

# Introduction to computability

Thursday, March 11, 2021 11:16 AM

Diophantine equation:

$$\rightarrow x^2 + y^2 = z^2$$

- give solutions in whole numbers
- give solutions in rational numbers

$$x^2 + y^2 = 1$$

$$3^2 + 4^2 = 5^2$$

$$5^2 + 12^2 = 13^2$$

$$x^3 + y^3 \stackrel{!}{=} z^3$$

$$x=1, y=0, z=1 \quad \times \text{ trivial}$$

$$\underline{x^y - y^x = 1}$$

$$3^2 - 2^3 = 1 \rightarrow \text{only solution}$$

MECHANICAL PROCEDURE??

The concept of algorithm was not yet formulated.

TODAY:

An algorithm is a process for computing something using finite resources, with a precise notion of primitive step; each

step involves a finite amount of information. SOMETIMES we require that the process halts (terminate).

1928 Hilbert & Ackerman asked for a mechanical procedure to determine if a formula of logic is valid or not.

VALID: True no matter how you interpret the variables.

$$\text{VALID} \left( (\forall x P(x) \Rightarrow Q(x)) \wedge P(a) \right) \Rightarrow Q(a)$$

ENTSCHEIDUNGSPROBLEME!

Both these problems are impossible!

(USA) ALONZO CHURCH 1935

(UK) ALAN TURING 1936

Need a notion of algorithm:

KURT GÖDEL → TOTAL RECURSIVE FUNCTIONS

ALONZO CHURCH → λ-CALCULUS

ALAN TURING → TURING M/C

A. MARKOV, E. POST, S. KLEENE, B. ROSSER . . . .

WE HAVE MANY EQUIVALENT SYSTEMS:

- (1) Partial recursive f<sup>ns</sup> (KLEENE)
- (2) FLOW CHARTS
- (3) "while" programs

- (4) ASSEMBLY LANGUAGE
- (5) RAM MACHINES
- (6) 2 STACK AUTOMATA
- (7) 2 COUNTER M/C
- (8) QUEUE M/C
- (9) 1000001 PROGRAMMING LANGUAGE
- (10) TURING MACHINES
- (11)  $\lambda$ -CALCULUS
- (12) PHRASE STRUCTURE GRAMMAR

## CHURCH - TURING THESIS

---

- (1) Algorithms can be expressed as a sequence of symbols and treated as data.
- (2) There is a universal machine.
- (3) There are unsolvable problems.

## THE HALTING PROBLEM:

ALONZO CHURCH - 1935

### BASIC ASSUMPTIONS:

Every algorithm can be coded as a program and represented as a string.

$P \rightarrow$  program  $\neq P \rightarrow$  source code

$P \rightarrow$  program,  $x \rightarrow$  input

$P(x) \rightarrow$  run  $P$  with  $x$  as input

$P(x, y) \rightarrow \dots \dots x, y, \dots$

PL has while loops, if-then-else  
and ;

What we want :  $H(\cdot, \cdot)$

$H(\#P, x)$   $\begin{cases} \rightarrow \text{TRUE if } P(x) \text{ halts} \\ \rightarrow \text{FALSE if } P(x) \text{ loops} \end{cases}$   
 $\hookrightarrow$  always halts.

**CLAIM : SUCH AN H CANNOT EXIST!**

Assume we have such an H.

Define a new program S:

$S(\#P)$  : if  $H(\#P, \#P)$  then loop  
else halts

S has source code  $\#S$

$S(\#S)$  : Does this halt?

if  $H(\#S, \#S)$  then loop else halt.

so if  $S(\#S)$  halts, H returns true,  
 we go to the then branch & loop (X)

if  $S(\#S)$  loops, H returns false  
 we go to the else branch & halts (X)

such an H cannot exist !

EXAMPLE

$COL(n) =$  if  $n=1$  then return 1  
 else if  $n$  is even  $COL(n/2)$   
 else  $COL(3n+1)$   
 $COL \rightarrow COLL AT 2$

$COL(11) \rightarrow COL(34) \rightarrow COL(17) \rightarrow (52)$   
 $\rightarrow 26 \rightarrow 13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow$   
 $8 \rightarrow 4 \rightarrow 2 \rightarrow 1 \quad HALT$

For  $n > 1$  does this always halt?

AS YET UNKNOWN!

---

RECAP Diagonalization

The set of infinite sequences of positive integers is not countable.

①	6	3	<u>37</u>	4	<u>29</u>	13	...
3	⑦	1	9	...	...	...	...
2	8	④	4	...	...	...	...
	⋮						

$\rightarrow 5 \ 3 \ 2 \ \dots$

The new list differs in its  $n^{\text{th}}$  place from the  $n^{\text{th}}$  element of the  $n^{\text{th}}$  list.

Hence it cannot be on the list!

The number of pairs of positive integers is countable.

<del>(1,1)</del>	<del>(1,2)</del>	<del>(1,3)</del>	(1,4)	...	.
<del>(2,1)</del>	<del>(2,2)</del>	<del>(2,3)</del>	(2,4)	...	.
<del>(3,1)</del>	(3,2)	(3,3)	(3,4)	...	.
(4,1)	(4,2)	(4,3)	(4,4)	...	.

(1,1) (2,1) (1,2) (3,1), (2,2), (1,3) - . . .