Undecidability

So far in this course
- algorithms, efficiency, polynomial vs. exponential time

Bad news: some problems seem to have only exponential time algorithms (NP-complete problems)

Next in this course

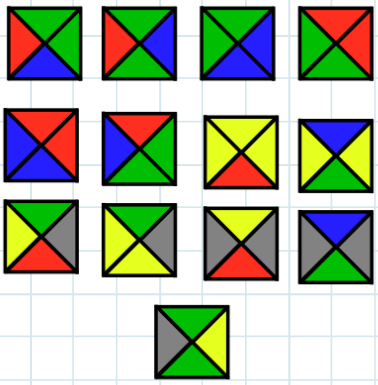
Worse news: some problems have no algorithm

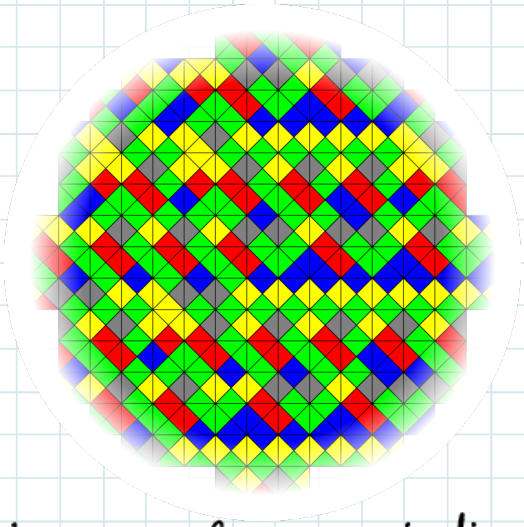
Examples

1. Program Equivalence: Given two programs, do they do the same thing. (i.e. produce same output for same input)

2. Tiling Given a finite set of tiles with coloured edges

Can you tile the plane? Colours must match when tiles touch. Can't rotate tiles. Can use infinitely many copies of each tile.





This set tiles the plane but only aperiodically.

The Halting Problem

Given a program, does it halt?

on all inputs?
some input?
specific input?

e.g.
```
while x ≠ 1 do
    x ← x-2
end
```

Halts iff x is an odd positive number

e.g.
```
while x ≠ 1 do
    if x is even then x ← x/2
    else x ← 3x+1
end
```

Does this halt on all inputs? No one knows.

Empirical evidence – the program has halted on every input tried.

"3x plus 1 problem"

we can convert a math statement about existence to a question about a program halting.

e.g. Does there exist a number x s.t. FOO(x) holds?

```
x ← 1
while not FOO(x) do
    x ← x+1
end
```

If the existence question is open then no one knows if the program halts.

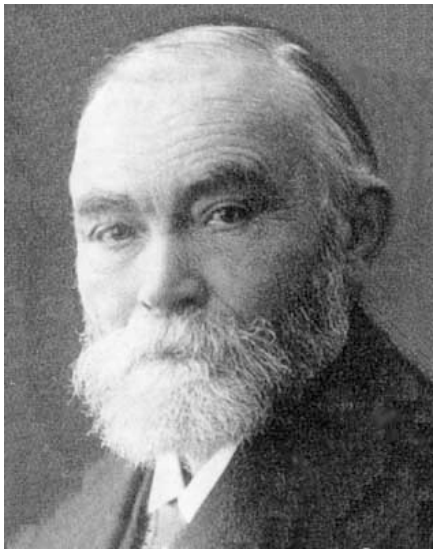
Defn A decision problem is undecidable if it has no algorithm.

Defn (more general, not just for decision problems)
A problem is unsolvable if it has no algorithm.

What is an algorithm?
as discussed before, any of the following, all equivalent
- Java/ C programs
- pseudo-code
- Random Access Machine
- Turing machine.

We will study some undecidable problems.

History of Undecidability



Gottlob Frege (1848 - 1925)

- tried to put all of mathematics on a firm foundation
- foiled by Russell's paradox

tried to formulate a set of axioms (axioms of Peano arithmetic, axioms of set theory) such that every true mathematical statement could be proved from the axioms using basic rules of logic

Russell's paradox:

Let S = the set of all sets that do not contain themselves.

(this is like the bibliography of bibliographies that do not list themselves)

Is S a member of itself ?

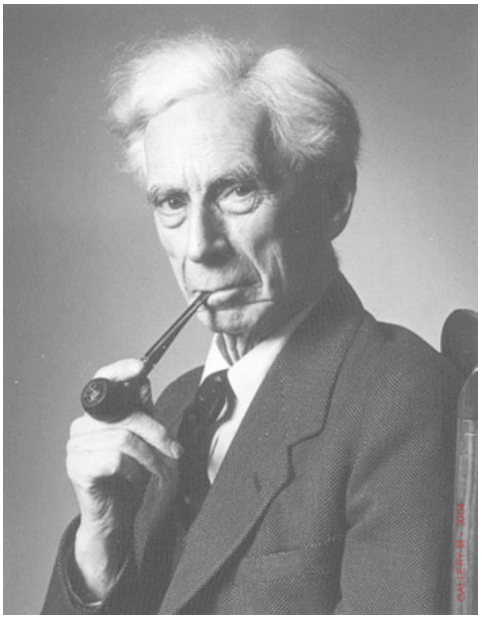
Either answer (YES or NO) gives a contradiction.

Once a system is powerful enough, you can get self-references that lead to contradictions

Set theory — what is a set? a collection of things?
Too powerful!

Moral (in this case) — we must define sets more carefully.

Application of Russell's paradox to computing —
If algs. were powerful enough to decide all problems we can decide something self-referential and get a contradiction.



Bertrand Russell (1872 - 1970)

FREGE ON RUSSELL'S PARADOX

Grundgesetze der Arithmetik, Vol. ii, Appendix, pp. 253-65

HARDLY anything more unfortunate can befall a scientific writer than to have one of the foundations of his edifice shaken after the work is finished.

This was the position I was placed in by a letter of Mr. Bertrand Russell, just when the printing of this volume was nearing its completion. It is a matter of my Axiom (V).[A] I have never disguised from myself its lack of the self-evidence that belongs to the other axioms and that must properly be demanded of a logical law. And so in fact I indicated this weak point in the Preface to Vol. i (p. VII). I should gladly have dispensed with this foundation if I had known of any substitute for it. And even now I do not see how arithmetic can be scientifically established; how numbers can be apprehended as logical objects, and brought under review; unless we are permitted—at least conditionally—to pass from a concept to its extension. May I always speak of the extension of a concept—speak of a class? And if not, how are the exceptional cases recognized? Can we always infer from one concept's coinciding in extension with another concept that any object that falls under the one falls under the other likewise? These are the questions raised by Mr. Russell's communication.

Solatium [*sic*] *miseris socios habuisse malorum.* I too have this comfort, if comfort it is; for everybody who has made use in his proofs of extensions of concepts, classes, sets,* is in the same position as I. What is in question is not just my particular way of establishing arithmetic, but whether arithmetic can possibly be given a logical foundation at all.

But let us come to the point. Mr. Russell has discovered a contradiction which may now be stated.

* Herr R. Dedekind's 'systems' also come under this head.

[A] Vol. i, § 3, § 20. Cf. also Frege's *Function und Begriff* for an explanation of the ideas used; especially pp. 9–10, 18. For any (first-level) function of one argument, there is some object that is its *value-range*; and two such functions are by Axiom (V) equal in value-range if and only if their values always equal for any given argument. Concepts (ibid., pp. 15–16) are functions whose values can only be the True or the False. For the value-ranges of concepts, which are called their *extensions*, the principle runs thus: Two concepts are equal in extension if and only if whatever falls under either falls under the other.

Nobody will wish to assert of the class of men that it is a man. p. 254] We have here a class that does not belong to itself. I say that something belongs to a class when it falls under the concept whose extension the class is. Let us now fix our eye on the concept: *class that does not belong to itself.* The extension of this concept (if we may speak of its extension) is thus the class of classes that do not belong to themselves. For short we will call it the class K. Let us now ask whether this class K belongs to itself. First, let us suppose it does. If anything belongs to a class, it falls under the concept whose extension the class is. Thus if our class belongs to itself, it is a class that does not belong to itself. Our first supposition thus leads to self-contradiction. Secondly, let us suppose our class K does not belong to itself; then it falls under the concept whose extension it itself is, and thus does belong to itself. Here once more we likewise get a contradiction!

What attitude must we adopt towards this? Must we suppose that the law of excluded middle does not hold good for classes? Or must we suppose there are cases where an unexceptionable concept has no class answering to it as its extension? In the first case we should find ourselves obliged to deny that classes are objects in the full sense. For if classes were proper objects, the law of excluded middle would have to hold for them. On the other hand, there is nothing 'unsaturated' or predicative about classes that would characterize them as functions, concepts, or relations. What we usually consider as a name of a class, e.g. 'the class of prime numbers,' has rather the nature of a proper name; it cannot occur predicatively, but *can* occur as the grammatical subject of a singular proposition, e.g. 'the class of prime numbers contains infinitely many objects.' If we were going to dispense classes from the law of excluded middle, we might think of regarding them (and, in fact, value-ranges generally) as improper objects. These could then not be allowed as arguments for all first-level functions. But there would also be functions that could have as arguments both proper and improper objects. At least the relation of equality (identity) would be a function of this sort. (An attempt might be made to escape this by assuming a special sort of equality for improper objects. But that is certainly ruled out. Identity is a relation given to us in such a specific form that it is inconceivable that various kinds of it should occur.)

Hilbert and the Entscheidungsproblem

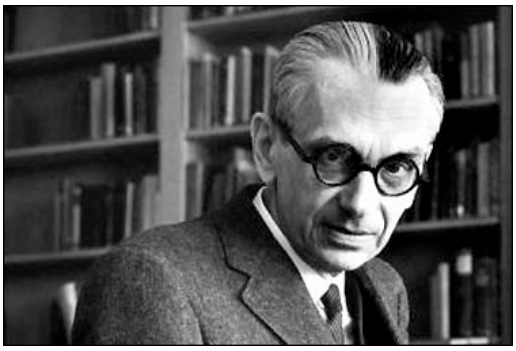
Hilbert asked (1928):

- Is mathematics **complete,** in the sense that every statement can be either proved or disproved?
- Is mathematics **consistent**, in the sense that one can never arrive at a contradiction such as 0=1 by a sequence of valid proof steps?
- Is mathematics **decidable**, in the sense that there exists a mechanical procedure that will determine the truth of a statement?

Goedel showed (1931):

- Every sufficiently powerful mathematical system is either **inconsistent** or **incomplete**
- such a system cannot be proved consistent within its own axioms
- third question remained open with "provability" substituted for "truth"

David Hilbert (1862 - 1943)



Kurt Goedel (1906 - 1978)

X

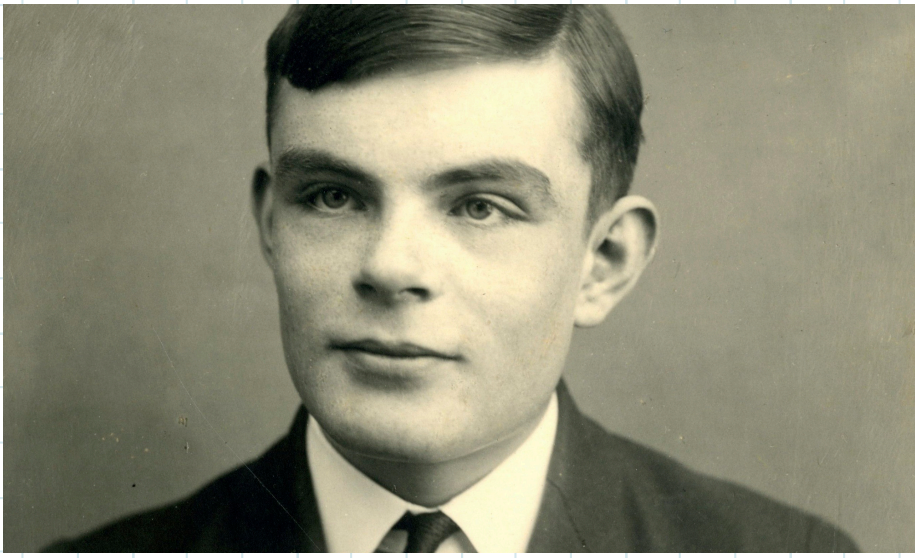
Hilbert's third question:

Is there a mechanical procedure that will determine the provability of a statement?

was answered by Turing in 1931:

No, there is no algorithm for this.

X



Alan Turing
1912 – 1954

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.

The first problem we will show is undecidable:

The Halting Problem Given an {algorithm / program} A and input w, does A halt on w?

Main idea used in the proof: self-reference to get a contradiction as in Russell's paradox.

Pf By contradiction

Suppose there is a program H for the halting problem

Input for H: program A and input w

Output: YES/NO, does A halt on w.

Can we have a program as input? Sure, compilers do it all the time.

Out of H, we can make a new program H'

Input for H': a program B

```
begin
    call H(B,B)
    if it returns NO then halt
    else loop forever
end
```

H' is like Russell's set S.

Russell's question (Does S contain S?) becomes

Does H' halt on input H'?

if YES
H' halts on input H'
then H(H', H') returns YES (by defn of H)
so (looking at code of H') H' on input H' loops forever
Contra

if NO
H' does not halt on input H'
then H(H', H') returns NO (by defn of H)
So (looking at code of H') H' on input H' halts
Contra.

We get a contradiction either way.
What's wrong?
Our assumption that program H exists must be wrong.
∴ there is no algorithm for the Halting Problem.

How to show that other decision problems are undecidable — reductions.

Thm Let P, Q be decision problems.
If P is undecidable and $P \leq Q$ (P reduces to Q)
then Q is undecidable.

(no restriction to poly. time)

If By contra.
Suppose there's an algorithm for Q.
$P \leq Q$ means we can use that alg. as a subroutine to make an alg. for P Contra.

More undecidable problems.

The Halt-no-input problem: Given a program A (no input), does A halt?

Thm Halt-no-input is undecidable

Pf. We will show

Halting Problem $\leq$ Halt-no-input

Suppose we have an algorithm X to decide Halt-no-input.

i.e. Input for X: program A

Output: YES/NO — does A halt?

Make an alg. for the Halting Problem

Input: program B, input w

Output: YES/NO — does B halt on input w.

```
begin
  Modify the code to make a program B' that
    has w hard-coded inside it and runs B on w.
  Call X(B')
  output the YES/NO answer.
end
```

Observe that B' halts (on no input) iff B halts on input w

(that's correctness, and we don't need to argue run-time)

Program Verification: Given a program and specification of inputs and corresponding outputs, does the program meet the specifications.

(so automatic program checking is impossible)

Note that we need a finite specification of inputs and outputs.

Thm Program Verification is undecidable.

Proof Halt-no-input $\leq$ Program Verification

Assume we have an algorithm V that decides Program Verification

Input: program + input/output specs.

Output: YES/NO — does the program meet the specs.

We want an alg. to decide Halt-no-input

Input: program A

Output: YES/NO — does A halt?

Here's the alg:

1. modify program A as follows [don't run it, just change the code]

program A' {
- read input (and ignore it)
- call A (with no input)
- output 1

2. Call V(A', specs: output is 1 for all input)
3. Output the YES/NO answer.

correctness

Claim This alg. correctly decides Halt-no-input.

i.e. V(A', output is 1 for all input) = YES iff A halts.

Pf. V outputs YES $\Rightarrow$ A' outputs 1 for any input $\Rightarrow$ A halts

A halts $\Rightarrow$ A' outputs 1 for any input $\Rightarrow$ V outputs YES.

Program Equivalence: Given two programs, do they do the same thing. (i.e. produce same output for same input)

Thm Program Equivalence is undecidable.

Pf. Halt-no-input $\leq$ Program Equivalence

The reduction is very similar too above

construct A' as above

construct program B: read input (and ignore)
output 1

Then ask if A' and B are equivalent

They are iff A halts.

Ex. fill in details.

Note idea: testing if a student program is correct is equivalent to testing if it matches the model solution program.

Other Undecidable Problems (no pfs)

1. Given a finite set L of 3×3 matrices over $\mathbb{Z}$ (integers) can the 0 matrix be written as a product (duplicates allowed) of matrices from L.

2. Recall context-free grammar

   Sentence → Noun-phase Verb-phrase
   Noun-phrase → Adjectives Noun
   Noun → car | house | flower ...
   Adjectives → Adjective Adjectives | ε
   Adjective → red | big | blue ...

   Used to specify syntax of programming languages and as a basis for parsing.

   Undecidable: given two context Free Grammars, do they generate the same set of strings?

3. Hilbert's 10th problem

Given a polynomial $P(x_1, \ldots x_n)$ in n variables with integer coefficients, does $P(x_1 \ldots x_n) = 0$ have a solution in integers?

You know how to solve eqns of form $ax^2 + bx + c = 0$ and you can check if there is an integer soln.

But $ax^2 + by + c = 0$ is NP-complete. !!

(i.e. does it have integer soln x, y)

http://dl.acm.org/citation.cfm?id=803627

The problem is in P if all exponents are 1 or if there is only 1 variable and the eqn is of the form $ax^k = c$

You could start trying integers. If there's a solution, you will eventually find it — But how do you know when to stop looking? There is no computable bound on size of solution.

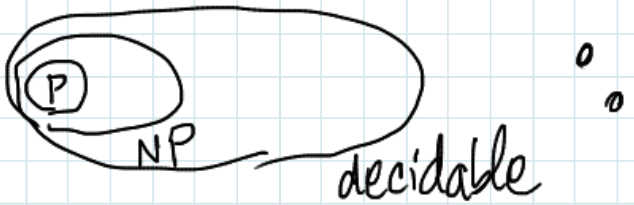
e.g. $x^2 = 991y^2 + 1$ has least soln x, y of ≥ 29 digits

$x = 379,516,400,906,811,930,638,014,896,080$

$y = 12,055,735,790,331,359,447,442,538,767$

The proof of undecidability took a long time: Hilbert posed the problem in 30's but final pf. was in 1971.

Are there interesting classes between P and decidable?



$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subsetneq$ decidable.

↑ poly. space

↑ runtime is $O(2^{poly(n)})$

known: $P \neq EXP$, but all else is open.

Is $P = PSPACE$?