Alan Turing and the Unsolvable Problem
To Halt or Not to Halt—That Is the Question

Cristian S. Calude

26 April 2012

Alan Turing

Alan Mathison Turing was born in a nursing home in Paddington, London, now the Colonnade Town House

on 23 June 1912, less than three months after the sinking of the Titanic.

He was to become, like Freud, who stayed briefly in the House in 1938, an explorer of the mind. But his life would not make him as famous as Freud, at least for a while.

Sherborne School

Turing started at Sherborne School, a classic English “public school” in the town of Sherborne, Dorset, in south-west England.

Sherborne was a boys boarding school and isolation from family had a life-long affect.

With very bad reports he was nearly stopped from taking the School Certificate.

Reports

English: I can forgive his writing, though it is the worst I have ever seen, and I try to view tolerantly his unswerving inexactitude and slipshod, dirty, work, inconsistent though such inexactitude is in a utilitarian; but I cannot forgive the stupidity of his attitude towards sane discussion on the New Testament. Bottom of the class.

Maths and science: His work is dirty.

Maths and science

In spite of the difficult start, in 1928 he was admitted to enter the sixth form of Sherborne School to study mathematics and science.

He read mathematics as an undergraduate student at King's College, Cambridge, 1931–1934.

He read J. von Neumann (quantum physics) and B. Russell (logic).
He was elected fellow of King's College, 1935.

Changing the foundations of mathematics: David Hilbert

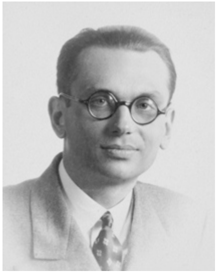
Hilbert's programme (1920): formalise mathematics as a set of fundamental truths (axioms) and rules of deduction and demonstrate that the system is consistent—i.e. free of contradictions.

Hilbert's Entscheidungsproblem (1928): find "a purely mechanical mindless procedure" to decide whether an arbitrary mathematical statement is provable in the system, i.e. true.

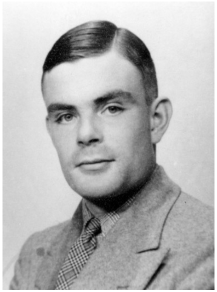
Changing the foundations of mathematics: Kurt Gödel

In 1931 Kurt Gödel published his incompleteness theorem: any computable axiomatic system that is powerful enough to describe the arithmetic of the natural numbers cannot be both consistent and complete (every true statement is provable in the system).

Oversimplifying, incompleteness "killed" Hilbert's programme.



Changing the foundations of mathematics: Alonzo Church and Alan Turing



In 1936–1937 Alonzo Church—an established logician—and Alan Turing, a recent graduate, 9 years younger, proved independently that the Entscheidungsproblem has no solution. Max Newman's role.

Turing's idea of computation

The main concept used by Turing's proof is the "a(utomatic) –machine", nowadays called Turing machine, which is a simple and abstract model of human computation.

Turing imagines a person—not a mechanism—whom he calls the "computer", who executes mechanical rules slavishly, "in a desultory manner". He then develops the mechanism into a class of abstract machines.

The Turing machine

The Turing machine mechanically operates on a potentially infinite tape on which the machine can read and write (one at a time) symbols using a tape head, which can move left or right.

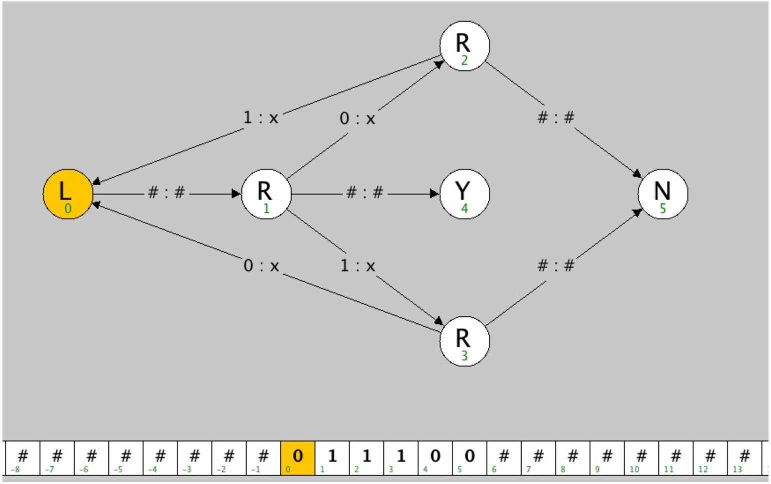
Operation is fully determined by a finite list of instructions—the instruction table—such as

> *"in state 17, if the symbol seen is 0, write a 1, move the head left and change to state 6".*

compactly written as (17, 0, 1, L, 6).

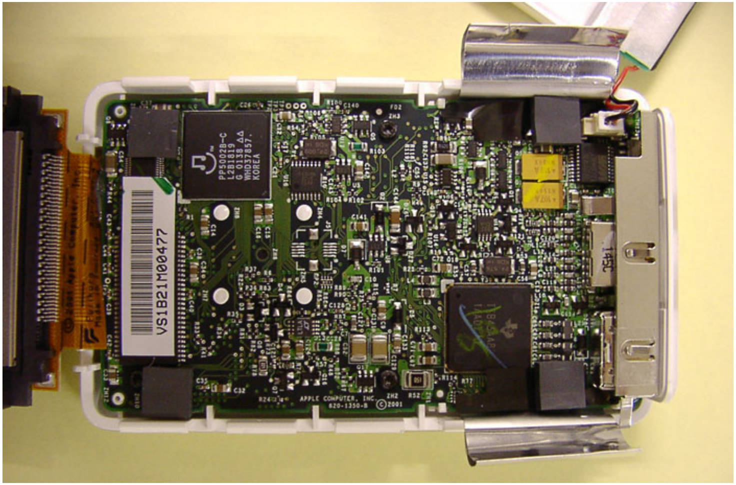
Turing's examples of machines are full of mistakes although in principle correct. This is benign for theory, but disastrous for practice.

A Turing machine given by a state diagram



Equality checker

iPod: Steve Jobs' fixed program, special purpose Turing machine



The concept of universal Turing machine (UTM)

Formal languages for mathematics are incomplete (Gödel) but programming languages are universal (Turing).

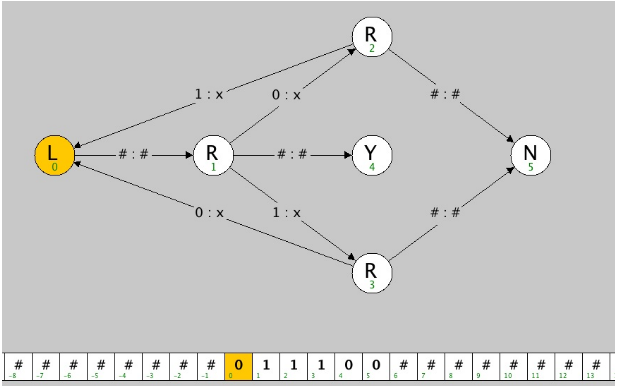
Turing's key idea: think of the description of a Turing machine itself as data.

Construct a specific Turing machine—called universal—that can simulate the behaviour of any Turing machine (including itself!).

Formally:

UTM(T,i) = T(i), for every Turing machine T and input i.

Coding—à la Gödel—of the equality checker

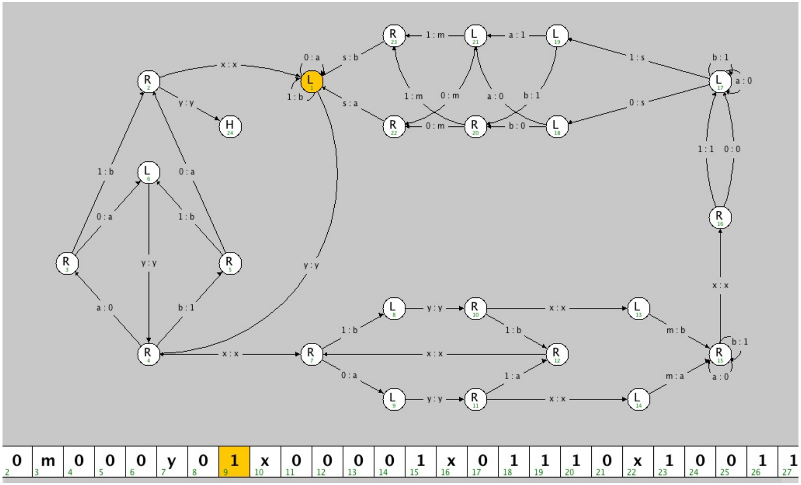


$0L|1R|2R|3R|4Y|5N|$

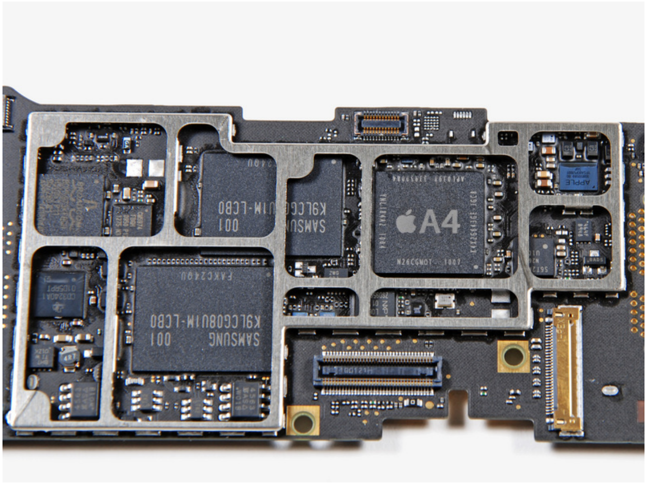
$01\#\#|120x|131x|14\#\#|201x|25\#\#|300x|35\#\#|$

$\#\#011100\#\#$

A universal Turing machine (UTM)

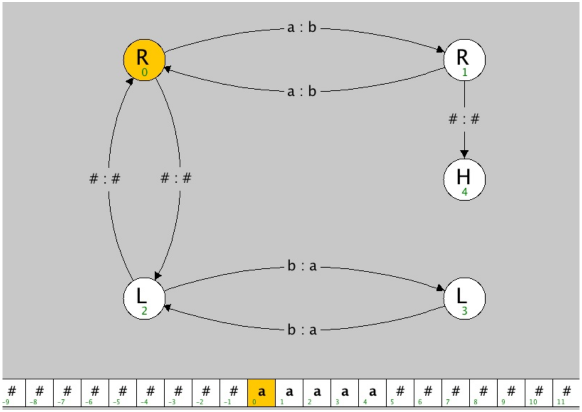


iPad: Steve Jobs' universal Turing machine can obey any program



and we know what an amazing variety of apps are available...

Does this Turing machine



halt on “aaaaa”?

But on *aaaaaaaaaaaaaaaaaaaaaaaaaaaa*?

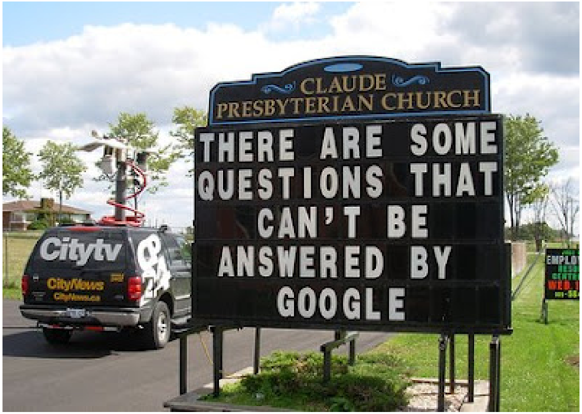
The halting problem

Can
any fixed Turing machine
decide whether
an arbitrary Turing machine
halts or not?

The halting problem in detail

Can we modify a UTM to obtain a Turing machine H that can inspect all other Turing machines T and inputs i and decide whether or not T will halt on input i, by returning “yes” or “no” and then halting itself?

Maybe Google can solve the halting problem?


CLAUDE PRESBYTERIAN CHURCH
THERE ARE SOME QUESTIONS THAT CAN'T BE ANSWERED BY GOOGLE
Citytv
CityNews
CityNews.ca

Turing (1936): The halting problem cannot be solved by any Turing machine

Assume that, similar to UTM(T,i), there exists a Turing machine H(T,i) deciding whether an arbitrary Turing machine T eventually halts on input i. Modify the program for H to H* as follows:

- if H(T,i) produces “yes”, then H*(T,i) goes into an infinite loop and never halts,
- if H(T,i) produces “no”, then H*(T,i)=H(T,i), i.e. it produces also “no” and stops.

Turing (1936): The halting problem cannot be solved by any Turing machine

Now apply H* to itself and input 0:

- if H* halts on 0, then H*(H*,0) never halts,
- if H* does not halt on 0, then H*(H*,0) halts,

so in both cases we have obtained a contradiction.

The proof is sound, but is this argument reasonable? After all, why should we allow H* to work on itself?

The [Chaitin] Omega number of a UTM

Fix a UTM U whose programs have an "halt" instruction and consider the set of all such programs p. What is the probability that one p (of any length), selected at random, halts?

This probability exists and is called the halting probability/the Omega number of U. In writing:

$$\Omega_U.$$

There exists one π, but infinitely many Omega's.

The double face of [every] Omega

1. The bits of Ω_U are in principle computable: run in parallel all programs and—like in a Feynman's path integral—add $2^{-\text{length}(p)}$ for each program p which halts on U.

2. No machine can compute more than finitely many scattered bits any Omega, because for almost all bits you need to wait an infinite time to get their value. For some Omega's, no bit can be computed.

Omega can be known of, but not known

Worse, every Omega is incompressible, i.e. its sequence of bits cannot be expressed more compactly than the number itself.

If you know Omega ...

then you know whether the Goldbach's conjecture—every even integer greater than 2 can be expressed as the sum of two primes—is true or not.

> *Program G enumerates every even integer $i > 2$ and checks whether i is a sum of two primes. Stop if an integer which is not a sum of two primes is found; otherwise, continue.*
>
> G stops if and only if the Goldbach's conjecture is false.

If you know the first N bits of Omega—where N is the size in bits of G—then you can decide whether G stops, so you solve the Goldbach's conjecture.

“I am the alpha and the omega, says the Lord, who is, who was, and who is to come” (Revelation, 1:8).

Omega is a “know it all” when it comes the truth of many mathematical statements.

Reason: with the first N bits of Omega we can calculate all halting programs of length at most N.

The complexity of some famous mathematical statements

[C. Calude, E. Calude, M. Dinneen, 2006–2012] One can evaluate the complexity of a problem according to the minimal number of Omega bits necessary to solve the problem:

- Fermat's last theorem is in class 1, i.e. less then 2^{10} bits are sufficient,
- The Riemann hypothesis is in class 3,
- The four colour theorem is in class 4.

Omega in CBS drama TV show Numb3rs

You might think that Omega is useless because it is unknowable
. . . but the first 64 bits of an Omega number calculated by
[Calude, Dinneen and Shu,2002] helped to solve "a crime".

season 5/episode 5/scene 6: `www.cbs.com/primetime/numb3rs`

Is there a probabilistic solution for the halting problem?

[Calude, Stay, 2008] There exists a Turing machine which stops on every input (T,i) and outputs either:

- "T halts on i", and in this case the result is correct, or
- "T does not halt on i", and in this case the result may be wrong, but with probability less than an arbitrarily small number.

Church-Turing Thesis

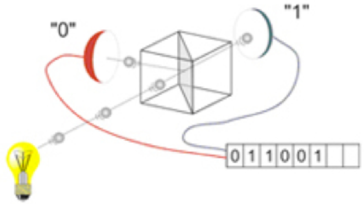
The Turing machine is just one way of capturing the essence of mechanical computation.

There have been many other approaches, such as Church's or your favourite computer, but they are all equivalent in the sense that they compute the same functions.

Are there more powerful calculating machines than Turing machines? Church-Turing Thesis states that the answer is negative. Hypercomputation challenges this view.

Hypercomputation revisited

Quantum randomness, produced with a simple optical experiment,



is highly-incomputable [Calude, Svozil, 2008].

Turing machines working with quantum random bits trespass the Turing barrier. But, how powerful are such machines?