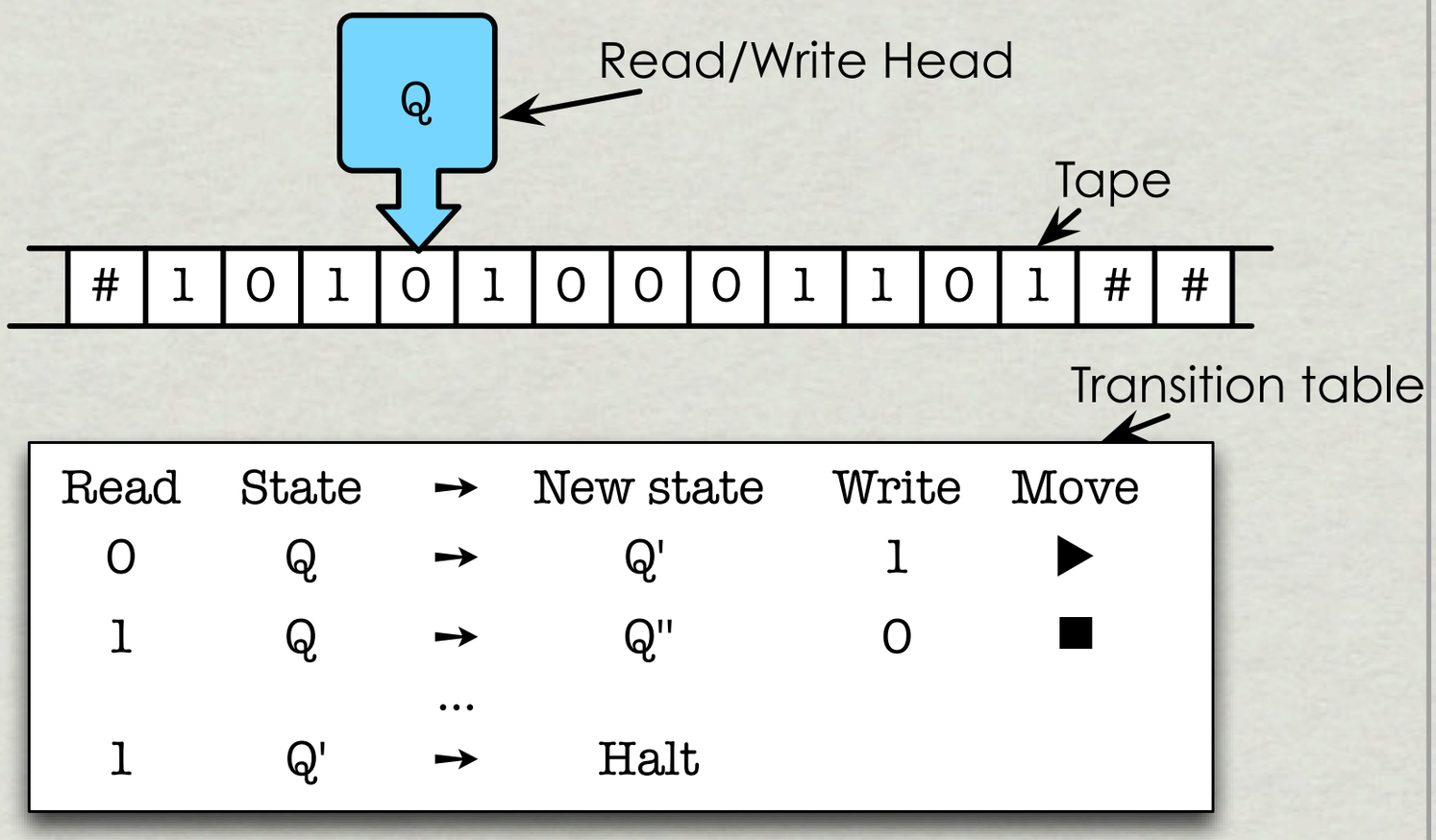
- **Theorem:** One can encode the termination of any computation into an arithmetic proposition

- **Corollary:** There are theorems (true propositions) that cannot proven nor denied

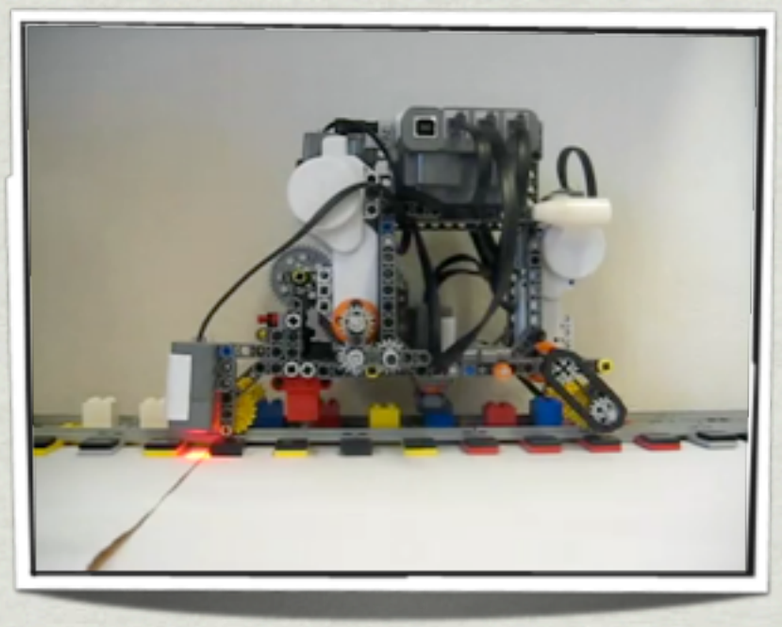
Mathematics cannot be complete

Strike 3: Turing (1935-)

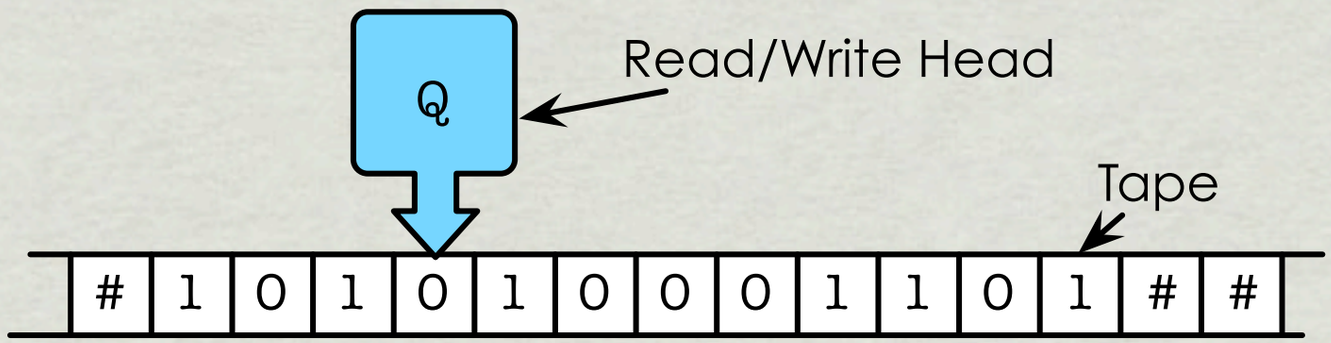
- **Alan Turing (1936) thesis:** A simple computation model that can emulate *all* possible computations



Strike 3: Turing (1935-)



- **Alan Turing (1936) thesis:** A simple computation model that can emulate *all* possible computations

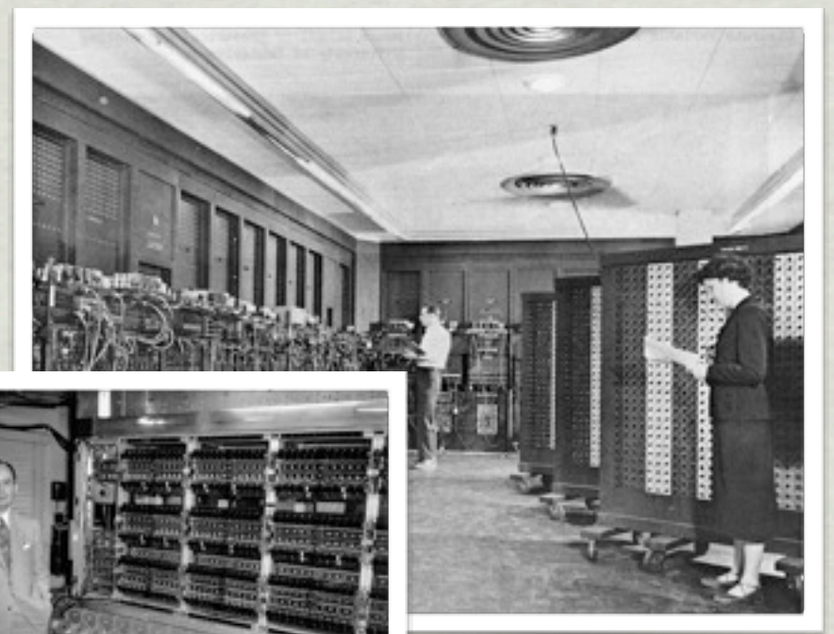


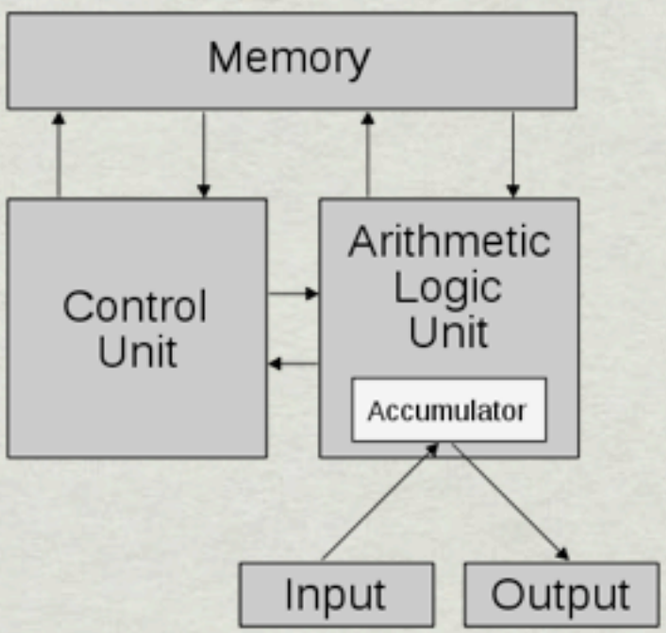
Transition table

Read	State	→	New state	Write	Move
0	Q	→	Q'	1	▶
1	Q	→	Q''	0	■
		...			
1	Q'	→	Halt		

Turing model: precursor of modern computers

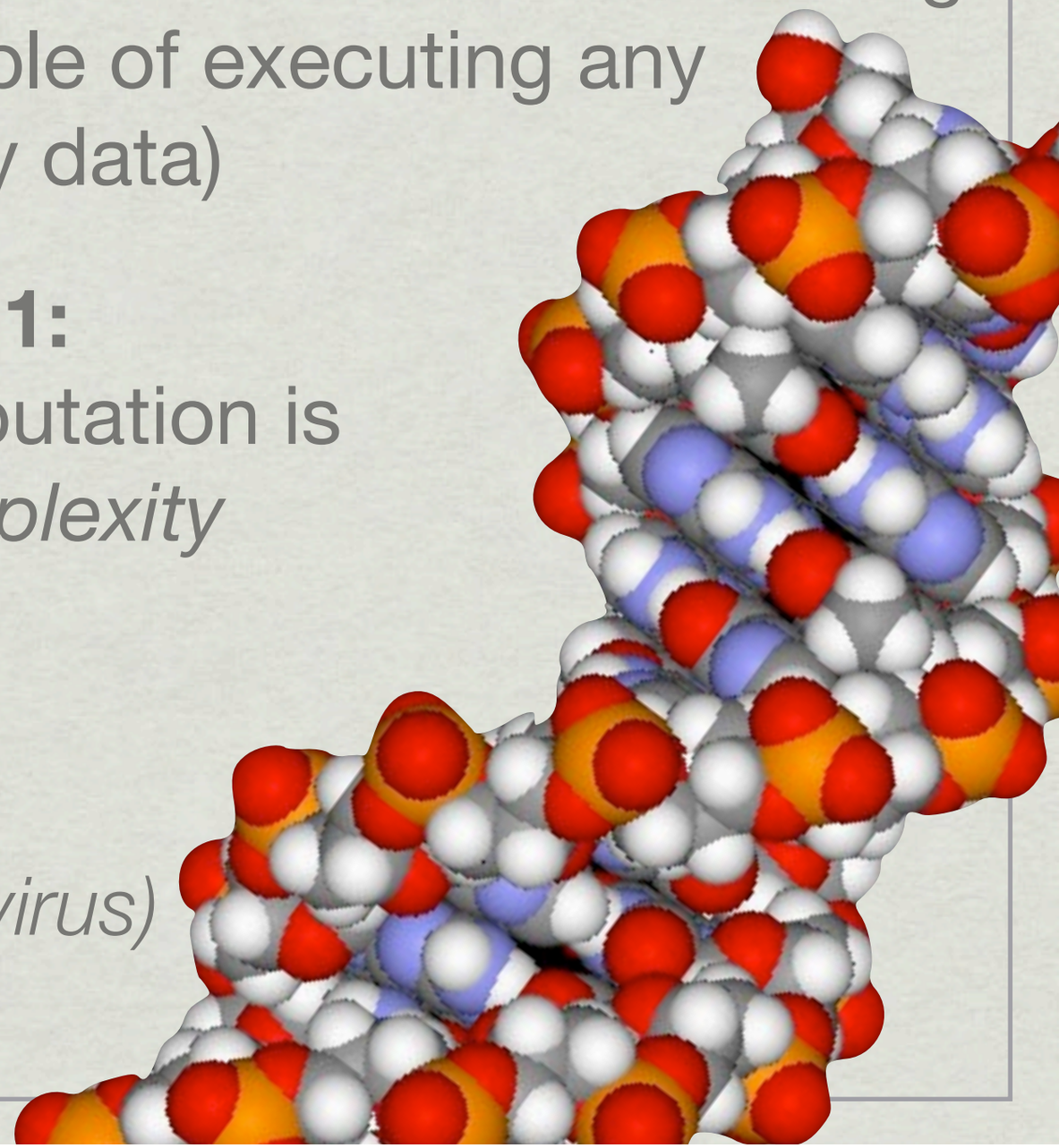
- **Zuse (1941):** Z3, first Turing complete binary computer
- **von Neumann (1946):** ENIAC, Turing complete decimal computer with modern architecture (RAM)





Strike 3: Turing (1935-)

- **Theorem:** There exists an *universal* Turing machine (capable of executing any program on any data)



- **Consequence 1:** Universal computation is of *modest complexity*

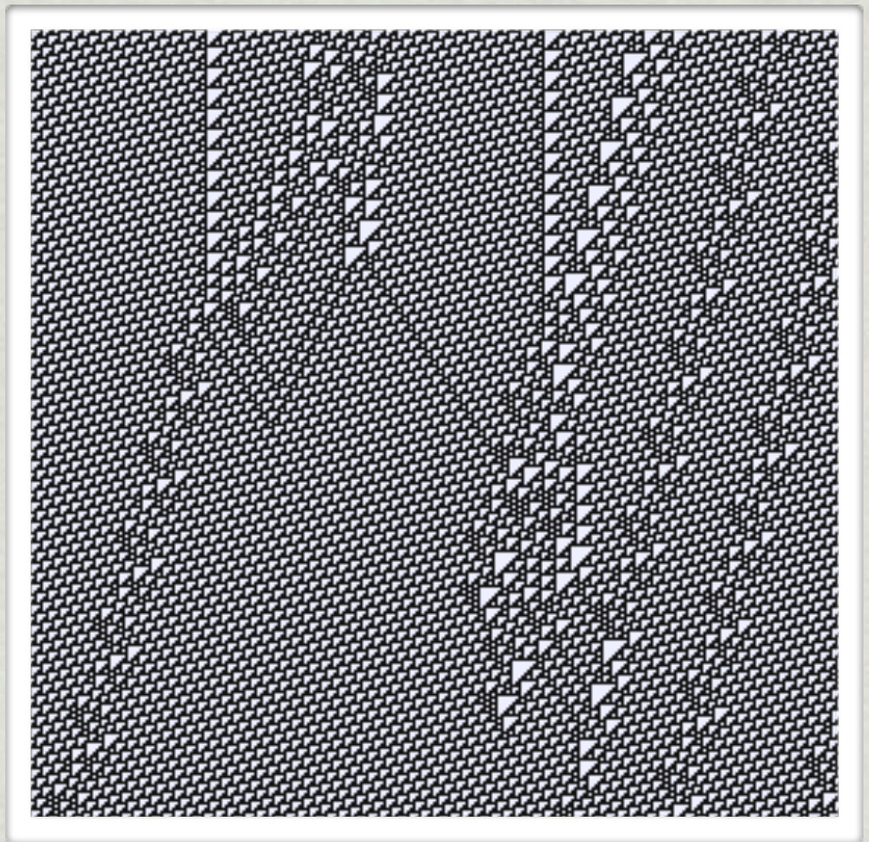
- **Consequence 2:**

  Data ⇆ Program

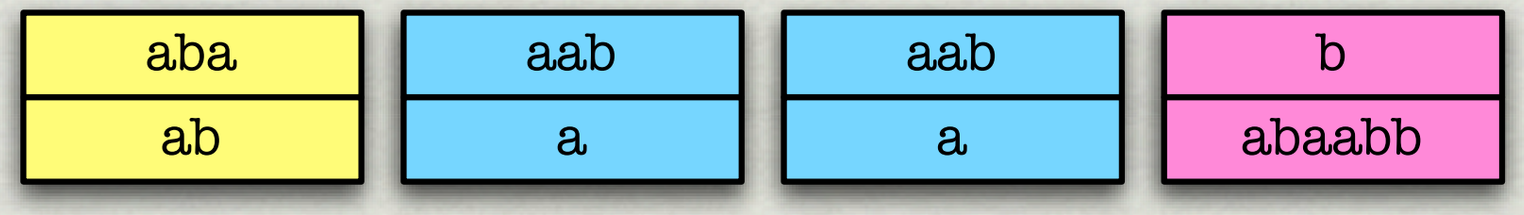
  (ex: biological & computer virus)

Universal computation

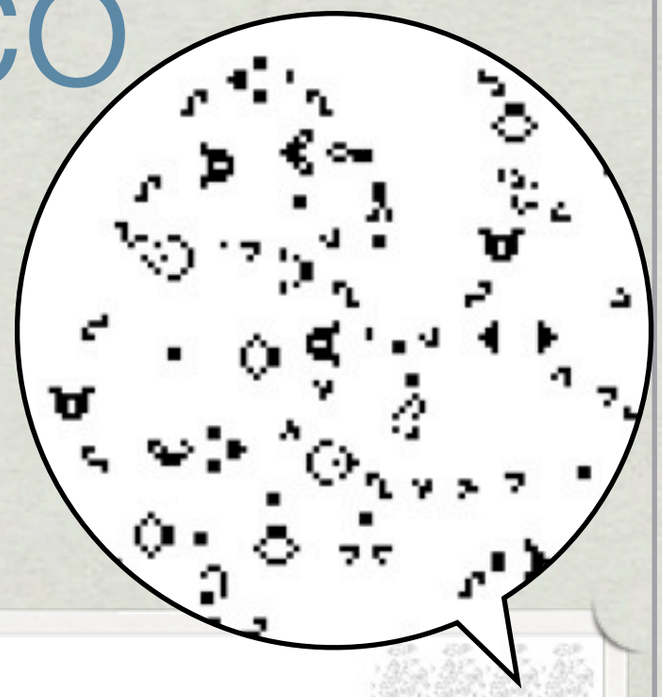
- Some small known universal machines:
  - **Cellular automata 110:** only 2 states, 2 neighbors
  - **Life:** 2 states and 8 neighbors
  - **Turing machine:** only 4 states and 5 letters
  - **Post correspondence:** only 7 tiles and it is undecidable



aba-aab-aab-b = ab-a-a-abaabb



Strike 3: Turing & co (1935-)



- **Theorem:** It is impossible to decide beforehand if a program will stop
- **Rice's theorem:** Every non-trivial proposition is undecidable

Almost all desirable properties are <u>unprovable</u> on powerful enough dynamical systems

Strike 4: Are decidable properties *predictable*?

- ***Not even!***

  Theorem: Every system capable of emulating boolean circuits is *P*-complete

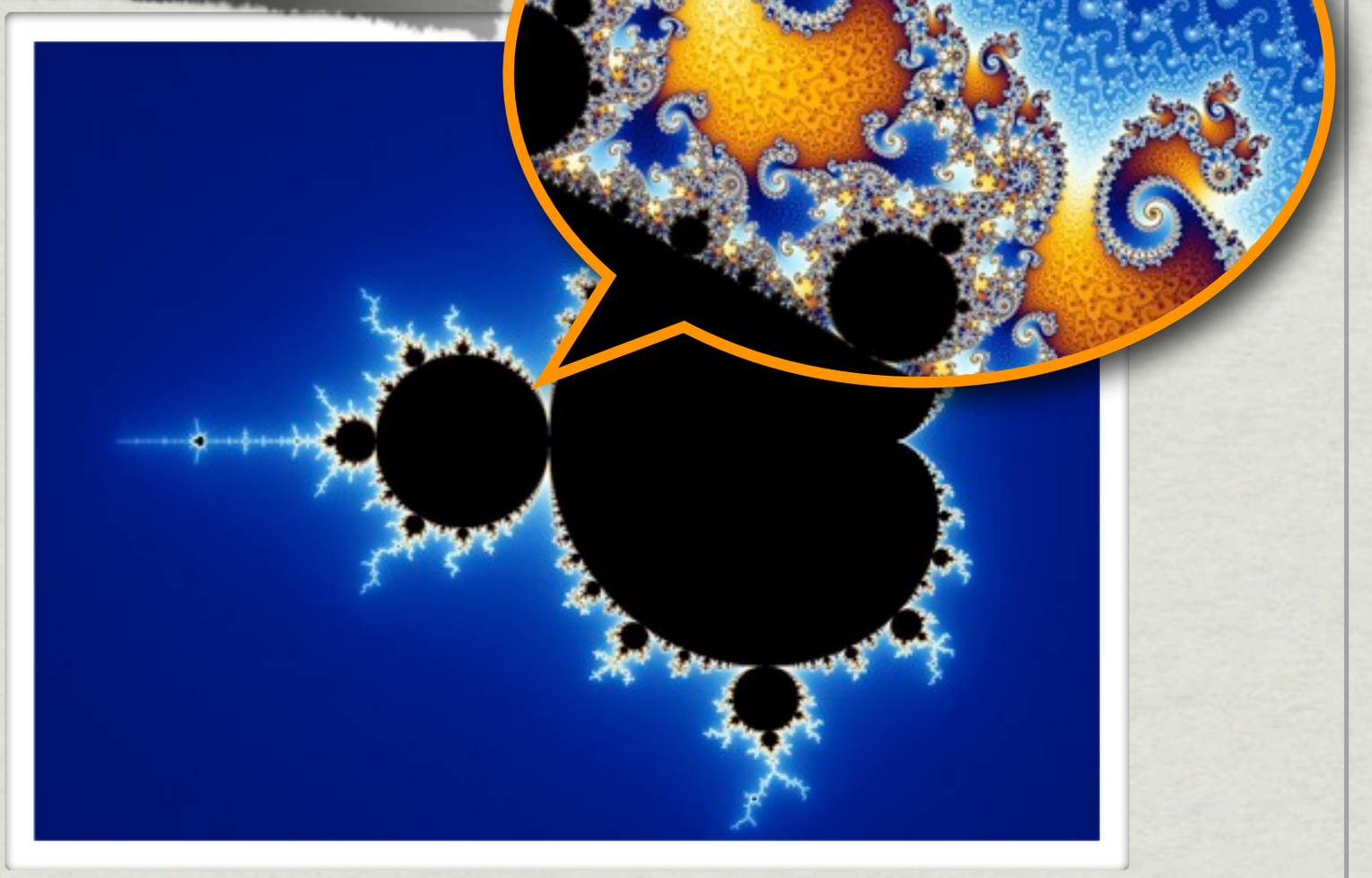
- *i.e.: accelerating its computation would imply solving a lot of other problems, more efficiently than widely believed*

- *(Hardness related to conjectures)*

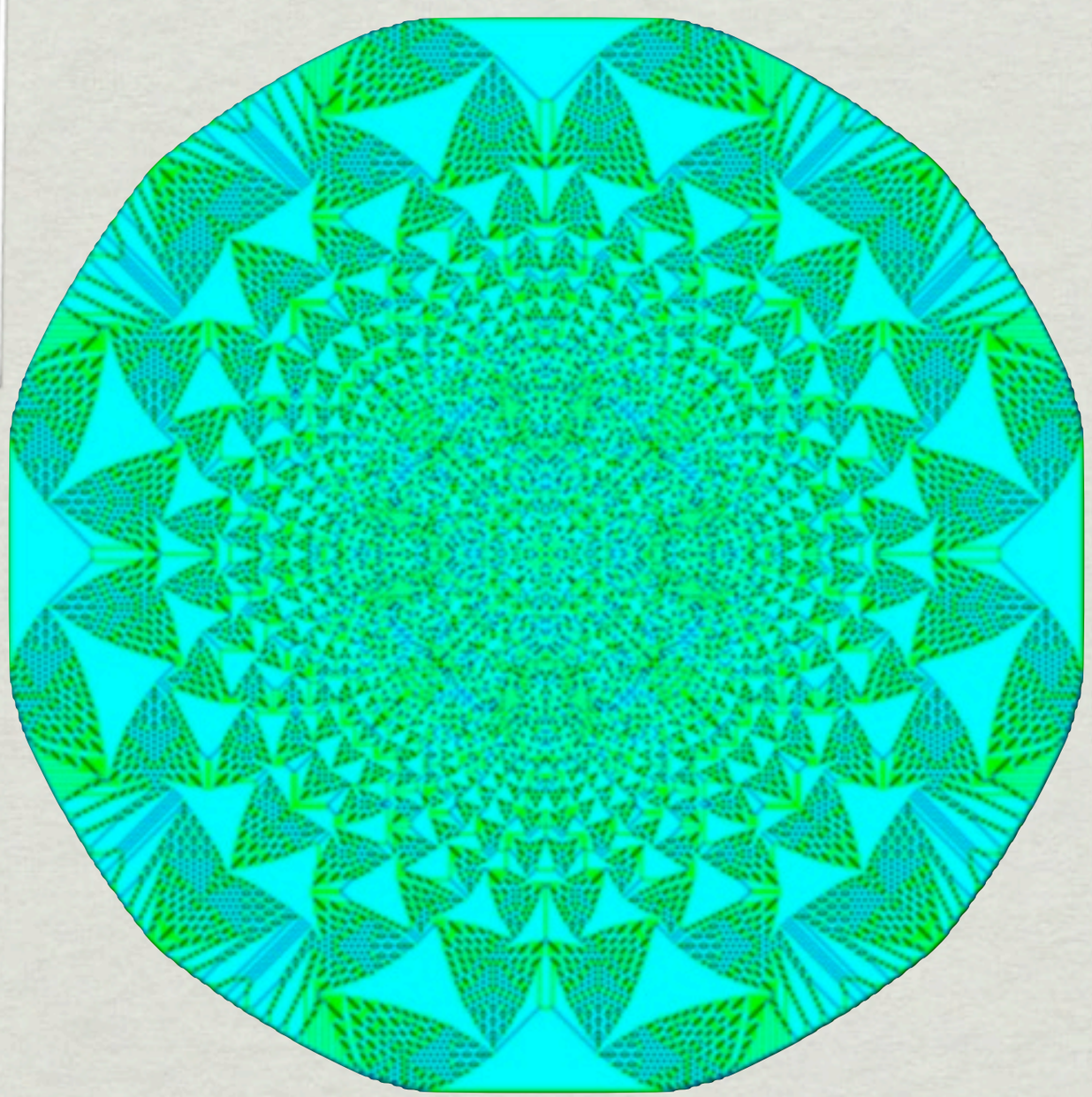
Simple objects with very complex structures

FractaleDeMandelbrot($n : \mathbb{N}$) :
for all $c \in [-2, 1] \times [-1, 1]$ **do**
let $f_c = z \mapsto (z^2 + c)$ **in**
if $|f_c^n(0)| \leqslant 2$ **then** paint c in noir.

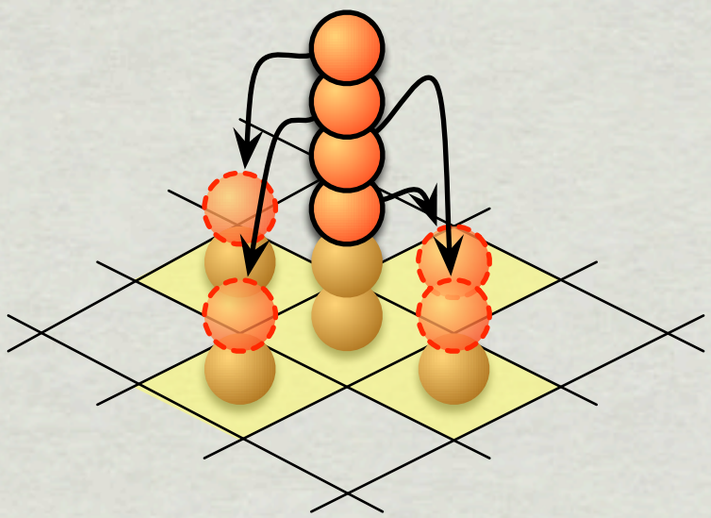
- **Fractals**
- Sand piles
- Cellular automata



Simple objects with very complex structures

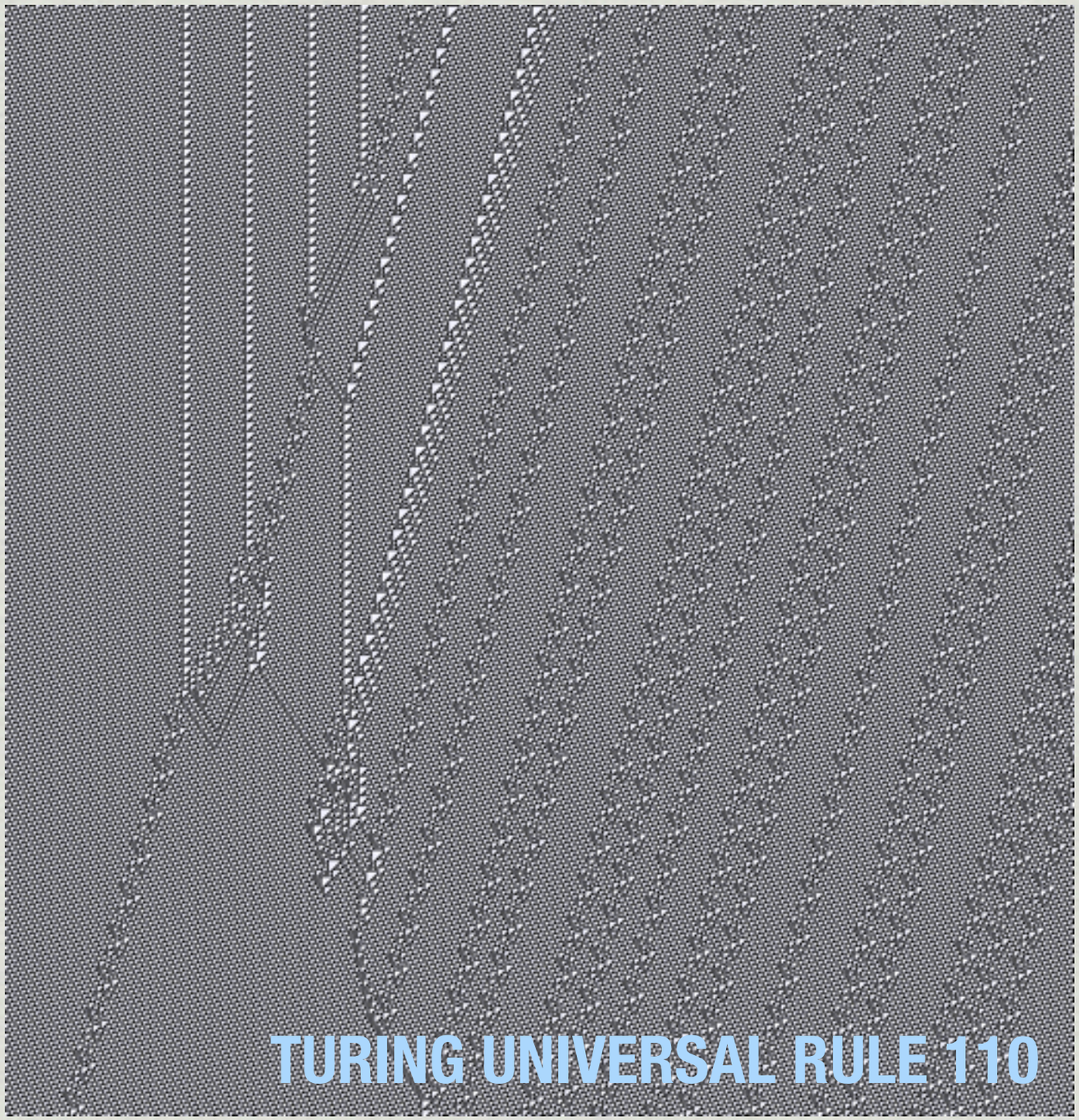


- Fractals
- **Sand piles**
- Cellular automata



Simple objects with very complex structures

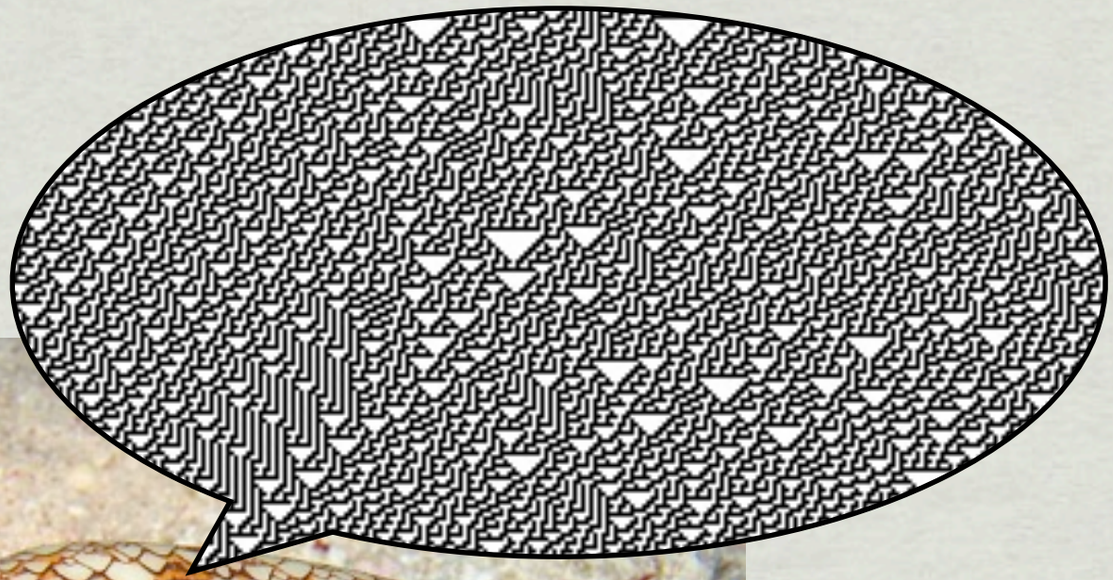
- Fractals
- Sand piles
- **Cellular automata**

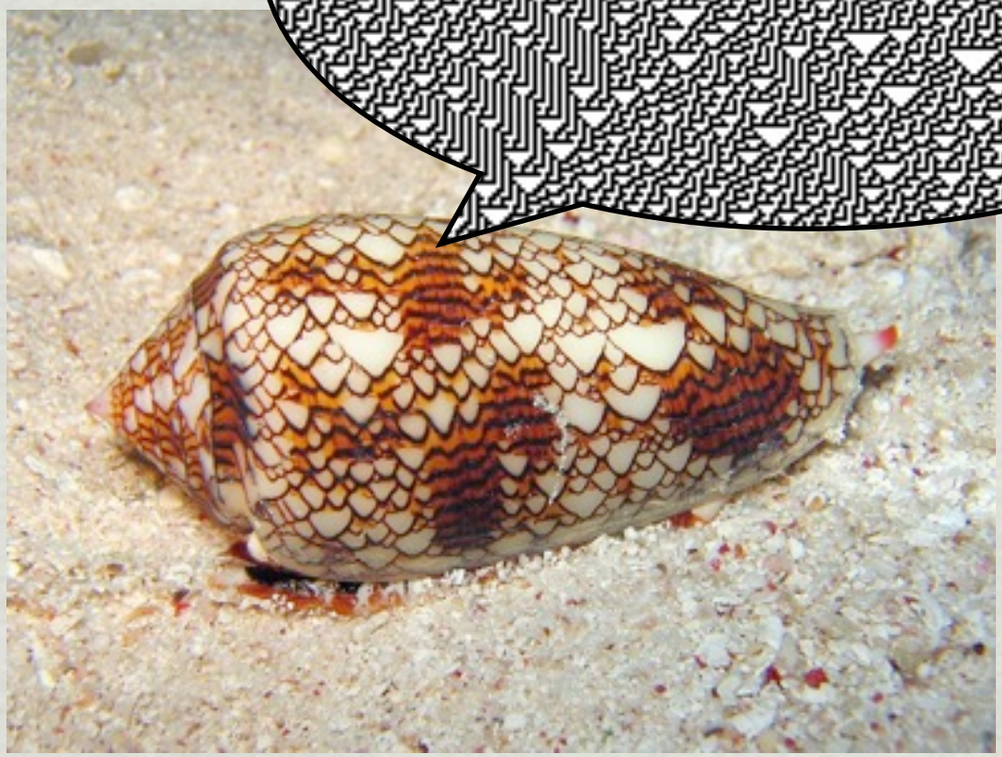


Simple objects with very complex structures

- Fractals
- Sand piles
- **Cellular automata**

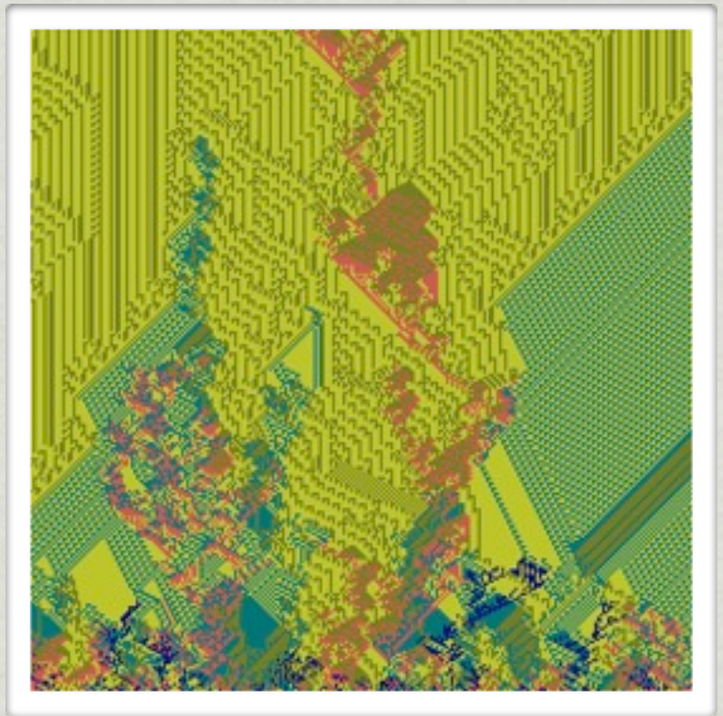
CA RULE 30





So...

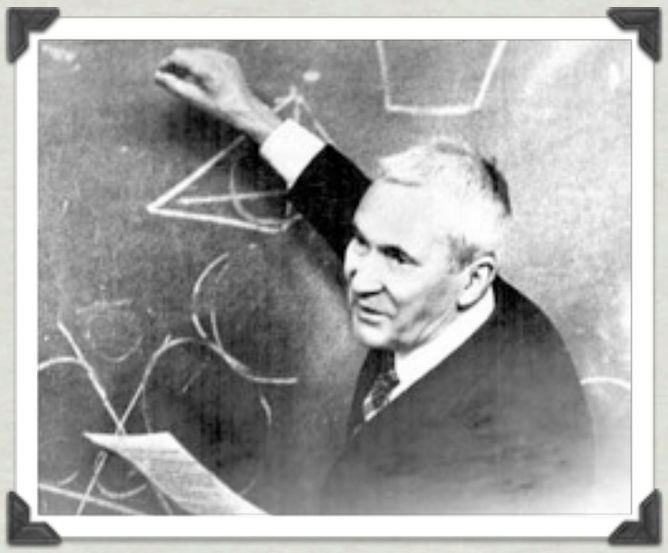
- Universal computation is problematic because:
  - it requires only small complexity
  - everything becomes undecidable
  - it can even be generated with probability 1-o(1)
- and... even decidable stuff may be unpredictable
  (like predicting the state of a cell after t steps in Life)



So... we have to be very careful when designing models!

Defining complexity and... randomness

Complexity measure: Kolmogorov & co (1966)



- **First:** *everything that will ever been said is a finite set of symbols and thus... an finite word.*

- **Solomonoff (1965), Kolmogorov & Chaitin (1966):**

  The K-complexity of a finite word is the *length of the smallest TM-program* that writes it on the tape of a given universal Turing machine.

  K(**0111001**) ≤ 8, since « **print 0111001** » works

Why are complexity and randomness related?

- Not all strings can be compressed...

  ... but K(w) ≤ length(w)+1

- Strings with the highest complexity have the exact same statistical properties as **random strings**

- These is a deterministic definition of random strings!

Complexity ≈ Degree of randomness

Complexity measure: a lot of frustrations

- **Theorem:** It is undecidable to determine if the K-complexity of a word is ≤ k

- Some small systems look complex:

- **Poincaré & Chaos theory:** randomness in nature may only be the byproduct of small chaotic dynamical systems (thus with small K-complexity)

Systems get really complex really fast, and small systems look already very random to us

Can we characterize formally what "looking random to us" could mean ??

A glance at recent progress

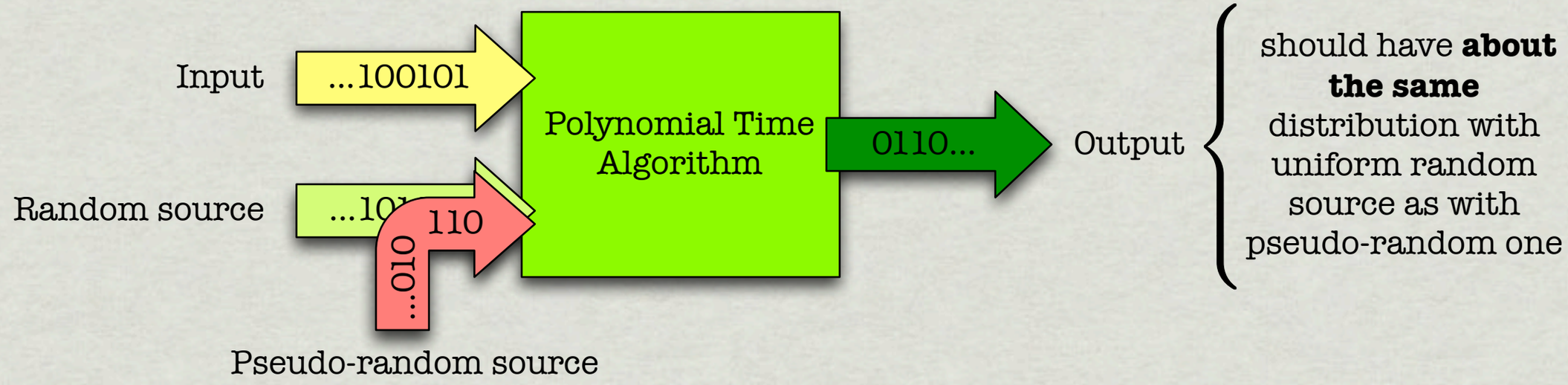
- We are computationally limited, as well as many small systems

- Computer science considers that the only accessible computation has to be conducted in polynomial time, i.e. polynomial in the size of the input.

Ex: searching for the max of n elements takes O(n) time

A glance at recent progress

- **Yao (1982):** Pseudo random strings = random strings that *fools* any polynomial algorithm



= deterministic chaos

- **Rabin (1981):** Unpredictable strings = random string whose *i*-th bit cannot be guessed by PT-algorithm with probability ≥ 1/2+ε

A glance at recent progress

PSEUDORANDOM STRINGS

UNPREDICTABLE STRINGS

PROVING THAT THERE EXISTS FUNCTIONS THAT ARE *REALLY* HARD TO COMPUTE

A glance at recent progress

PSEUDORANDOM STRINGS

UNPREDICTABLE STRINGS

PROVING THAT THERE EXISTS FUNCTIONS THAT ARE *REALLY* HARD TO COMPUTE

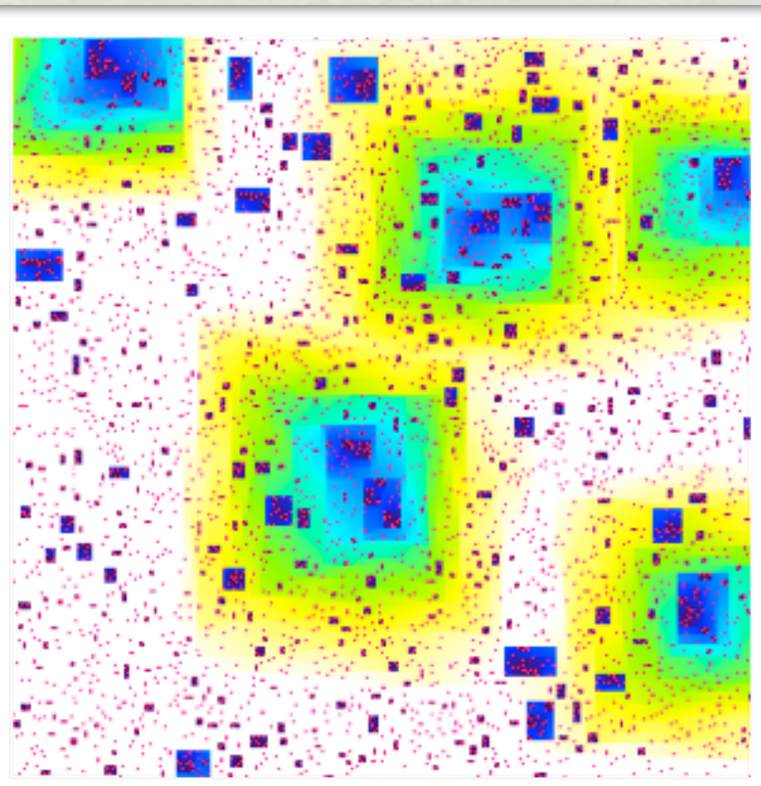
SIMILAR TO THE CONJECTURE P≠NP

Computer science & complex systems: *a long-term friendship*

Computer simulations

- Complex systems basically started with computer simulations
- Computer simulations are a non trivial subject

Ex: Holroyd's result on bootstrap percolation



Theorem (2002): $p(n)\ln n \sim \lambda = \pi^2/18$

≈ 0.548311

Simulations (1989): $\lambda = 0.245 \pm 0.015$

Turns out that correct prediction would require (2008): $n \geq 10^{500}$

Are simulations relevant?
But also, is theory relevant?

Other contributions of Computer Science related to Complex Systems

Contributions of C.S. related to C.S.

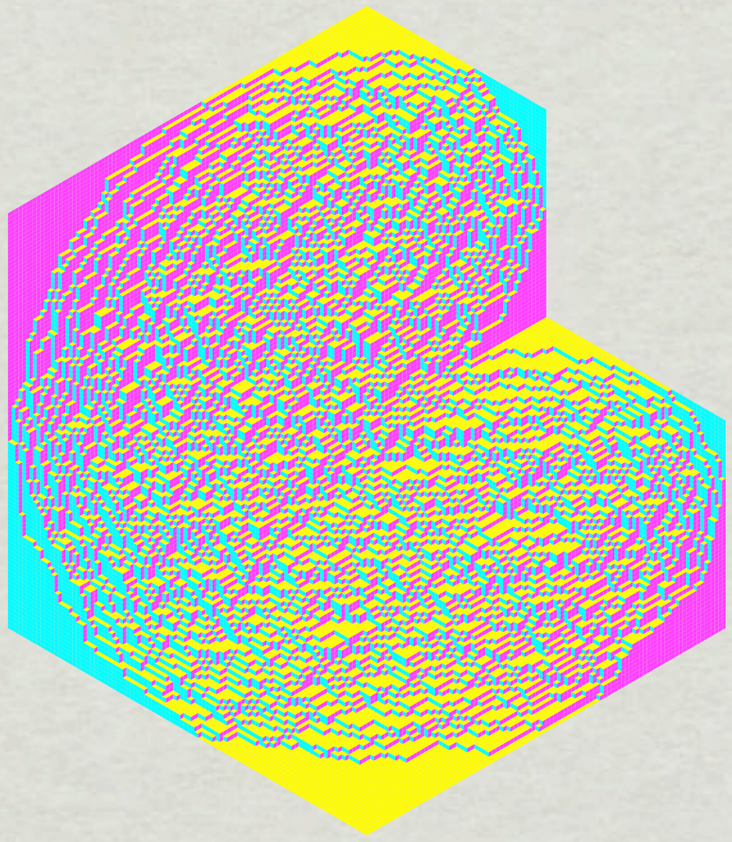
- Besides simulations, we have 4 categories:
  - Collecting, treating, and representing data
  - Mathematical tools
  - Conceptual: *things on which one may have the wrong intuition, enlarging thus the domain of possible*
  - Foundations: *things that help settling proper foundations of a complex system science in progress*

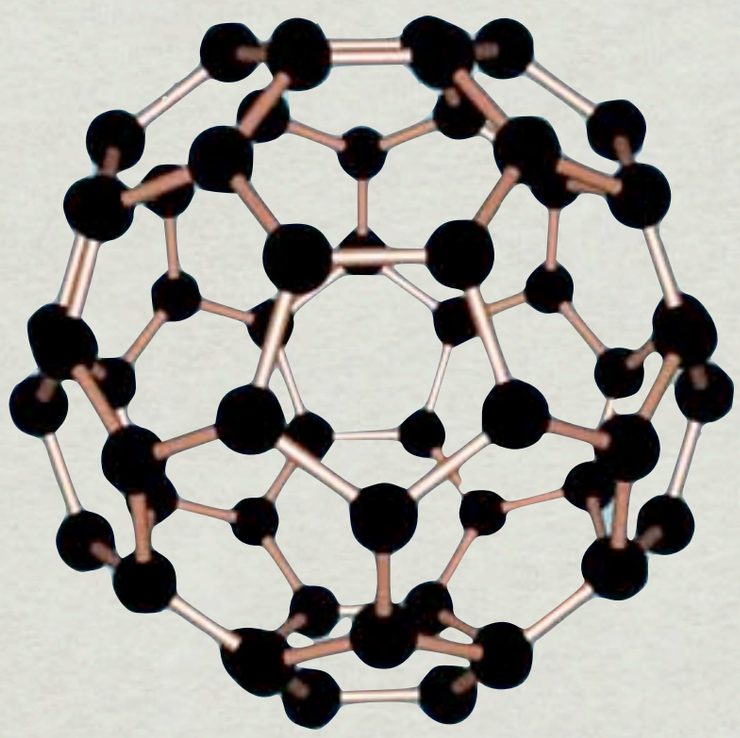
Collecting, treating, representing data

- Very large networks (billions of nodes and edges)
- Need for automatic technics to collect the data
- Need for specific algorithms to representing the data
- Need for specific algorithms to extract informations from the data

Mathematical tools

- Generating random elements in sophisticated combinatorial structures
- Proving the stability of molecules





Conceptual contributions

- Enlarge the exploration field of other domains
- The power of **randomness in computation**

Ex: random counters count up to n
with log(log n) bits only!

Ex: Self-correcting codes

Ex: Zero-knowledge

Contributions to foundations

- Working on defining what complex means
- Recent progresses relating the existence of pseudo-random sequences to old classic conjectures in deterministic computation (circuit lower bounds)

Thank you & Feliz Cumpleaños Eric !