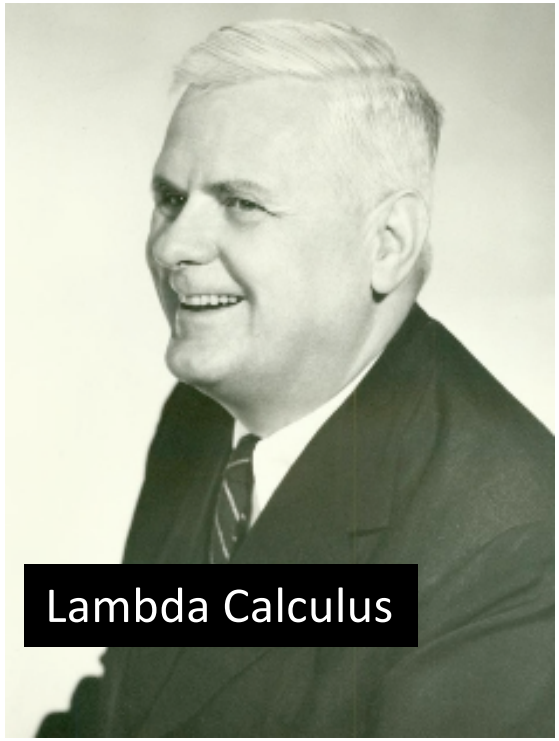
Lambda Calculus

CSE 340 FALL 2021

Rida Bazzi

Part of the presentation is adopted from types and Programming languages
by Benjamin C. Pierce, MIT Press

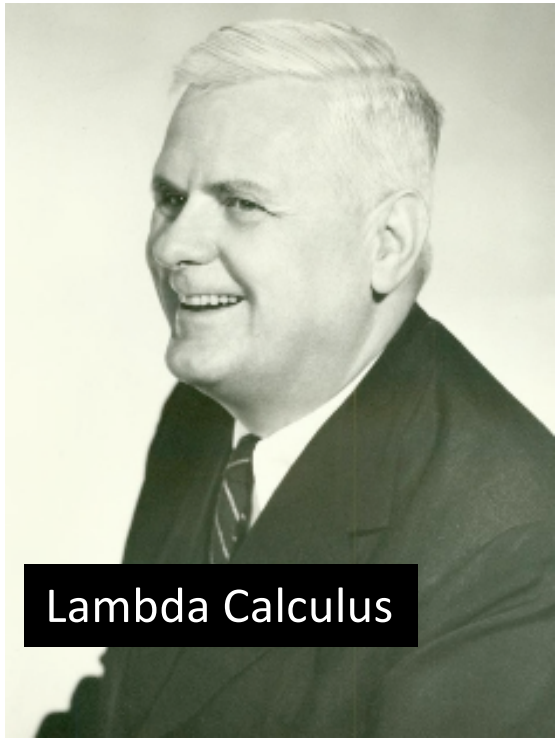
Alonzo Church



Founders of computability theory

Founders of computability theory

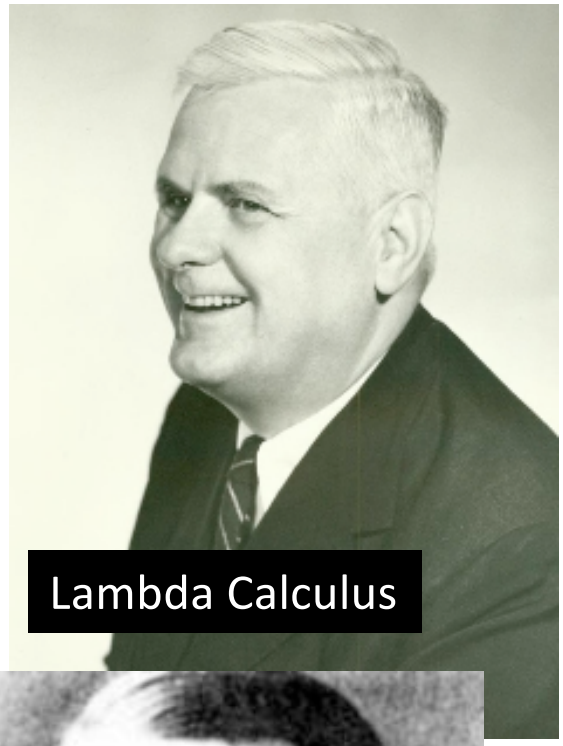
Alonzo Church



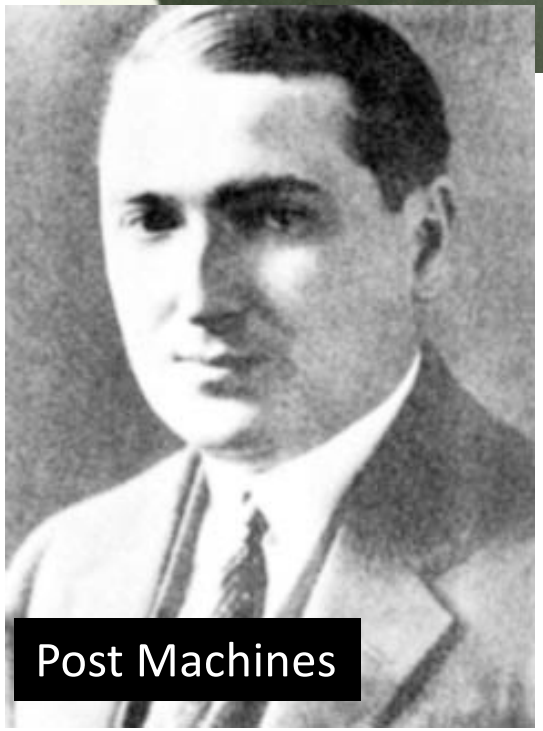
Alan Turing

Founders of computability theory

Alonzo Church



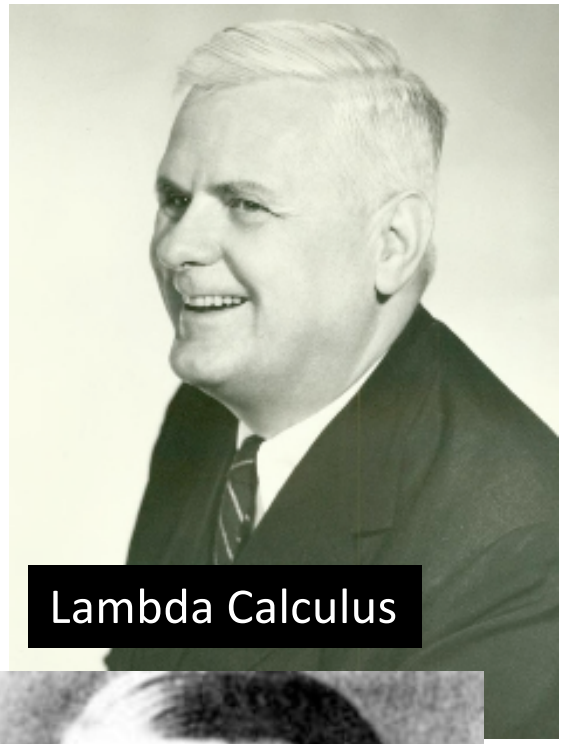
Alan Turing



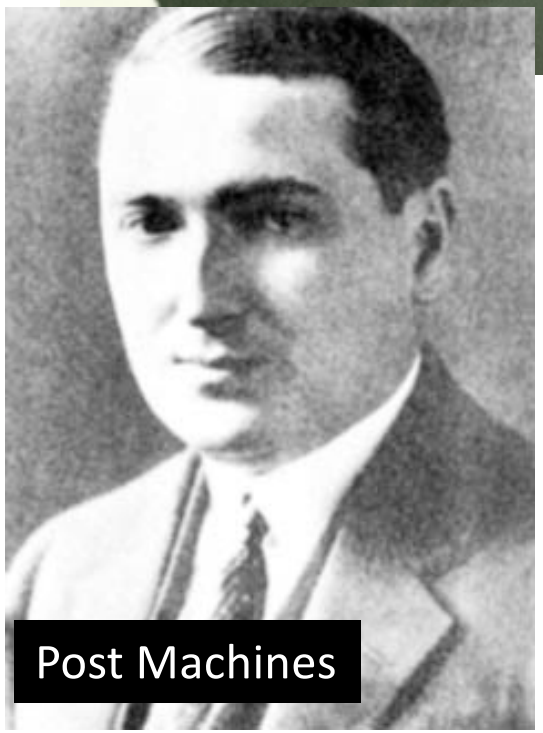
Emil Post

Founders of computability theory

Alonzo Church



Lambda Calculus



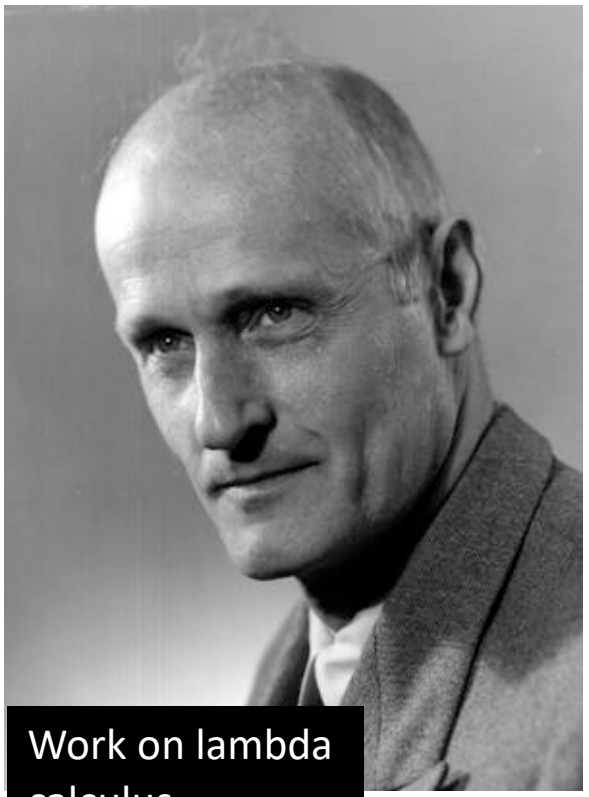
Post Machines

Emil Post

Turing Machines

Alan Turing

Stephen Kleene

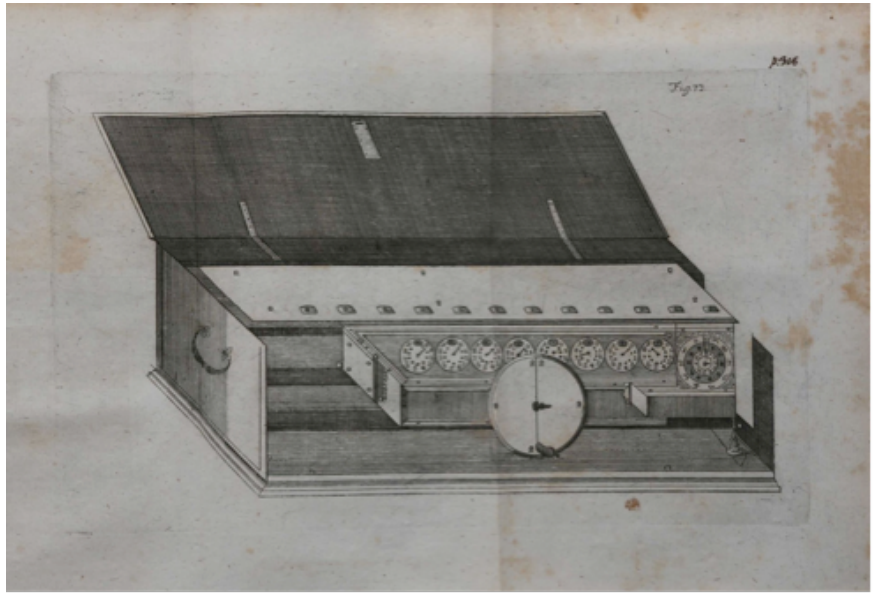


Work on lambda calculus

invented regular expressions

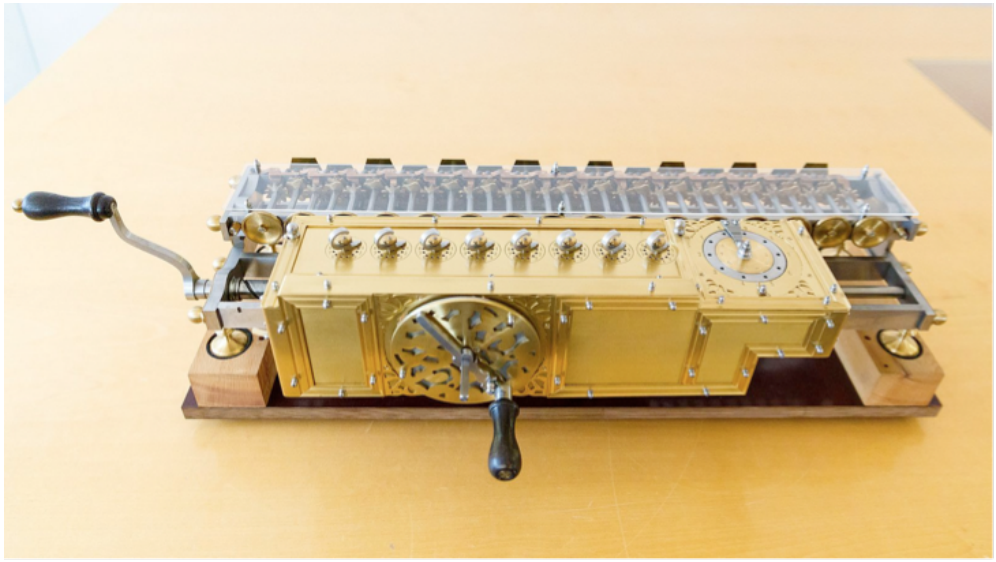
Gottfried Wilhelm Leibniz

Gottfried Wilhelm Leibniz
(wearing hair)



Drew a calculating machine

Gottfried Wilhelm Leibniz



Built a calculating machine

Gottfried Wilhelm Leibniz

Entscheidungsproblem Precursor

Was interested in a universal language to specify mathematical statements and the possibility of building a machine to test whether or not a statement written in the formal language is true or false

Gottfried Wilhelm Leibniz

Entscheidungsproblem Precursor

Was interested in a **universal language to specify mathematical statements** and the possibility of building a machine to test whether or not a statement written in the formal language is true or false

Gottfried Wilhelm Leibniz

Entscheidungsproblem Precursor

Was interested in a **universal language to specify mathematical statements** and the possibility of building a machine to test whether or not a statement written in the formal language is true or false

Gottfried Wilhelm Leibniz

David Hilbert

David Hilbert
(wearing hat)

Entscheidungsproblem

David Hilbert

Entscheidungsproblem

- Is there an algorithm to determine for any first-order arithmetic formula if the statement is provable from Peano's axioms using first order logic?

David Hilbert

Entscheidungsproblem

- Is there an algorithm to determine for any **first-order arithmetic formula** if the statement is provable from Peano's axioms using first order logic?

David Hilbert

Entscheidungsproblem

- Is there an algorithm to determine for any **first-order arithmetic formula** if the statement is provable from Peano's axioms using first order logic?

David Hilbert

$$prime(n) \equiv \forall x\, y\ \ (\, x\, y = n \Rightarrow (x = 1 \wedge y = n) \vee (x = n \wedge y = 1)\,)$$

Entscheidungsproblem

- Is there an algorithm to determine for any first-order arithmetic formula if the statement is provable from Peano's axioms using first order logic?

David Hilbert

$$prime(n) \equiv \forall x\, y\ (\, x\, y = n \Rightarrow (x = 1 \wedge y = n) \vee ((x = n \wedge y = 1)\,)$$

In English: n is prime if the only two factors of n are 1 and n

Entscheidungsproblem

- Is there an algorithm to determine for any first-order arithmetic formula if the statement is provable from Peano's axioms using first order logic?

David Hilbert

$$prime(n) \equiv \forall x\, y\ (\ x\, y = n \Rightarrow (x = 1 \wedge y = n)\ \vee((x = n \wedge y = 1)\ )$$

$$\forall x\ \exists n\ (n > x \wedge prime(n))$$

Entscheidungs problem

- Is there an algorithm to determine for any first-order arithmetic formula if the statement is provable from Peano's axioms using first order logic?

David Hilbert

$\text{prime}(n) \equiv \forall x\, y\ (x\, y = n \Rightarrow x = 1 \vee y = 1)$

$\forall x\, \exists n\, (n > x \wedge \text{prime}(n))$

Entscheidungsproblem

- Is there an algorithm to determine for any first-order arithmetic formula if the statement is provable from Peano's axioms using first order logic?

definitely more modest goal!

David Hilbert

Entscheidungsproblem

- Is there an algorithm to determine for any first-order arithmetic formula if the statement is provable from Peano's axioms using first order logic?

definitely more modest goal and wardrobe!

David Hilbert

Decision ~~Entscheidungsproblem~~

- Is there an algorithm to determine for any first-order arithmetic formula if the statement is provable from Peano's axioms using first order logic?

definitely more modest goal and wardrobe!

David Hilbert

Decision problem

- Is there an algorithm to determine for any first-order arithmetic formula if the statement is provable from Peano's axioms using first order logic?

David Hilbert

Relevant for program checking and circuit testing

Decision problem

- Is there an algorithm to determine for any first-order arithmetic formula if the statement is provable from Peano's axioms using first order logic?

David Hilbert

Can we tell if some code has buffer overflow vulnerability?

Decision problem

- Is there an algorithm to determine for any first-order arithmetic formula if the statement is provable from Peano's axioms using first order logic?

David Hilbert

Can we tell if some code has buffer overflow vulnerability?

What is an algorithm?

What is an algorithm?

- The problem posed by Hilbert did not give a definition of what an algorithm is

What is an algorithm?

- The problem posed by Hilbert did not give a definition of what an algorithm is

- Finding a solution requires defining what algorithms are

What is an algorithm?

- The problem posed by Hilbert did not give a definition of what an algorithm is
- Finding a solution requires defining what algorithms are
- This is especially important if the answer is negative (as it turned out to be)

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

1936

By Alonzo Church.

Scottsdale in 1936

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By ALONZO CHURCH.

---

2. Conversion and λ-definability. We select a particular list of symbols, consisting of the symbols { , }, (,), λ, [,], and an enumerably infinite set of symbols *a*, *b*, *c*, · · · to be called *variables*. And we define the word *formula* to mean any finite sequence of symbols out of this list. The terms *well-formed formula*, *free variable*, and *bound variable* are then defined by induction as follows. A variable x standing alone is a well-formed formula

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By ALONZO CHURCH.

---

2. Conversion and λ-definability. We select a particular list of symbols, consisting of the symbols { , }, (,), λ, [,], and an enumerably infinite set of symbols *a*, *b*, *c*, · · · to be called *variables*. And we define the word *formula* to mean any finite sequence of symbols out of this list. The terms *well-formed formula*, *free variable*, and *bound variable* are then defined by induction as follows. A variable ***x*** standing alone is a well-formed formula

Syntactically correct program

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By ALONZO CHURCH.

---

2. Conversion and λ-definability. We select a particular list of symbols, consisting of the symbols { , }, (,), λ, [,], and an enumerably infinite set of symbols a, b, c, $\cdots$ to be called *variables*. And we define the word *formula* to mean any finite sequence of symbols out of this list. The terms *well-formed formula*, *free variable*, and *bound variable* are then defined by induction as follows. A variable $\boldsymbol{x}$ standing alone is a well-formed formula

Syntactically correct program

declarations and scope

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

Theorem XVI. *Every recursive function of positive integers is λ-definable.*[16]

Theorem XVII. *Every λ-definable function of positive integers is recursive.*[17]

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

Theorem XVI. *Every recursive function of positive integers is λ-definable.*[16]

Theorem XVII. *Every λ-definable function of positive integers is recursive.*[17]

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

Theorem XVI. *Every recursive function of positive integers is λ-definable.*[16]

Theorem XVII. *Every λ-definable function of positive integers is recursive.*[17]

lambda calculus can be used to define any recursive function of integers and vice versa

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

Theorem XVI. *Every recursive function of positive integers is λ-definable.*[16]

Theorem XVII. *Every λ-definable function of positive integers is recursive.*[17]

Lambda calculus is a model for **functional programming**

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

Lambda calculus is a model for **functional programming**

Lisp, ML, Haskell are influenced by λ-calculus

Java SE-8 has *lambdas* (anonymous functions)

Can result in more compact code and even allow for some optimizations

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

---

7. The notion of effective calculability. We now define the notion, already discussed, of an *effectively calculable* function of positive integers by identifying it with the notion of a recursive function of positive integers[18] (or of a λ-definable function of positive integers). This definition is thought to be justified by the considerations which follow, so far as positive justification

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By ALONZO CHURCH.

---

7. The notion of effective calculability. We now define the notion, already discussed, of an *effectively calculable* function of positive integers by identifying it with the notion of a recursive function of positive integers[18] (or of a λ-definable function of positive integers). This definition is thought to be justified by the considerations which follow, so far as positive justification

Translation: a function is effectively computable if it can be computed in λ calculus

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

THEOREM XVIII. *There is no recursive function of a formula* $\mathbf{C}$, *whose value is* 2 *or* 1 *according as* $\mathbf{C}$ *has a normal form or not.*

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

THEOREM XVIII. *There is no recursive function of a formula* **C**, *whose value is* 2 *or* 1 *according as* **C** *has a normal form or not.*

Translation: There is no λ expression that can compute whether or not a general λ expression has normal form

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

Theorem XVIII. *There is no recursive function of a formula* **C**, *whose value is* 2 *or* 1 *according as* **C** *has a normal form or not.*

Translation: There is no λ expression that can compute whether or not a general λ expression has normal form

Better translation: There is no λ expression that can compute whether or not the evaluation of a λ expression can ever terminate

AN UNSOLVABLE PROBLEM OF ELEMENTARY NUMBER THEORY.[1]

By Alonzo Church.

Theorem XVIII. *There is no recursive function of a formula* **C**, *whose value is* 2 *or* 1 *according as* **C** *has a normal form or not.*

Translation: There is no λ expression that can compute whether or not a general λ expression has normal form

Better translation: There is no λ expression that can compute whether or not the evaluation of a λ expression can ever terminate

Even better translation: In general we cannot tell if a program will ever halt

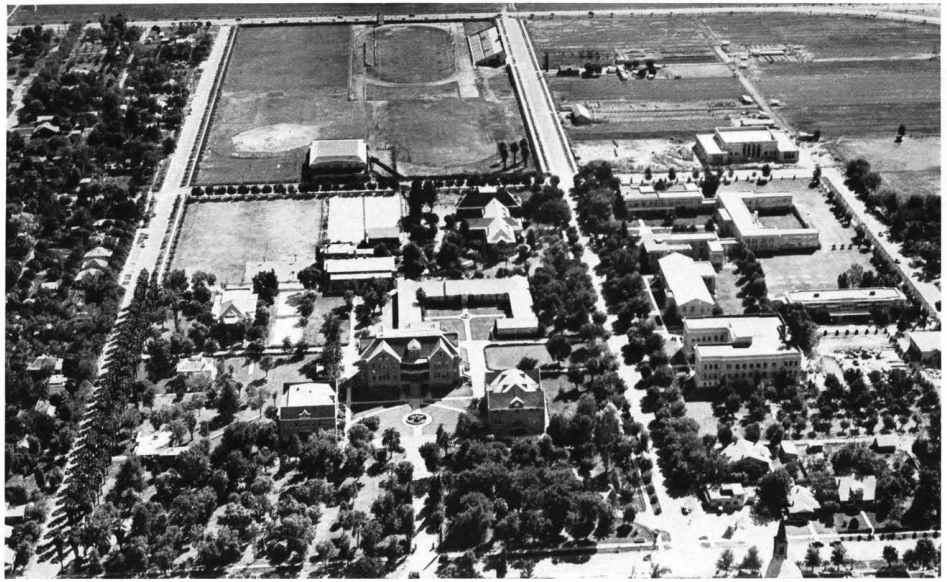
undecidable

1936

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

[Received 28 May, 1936.—Read 12 November, 1936.]



ASU in 1939

ARIZONA STATE TEACHERS COLLEGE

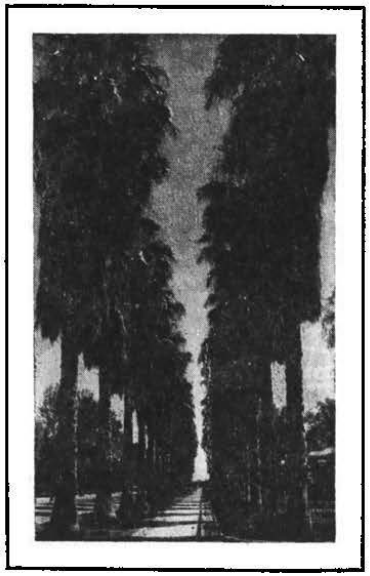
AT TEMPE

BULLETIN

CATALOGUE ISSUE

FOR THE SESSION OF

1939-1940



TEMPE, ARIZONA

THE VENERABLE CAMPUS PALMS

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

Translation: If something is computable, it is computable by a Turing machine

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

Translation: If something is computable, it is computable by a Turing machine

Turing machines are a formal model for **imperative programming**

state = instruction to execute
tape = memory (can only be accessed sequentially)
transition function = computation step

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

Turing machine influenced Von Neumann architecture

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

have valuable applications. In particular, it is shown (§ 11) that the Hilbertian Entscheidungsproblem can have no solution.

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

have valuable applications. In particular, it is shown (§ 11) that the Hilbertian Entscheidungsproblem can have no solution.

's decision problem 😢

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

have valuable applications. In particular, it is shown (§ 11) that the Hilbertian Entscheidungsproblem can have no solution.

's decision problem 😢

The 'decision problem' of the guy with the fancy hat is not solvable in general

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

proved equivalence of λ calculus and Turing machines in terms of computation power

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

To show that every λ-definable sequence γ is computable, we have to show how to construct a machine to compute γ. For use with machines it

proved equivalence of λ calculus and Turing machines in terms of computation power

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

To show that every λ-definable sequence γ is computable, we have to show how to construct a machine to compute γ. For use with machines it

To prove that every computable sequence γ is λ-definable, we must

proved equivalence of λ calculus and Turing machines in terms of computation power

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

Added 28 August, 1936.

To show that every λ-definable sequence γ is computable, we have to show how to construct a machine to compute γ. For use with machines it

To prove that every computable sequence γ is λ-definable, we must

The Graduate College,
Princeton University,
New Jersey, U.S.A.

proved equivalence of λ calculus and Turing machines in terms of computation power while visiting Church at Princeton

Turing Completeness

- λ definability $\equiv$ Turing computability

Turing Completeness

- λ definability $\equiv$ Turing computability

- A programming language is Turing-complete if it is equivalent in expressiveness/computing power to Turing machines

Turing Completeness

- λ definability $\equiv$ Turing computability
- A programming language is Turing-complete if it is equivalent in expressiveness to Turing machines
- Assembly language is Turing-complete

Turing Completeness

- λ definability $\equiv$ Turing computability
- A programming language is Turing-complete if it is equivalent in expressiveness to Turing machines
- Assembly language is Turing-complete
- All common programming languages such as C, Java, Python, Haskell, ... are Turing-complete

Turing Completeness

- λ definability $\equiv$ Turing computability
- A programming language is Turing-complete if it is equivalent in expressiveness to Turing machines
- Assembly language is Turing-complete
- All common programming languages such as C, Java, Python, Haskell, ... are Turing-complete
- lambda-calculus is Turing-complete

Turing Completeness

- λ definability $\equiv$ Turing computability
- A programming language is Tu… mplete if it is equivalent in expressivenes… ing machines
- Assembly language i… g-complete
- All common … mming languages such as C, Java, Python, H… … are Turing-complete
- la… alculus is Turing-complete

Assuming that there is no bound on memory

Summary

- Two different approaches to define computation
  - Turing machine: imperative programming
  - lambda calculus: functional programming
- Same expressiveness
  - whatever can be computed by a Turing machine can be computed by some lambda function and vice versa
- Church-Turing Thesis
  - Anything that is effectively computable is computable with a Turing machine/Lambda Calculus.