Theory of Computation

(Introduction)

Pramod Ganapathi
Department of Computer Science
State University of New York at Stony Brook

January 24, 2021

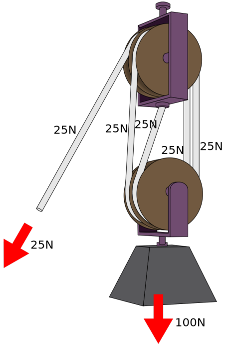
Hand-axe

Longest-used tool in human history (1.5 million years)

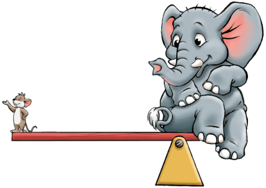
Wheel

Idea behind transportation revolution
(Other uses: potter's wheel, steering wheel, flywheel)

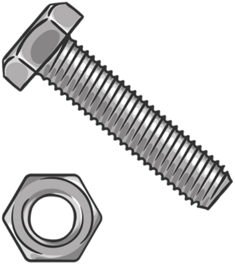
Simple machines



Pulley



Lever



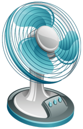
Screw

Machines










Computing

Counting cattle

1

2

3

4

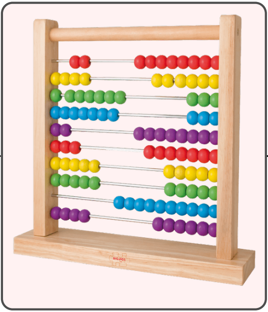
5

Machine for computing

- How do humans compute or calculate or solve problems?
- Is it possible to build a computing machine that can mechanically (i.e., without thinking) simulate the computations performed by a human brain like that of Galileo or Newton or Einstein?
- If so, what problems can or cannot be solved by such a computing machine?

< 2000 BC: Abacus

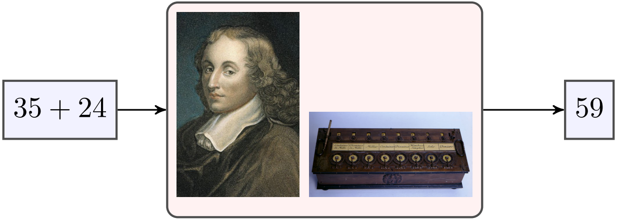
$35 - 24$ →



→ 14

- Not automatic
- Operations: Addition, subtraction, multiplication, and division

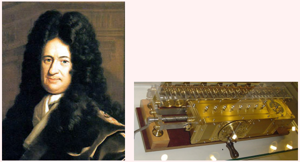
1643: Pascal's calculator (Pascaline)



Source: Computer Museum History Center

- Inventor: Blaise Pascal
- Operations: Addition and subtraction
- World's first mechanical calculator

1694: Leibniz' calculator (Step reckoner)

35×24 →  → 840

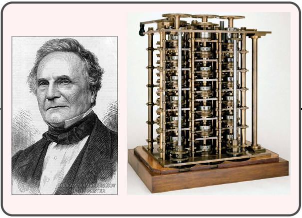
- Inventor: Gottfried Wilhelm Leibniz
- Operations: Addition, subtraction, multiplication, and division

1820: Colmar's calculator (Arithmometer)

$\sqrt{35}$ →   → 5.9161

- Inventor: Thomas de Colmar
- Operations: Addition, subtraction, multiplication, division, square root, involution, resolution of triangles, etc
- Applications: Financial organizations

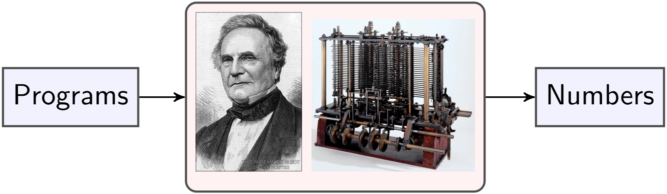
1822: Babbage's calculator (Difference engine)

$\log 35$ →  → 1.5441

Source: Science Museum London

- Designer: Charles Babbage
- The system was never built due to conflicts and insufficient funding
- Operations: Addition, subtraction, multiplication, division, logarithmic, trigonometric functions, etc

1833: Babbage's computer (Analytical engine)



Source: Science Museum London

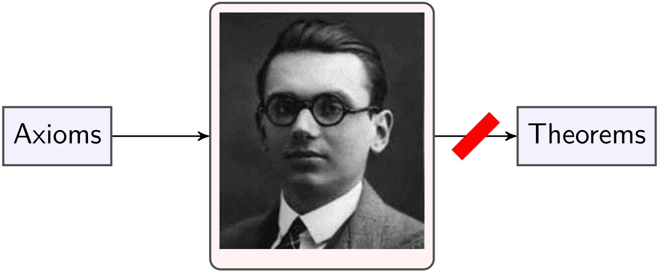
- Designer: Charles Babbage
- The system was never built due to conflicts and insufficient funding
- World's first general-purpose computer (Turing-complete)
- Components: arithmetic logic unit, control flow in the form of conditional branching and loops, and integrated memory

1843: Lovelace's algorithm

Pic by: Antoine Claudet

- Designer: Ada Lovelace
- World's first programmer
- Published the first algorithm to be implemented on a computer
- The algorithm was used to compute Bernoulli numbers

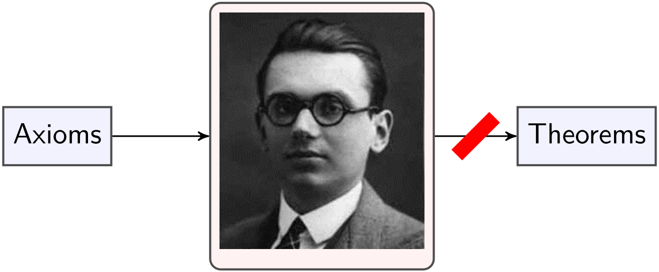
1931: Gödel's proof



Source: geni.com

- Discoverer: Kurt Gödel
- Some mathematical truths cannot be proved

1931: Gödel's proof



Source: geni.com

- Discoverer: Kurt Gödel
- Some mathematical truths cannot be proved
  (If you cannot prove a mathematical statement, then how do you know that the statement is true?)

1936: Turing machine

- Discoverer: Alan Mathison Turing
- Creator of computer science
- Turing machine – the simplest, the most intuitive, the most generic, and the most powerful mathematical model of a computing human brain and a computer
- Algorithm and computation

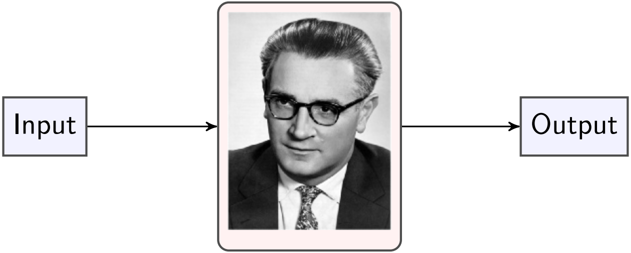
1936: Turing's proof

- Discoverer: Alan Mathison Turing
- Some computational problems cannot have algorithms

1936: Turing's proof

- Discoverer: Alan Mathison Turing
- Some computational problems cannot have algorithms
  (If you cannot mechanically compute a computational problem, then why is it called a computational problem?)

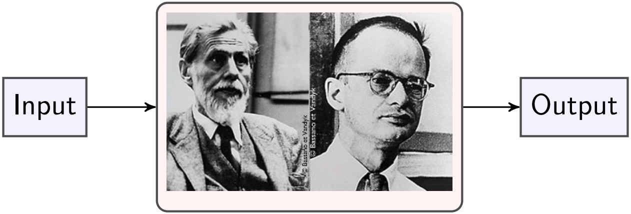
1941: Zuse's Z3



Source: http://www.horst-zuse.homepage.t-online.de/

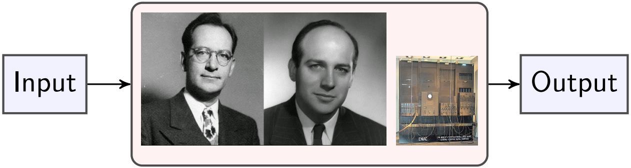
- Designer: Konrad Zuse
- World's first working programmable, fully automatic digital computer (Turing-complete)

1943: McCulloch and Pitts' finite automata



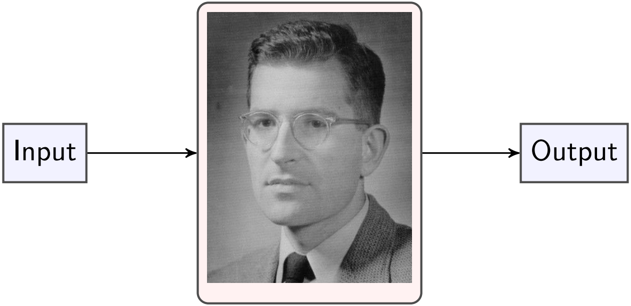
- Designers: Warren McCulloch and Walter Pitts
- Finite automata – simple model of computation

1945: Mauchly and Eckert's ENIAC



- Designers: John Mauchly, J. Presper Eckert
- World's first electronic general-purpose computer (Turing-complete)

1957: Chomsky's grammars



- Designer: Noam Chomsky
- Context-free grammar and context-sensitive grammar – models of computation

1985: Deutsch's quantum machine

Source: twitter

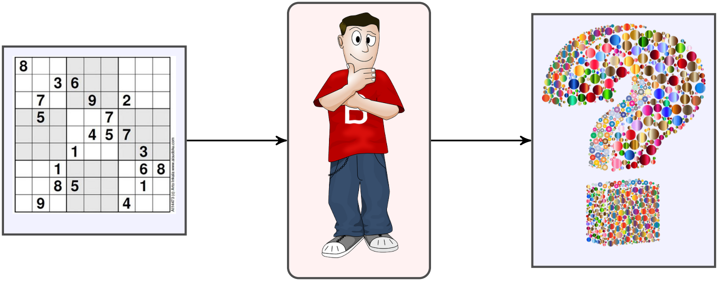
- Discoverer: David Deutsch
- Quantum model of computation
- Model based on quantum physics and not classical physics
- Exponentially faster than classical computing for some problems

1989: Lee's world wide web

Source: CERN

- Designer: Tim Berners Lee
- World wide web – led to Internet revolution

What is a computer/computation/algorithm?



What is a computer/computation/algorithm?

What is an alphabet?

Definition

- An alphabet, denoted by Σ, is a finite, non-empty set of symbols.

Examples

- $\Sigma = \{a, b\}$
- Unary alphabet $\Sigma = \{1\}$
- Binary alphabet $\Sigma = \{0, 1\}$
- English alphabet $\Sigma = \{a, \ldots, z, A, \ldots, Z\}$
- Alphanumeric alphabet $\Sigma = \{a\text{-}z, A\text{-}Z, 0\text{-}9\}$
- Morse code alphabet $\Sigma = \{\text{dot}, \text{dash}, \text{pause}\}$
- DNA alphabet $\Sigma = \{A, C, G, T\}$
- Java programming language alphabet
  $\Sigma = \{a\text{-}z, A\text{-}Z, 0\text{-}9, (,), \{, \}, \ldots, ;\}$
- $\{1, 2, 3, \ldots\}$ is not an alphabet as the set is not finite

Powers of an alphabet

Definition

- Σ = Some alphabet
- Σ^k = Set of all strings of length k over Σ
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots =$ Set of all strings over Σ
  Σ^* is the universal set of all strings
- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots =$ Set of nonempty strings over Σ

Examples

- Let $\Sigma = \{a, b\}$
- $\Sigma^0 = \{\epsilon\}$
  $\Sigma^1 = \{a, b\}$
  $\Sigma^2 = \{aa, ab, ba, bb\}$
- $\Sigma^* = \{\epsilon, a, b, aa, ab, ba, bb, \ldots\}$
  This ordering is called canonical ordering, which is different from lexicographic ordering
- $\Sigma^+ = \{a, b, aa, ab, ba, bb, \ldots\}$

What is a string?

Definition

- A string or word is a finite sequence of symbols chosen from Σ. A string $x \in \Sigma^*$. An empty string is denoted by ϵ.
- $|x|$ = length of string x
- $n_\sigma(x)$ = #occurrences of symbol $\sigma \in \Sigma$ in the string x

Examples

- $x = abaaabb$ from $\Sigma = \{a, b\}$
- $x = 111$ from $\Sigma = \{0, 1\}$
- $x = \epsilon$ from $\Sigma = \{a, \ldots, z, A, \ldots, Z\}$
- $x = Bond007$ from $\Sigma = \{a - z, A - Z, 0 - 9\}$
- $x = CGGTCCGC$ from $\Sigma = \{A, C, G, T\}$
- x = a simple hello world C program from $\Sigma = \{\text{if}, \text{main}, \text{return}, \text{for}, (,), \{, \}, \ldots, ;\}$

What is a language?

Definition

- A language over Σ is a subset of Σ^*.

Examples

- The empty language ϕ.
- $\{\epsilon, a, aab\}$ - a finite language.
- Language of palindromes over $\{a, b\}$
- $\{x \in \{a, b\}^* \mid n_a(x) > n_b(x)\}$.
- $\{x \in \{a, b\}^* \mid |x| \geq 2$ and x begins and ends with $b\}$

What is a language?

Examples (continued)

- Language of your favorite quotations
- Language of legal Java identifiers
- Language of legal algebraic expressions involving the identifier a, the binary operations $+$ and $*$, and parentheses (strings: $a, a + a * a$, and $(a + a * (a + a)))$)
- Language of balanced strings of parentheses. (strings: $\epsilon, ()(())$, and $((((()))))$)
- Language of numeric "literals" in Java (e.g: $-41, 0.03, 5.0E-3$).
- Language of legal Java programs.
- Language of theorems (true statements) in arithmetic
- Language of theorems (true statements) in geometry

How can we represent information?

Representation

- Strings can be used to represent all types of information
- Strings can encode information about names, numbers, dates, text documents, images, videos, and literally any type of data
- Binary strings are the simplest type of strings that can encode any information
- Binary strings can also be viewed as numbers
- Hence, numbers can also be used to represent all types of information

Three major concepts in Theory of Computation

Concept	Meaning
Model of computation	An abstract but physically realistic machine that does computation
Language	Set of all strings that the computational model accepts
Grammar	Set of rules to derive any string from the language

Core idea of Theory of Computation

Computation model	Language	Grammar
Finite automaton	Regular language	Regular grammar
Pushdown automaton	Context-free language	Context-free grammar
Linear-bounded automaton	Context-sensitive language	Context-sensitive grammar
Turing machine	Recursively enumerable language	Unrestricted grammar
No computer or no algorithm	Undecidable language	?

- We will spend an entire semester for this course trying to understand this table.

Three major topics of Theory of Computation

Covered topic	Questions
Automata theory	What can be computed with extremely limited space?
Computability theory	What can be computed? Can a computer solve all computational problems, given enough (finite) time and space?
Complexity theory	How fast can we solve a problem? How small space can we use to solve a problem?
Not covered topic	**Questions**
Algorithms	How can a given computational problem be solved efficiently (less time and space)?

What can be computed?

Problem	DFA	PDA	TM
Draw money from ATM	✓	✓	✓
Check if a string is present in another string	✓	✓	✓
Linux regular expressions	✓	✓	✓
Parse if-else blocks and for loops in C/C++/Java programs	✗	✓	✓
Parse nested arithmetic expressions	✗	✓	✓
Parse markup languages such as HTML	✗	✓	✓
Multiply two integers	✗	✗	✓
Factorize an integer into two integers	✗	✗	✓
Find a shortest path between two cities	✗	✗	✓
Check if a computer program halts or terminates	✗	✗	✗
Check if a computer program crashes	✗	✗	✗
Check if a computer program is correct	✗	✗	✗

- DFA: Deterministic Finite Automaton
- PDA: Pushdown Automaton
- TM: Turing Machine

Applications of Theory of Computation

Topic	Applications
Finite automaton	Regular expressions Traffic signals, Vending machines, ATMs String matching Lexical analysis in a compiler Combination/sequential digital logic circuits Spell checkers
Pushdown automaton	Stack applications Balanced parentheses Syntax analysis in a compiler Evaluating arithmetic expressions
Linear-bounded automaton	Variable declaration and definition in a compiler Genetic programming
Turing machine	Understanding computation Mother of classical computers and algorithms Grandmother of quantum computers
Complexity theory	Cryptography

Turing-complete systems

Time	Turing-complete system	Designer
1830s	Analytical engine	Charles Babbage
1930s	Recursive functions	Stephen Kleene
	λ-calculus	Alonzo Church
	Turing machine	Alan Turing
–	Unrestricted grammar	–
1940s	Z3	Konrad Zuse
	Tag systems	Emil Leon Post
1960s	Markov's algorithms	Andrey Markov, Jr.
	Unlimited register machines	John Shepherdson, Howard Sturgis
1970s	C	Dennis Ritchie
	Game of life	John Conway
1980s	Rule 110	Stephen Wolfram
	Quantum computers	David Deutsch

What can be computed?

Problems

- [Halting program]
  Write a computer program that takes a computer program P as input and outputs whether P halts (i.e., terminates) or not.
- [Correctness program]
  Write a computer program that takes a computer program P and a specification s for P as input and outputs whether P is correct or not (i.e., if P follows the input-output specification s or not).
- [Equivalence program]
  Write a computer program that takes two computer programs P_1 and P_2 as input and outputs whether P_1 is functionally equivalent to P_2 or not.
- [Self-replicating program]
  Write a computer program that does not take any input and outputs its own source code.

What can be computed?

Problems

- [Halting program] ▷ Impossible
  Write a computer program that takes a computer program P as input and outputs whether P halts (i.e., terminates) or not.
- [Correctness program] ▷ Impossible
  Write a computer program that takes a computer program P and a specification s for P as input and outputs whether P is correct or not (i.e., if P follows the input-output specification s or not).
- [Equivalence program] ▷ Impossible
  Write a computer program that takes two computer programs P_1 and P_2 as input and outputs whether P_1 is functionally equivalent to P_2 or not.
- [Self-replicating program] ▷ Possible
  Write a computer program that does not take any input and outputs its own source code.