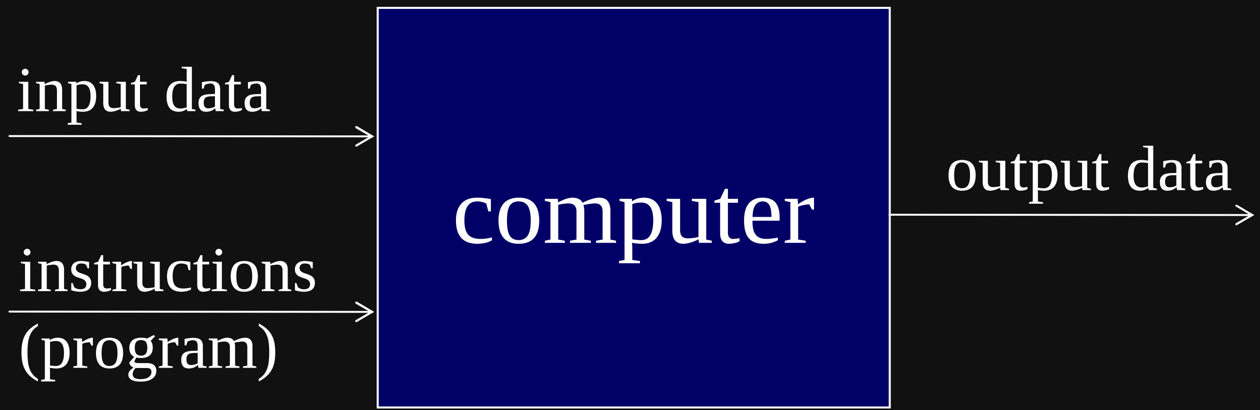
CSC 3130: Automata theory and formal languages

Turing Machines

Andrej Bogdanov

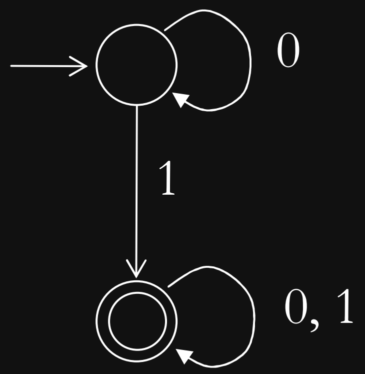
http://www.cse.cuhk.edu.hk/~andrejb/csc3130

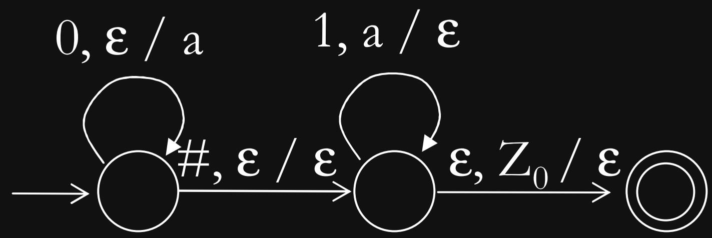
What is a computer?



A computer is a machine that manipulates data according to a list of instructions.

Are DFAs and PDAs computers?



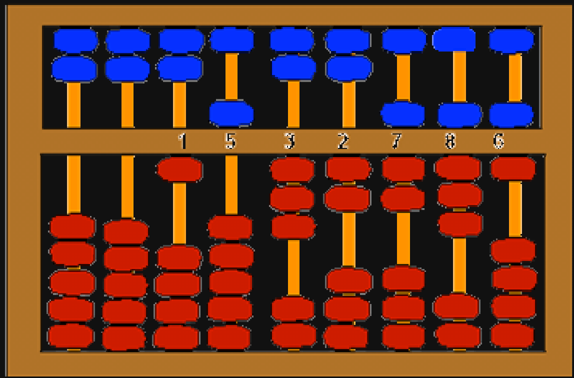


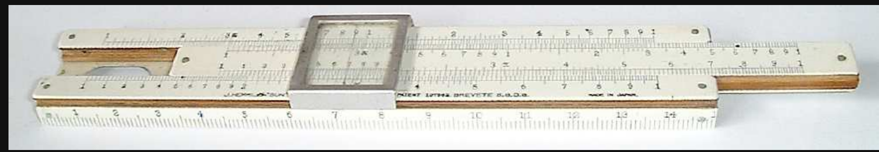
yes

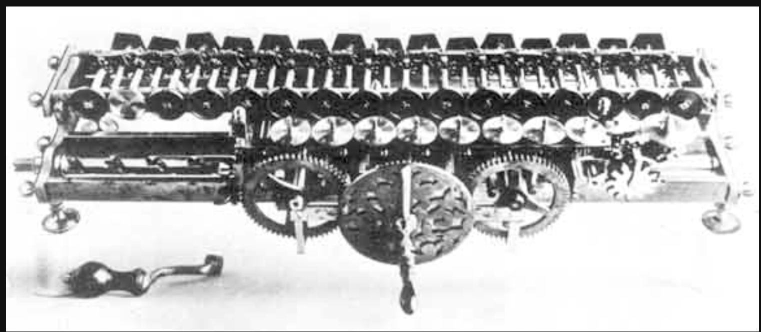
no!

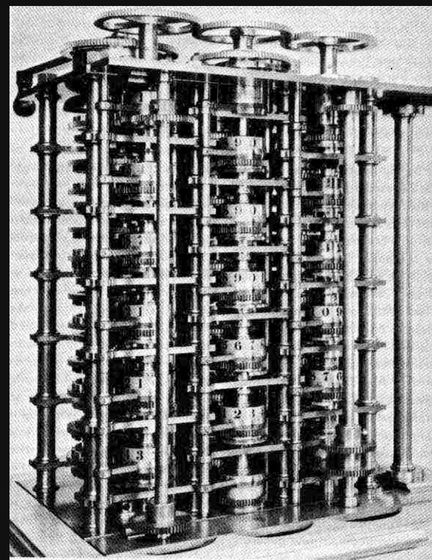
A computer is a machine that manipulates data according to a list of instructions.

A short history of computing devices


abacus
(Babylon 500BC, China 1300)

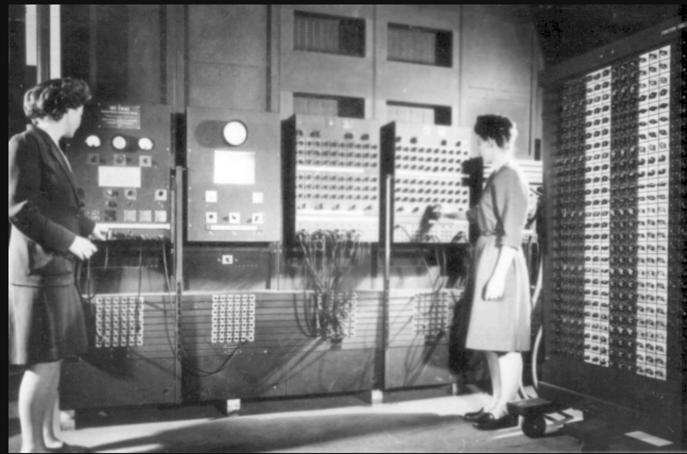

slide rule (1600s)


Leibniz calculator (1670s)


Babbage’s
mechanical
engine
(1820s)

A brief history of computing devices

Z3 (Germany, 1941)

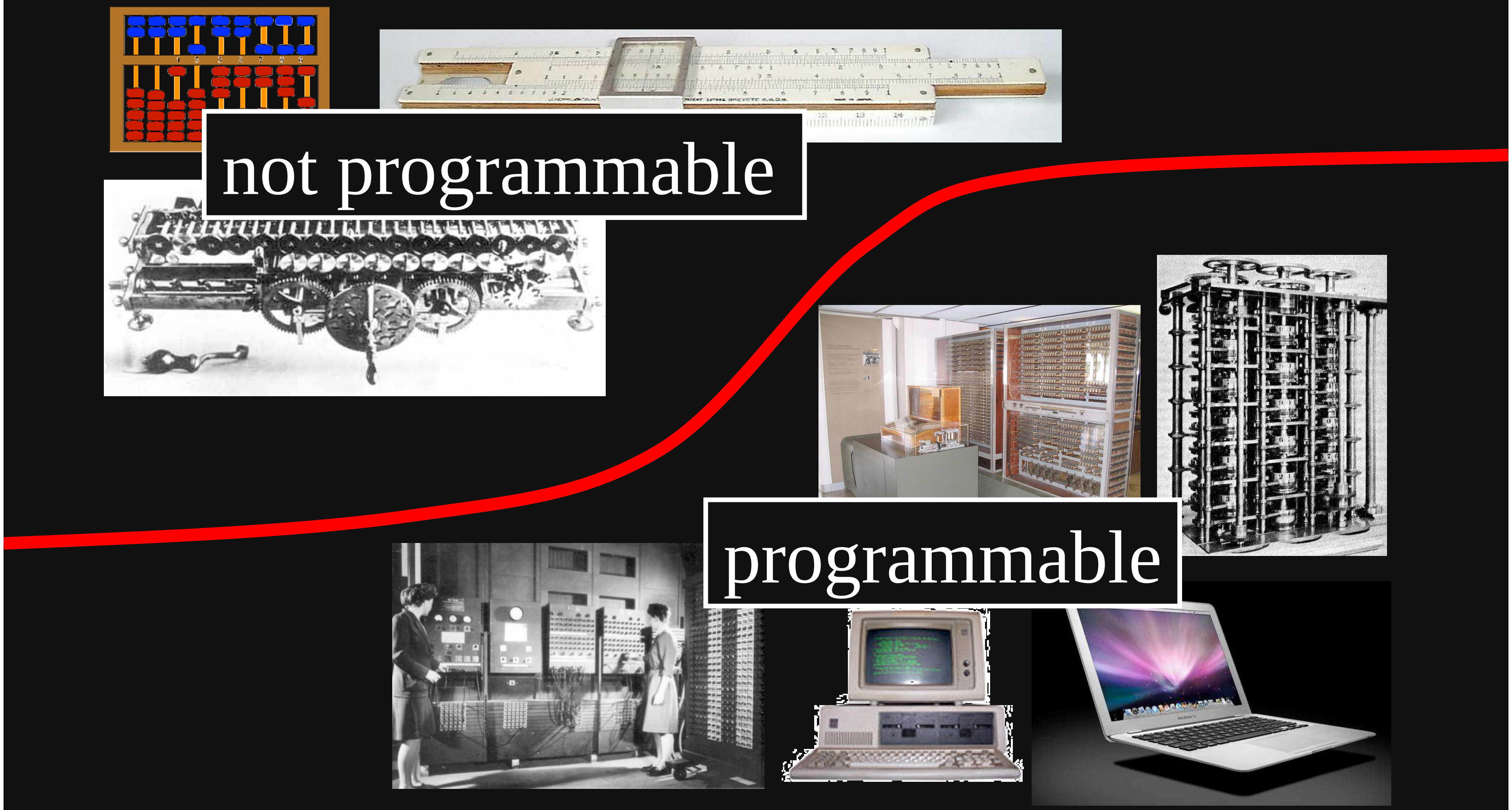


ENIAC (Pennsylvania, 1945)

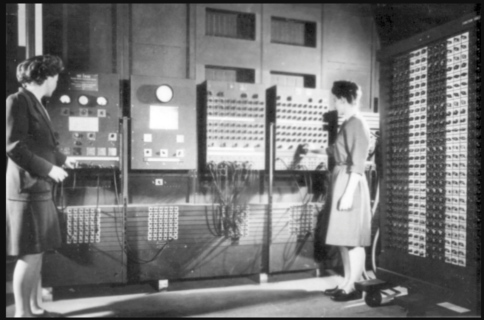
PC (1980)

MacBook Air (2008)

Which of these are "real" computers?



Computation is universal



In principle, all these computers
have the same problem-solving ability

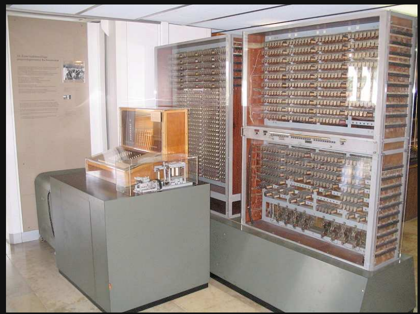
C++

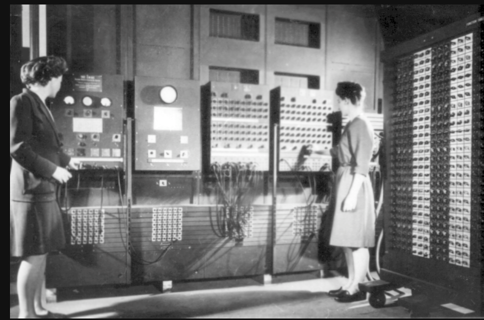
java

python

LISP

The Turing Machine





Turing Machine

C++

java

python

LISP

The Turing Machine

- The Turing Machine is an abstract automaton that is meant to model any physically realizable computer
  - Using any technology (mechanical, electrical, nano, bio)
  - Under any physical laws (gravity, quantum, E&M)
  - At any scale (human, quantum, cosmological)

- How come there is one model that can capture all these vastly different settings?

- Why do we need an abstract model of a computer?

Computation and mathematical proofs

- Prove that...

$\sqrt{2}$ is irrational.
(Archimedes' Theorem)

The equation
$$x^n + y^n = z^n,\ n \geq 2$$
has no integer solutions.
(Fermat-Wiles Theorem)

$G = (L, R)$ has a perfect matching iff $|\Gamma(S)| \geq |S|$ for every $L \subseteq S$.
(Hall's Theorem)

The language $0^n 1^n$ is not regular.

$L \subseteq \{1\}^*$ is regular iff
$L = L_0 \cup L_{q1,r} \cup \ldots \cup L_{qk,r}$

Math is hard, let's go shopping!

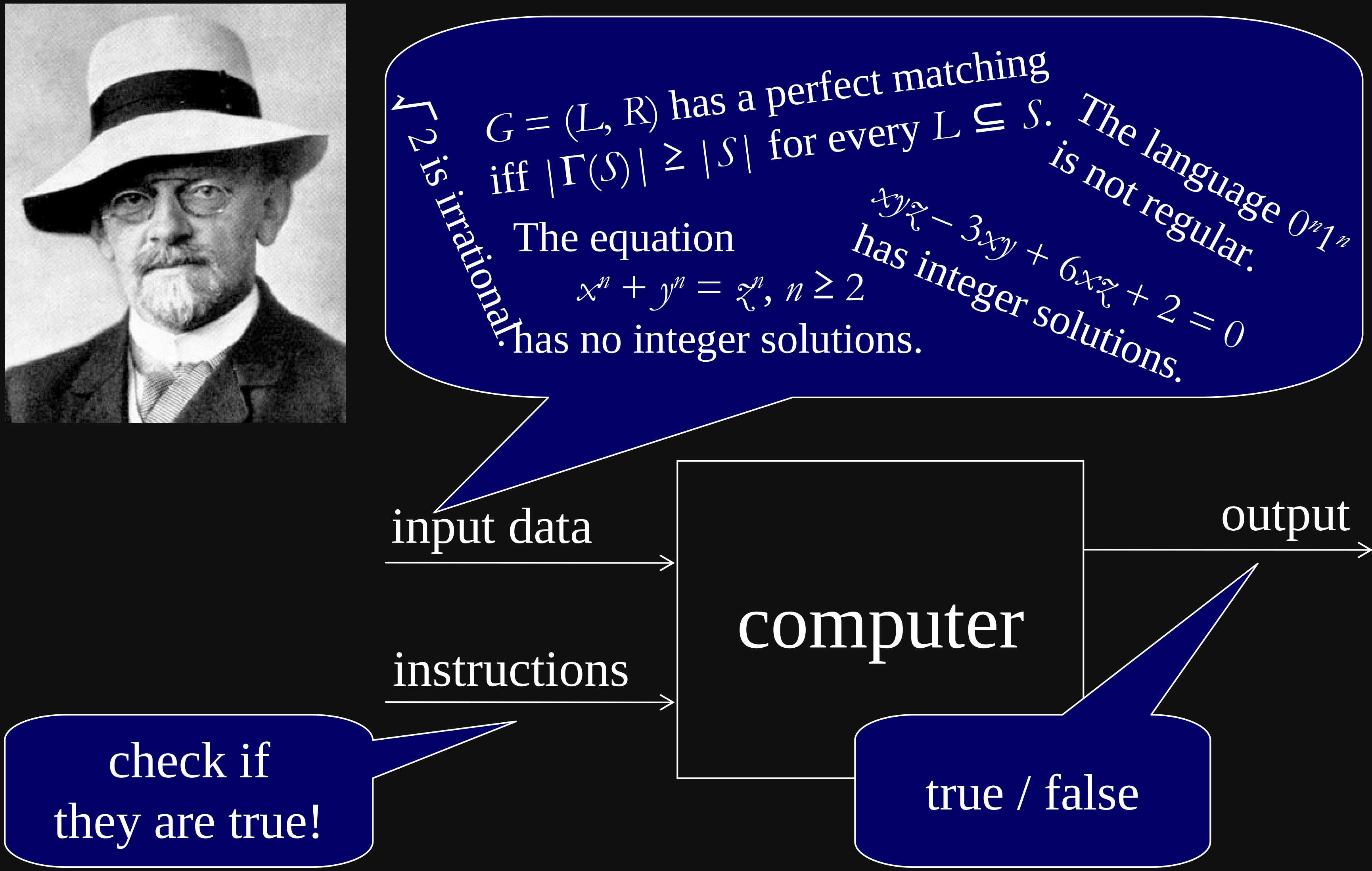
Hilbert's list of 23 problems

David Hilbert
(1862-1943)

- Leading mathematician in the 1900s
- At the 1900 International Congress of Mathematicians, proposed a list of 23 problems for the 20th century
- Hilbert's 10th problem:

Find a method for telling if an equation like $xyz - 3xy + 6xz + 2 = 0$ has integer solutions

Automated theorem-proving



Automated theorem proving

- How could automated theorem proving work?

- Let's look at a typical proof:

Theorem: $\sqrt{2}$ is irrational.

Proof: Assume not.
Then $\sqrt{2} = m/n$.
Then $m^2 = 2n^2$
But m^2 has an even number of prime factors
and $2n^2$ has an odd number of prime factors.
Contradiction.

First idea of automated theorem proving

① Checking that a proof is correct is much easier than coming up with the proof

Proof:

Assume not.
Then $\sqrt{2} = m/n$.
Then $m^2 = 2n^2$
But m^2 has an even ...
and $2n^2$ has an odd ...
Contradiction.

$\sqrt{2} = m/n$	both sides $\times\, n$
$(\sqrt{2})n = m$	square
$(\sqrt{2})^2 n^2 = m^2$	def of sqrt
$2n^2 = m^2$	

... if we write out the proof in sufficient detail

First idea of automated theorem proving

① Checking that a proof is correct is much easier than coming up with the proof

A proof, if written out in sufficient detail, can be checked mechanically

- In principle this is uncontroversial
- In practice, it is hard to do because writing proofs in detail sometimes requires a lot of work

Second idea of automated theorem proving

② To find if statement is true, why don't we try all possible proofs

After all, a proof is just a sequence of symbols (over alphabet $\{0, 1, 2, m, n, \sqrt{\ }, {}^2, \text{etc.}\}$)

If statement is true, it has a proof of finite length, say k

To find it, let's just try all possible proofs of length 1, 2, up to k

$$\sqrt{2} = m/n$$
$$(\sqrt{2})n = m$$
$$(\sqrt{2})^2 n^2 = m^2$$
$$2n^2 = m^2$$
$$\cdots$$

Königsberg, 1930

David Hilbert
(1862-1943)

Kurt Gödel
(1906-1978)

September 8: Hilbert gives a famous lecture
"Logic and the understanding of nature"

Königsberg, 1930

David Hilbert
(1862-1943)

Kurt Gödel
(1906-1978)

John von Neumann
(1903-1957)

It is all over!

I reached the conclusion that in any reasonable formal system in which provability in it can be expressed as a property of certain sentences, there must be propositions which are undecidable in it.

What did Gödel say?

② To find if statement is true, why don't we try all possible proofs

After all, a proof is just a sequence of symbols (over alphabet $\{0, 1, 2, m, n, \sqrt{\ }, {}^2$, etc.$\}$)

NO!

If statement is true, it has a proof of finite length, say k

To find it, let's just try all possible proofs of length 1, 2, up to k

Gödel’s incompleteness theorem

Some mathematical statements are true but not provable.

What are they?

Example of incompleteness

- Recall Hilbert's 10th problem:

  Find a method for telling if an equation like $xyz - 3xy + 6xz + 2 = 0$ has integer solutions

- Matyashievich showed in 1970 that there exists a polynomial p such that

  $p(x_1, ..., x_k) = 0$ has no integer solutions, but this cannot be proved

Another example of incompleteness

- Recall ambiguous context free grammars

- We did exercises of the type

"Show that G is ambiguous"

- However there exists a CFG G such that

G is unambiguous, but this cannot be proved

Gödel's incompleteness theorem

Some mathematical statements are true but not provable.

What does this have to do with computer science?

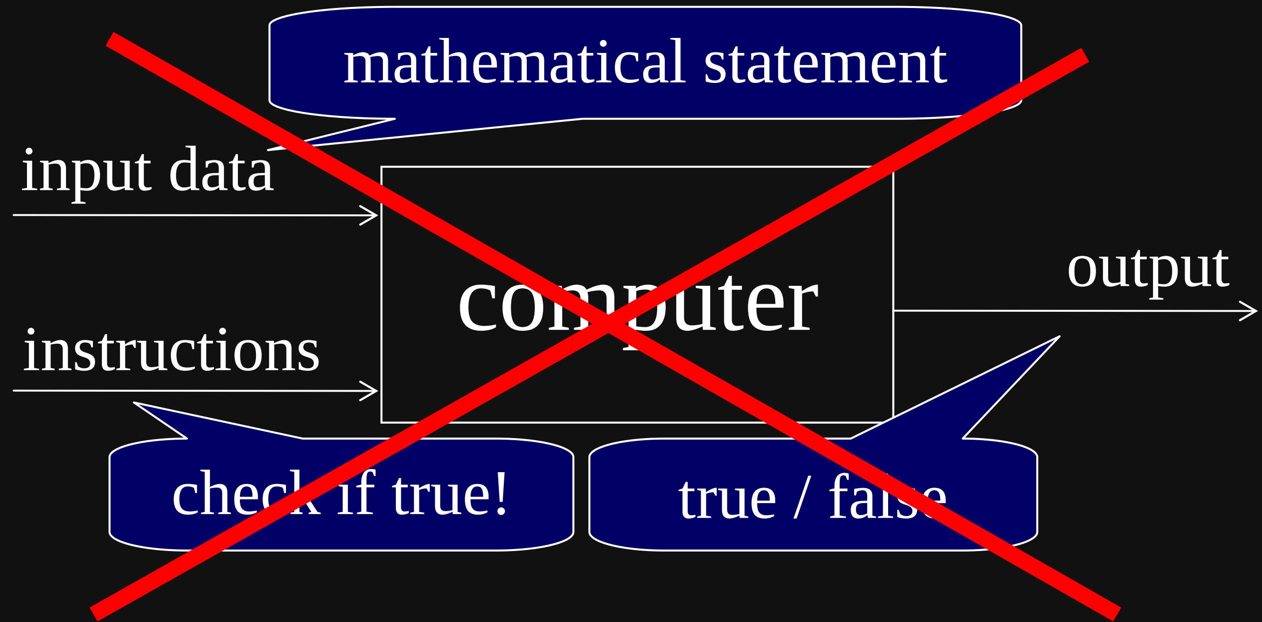
The father of modern computer science

Alan Turing
(1912-1954)

- Invented the Turing Test to tell humans and computers apart
- Broke German encryption codes during World War II
- The Turing Award is the "Nobel prize of Computer Science"

1936: "On Computable Numbers, with an Application to the *Entscheidungsproblem*"

Turing's angle on incompleteness



Gödel's theorem says that a computer cannot determine the truth of mathematical statements

But there are many other things!

Computer program analysis

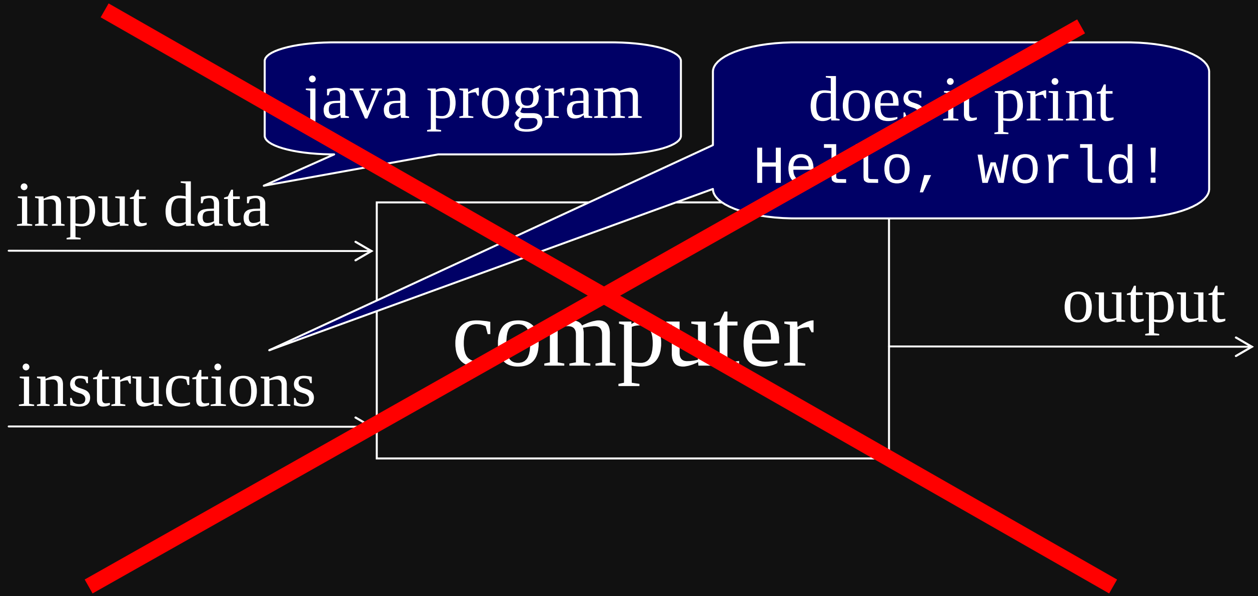
```
public static void main(String args[]) {
  System.out.println("Hello World!");
}
```

What does this program do?

```
public static void main(String args[]) {
  int i = 0;
  for (j = 1; j < 10; j++) {
    i += j;
    if (i == 28) {
      System.out.println("Hello World!");
    }
  }
}
```

How about this one?

Computers cannot analyze programs!



The essence of Gödel’s theorem is that computers cannot analyze computer programs

How do you argue things like that?

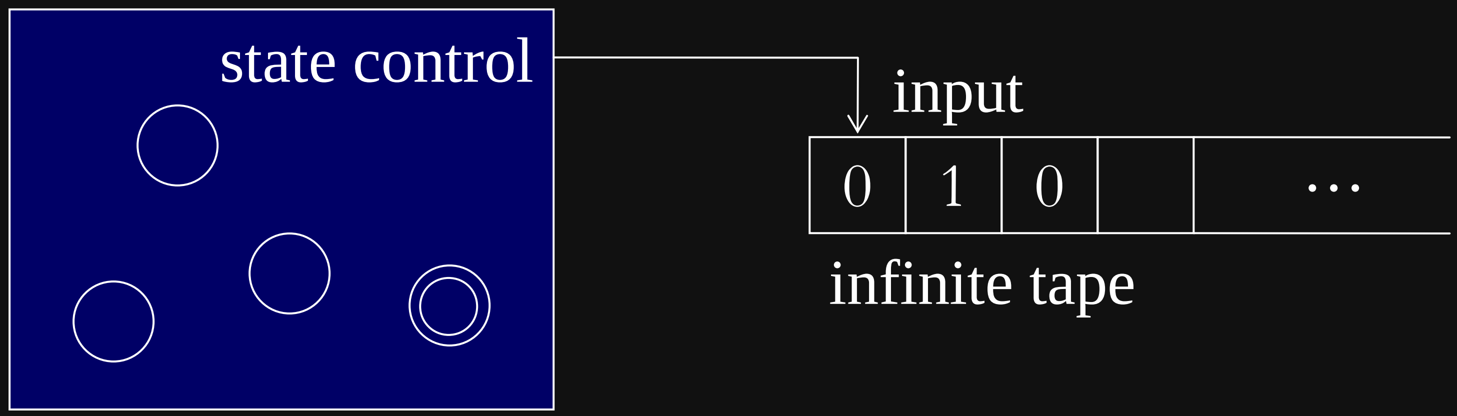
To argue what computers cannot do, we need to have a precise definition of what a computer is.

1936: "On Computable Numbers, with an Application to the *Entscheidungsproblem*"

Section 1. Computing Machines

Turing Machines

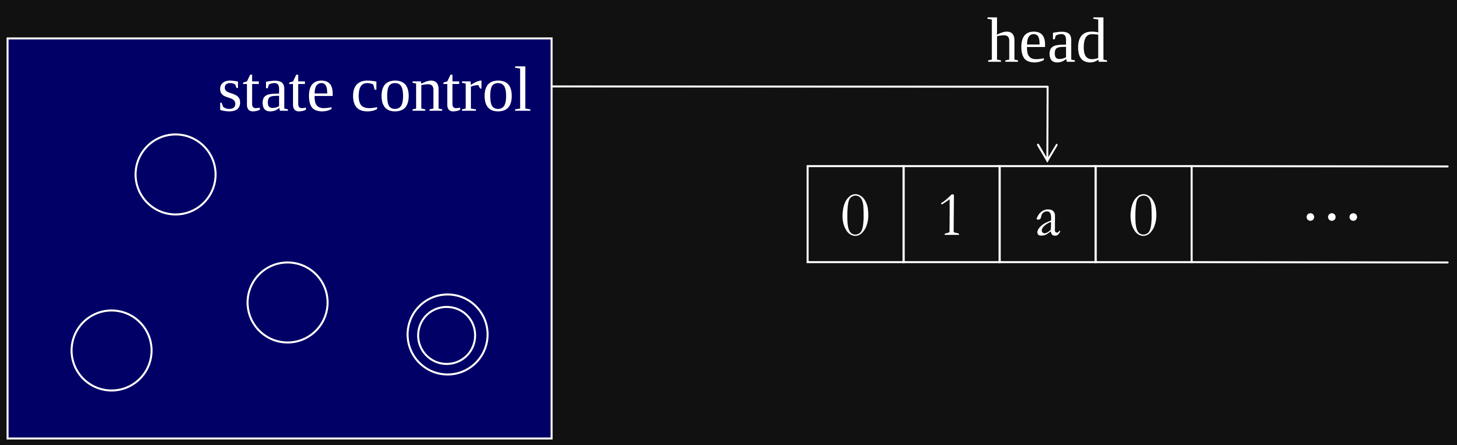
- A Turing Machine is a finite automaton with a two-way access to an infinite tape



- Initially, the first few cells contain the input, and the other cells are blank

Turing Machines

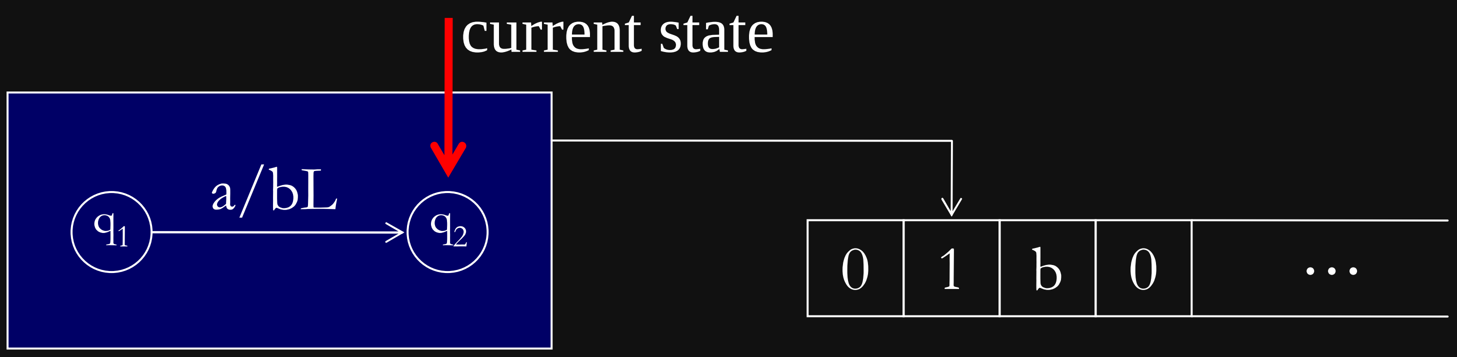
- At each point in the computation, the machine sees its current state and the symbol at the head



- It can replace the symbol on the tape, change state, and move the head left or right

Example

Replace a with b, and move head left



Formal Definition

A Turing Machine is $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$:

- Q is a finite set of states;
- Σ is the input alphabet;
- Γ is the tape alphabet ($\Sigma \subseteq \Gamma$) including the blank symbol $\square$
- q_0 in Q is the initial state;
- q_{acc}, q_{rej} in Q are the accepting and rejecting state;
- δ is the transition function

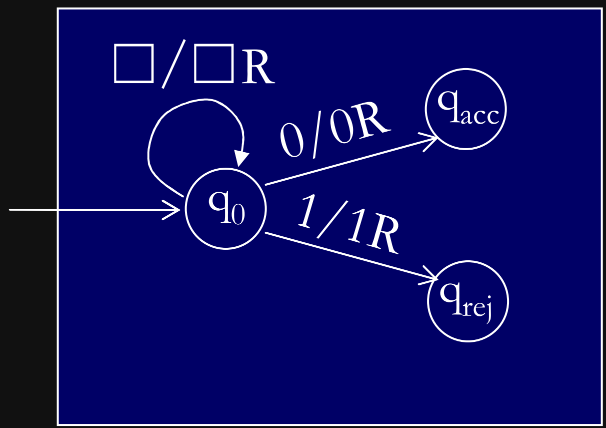
$$\delta: (Q - \{q_{acc}, q_{rej}\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

Turing Machines are deterministic

Language of a Turing Machine

- The language recognized by a TM is the set of all inputs that make it reach q_{acc}

- Something strange can happen in a Turing Machine:



$\Sigma = \{0, 1\}$

input: ε

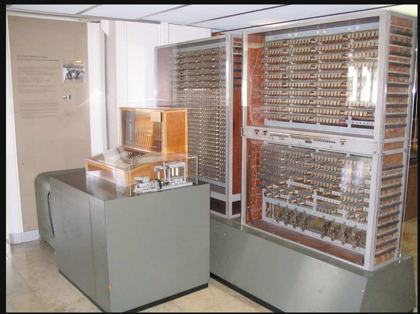
This machine never halts

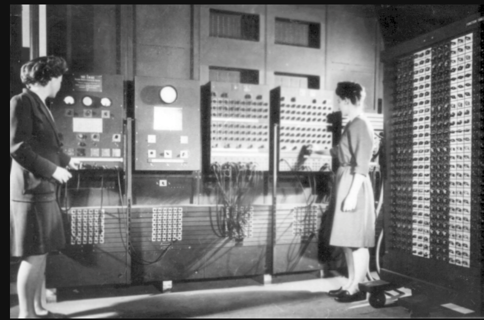
Looping behavior

- Inputs can be divided into three types:
  - 1. Reach the state q_{acc} and halt
  - 2. Reach the state q_{rej} and halt
  - 3. Loop forever

- The language recognized by a TM = type 1 inputs

The Church-Turing Thesis





Turing Machine

quantum computing

DNA computing

cosmic computing

The Church-Turing Thesis

All arguments [for the CT Thesis] which can be given are bound to be, fundamentally, appeals to intuition, and for this reason rather unsatisfactory mathematically.

The arguments which I shall use are of three kinds:

1. A direct appeal to intuition
2. A proof of the equivalence of two definitions
   (In case the new definition has greater intuitive appeal)
3. Giving examples of large classes of numbers which are computable.

1936: "On Computable Numbers, with an Application to the *Entscheidungsproblem*"

Section 9. The extent of the computable numbers