Exploring Complexity

In Science and Technology

Oct. 6, 2010

Jeff Fletcher

Logistics

- Lab1 (dynamics) due next Monday Oct. 11
  - Online--Blackboard
  - Word format ONLY!
- HW2 due next Wednesday Oct. 13
- No class on Monday Oct. 11
  - we'll shift schedule as needed
  - How many got my email about Blackboard?

Review 1

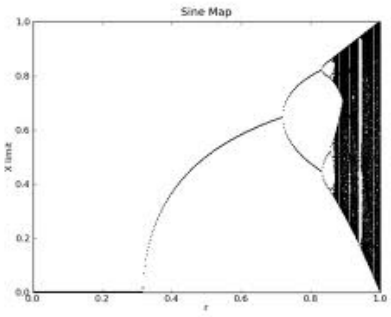
- Ways of Knowing (many)
- Science is a particular way of knowing
  - Natural explanations
  - Skepticism about mental biases
- Entropy (order vs. disorder)
- Information (degree of surprise)
  - Analogy between Entropy and Information
- Hierarchy of Matter, Energy, Information
- Boltzmann Entropy (physical)
  - Ordered macrostates have less microstates
- Shannon Entropy (information)
  - Bits
  - 1 to *i* are message symbols

$$S = K_b \sum_i p_i \log p_i$$

$$H = -\sum_i p_i \log_2 p_i$$

Review2: Another Chaotic Map

$$x_{t+1} = \frac{R}{4}\sin(\pi x_t)$$



- Note: Sine in radians, but Netlogo uses degrees. Multiply by 180/pi

Computation

- Motivating questions:
  - What does “computation” mean?
  - What are the similarities and differences between computation in computers and in natural systems?
  - What are the limits of computation? Are there things that cannot be “computed”?

Hilbert’s problems (1900)

- 23 unsolved problems in mathematics
- The most significant
  - Is mathematics complete?
  - Is mathematics consistent?
  - Is mathematics decidable?
- What do we mean by each of these questions?

David Hilbert 1862 − 1943

Hilbert's problems (1900)

- Is mathematics complete?
  - Can every mathematical statement be proved or disproved from a finite set of axioms?
- Is mathematics consistent?
  - Can only the true statements be proved?
- Is mathematics decidable?
  - Is there a "definite procedure" that can be applied to every statement that will tell us in a finite time whether the statement is true or false?

Gödel's Incompleteness Theorem (1930)

- At this time Hilbert (and many others) assumed Hilbert's questions would be answered "YES" in all three cases.
- Proved Arithmetic (and more complex systems of mathematics) must either be incomplete or inconsistent (or both)

Kurt Gödel, 1906-1978

Gödel's Incompleteness Theorem (1930)

- Cleverly was able to encode in the arithmetic system the statement: "This statement is unprovable."
- Why is this a problem?
  - If you can prove this statement, then math is inconsistent
  - If can't prove this statement, then math is incomplete
  - Conclusion: there are some statements in math that are true, but not provable.

The "Entscheidungs" (decision) problem

- Hilbert's 3rd Question
  - Is there a "definite procedure" that can be applied to every statement that will tell us in a finite time whether the statement is true or false?
- What is a "definite procedure"?

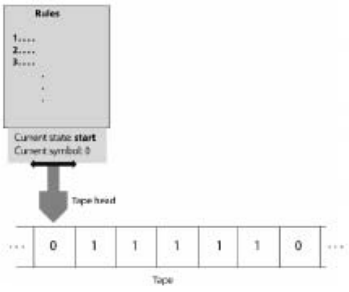
Turing Machines

- A way of implementing a "definite procedure"
- 3 parts
  - An infinite tape divided into cells that can contain symbols
  - A movable read/write head that can be in different states
  - A set of rules that tell the head what to do next
    - read/write, change state, move

Alan Turning, 1912-1954

Simple Example: Even or Odd?

- Symbols:
  - 0, 1, blank
- States:
  - Start, even, odd, halt
- What is a definite procedure for solving this classification problem?



A Definite Procedure

1. If you are in the **start** state and read a 0, then change to the **even** state, replace the 0 with a *blank* (i.e., erase the 0), and move one cell to the right.
2. If you are in the **even** state and read a 1, change to the **odd** state, replace the 1 with a *blank*, and move one cell to the right.
3. If you are in the **odd** state and read a 1, change to the **even** state, replace the 1 with a *blank*, and move one cell to the right.
4. If you are in the **odd** state and read a 0, replace that 0 with a 1 and change to the **halt** state.
5. If you are in the **even** state and read a 0, replace that 0 with a 0 (i.e., don't change it) and change to the **halt** state.

Encoding a Turing Machine

- Goal: encode TM rules as a binary string
- Current State—Current Symbol—New State—New Symbol—Motion
- Example from rule 1
- start—0—even—blank--right
- Possible code:

States	Symbols	Actions
start = 000	'0' = 000	move_right = 000
even = 001	'1' = 001	move_left = 111
odd = 010	'blank' = 100	
halt = 100		

- separator (—)= 111

Encoding a Turing Machine

- Rule 1: start—0—even—blank--right
- Possible code:

States	Symbols	Actions
start = 000	‘0’ = 000	move_right = 000
even = 001	‘1’ = 001	move_left = 111
odd = 010	‘blank’ = 100	
halt = 100		
	separator (—)= 111	

- What does rule 1 look like using this code?

Universal Turing Machine

- Special “universal” Turing machine U
- Input on U’s tape is code for some other Turing machine M, plus input **I** for that Turing machine
- Universal TM U “runs M on **I**”
- In other words, U is simply a programmable computer!
  - Where M is a program or “definite procedure”

Turing's solution to the Entscheidungs problem

- Recall the Entscheidungs problem:
  - Is there always a definite procedure that can decide whether a statement is true?
- What Turing did:
  - 1. Proposed a particular statement
  - 2. Assume there exists a definite procedure (a Turing machine H) for deciding the truth or falsity of this statement
  - 3. Prove that H cannot exist.

Turing's proof that H cannot exist

- Let **U(M, I)** denote the results of having universal TM U run M on **I**
- Turing's key insight: can have **I** = M'
  - That is, can have U(M, M'**)**
- Example of this in your day-to-day experience?
- Can also have U(M, M**)**
  - Example?

Turing's proof that H cannot exist

- Proof Setup:
  - Statement: "Turing Machine M, when run on input **I**, will halt after a finite number of time steps."
    - (Example of machine that does not halt?)
  - Assume H exists, such that H(M, **I**) will answer "yes" if M halts on **I**; "no" if it does not, for any M, **I**.
  - So H is an infinite loop detector
  - We will later show this leads to a contradiction.

Problem of Designing H is the "Halting Problem"

- Not clear how to design H, but let's assume it exists.
- Given H, let's create H' as follows:
  - H' takes as input the code of a Turing machine M
  - H' then runs H(M,M)
  - If H(M,M) answers "yes" (i.e., "yes, M halts on input M), then H' goes into an infinite loop.
  - If H(M,M) answers "no" (i.e., "no, M does not halt on input M), then H' halts.
- Now, Turing asked, does H' halt when given H' as input?
- Can you spot the contraction?

Proof by Contradiction

- Assume H'(H') does not halt
  - But H'(M) does not halt only if M halts
  - Here H' as input = M
  - So H'(H') does not halt only if H' halts!
- Assume H'(H') does halt
  - But H'(M) halts only if M does not halt
  - So H'(H') halts only if H' does not halt!
- So H' cannot exist

Proof by Contradiction

- But everything H' does is allowed for in a Turing Machine
- So H also cannot exist
- There is no “definite procedure” for solving the halting problem
- So therefore some statements are undecidable
- So NO to Hilbert’s 3rd question
- Note that Turning’s proof uses same kind of logic as Gödel’s: self-referential

NetLogo Code for Sine Map

- Changing calculation of x_new and x_new'
  - In setup and iterate procedures
  - Radians to degrees
- Changing plot name
  - And reference to plot name
- Changing calculation of x
  - In draw_parobola procedure