

## 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”?

## Some history...

### Hilbert’s problems (1900)

- Is mathematics complete?
- Is mathematics consistent?
- Is mathematics decidable?



David Hilbert, 1862 – 1943

### Some history...

#### Hilbert's problems (1900)

- Is mathematics complete?  
Can every mathematical statement be proved or disproved from a finite set of axioms?
- Is mathematics consistent?
- Is mathematics decidable?



David Hilbert, 1862 – 1943

### Some history...

#### 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?



David Hilbert, 1862 – 1943

## Some history...

### 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?

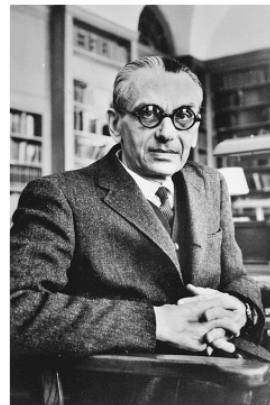
– Is mathematics decidable?

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?



David Hilbert, 1862 – 1943

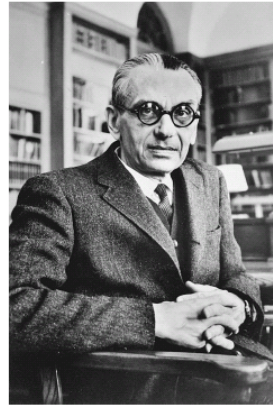
**Gödel's incompleteness theorem: Arithmetic (and more complex systems of mathematics) must either be incomplete or inconsistent (or both)**



Kurt Gödel, 1906 – 1978

**Gödel's incompleteness theorem: Arithmetic (and more complex systems of mathematics) must either be incomplete or inconsistent (or both)**

**"This statement is unprovable."**



Kurt Gödel, 1906 – 1978

### **The "Entscheidungs" (decision) problem**

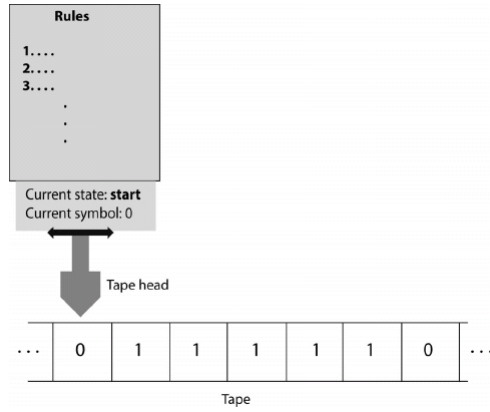
Is there a "definite procedure" that can be applied to every statement that will tell us in a finite time whether the statement is true or false?

What is a "definite procedure"?

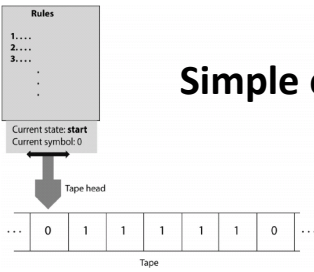
# Turing machines

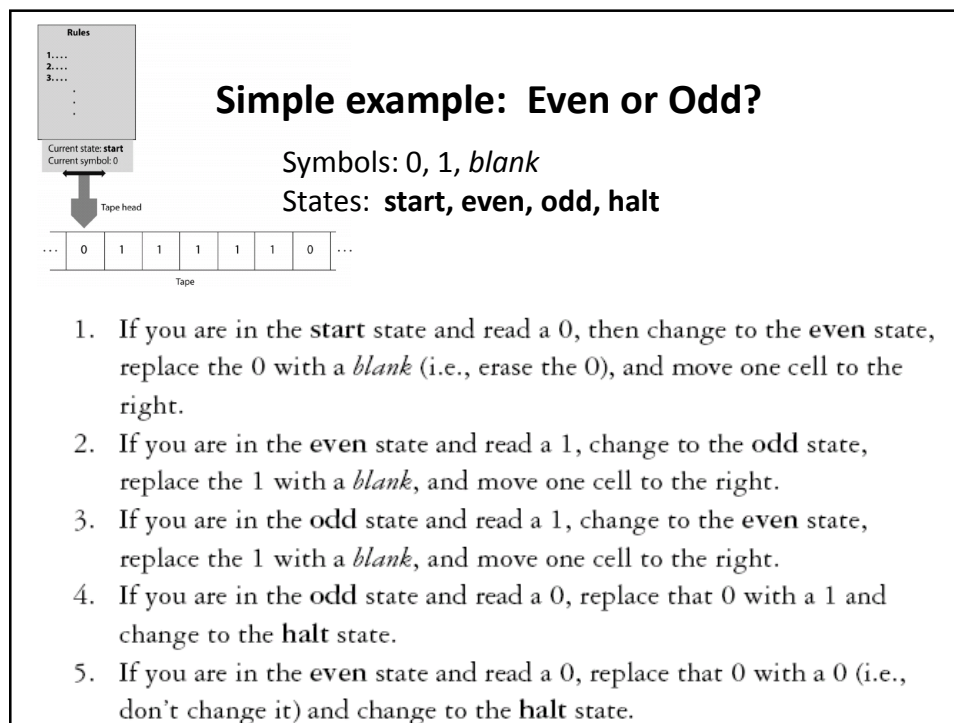
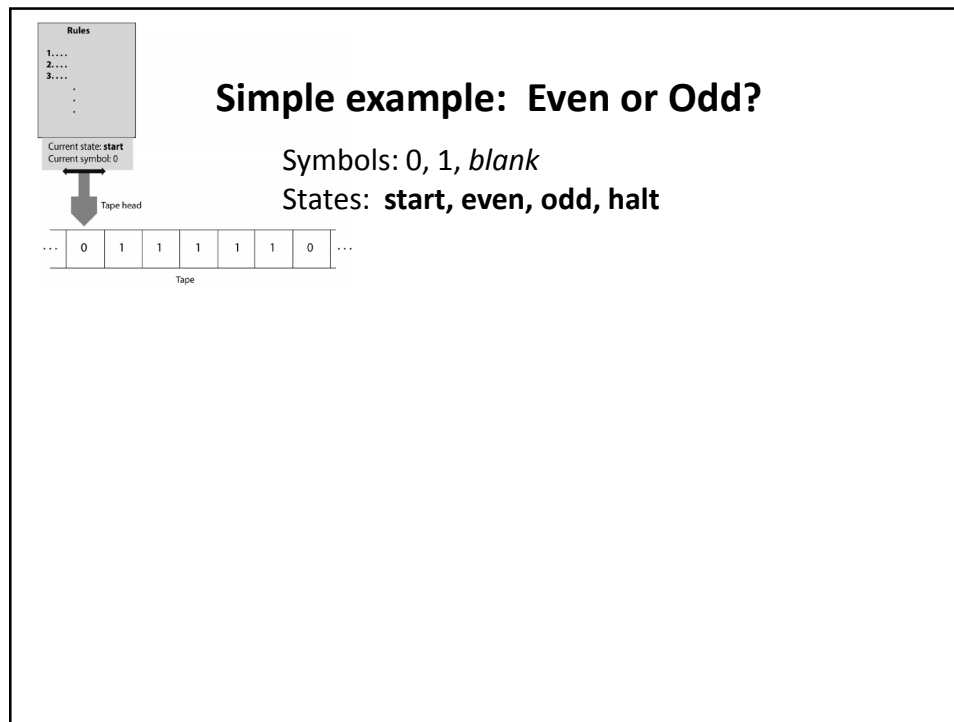


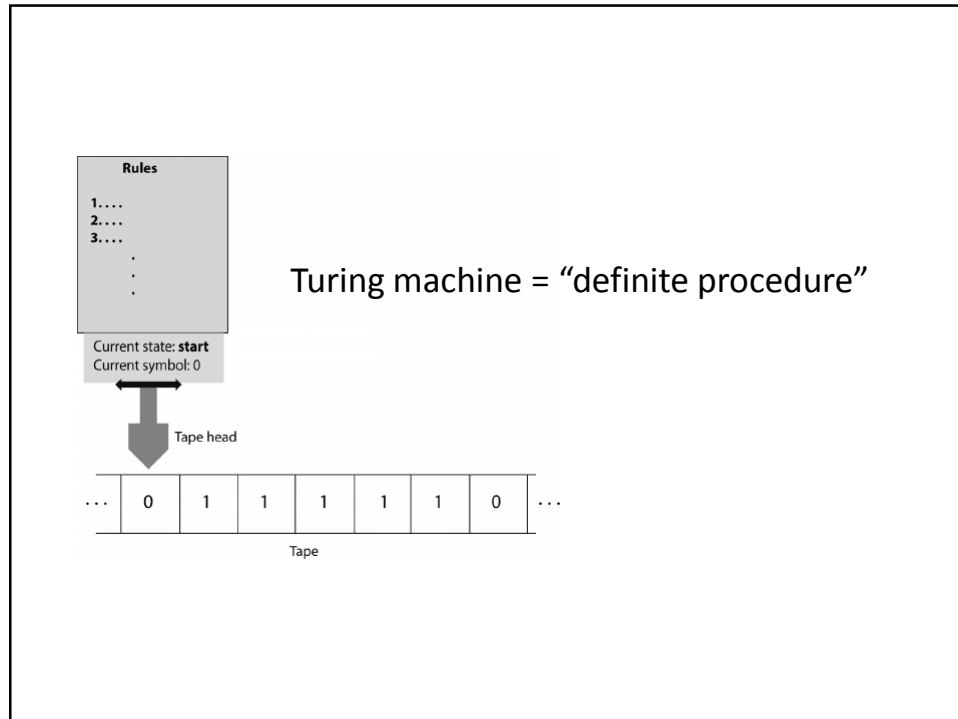
Alan Turing, 1912 – 1954



## Simple example: Even or Odd?







## Encoding a Turing Machine

Goal: encode TM rules as a binary string

—Current State—Current Symbol—New State—New Symbol—Motion—

In this shorthand, Rule 1 above would be written as:

—start—0—even—*blank*—right—

## Encoding a Turing Machine

Goal: encode TM rules as a binary string

—Current State—Current Symbol—New State—New Symbol—Motion—

In this shorthand, Rule 1 above would be written as:

—start—0—even—*blank*—right—

### Possible code:

<u>States</u>	<u>Symbols</u>	<u>Actions</u>
<b>start</b> = 000	'0' = 000	move_right = 000
<b>even</b> = 001	'1' = 001	move_left = 111
<b>odd</b> = 010	'blank' = 100	
<b>halt</b> = 100		
	separator (—) = 111	

## Encoding a Turing Machine

Goal: encode TM rules as a binary string

—Current State—Current Symbol—New State—New Symbol—Motion—

In this shorthand, Rule 1 above would be written as:

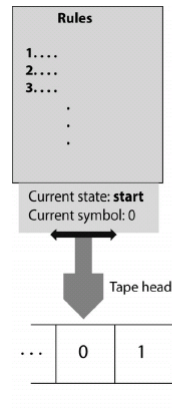
—start—0—even—*blank*—right—

### Possible code:

<u>States</u>	<u>Symbols</u>	<u>Actions</u>
<b>start</b> = 000	'0' = 000	move_right = 000
<b>even</b> = 001	'1' = 001	move_left = 111
<b>odd</b> = 010	'blank' = 100	
<b>halt</b> = 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: 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!

## 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 (= 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:

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  $M, I$ . (Will later show this leads to a contradiction.)

- Problem of designing **H** is called 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?**

### Recap

- In the 1930s, Turing formalized the notion of “definite procedure”
- This formalization opened the door for the invention of programmable computers in the 1940s. Turing machines were blueprints for the “von-Neumann architecture”.
- Turing also showed that there are limits to what can be computed.
- 19<sup>th</sup> century: all seemed possible in science and mathematics
- 20<sup>th</sup> century: limits to prediction and determinism (quantum mechanics, chaos); limits to mathematics and computation
- 21<sup>st</sup> century: ?