From Turing to Type Theory

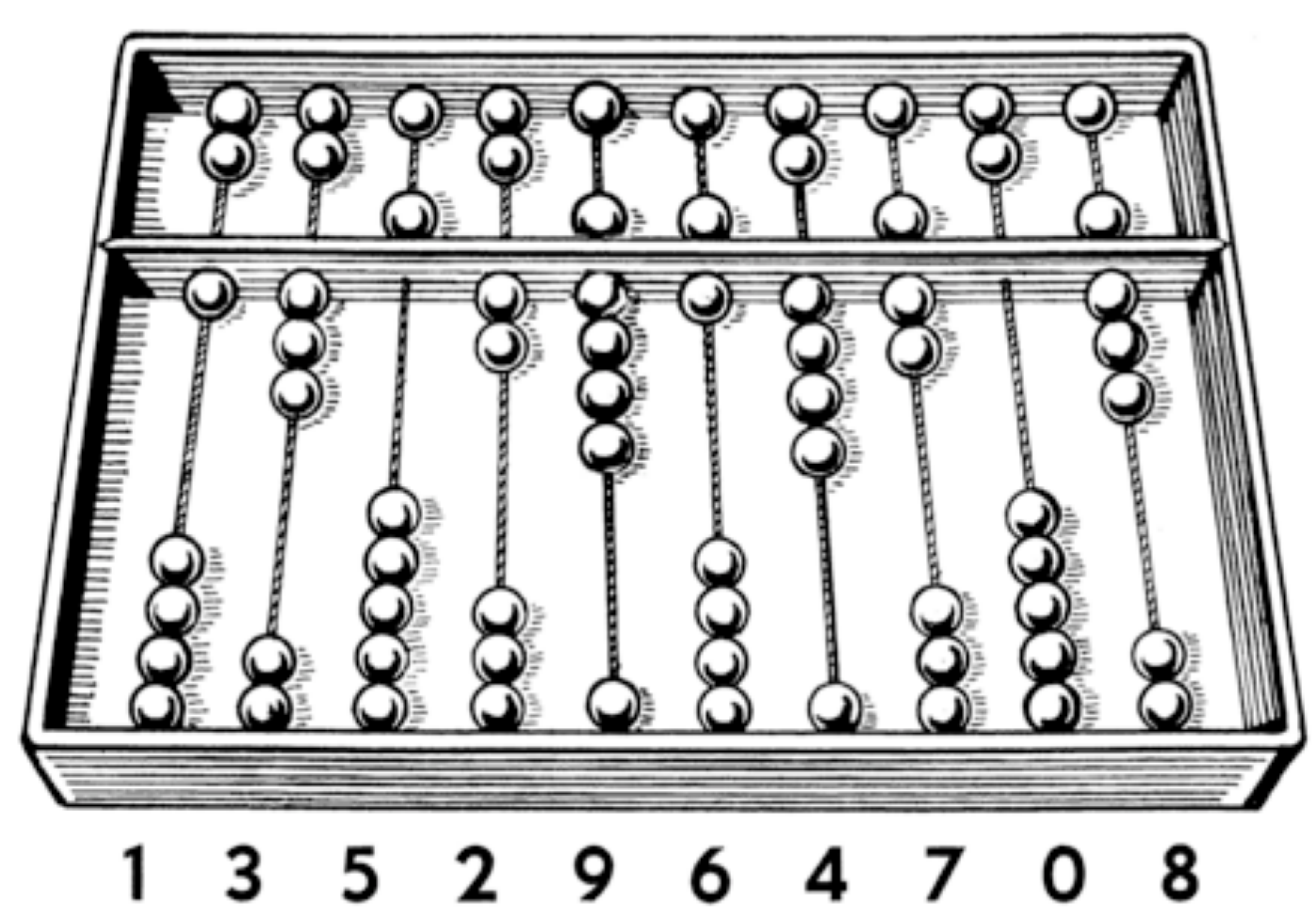
The Rich Historical Context of Computation

Pedro Abreu

Brief History of (Pre-)Computer

- Abaacus - 2700 - 2300 A.C. - Sumerians
- Antikythera Mechanism - 2nd Century BC II
  - Analog Computer for eclipses and planetary positions
- Analog Computers at the Islamic Medieval World (10th Century)
  - Automatic Flute Player
  - Mechanism Astrolable
  - Torquetum (astronomical observations)
- Astronomical Watches - 14th Century

Abacus

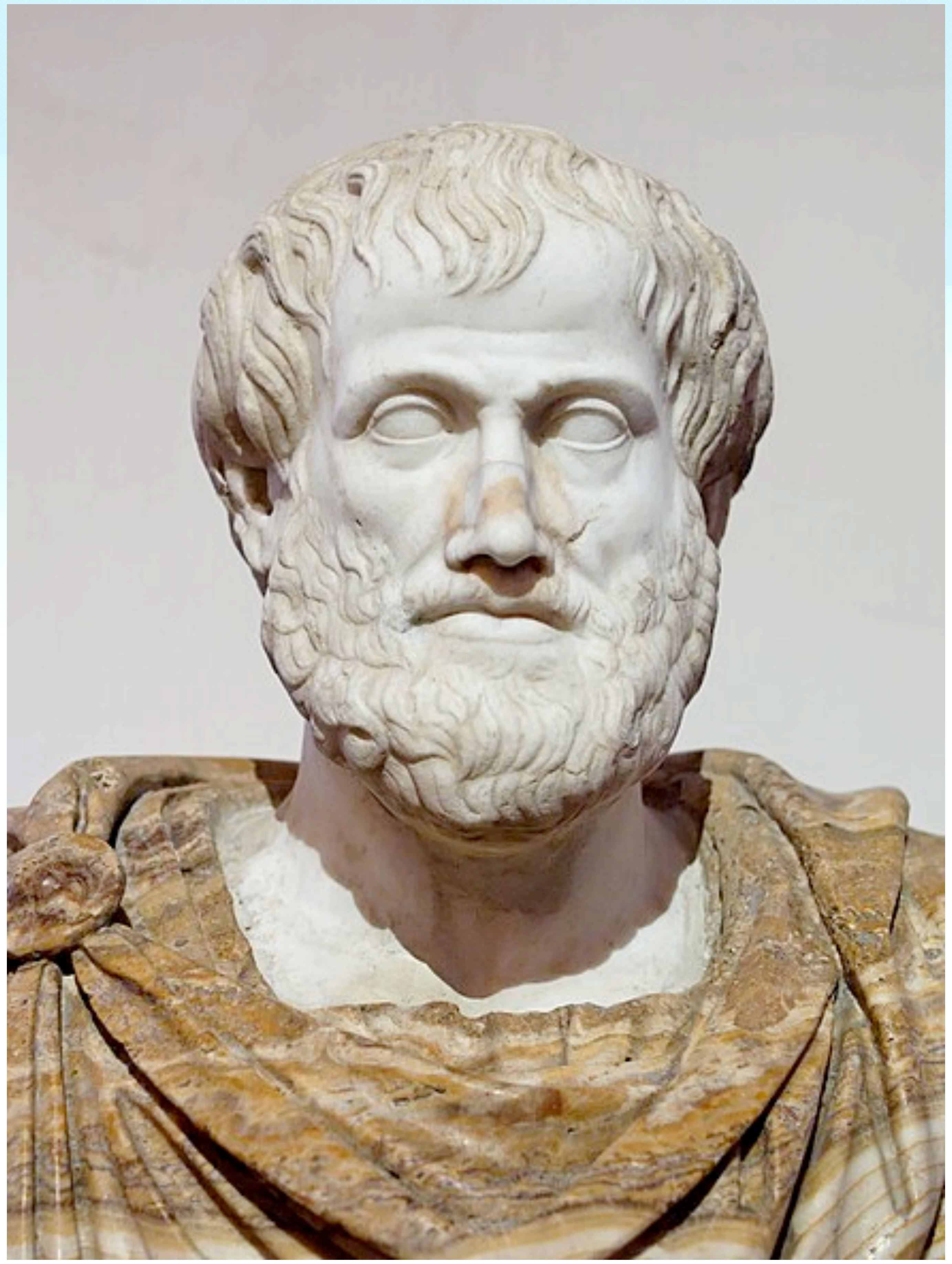


Antikythera Mechanism

Astrolable

Aristotle

384 BC - 322 BC (Ancient Greece)



Aristotle

384 BC - 322 BC

- Logical Argument
- Deduction vs Induction
- Universal vs Particular Reasonings
- Counter-Examples
- Etc.

Gottfried Leibniz

1646 - 1716 (Germany)

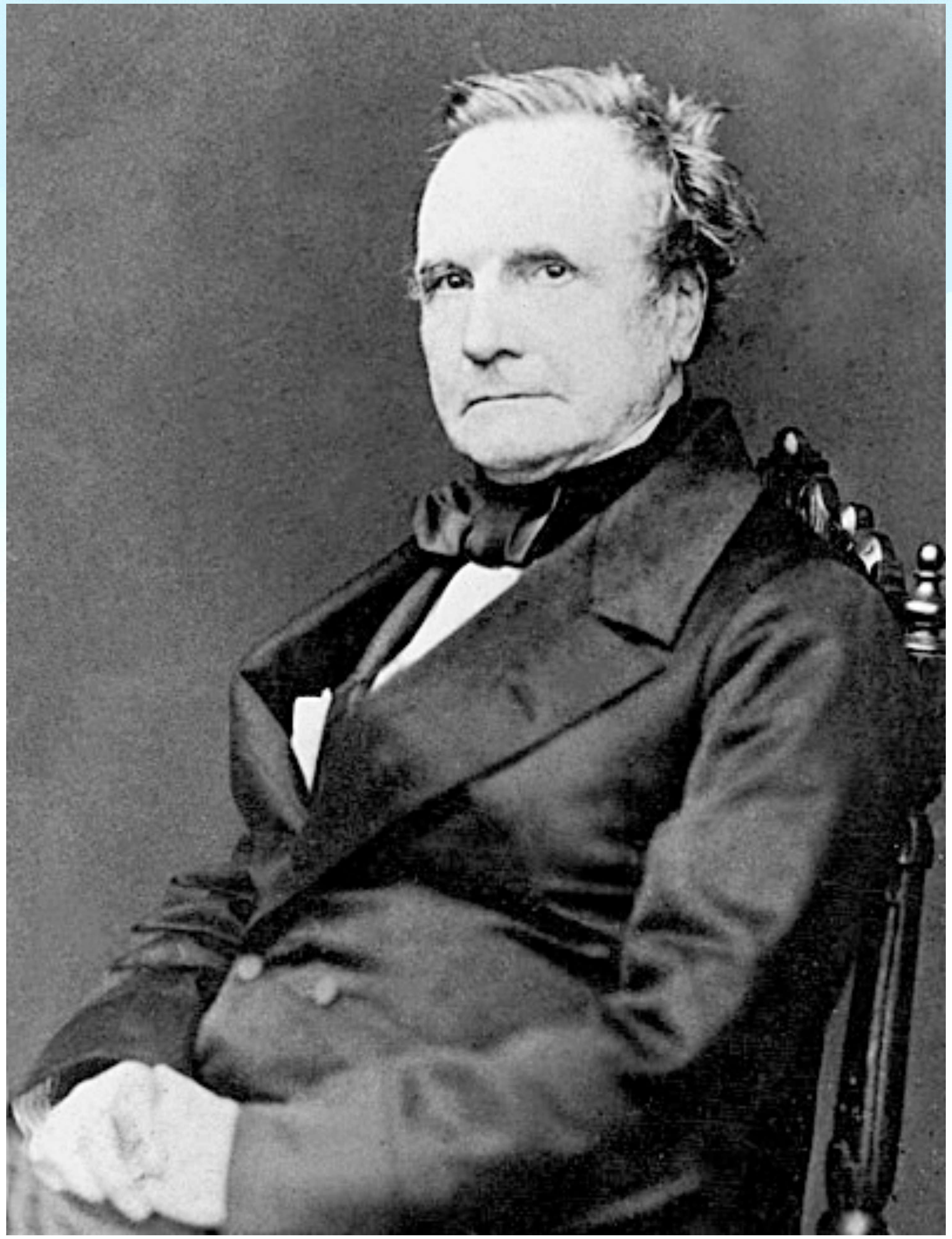
Gottfried Leibniz

1646 - 1716 (Germany)

- Leibniz Stepped Reckoner - 1694 - calculator (not very trustworthy)
  - 1702 - Begins the development of formal logic
  - Simplified the binary system and articulated its formal properties
    - conjunction
    - disjunction
    - negation
    - identity, inclusion, empty
  - "Calculus Ratiocinator" anticipated the Turing Universal Machine
    - Only 154 years later it would be possible to model computational systems with the publications of George Boole (1854)

Charles Babage

1791 - 1871 (England)



Charles Babage

1791 - 1871 (Inglaterra)

- 1837 Babage described “The Analytical Machine” but was never able to finish building it
- Modern design far ahead of its time
  - Memory
  - Arithmetic Unit
  - Logic with loops and conditionals
  - Programmed with Punch Cards
- Ada Lovelace wrote the first program in history for this machine

David Hilbert

1862 - 1943 (Germany)

Hilbert Problems

Proposed in 1900

- 23 most important open problems in math
- Second problem: To show that math is consistent
- Define a set of axioms able to define math precisely
- Show that these axioms aren’t self-contradictory

Kurt Gödel

1906 - 1978 (Austria-Hungary, now Czech Republic)

Gödel is friends with Einstein

Gödel's Incompleteness Theorems

1931 - Princeton

- First Incompleteness Theorem
- There is no consistent system of axioms that can be generated by an algorithm capable of proving all truths about arithmetics of natural numbers
- There will always be propositions about natural numbers that are true but impossible to be proved within such a system
- For Example: This proposition can't be proved

Gödel’s Second Incompleteness Theorem

- Such a system is not capable of proving it’s own correctness

Alan Turing

1912 - 1954 (England)

Alan Turing

1936 - Cambridge

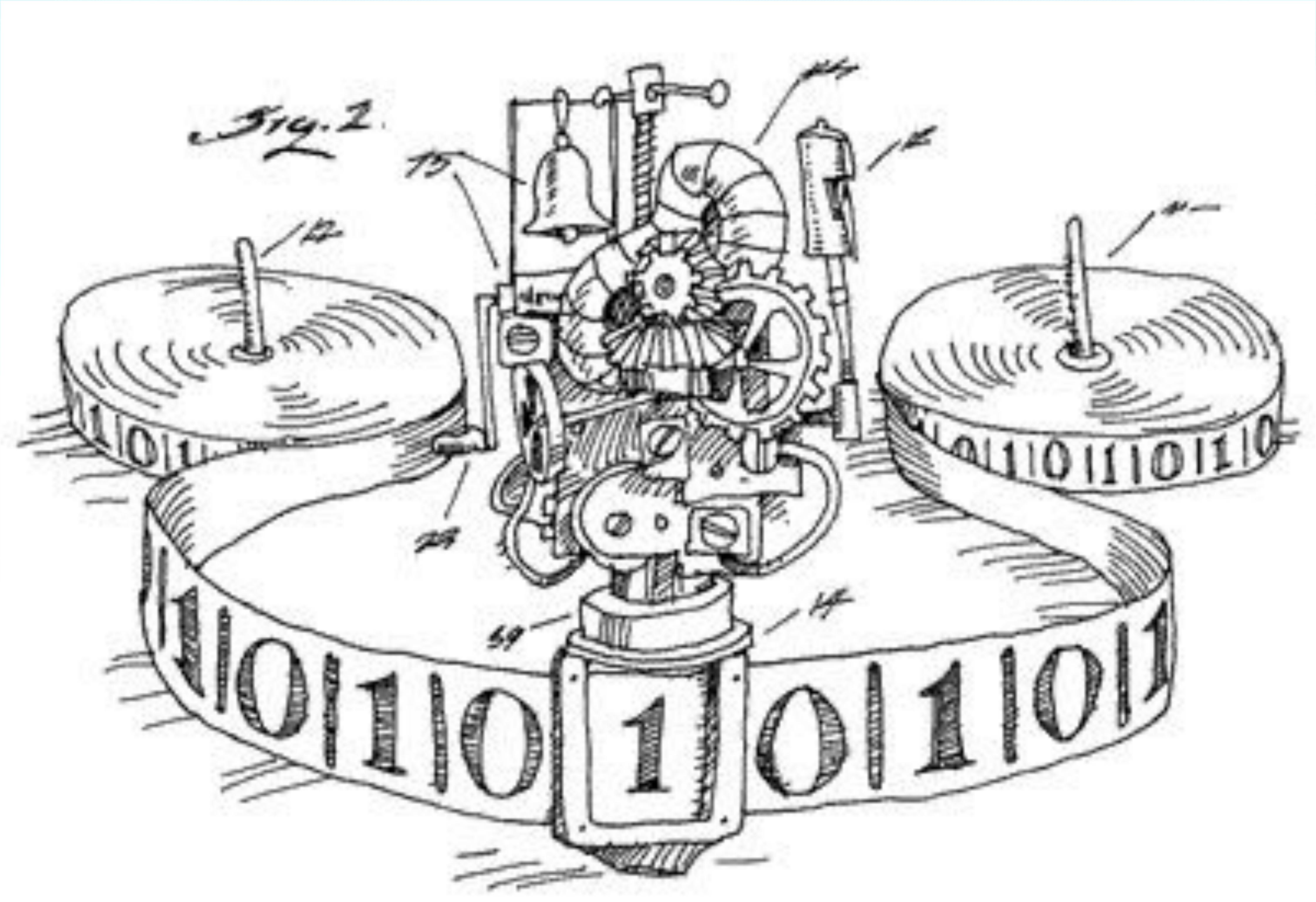
- *On the Computable Numbers with an Application to the Entscheidungsproblem*

Entscheidungsproblem

Decidability Problem

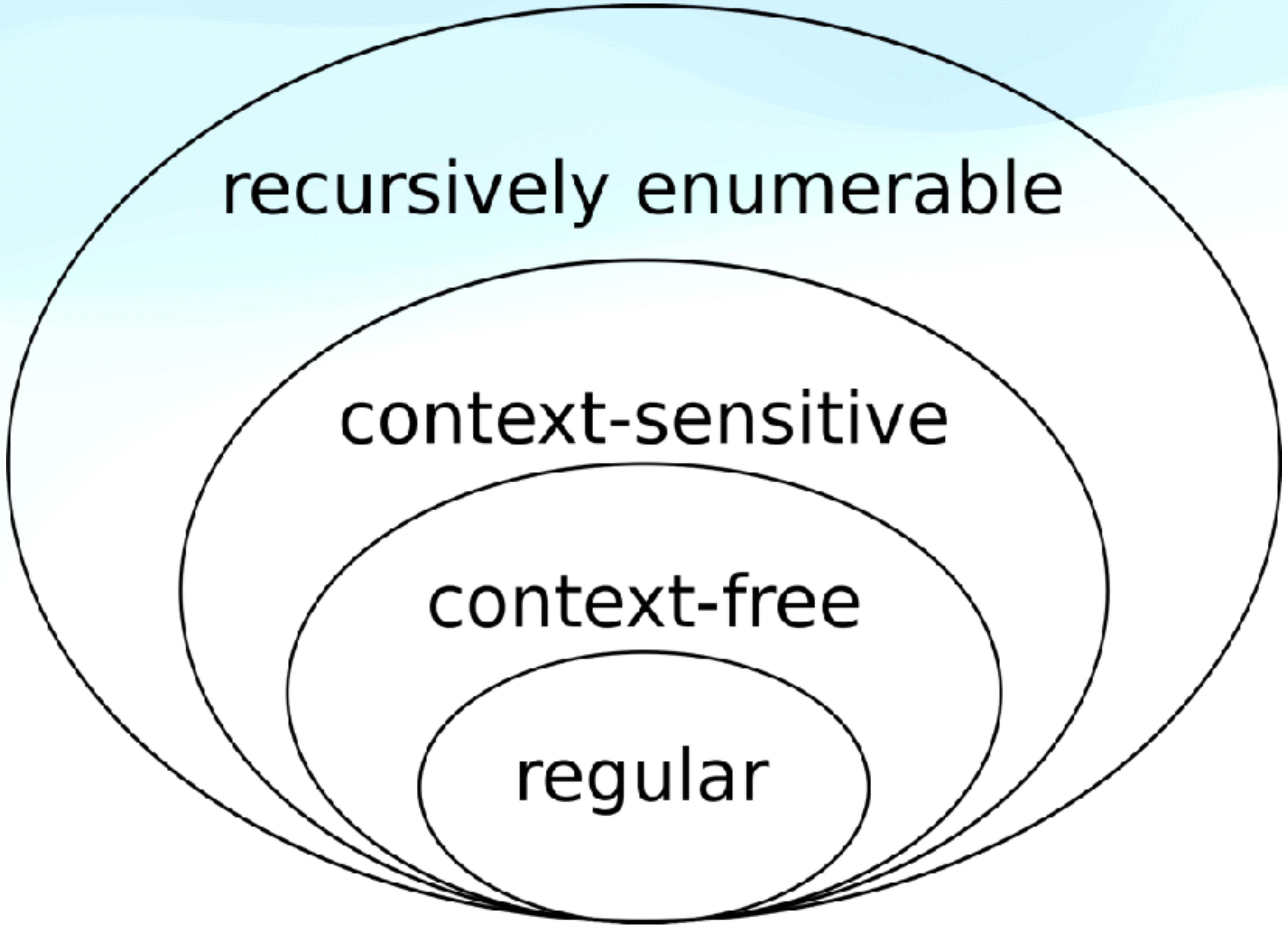
- Decide if the answer to a given problem is ‘yes’ or ‘no’
- Example: An Algorithm decides if a given number is prime or not

Universal Machine (Turing Machine)



Chomsky Hierarchy

1956



Alonzo Church

1903 - 1995 (USA)

Alonzo Church

1936 - Princeton

- *An unsolvable problem of elementary number theory*
- Untyped Lambda Calculus

Introduction to the Lambda Calculus

$$
\begin{array}{llll}
MN & ::= & y & \text{(variable)} \\
 & ::= & MN & \text{(application)} \\
 & ::= & \lambda x.M & \text{(abstraction)}
\end{array}
$$

Introduction to the Lambda Calculus

Lambda expression
$0 = \lambda f.\, \lambda x.\, x$
$1 = \lambda f.\, \lambda x.\, f\ x$
$2 = \lambda f.\, \lambda x.\, f\ (f\ x)$
$3 = \lambda f.\, \lambda x.\, f\ (f\ (f\ x))$
$\vdots$
$n = \lambda f.\, \lambda x.\, f^{\circ n}\ x$

Introduction to the Lambda Calculus

Function definition	Lambda expressions	
$\operatorname{succ}\ n\ f\ x = f\ (n\ f\ x)$	$\lambda n.\lambda f.\lambda x.f\ (n\ f\ x)$	...
$\operatorname{plus}\ m\ n\ f\ x = m\ f\ (n\ f\ x)$	$\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$	$\lambda m.\lambda n.n\operatorname{succ}m$
$\operatorname{multiply}\ m\ n\ f\ x = m\ (n\ f)\ x$	$\lambda m.\lambda n.\lambda f.\lambda x.m\ (n\ f)\ x$	$\lambda m.\lambda n.\lambda f.m\ (n\ f)$
$\exp\ m\ n\ f\ x = (n\ m)\ f\ x$	$\lambda m.\lambda n.\lambda f.\lambda x.(n\ m)\ f\ x$	$\lambda m.\lambda n.n\ m$
$\operatorname{if}(n == 0)\ 0\ \operatorname{else}\ (n-1)$	$\lambda n.\lambda f.\lambda x.n\ (\lambda g.\lambda h.h\ (g\ f))\ (\lambda u.x)\ (\lambda u.u)$	
$\operatorname{minus}\ m\ n = (n\operatorname{pred})\ m$	...	$\lambda m.\lambda n.n\operatorname{pred}m$

Church-Turing Thesis

- Turing goes to Princeton between the years of 1936 and 1938 to be supervised by Church. Together they prove several theorems about the theory of computation
- In particular, they prove that the Lambda Calculus and the Turing Machine has exactly the same computational power
    - They also postulate that any other computational mechanism is equivalent to the Turing Machine.

Curry-Howard Isomorphism
1980

- There is a direct relationship between formal logic and programs

Curry-Howard Isomorphism

1980

Logic	**Programming Languages**
Propositions	Types
$P \supset Q$	$P \to Q$
$P \wedge Q$	$P \times Q$
Proof of Proposition P	Term t of type P
It is possible to prove Proposition P	Type P is inhabited

Curry-Howard Isomorphism

1980

- Mathematical Proofs and Programs are equivalent

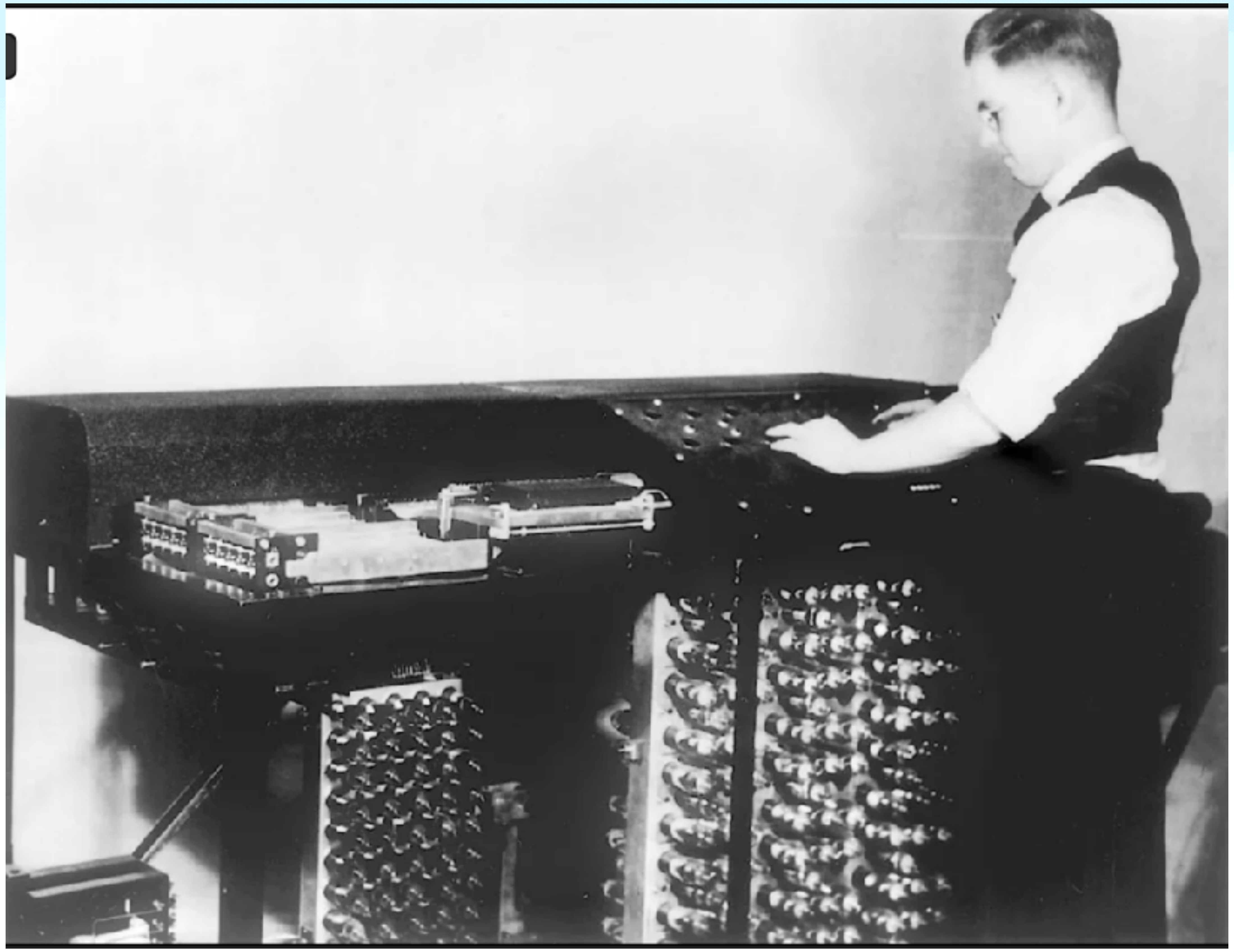
```
plus_comm =
fun n m : nat =>
nat_ind (fun n0 : nat => n0 + m = m + n0)
  (plus_n_0 m)
  (fun (y : nat) (H : y + m = m + y) =>
   eq_ind (S (m + y))
     (fun n0 : nat => S (y + m) = n0)
     (f_equal S H)
     (m + S y)
     (plus_n_Sm m y)) n
     : forall n m : nat, n + m = m + n
```

Theorem Provers

- The strongest guarantee of software correctness
- Interactive Theorem Provers
  - Coq, Lean, Agda, Isabelle, etc.
- Automated Theorem Provers
  - Z3, CVC6, Dafny, F*

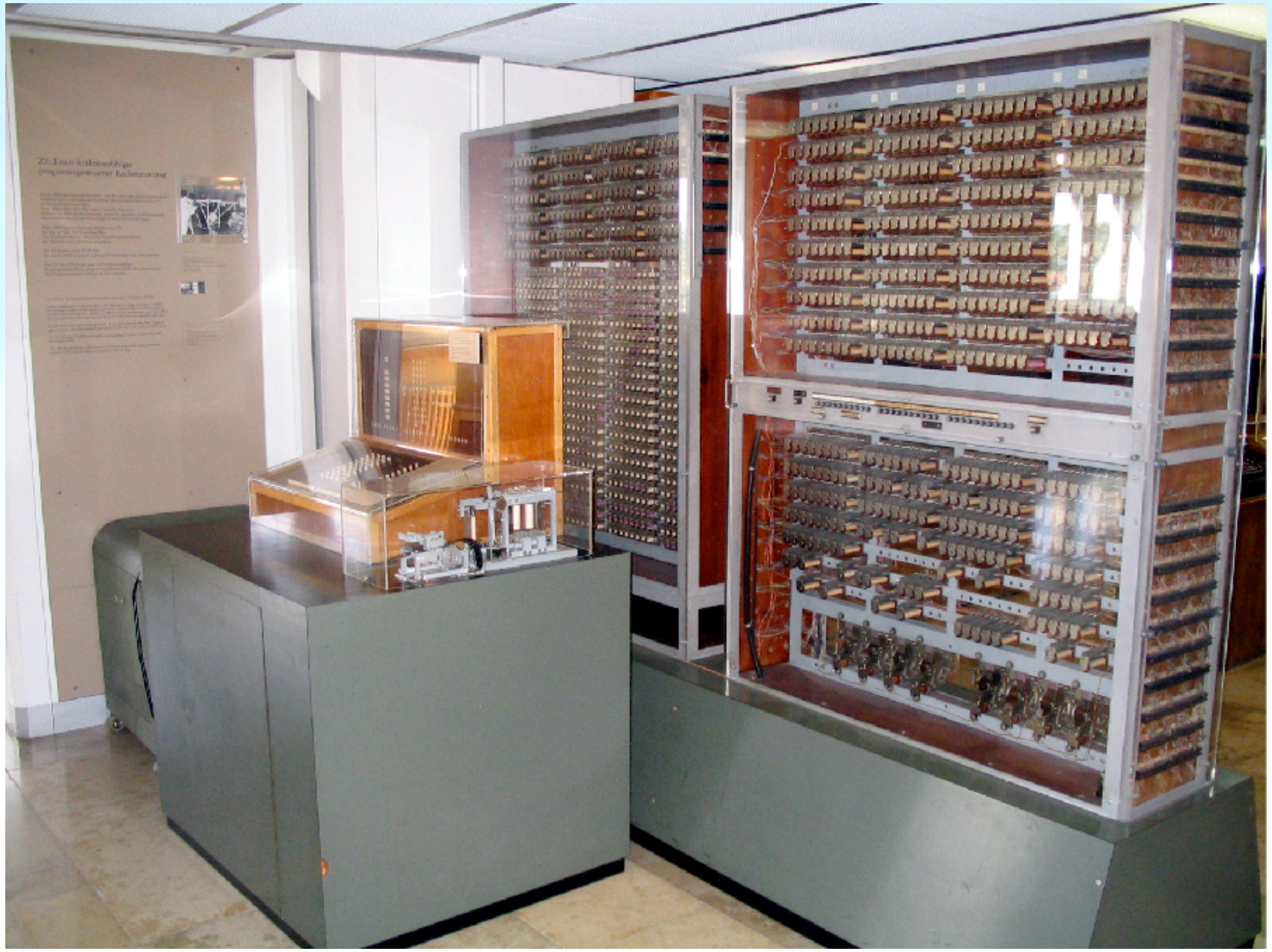
The First Electronic Computer

Atanasoff-Berry Computer 1937 - 1942 (Iowa State University)



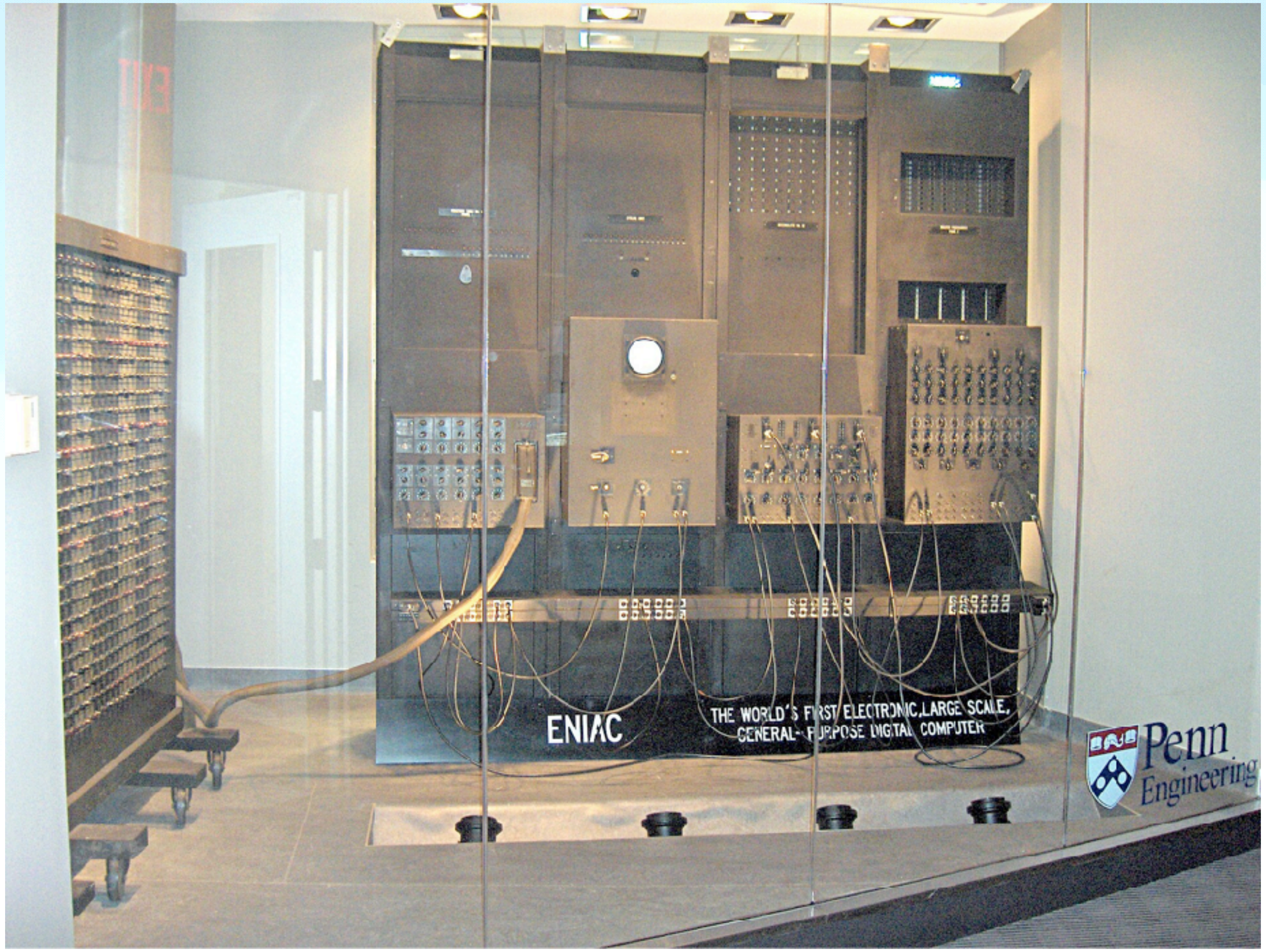
Z3

First Commercial Computer by Konrad Zuse (41, Alemão)



Eniac

1945 - First electronic, programable, and of general purpose (UPenn)



First Computer Science Curriculum

- University of Cambridge in 1953

First Computer Science Department

- Purdue University - 1962

Reading List

- Alan Turing: The Enigma
- Godel Escher Bach: An Eternal Golden Braid
- Software Foundations Series
- Types and Programming Languages

LINKS

- https://pedroabreu0.github.io
- https://typetheoryforall.com
- Instagram: @p_droabreu
- Twitter: @p_droabreu0