Universality and Self-Reference: Alan Turing's Legacy in Logic and Computing

Dexter Kozen
Computer Science Department
Cornell University

Braga, July 19, 2012

Alan Mathison Turing (23 June 1912 – 7 June 1954)

Chief contributions

- The Turing machine, including a definition of computability, the concepts of universality and the stored program computer (1936)
- Cracking the Enigma code (1939–42)
- The Turing test and contributions in artificial intelligence and brain modeling (1948–50)

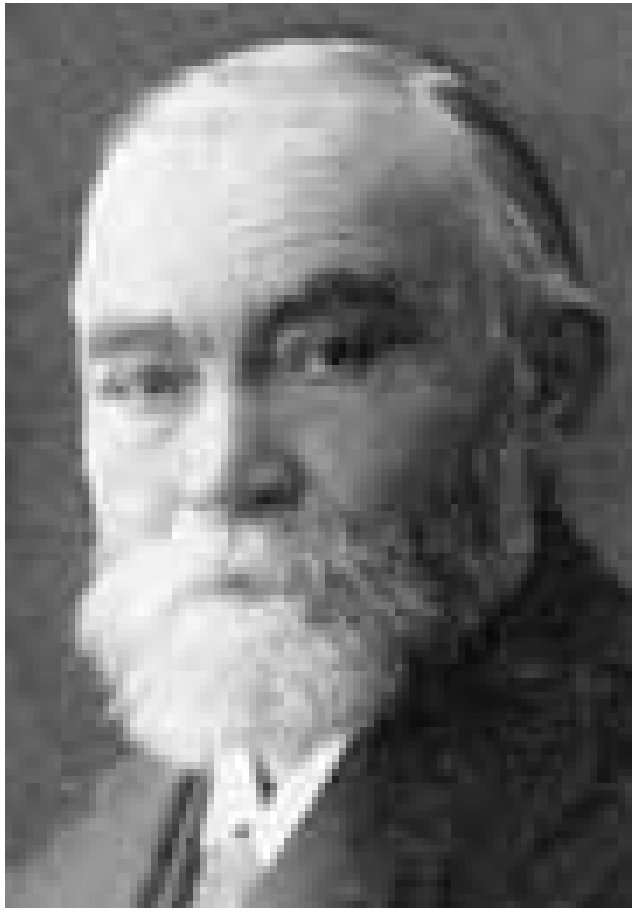
Brief Chronology of Alan Turing's Life

1912, 23 June: Birth, Paddington, London
1931-34: Undergraduate at King's College, Cambridge University
1932-35: Quantum mechanics, probability, logic
1935: Elected fellow of King's College, Cambridge
1936: The Turing machine, computability, universal machine
1936-38: Princeton, PhD in logic, algebra, number theory
1938-39: Cambridge, Bletchley Park, work on German Enigma
1939-40: The Bombe, machine for Enigma decryption
1939-42: Breaking of U-boat Enigma, saving battle of the Atlantic
1943-45: Anglo-American crypto consultant
1945: National Physical Laboratory, London
1946: Computer and software design
1947-48: Programming, neural nets, and artificial intelligence
1948: Manchester University
1949: First serious mathematical use of a computer
1950: The Turing Test for machine intelligence
1951: Elected Fellow of the Royal Society
1952: Arrested for sexual deviancy, loss of security clearance
1954, 7 June: Suicide by cyanide poisoning, Wilmslow, Cheshire

This Talk

- A. M. Turing. On computable numbers with an application to the Entscheidungsproblem. Proc. London Math. Soc. 42 (1936), pp. 230Ð265. Erratum: Ibid. 43 (1937), pp. 544Ð546.
- The context
- Universality and self-reference in competing formalisms of the time
- Turing's definition of computable reals

Prehistory



Friedrich Ludwig
Gottlob Frege
(1848–1925)

German mathematician,
logician, and philosopher

Frege's Concept of "Proof"

A sequence of statements in a formal language

$$\varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n$$

such that each φ_i is either

- an axiom (basic assumption)
- follows from statements earlier in the list by a rule of inference

Prehistory

Georg Ferdinand
Ludwig Philipp Cantor
(1845–1918)

Sets, Ordinals, and Cardinals

- Cantor introduced the concept of cardinality (size) of a set
- Two sets are the same size if there is a one-to-one correspondence between them
- There are infinitely many different sizes of infinite sets

No one shall expel us from the paradise that Cantor has created for us.
—David Hilbert

Cantor's Diagonalization Argument

.0 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 1 0 0 ...
.0 1 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
.1 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 ...
.1 0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 1 ...
.0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 1 ...
.0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
.0 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 1 0 0 ...
.0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
.1 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 ...
.1 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 1 ...
.0 1 0 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 1 ...
.0 1 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
...

Cantor's Diagonalization Argument

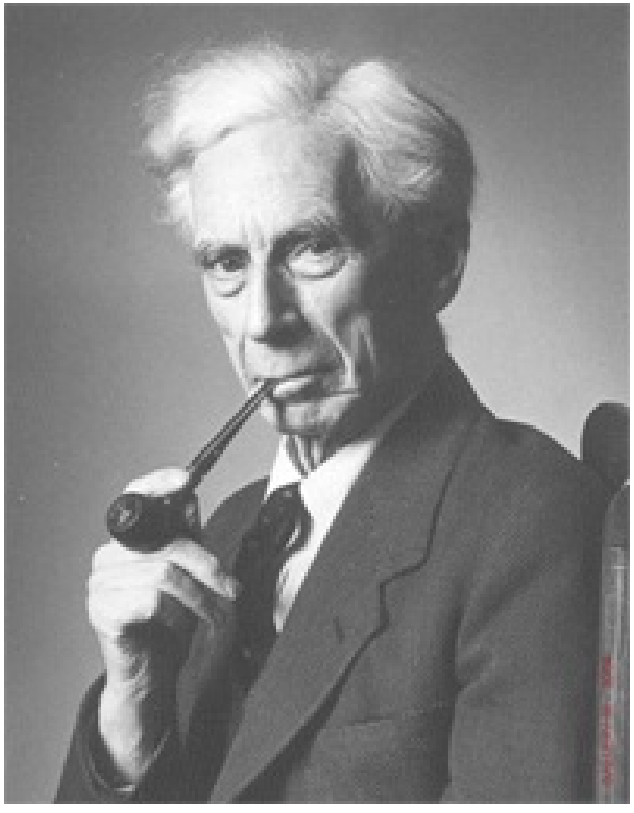
.0 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 1 0 0 ...
.0 1 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
.1 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 ...
.1 0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 1 ...
.0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 1 ...
.0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
.0 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 1 0 0 ...
.0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
.1 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 ...
.1 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 1 ...
.0 1 0 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 1 ...
.0 1 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
...

Cantor's Diagonalization Argument

.1 0 1 1 1 1 0 1 0 1 0 1 ...

.0 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 1 0 0 ...
.0 1 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
.1 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 ...
.1 0 0 0 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 1 ...
.0 1 0 0 0 1 1 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 1 ...
.0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
.0 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 1 0 0 ...
.0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
.1 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 1 0 1 1 0 1 ...
.1 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 1 ...
.0 1 0 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1 0 1 ...
.0 1 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 ...
...

Turn of the Century — The Quest for Formalization



Bertrand Arthur William Russell
3rd Earl Russell
(1872–1970)

Welsh logician,
mathematician
and philosopher

Comprehension

If $\varphi(x)$ is a property, one can form its extension

$$\{x \mid \varphi(x)\}.$$

For any y,

$$y \in \{x \mid \varphi(x)\} \quad \Leftrightarrow \quad \varphi(y).$$

Russell's Paradox

Let

$$b = \{x \mid x \notin x\}.$$

Then

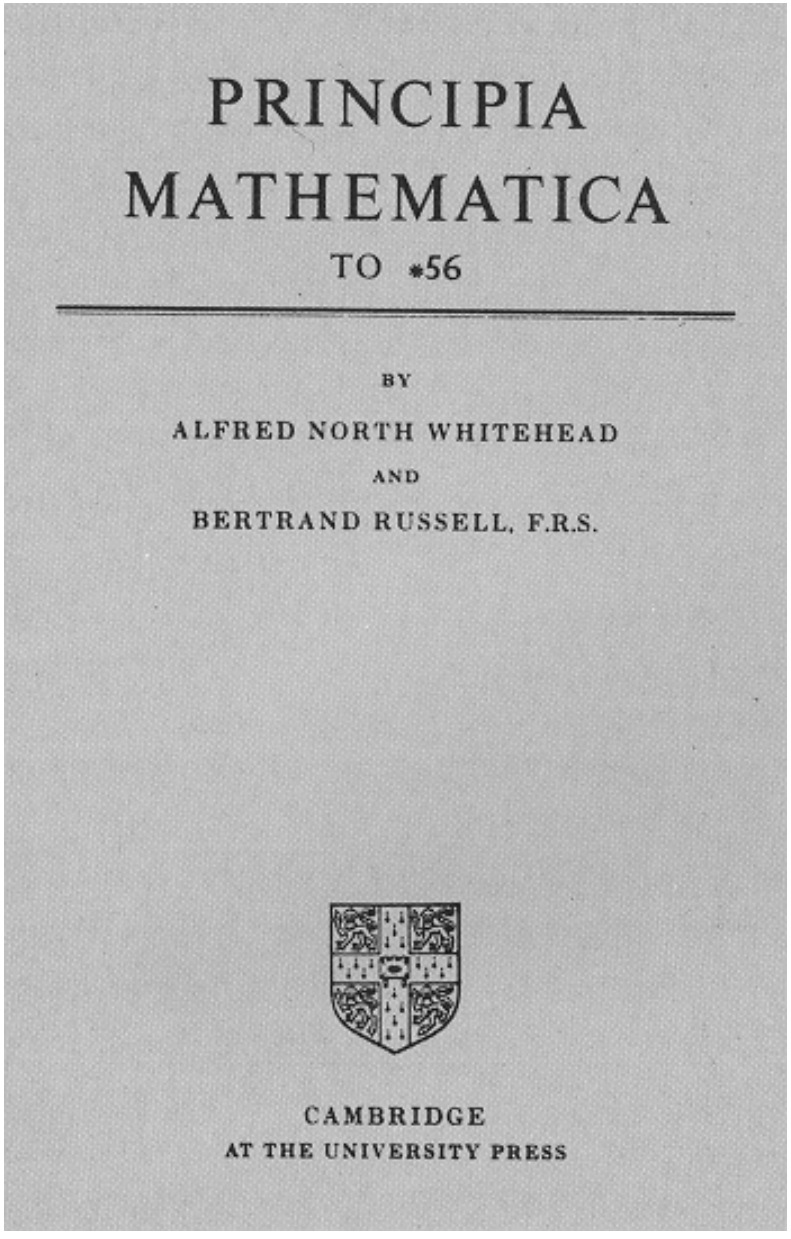
$$b \in b \quad \Leftrightarrow \quad b \in \{x \mid x \notin x\} \quad \Leftrightarrow \quad b \notin b.$$

Other Paradoxes

Burali–Forti paradox The ordinal number of the set of all ordinals must be an ordinal, and must be the largest ordinal; but there is no largest ordinal.

Cantor's paradox The cardinal number of the set of all sets must be the greatest possible cardinal. But for any set A, the cardinal number of the power set of A (set of all subsets of A) $>$ cardinal number of A.

Turn of the Century — The Quest for Formalization


PRINCIPIA
MATHEMATICA
TO *56

BY

ALFRED NORTH WHITEHEAD

AND

BERTRAND RUSSELL, F.R.S.

CAMBRIDGE
AT THE UNIVERSITY PRESS

Alfred North Whitehead
(1861–1947)

***54·43.** $\vdash :. \alpha, \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 . \supset \vdash :. \alpha = \iota'x . \beta = \iota'y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

[*51·231] $\equiv . \iota'x \cap \iota'y = \Lambda .$

[*13·12] $\equiv . \alpha \cap \beta = \Lambda \qquad (1)$

$\vdash . (1) . *11{\cdot}11{\cdot}35 . \supset$

$\vdash :. (\exists x, y) . \alpha = \iota'x . \beta = \iota'y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda \qquad (2)$

$\vdash . (2) . *11{\cdot}54 . *52{\cdot}1 . \supset \vdash . \text{Prop}$

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

Turn of the Century — The Quest for Formalization

David Hilbert
(1862–1943)

German
mathematician

Hilbert's Second Problem

"When we are engaged in investigating the foundations of a science, we must set up a system of axioms which contains an exact and complete description of the relations subsisting between the elementary ideas of that science... But above all I wish to designate the following as the most important among the numerous questions which can be asked with regard to the axioms: To prove that they are not contradictory, that is, that a definite number of logical steps based upon them can never lead to contradictory results. In geometry, the proof of the compatibility of the axioms can be effected by constructing a suitable field of numbers, such that analogous relations between the numbers of this field correspond to the geometrical axioms... On the other hand a direct method is needed for the proof of the compatibility of the arithmetical axioms."

Hilbert's Second Problem (Paraphrased)

Show that Peano Arithmetic is consistent (not self-contradictory).

That is, show that there is no Frege-style proof

$$\varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n, \ldots, \neg\varphi_n$$

Hilbert's Tenth Problem

Is there a procedure to determine whether a given a set of Diophantine equations are solvable?

The Entscheidungsproblem (Hilbert, 1928)

Given a deductive system for a logic and a sentence φ in that logic, can one decide whether φ is provable?

"For the mathematician there is no Ignorabimus, and, in my opinion, not at all for natural science either... The true reason why [no one] has succeeded in finding an unsolvable problem is, in my opinion, that there is no unsolvable problem. In contrast to the foolish Ignoramibus, our credo avers:

Wir müssen wissen, wir werden wissen!
(We must know, we shall know.)"

—David Hilbert, Königsberg, September 1930, opening address to the Society of German Scientists and Physicians

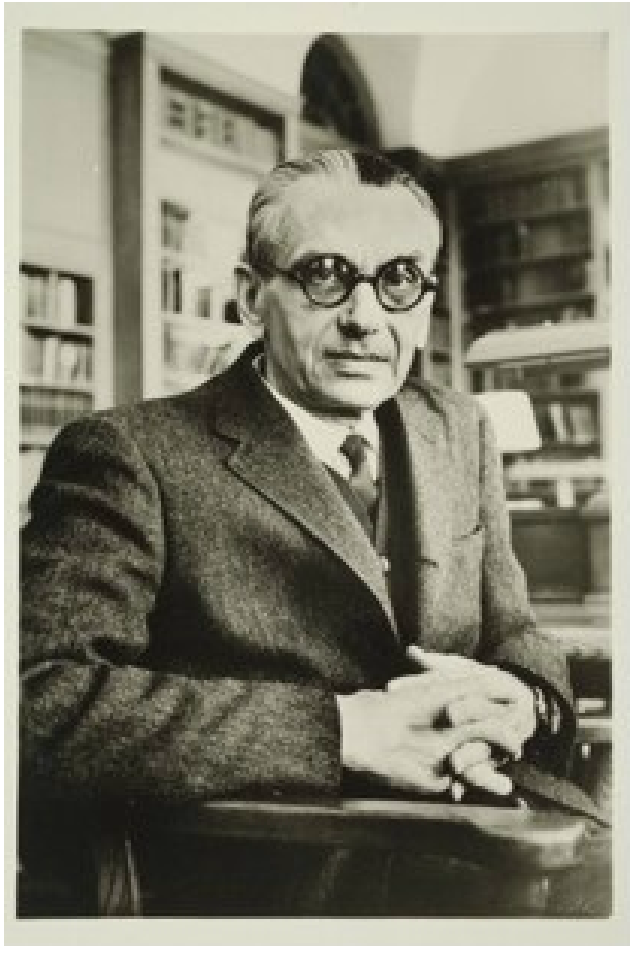
This question was the main inspiration for Turing's work.

Formalizing Computation

- proved that the Entscheidungsproblem is undecidable for Peano arithmetic
- articulated Church's thesis, later called the Church–Turing thesis
- created the λ-calculus

Alonzo Church
(1903–1995)
American mathematician
and logician

Formalizing Computation



Kurt Gödel (1906–1978)
Austrian logician, mathematician,
philosopher of mathematics

- the completeness theorem
- the incompleteness theorem
- μ-recursive functions

The μ-Recursive Functions

A collection of number-theoretic functions $\mathbb{N}^k \to \mathbb{N}$.

- Successor $s : \mathbb{N} \to \mathbb{N}$, $s(x) = x + 1$ is computable.
- Zero $z : \mathbb{N}^0 \to \mathbb{N}$, $z(\ ) = 0$ is computable.
- Projections $\pi_n^k : \mathbb{N}^n \to \mathbb{N}$, $\pi_n^k(x_1, \ldots, x_n) = x_k$, $1 \leq k \leq n$, are computable.
- Composition If $f : \mathbb{N}^k \to \mathbb{N}$ and $g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$ are computable, then so is $f(g_1, \ldots, g_k) : \mathbb{N}^n \to \mathbb{N}$ that on $\overline{x} = x_1, \ldots, x_n$ gives $f(g_1(\overline{x}), \ldots, g_k(\overline{x}))$.
- Primitive recursion If $h_i : \mathbb{N}^{n-1} \to \mathbb{N}$ and $g_i : \mathbb{N}^{n+k} \to \mathbb{N}$ are computable, $1 \leq i \leq k$, then so are $f_i : \mathbb{N}^n \to \mathbb{N}$, $1 \leq i \leq k$, defined by mutual induction:
  - $f_i(0, \overline{x}) = h_i(\overline{x})$,
  - $f_i(x + 1, \overline{x}) = g_i(x, \overline{x}, f_1(x, \overline{x}), \ldots, f_k(x, \overline{x}))$,

  where $\overline{x} = x_2, \ldots, x_n$.

The μ-Recursive Functions

- Unbounded minimization If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ is computable, then so is $f : \mathbb{N}^n \to \mathbb{N}$ that on input $\overline{x} = x_1, \ldots, x_n$ gives the least y such that $g(z, \overline{x})$ is defined for all $z < y$ and $g(y, \overline{x}) = 0$ if such a y exists and is undefined otherwise. We denote this by $f(\overline{x}) = \mu y.(g(y, \overline{x}) = 0)$.

Church (1936) published a proof of the equivalence of the μ-recursive functions and the λ-calculus, but attributed the result to Kleene.

Other Equivalent Formalisms

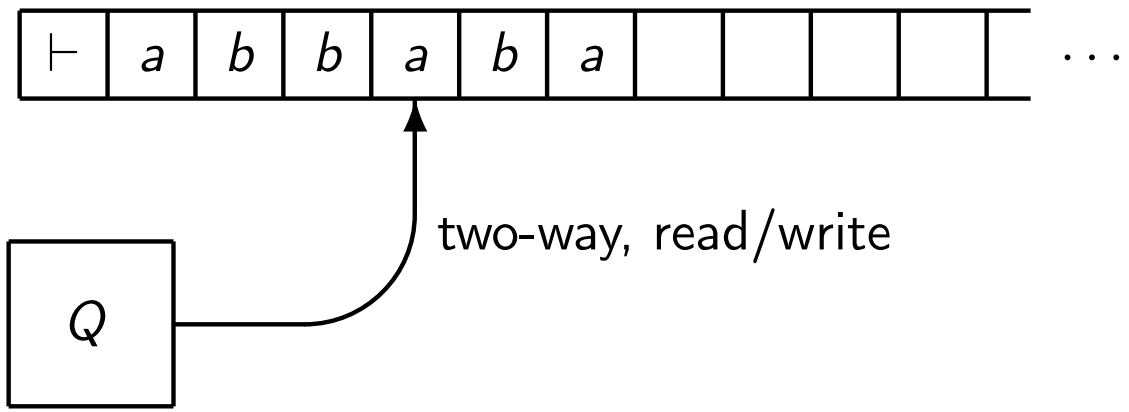
- Post systems (Emil Post 1936)
- type 0 grammars (Noam Chomsky 1956)
- combinatory logic (Schönfinkel 1925, Haskell Curry 1929)

Turing's 1936 Paper

A. M. Turing. On computable numbers with an application to the Entscheidungsproblem. Proc. London Math. Soc. 42 (1936), pp. 230Ð265. Erratum: Ibid. 43 (1937), pp. 544Ð546.

- introduced Turing machines
- proved the existence of a universal Turing machine
- defined computable reals and showed that all commonly known reals (e.g. e, π) are computable
- proved undecidability of the halting problem (and other problems) using a direct application of Cantor's diagonalization argument
- showed that Turing machines and the λ-calculus are computationally equivalent, thereby giving force to Church's thesis
- gave a much simpler proof of the undecidability of the Entscheidungsproblem

The Turing Machine (Basic Model)



Nondeterminism

If at each stage the motion of a machine (in the sense of §1) is completely determined by the configuration, we shall call the machine an "automatic machine" (or *a*-machine). For some purposes we might use machines (choice machines or *c*-machines) whose motion is only partially determined by the configuration (hence the use of the word "possible" in §1). When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice has been made by an external operator. This would be the case if we were using machines to deal with axiomatic systems. In this paper I deal only with automatic machines, and will therefore often omit the prefix *a*-.

Universality, or Programs as Data

Each of these systems is powerful enough that programs can be written that manipulate other programs encoded as data

system	*programs*	*data*
λ-calculus	λ-terms	λ-terms
combinatory logic	combinator symbols	combinator symbols
μ-recursive functions	Gödel numberings	natural numbers
Turing machines	Turing machines	strings

In Turing machines: There is an encoding of Turing machines M as strings x and a universal simulator U that on input $x\#y$ simulates M_x (the machine with code x) on input y

Led to the notion of stored-program computer, the prevailing architecture of most computers in existence today

Self-Reference

If Turing machines can compute with the codes of other Turing machines, then they can compute with their own codes.

Example: Quines (self-printing programs)

C

```
char *s="char *s=%c%s%c;
main()printf(s,34,s,34,10);%c";
main()printf(s,34,s,34,10);
```

Standard ML

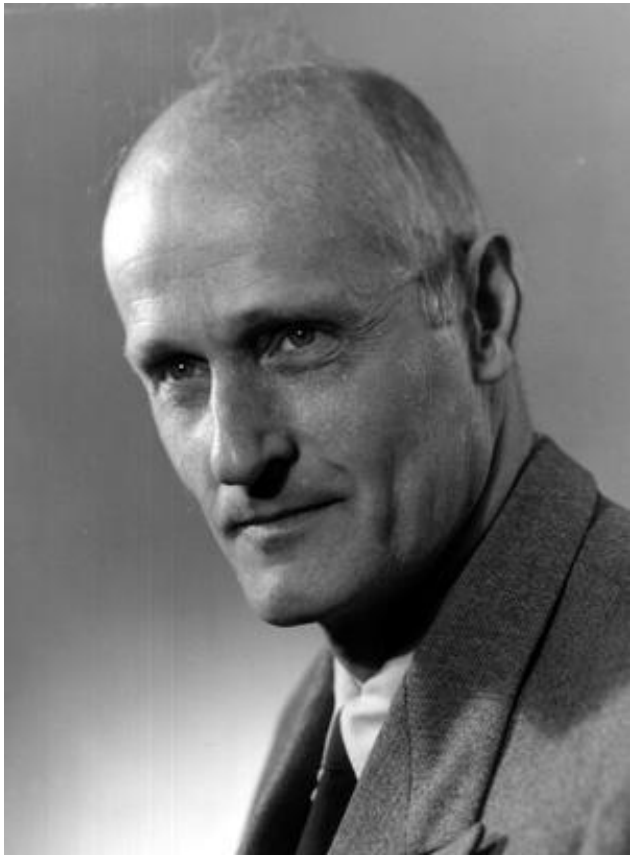
```
let val s="let val s=#$#
in String.translate(fn x=>if Char.ord x=36 then s
else implode [if Char.ord x=35 then Char.chr 34 else x])s end"
in String.translate(fn x=>if Char.ord x=36 then s
else implode [if Char.ord x=35 then Char.chr 34 else x])s end
```

Self-Reference

Java

```
public class PrintSelf
static StringBuffer self = new StringBuffer("public class PrintSelf
static StringBuffer self = new StringBuffer();
static char quote = '!';
public static void main(String[] args)
quote++; self.insert(69, self); self.insert(69, quote);
self.insert(318, quote); System.out.println(self); ");
static char quote = '!';
public static void main(String[] args)
quote++; self.insert(69, self); self.insert(69, quote);
self.insert(318, quote); System.out.println(self);
```

Self-Reference



Stephen Cole Kleene
(1909–1994)

- Kleene's theorem (equivalence of automata and regular expressions)
- Kleene's theorem (the recursion theorem)
- Kleene's theorem (inductive $= \Pi^1_1$ in $\mathbb{N}$)

The Recursion Theorem (Kleene, 1938)

Theorem

For any total computable function σ, there exists a Turing machine M_x such that M_x and $M_{\sigma(x)}$ compute the same (partial) function.

Proof.

Let M_v be a Turing machine that on input x computes the code of another Turing machine that on input y does the following.

1. Runs M_x on input x, using the universal simulator.
2. If it halts with output $M_x(x)$, apply σ to get $\sigma(M_x(x))$.
3. Runs the machine with that code on y, using the universal simulator.

Note that M_v does not do 1-3 itself, it just computes the code of a machine that does so. Then

$$M_{M_{\boldsymbol{v}}(x)}(y) = \begin{cases} M_{\sigma(M_{\boldsymbol{x}}(x))}(y) & \text{if } M_x(x) \text{ halts,} \\ \text{undefined,} & \text{otherwise.} \end{cases}$$

In particular, $M_{M_{\boldsymbol{v}}(v)}(y) = M_{\sigma(M_{\boldsymbol{v}}(v))}(y)$, since M_v is total. □

Fixpoints in the λ-Calculus

The traditional fixpoint combinator Y satisfies

$$Y\sigma = (\lambda x.\sigma(xx))(\lambda x.\sigma(xx))$$

This is a fixpoint of σ:

$$(\lambda x.\sigma(xx))(\lambda x.\sigma(xx)) \quad \rightarrow \quad \sigma((\lambda x.\sigma(xx))(\lambda x.\sigma(xx)))$$

This does not correspond to the Kleene construction. It is bad because it forces a lazy evaluation strategy. The one that corresponds to the Kleene construction is

$$v = \lambda x.\lambda y.\sigma(xx)y.$$

Then

$$vv = (\lambda x.\lambda y.\sigma(xx)y)v \rightarrow \lambda y.\sigma(vv)y$$
$$vvy \rightarrow (\lambda y.\sigma(vv)y)y \rightarrow \sigma(vv)y.$$

Gödel's Incompleteness Theorem (1933)

Gödel used exactly the same technology to prove his incompleteness theorem.

Theorem

1. *For Peano arithmetic or any sufficiently powerful deductive system for number theory, there exist sentences that are true but not provable*[a].
2. *No sufficiently powerful consistent deductive system can prove its own consistency.*

[a] unless the system is inconsistent, in which case everything is provable

The second theorem solves Hilbert's second problem negatively.

Gödel's Incompleteness Theorem (1933)

Let φ be a number-theoretic sentence. Let

$$\models \varphi \quad \Leftrightarrow \quad \varphi \text{ is true in } \mathbb{N}$$
$$\vdash \varphi \quad \Leftrightarrow \quad \varphi \text{ is provable in Peano arithmetic (PA).}$$

Can rewrite 1 as:

There exists φ such that $\models \varphi$ but $\nvdash \varphi$ (unless PA is inconsistent).

Gödel's Fixpoint Lemma

Let 'φ' be a numeric code for the formula φ. The coding scheme is chosed so that formulas of number theory can talk about other formulas in terms of their codes, manipulate them, do substitutions, etc.

Lemma (Gödel's Fixpoint Lemma)

For any formula σ with one free variable x, there exists a fixpoint φ of σ; that is, a sentence φ such that

$$\vdash \varphi \leftrightarrow \sigma(\text{'}\varphi\text{'}).$$

The sentence φ says, "I satisfy $\sigma(x)$." Moreover, the equivalence is provable in PA.

Proof of the Fixpoint Lemma

Let x_0 be a fixed variable. Let $\mathrm{subst}(x, y, z)$ be a formula with three free variables x, y, z that says:

> *"The number z is the code of the formula obtained by substituting the constant whose value is x for all free occurrences of the variable x_0 in the formula whose code is y."*

For example, if $'x_0 + 4x' = 724$, then $\mathrm{subst}(7, 724, 581)$ would be true if $'7 + 4x' = 581$.

Now define

$$\tau(x) = \forall y\ \mathrm{subst}(x, x, y) \rightarrow \sigma(y) \qquad \varphi = \tau('\tau(x_0)').$$

Proof of the Fixpoint Lemma

$$\tau(x) = \forall y\ \mathsf{subst}(x, x, y) \to \sigma(y) \qquad\qquad \varphi = \tau(\text{'}\tau(x_0)\text{'}).$$

The sentence φ is the desired fixpoint:

$$\begin{array}{lcl}
\varphi & \Leftrightarrow & \tau(\text{'}\tau(x_0)\text{'}) \\
& \Leftrightarrow & \forall y\ \mathsf{subst}(\text{'}\tau(x_0)\text{'}, \text{'}\tau(x_0)\text{'}, y) \to \sigma(y) \\
& \Leftrightarrow & \forall y\ y = \text{'}\tau(\text{'}\tau(x_0)\text{'})\text{'} \to \sigma(y) \\
& \Leftrightarrow & \forall y\ y = \text{'}\varphi\text{'} \to \sigma(y) \\
& \Leftrightarrow & \sigma(\text{'}\varphi\text{'}).
\end{array}$$

We have argued semantically, but the whole argument can be carried out in PA, thus

$$\vdash \quad \varphi \leftrightarrow \sigma(\text{'}\varphi\text{'}).$$

Proof of the First Incompleteness Theorem

Theorem (Gödel's First Incompleteness Theorem)

Suppose PA is consistent. There exists a number-theoretic sentence φ that is true but not provable.

Proof.

One can construct a sentence provable(x) such that

$$\vdash \varphi \quad \Leftrightarrow \quad \vdash \text{provable}(\text{'}\varphi\text{'}) \tag{1}$$

for all φ. Take φ to be a fixpoint of $\neg$provable(x). Then

$$\vdash \varphi \quad \leftrightarrow \quad \neg\text{provable}(\text{'}\varphi\text{'}) \tag{2}$$

by the fixpoint lemma[a]. Combining (1) and (2),

$$\vdash \varphi \quad \Rightarrow \quad \vdash \bot,$$

a contradiction, therefore φ is not provable. By (2), it is true. □

[a] φ says: "I am not provable."

Turing's Proof

Proof.

Let Con(PA) be the set of theorems of PA, and let Th($\mathbb{N}$) be the set of true statements of number theory.

Con(PA) is recursively enumerable: one could build a Turing machine that runs forever and constructs longer and longer proofs, emitting new theorems as they are discovered.

But Th($\mathbb{N}$) is not recursively enumerable: if it were, say by a machine M, then it would be decidable: to decide φ, just run M on both 'φ' and '$\neg\varphi$'. One of them is true and will eventually be enumerated. And we know it is not decidable, because we can express the halting problem.

So Con(PA) $\neq$ Th($\mathbb{N}$). Either there is a $\varphi \in \text{Th}(\mathbb{N}) \backslash \text{Con}(PA)$, in which case we are done, or there is a $\varphi \in \text{Con}(PA) \backslash \text{Th}(\mathbb{N})$, in which case $\neg\varphi \in \text{Th}(\mathbb{N}) \backslash \text{Con}(PA)$, and we are also done. $\square$

Q. Why are Turing Machines So Great?

Q. Why are Turing Machines So Great?

A. Programmability!

Turing's Notion of Computable Real Numbers

Definition (Turing 1936)

A computable real α is given by a Turing machine that eventually writes down all the digits in the binary expansion of α.

Alternative definitions:

Definition

A computable Cauchy sequence consists of computable total functions $\alpha : \mathbb{N} \to \mathbb{Q}$ and $f : \mathbb{N} \to \mathbb{N}$ such that for all n and for all $m, k \geq f(n)$, $|\alpha_m - \alpha_k| < 1/n$.

Definition

A computable convergent interval sequence consists of a computable total interval-valued function $[\alpha, \beta] : \mathbb{N} \to \mathbb{Q}^2$ such that for all n, $[\alpha_{n+1}, \beta_{n+1}] \subseteq [\alpha_n, \beta_n]$ and $\beta_n - \alpha_n < 1/n$.

Turing's Notion of Computable Real Numbers

The latter two definitions (Cauchy sequences and interval sequences) are equivalent: one can go back and forth effectively between representations.

Unfortunately, the Turing representation (enumeration of digits) effectively determines the other two, but not vice versa!

Q. Does this mean there are fewer Turing reals than Cauchy sequences and interval sequences?

Turing's Notion of Computable Real Numbers

The latter two definitions (Cauchy sequences and interval sequences) are equivalent: one can go back and forth effectively between representations.

Unfortunately, the Turing representation (enumeration of digits) effectively determines the other two, but not vice versa!

Q. Does this mean there are fewer Turing reals than Cauchy sequences and interval sequences?

A. Yes. . . and no. . .

Turing's Notion of Computable Real Numbers

Theorem (Turing 1936)

Every Cauchy sequence converges to a Turing real.

Proof.

Let α be the limit of the Cauchy sequence. If α is rational, then it is clearly computable, as its digit sequence is ultimately periodic. If α is irrational, then eventually arbitrarily long prefixes of the expansion are determined. $\square$

Turing's Erratum (1937) Fixes the Problem (?)

Use a different representation: output a sequence $c_n \in \{-1, 1\}$. The number represented is

$$c_0 \cdot m + \sum_{i=m}^{\infty} c_i \cdot (2/3)^i$$

where m is the number of leading -1's!

Obrigado!