ALAN TURING YEAR

2012

ucalgary.ca/turing

Alan Turing and the Decision Problem

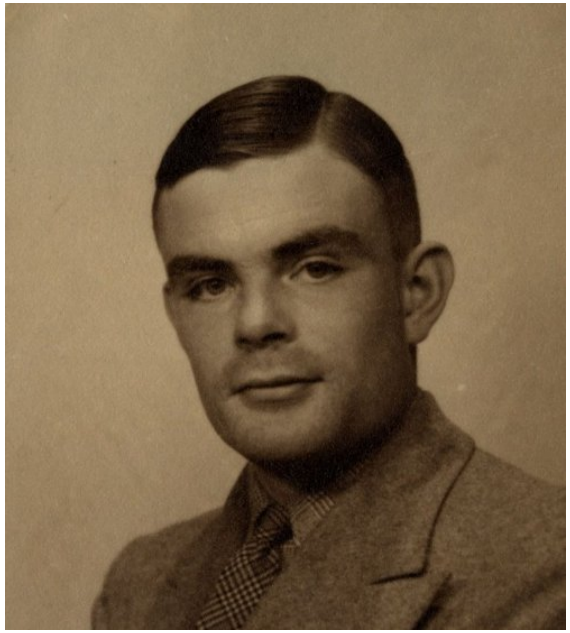
Richard Zach

University of Calgary, Canada
www.ucalgary.ca/~rzach/

January 24, 2012

UNIVERSITY OF CALGARY
PHILOSOPHY

Alan Mathison Turing, OBE, FRS (1912–1954)



- Mathematics at Cambridge, "On computable numbers" (1936)
- Wartime work on breaking German codes ("Enigma") at Bletchley Park
- After war, worked on
  - design of electronic computers,
  - artificial intelligence ("Turing Test"),
  - mathematical biology
- Openness about homosexuality resulted in criminal conviction
- Death from cyanide poisoning in 1954

Outline

Axioms and Primitives

- Axiomatic method systematizes a domain of knowledge
  - identifying primitive concepts and relations
  - collecting basic propositions (axioms)
- All truths of domain can be
  - formulated in terms of primitive concepts
  - proved from axioms
- Axioms completely describe relationships between primitives
- Eliminates “intuition” from proofs

Hilbert's Geometry

- David Hilbert (1862–1943)
- Worked at University of Göttingen
- Foremost mathematician of his time
- *Foundations of Geometry* (1899)
- Primitives:
  - point, line, plane
  - betweeness, containment, congruence

Hilbert's Geometry: Axioms

- Two distinct points determine a straight line, i.e., for any two distinct points A and B there is one and only one line g which contains both A and B
- If A is between B and C, then A is also between C and B
- If a line g and a point A are both contained in a plane α, but A is not contained in g, then there is one and only one line h contained in α which contains A but has no point in common with g. ("Parallel Axiom")
- …

Development of Axiomatic Method

- Euclid's *Elements* (c. 300 BCE)
- More rigorous theories for more areas of science (19th/20th C.)
  - Arithmetic (Dedekind)
  - Set theory (Cantor, Zermelo)
  - Geometry (Riemann, Lobachevsky, Pasch, Hilbert)
  - Probability theory (Kolmogorov)
  - Thermodynamics, kinetic theory of gases (Hilbert)
  - …
- Formalization
  - Logic (Frege, Peano, Russell, Hilbert)
  - Formal axiom systems for mathematics, physics

Axiom Systems and Logical Calculus

- Hilbert and his school developed logical calculus which could be applied to arbitrary axiom systems
- Allowed formalization of axiomatic systems as collections of formulas in an artificial language
- A proposition follows from the axioms if, in the logical calculus, there is a derivation of the (formalized) proposition from the (formalized) axioms.
- Derivations purely formal, sequences of symbols
- Decision Problem (1921): Show that question of whether a formula can be derived from axioms has a systematic, mechanical solution

Alan Turing's "On Computable Numbers" (1936)

230 A. M. TURING [NOV. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

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

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, 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.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

- Introduces Turing Machines as a way to make "mechanical procedure" precise
- Applies theory to decision problem (shows cannot be solved)
- Read original here

Computable Numbers

- Every real number can be written as an infinite decimal, e.g,

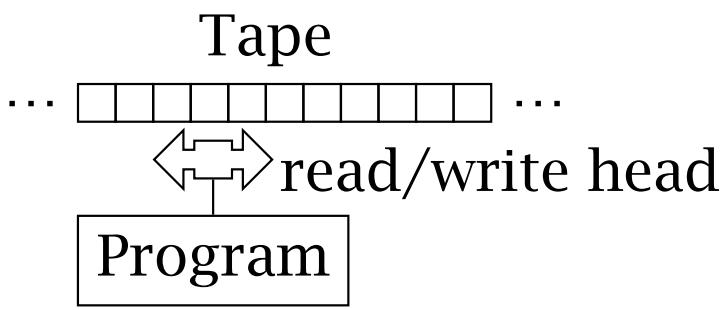
$$1/2 = 0.5000000\ldots$$
$$1/3 = 0.3333333\ldots$$
$$\pi - 3 = 0.1415926\ldots$$

- For simplicity, we may also write numbers as infinite binary decimals. In binary,

$$1/2 = 0.1000000000000000000000000\ldots$$
$$1/3 = 0.0101010101010101010101010101\ldots$$
$$\pi - 3 = 0.0010010000111111101101010\ldots$$

- A real number is computable if the infinite sequence of 0's and 1's can be produced mechanically (by a machine/program)

Turing's Abstract "Computing Machines"



- Tape can contain symbols (e.g., blank (#), **0**, **1**)
- Read/write head can read one square at a time, replace symbol on it, move one square to left or right
- Program tells machine what to do, depending on which "state" it is in, and what it currently reads

Machines Computing Numbers

Turing Machine starts on empty tape, writes infinite sequence of 0's and 1's on its tape, never halts

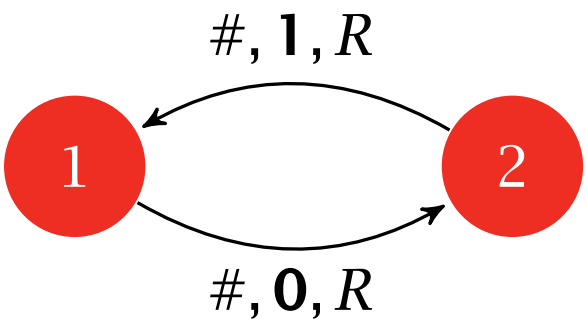
- Sequence of **0**'s and **1**'s produced by TM: real number in binary
- Machine computes a computable number

Machines Computing Output from Input

Turing machine starts on tape (which contains input), eventually halts

- Input may be a number n represented by a sequence of n **1**'s on the tape
- Machine halts if program doesn't say what to do
- Contents of tape after machine halts: output
- Such a TM computes a function
- If output is always just a single **0** or **1**:
  TM tests if input satisfies a condition (**1** if it does, **0** if not)

Turing Machine Program for 1/3

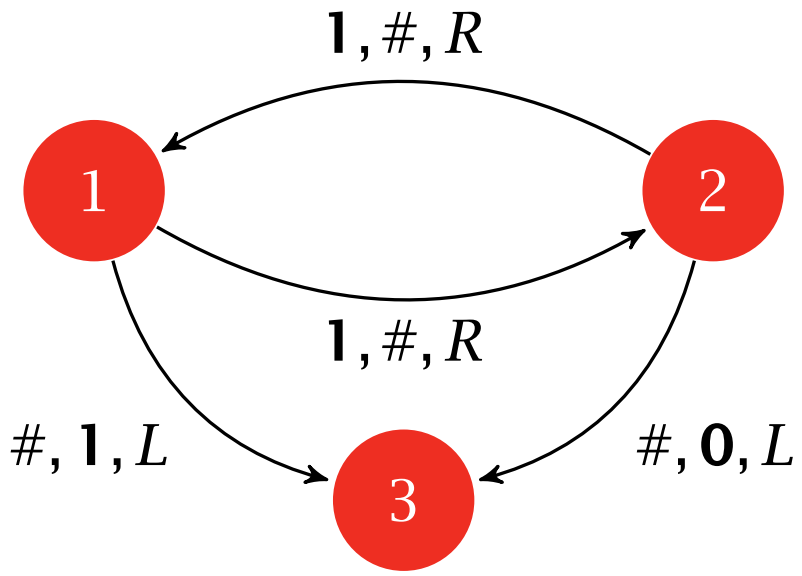


$\langle 1, \#, \mathbf{0}, R, 2\rangle$;

$\langle 2, \#, \mathbf{1}, R, 1\rangle$

- Start on empty tape, in state 1
- Since tape blank, always reads #
- Machine alternates between state 1 and 2
- Writes **0** and moves right, then **1** and moves right, …

Turing Machine Program for Testing if n is Even



$\langle 1, \mathbf{1}, \#, R, 2\rangle$;

$\langle 1, \#, \mathbf{1}, L, 3\rangle$;

$\langle 2, \mathbf{1}, \#, R, 1\rangle$;

$\langle 2, \#, \mathbf{0}, L, 3\rangle$

- Machine starts at left of sequence of **1**'s in state 1
- Keeps going right until it reads a blank
- While it does, it alternates between states 1 and 2, and erases tape
- If it stops reading **1**'s in state 1, it's read an even number of **1**'s

Running the Turing Machines

- Open xTuringMachine
- Select program
- Set state to **1**
- Click "run"

Descriptions of Turing Machines

- A TM is described entirely by its program
- That program is a list of instruction
- We can associate with each instruction a number:

$$\begin{array}{ccccccccccc} \langle & i, & & S_j, & & S_k, & & L/R, & & m & \rangle \\ & 3\underbrace{1\ldots1}_{i} & & 3\underbrace{2\ldots2}_{j} & & 3\underbrace{2\ldots2}_{k} & & 4/5 & & 3\underbrace{1\ldots1}_{m} & \end{array}$$

- and with a program (list of instruction), the number corresponding to each instruction, separated by 7's.
- For instance, $\langle 1, \#, \mathbf{0}, R, 2\rangle; \langle 2, \#, \mathbf{1}, R, 1\rangle$ has number

$$31\ 32\ 322\ 5\ 311\ 7\ 311\ 32\ 3222\ 5\ 31$$

($\# = S_1$, $\mathbf{0} = S_2$, $\mathbf{1} = S_3$)

The Universal Turing Machine

- The Universal Turing Machine U
  - takes as part of input the number k corresponding to description of a turing machine T
  - when started on a tape containing k **1**'s (plus other input), it does exactly what T would have done on the input
- If T computes a number, U started on input k computes the same number
- If we write $T(n)$ for he output of T started on input n, then

$$T(n) = U(k, n)$$

  for all n.
- U is an interpreter for TM programs

Testing TM Codes

- Since TM programs can be identified with their corresponding numbers, we can use TM programs as inputs to other TMs
- We might ask: are there TMs that test TM programs for certain properties, e.g.,
  - Does T compute a number, i.e., does T produce an infinite sequence of **0**'s and **1**'s? ("Is T circle-free?")
  - Does T ever write a **0** if started on the blank tape?
- These decision problems would be solvable if some TM started on input k always produces **1** or **0** depending on the answer

Unsolvability of Decision Problems

- Turing showed that these problems are not solvable by Turing Machines
- One important problem of this sort is the Halting Problem:
  - Does T started on input n ever halt?
- Famously also unsolvable, but not mentioned by Turing!

Uncomputable Numbers

- Imagine the numbers corresponding to TM programs listed in ascending order
- Not all of these TMs compute numbers;
  leave off all those that don't
- Let k_i be the ith TM program code in that list
- TM k_i computes a number α_i (sequence of **0**'s and **1**'s)
- Let $\alpha_i(j)$ be the jth digit of α_i
- Let β be the number whose jth digit is $1 - \alpha_i(j)$
- This means that if the jth digit of α_i is 0, the jth digit of β is 1, and vice versa

An Uncomputable Number

$$
\begin{array}{lcl}
\alpha_1 & = & 0.\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \ldots \\
\alpha_2 & = & 0.\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ \ldots \\
\alpha_3 & = & 0.\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ \ldots \\
\alpha_4 & = & 0.\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ \ldots \\
\alpha_5 & = & 0.\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ \ldots \\
\alpha_6 & = & 0.\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \ldots \\
\alpha_7 & = & 0.\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ \ldots \\
\alpha_8 & = & 0.\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ \ldots \\
\vdots & & \\
\beta & = & 0.\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ \ldots
\end{array}
$$

- So β is not among the α_i's, and thus not computable
- But wait! Couldn't we compute β by computing $1 - \alpha_i(i)$ in sequence (using U)?

Undecidability of Circle-Freeity

- In order to compute β, we'd have to first find the code of the ith circle-free TM!
- Is there a TM C which, if started on input k, will eventually halt with
  - output **1** if the TM with number k is circle-free, and with
  - output **0** if it is not?
- If there were such a TM, we could use it to compute β
- Since β is not among the α_i's, it isn't computable
- So there can't be a TM like C

Logic and Formal Axioms

- Formal language and calculus of logic developed by the 1920s
- The question of whether a mathematical proposition is a consequence of some axiom system can be made precise:
  - Is there a derivation of the corresponding formula from the axioms in the logical calculus?
- For instance, in an axiom systems for number theory:
  - to ask whether Lagrange's Theorem "Every natural number is the sum of four squares" follows from axioms
  - is to ask if the formula

$$\forall x \exists y_1 \exists y_2 \exists y_3 \exists (y_4 x = y_1^2 + y_2^2 + y_3^2 + y_4^2)$$

can be proved in Hilbert's logical calculus from the axiom formulas of Peano Arithmetic.

The Logical Decision Problem

- The logical decision problem is the problem of finding a a general mechanical procedure which, for any formal axiom system and any formula, can decide if the formula can be derived from the axioms in the logical calculus
- Would suffice if there were a general mechanical procedure which decides if a formula can be derived in Hilbert's logical calculus alone (without axioms)
- [B can be proved from axioms A_1, …, A_n, if, and only if

$$(A_1 \,\&\, \ldots \,\&\, A_n) \rightarrow B$$

  can be proved in the logical calculus alone]
- Hilbert thought such a procedure can be found
- Partial successes in the 1920s gave cause for optimisim

Turing's Negative Solution to the Decision Problem

- A TM T and its computation on an empty tape may be seen as an axiomatic theory!
- Then the question of whether T, e.g., ever prints the symbol **0** can be seen to fall within the decision problem:
  Does the statement "TM T ever prints a **0**" follow from the axioms?
- So if the decision problem could be solved, we would also have a solution to the decision problems about Turing Machines.
- But as Turing showed, these are unsolvable.

Turing Machines as Axiom Systems

Primitives (same for all TMs):

- "configuration of T", "square of the tape", "state of machine", "symbol"
- "follows after" (for configrations of T), "right of", "left of" (for squares)
- "in configuration x, T is in state q_i
- "in configuration x, T is scanning square s"
- "in configuration x, square s contains symbol S_j"
- …

Turing Machines as Axiom Systems

Axioms (some depend on T):

- Facts about the execution of T, which depend on T's program, e.g.,
- If T contains the instruction

$$\langle q_i, S_j, S_k, R, q_m \rangle,$$

one of the axioms is

> "Whenever T is in a configuration where its state is q_i, it is scanning the square s of the tape, and square s contains symbol S_j, then in the next configuration T is in state q_m, the square s contains symbol S_k, and T is scanning the square to the right of s"

- Other facts about T, e.g., that the tape doesn't change outside of where T is working, etc.

Questions about Turing Machines

Turing Machine T ever prints **0**

if, and only if

"In some configuration of T some square s contains the symbol **0**" follows from the axioms describing T.

Importance of "On Computable Numbers"

Turing's paper has made immense impact in several respects

- Limitations of logic and the axiomatic method
- Foundation of theoretical computer science
- Insight into the nature of computation

Limitations of Logic

- Logic cannot be fully mechanized
- No general procedure for all mathematical problems
- Despite this, subsequent work in
  - automated deduction
  - decision procedures for specific theories (e.g., geometry)
- Also, no general procedure for answering questions about computer programs

Foundations of Computer Science

- Turing's work became foundation for mathematical theory of computability
- …which later became theoretical computer science
- Turing machines now used as main abstract model for computability and complexity of computation

Nature of Computation

- In his paper, Turing was first to provide reasons for taking Turing machine as analysis of computability
  - Reflection on what human computors can do
  - Equivalence of Turing machines to other definitions
  - Showing that large classes of numbers/functions are computable by TM
- Reason people became convinced of unsolvability of the decision problem after Turing
- Turing machine analysis of computability now widely accepted ("Church-Turing Thesis")

Thank You

- Next lectures:
  - Michael Williams: "Turing's Real Machines"
    Tuesday, February 28
  - John Ferris: "Alan Turing and Enigma"
    Tuesday, March 27
- Please visit

  ucalgary.ca/turing

  for information on more events!