What is an algorithm?

We saw many models of computation. TM, Variants of TM, Enumerator etc. They are all equivalent in power. They all have unrestricted access to unlimited memory.

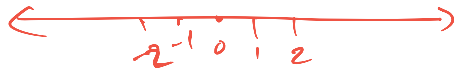
This is similar to programming languages. What we can compute with C, we can compute with PASCAL, and can compute with LISP etc.

Algorithms = What is decidable. Even though the computational models vary, they are of the same power.

Informally, Algorithm = Step-by-step instruction or "recipe".

In 1900, David Hilbert in ICM listed 23 problems. These are now known as Hilbert's problems. Many of these are important problems.

Hilbert's tenth Problem : To devise an algorithm to decide if a given polynomial has integer roots.



Given $P(x) = 7x^6 - 8x^4 + 13x^3 - x + 10$.

Is there an integer solution to $P(x) = 0$?

One approach: Check if $P(0)=0$, $P(1)=0$, $P(-1)=0$, $P(2)=0$, $P(-2)=0$, $P(3)=0$, $P(-3)=0$ ....

$$\boxed{\begin{array}{l} 6x^3yz^2 + 3xy^2 - x^3 - 10 = 0 \\ x=5,\ y=3,\ z=0 \end{array}}$$

This is a recognizer, not a decider. But can we convert this into a decider?

For a single variable polynomial $P(x)$, we can get an upper bound for x and convert it into a decider. (Problem 3.21) But not in the multivariate case.

Matiyasevich (1970)
(Building on Davis, Putnam and Robinson)

→ No algorithm exists for the 10th Problem.

In other words: Hilbert's 10^{th} is undecidable.

To show this, the basis was provided by Turing's work on computability theory

Church-Turing thesis:

$$\boxed{\left.\begin{array}{c}\text{Intuitive notion}\\ \text{of Algorithms}\end{array}\right\} = \left\{\begin{array}{c}\text{Turing machine}\\ \text{Algorithms}\end{array}\right.}$$

<u>How do we specify inputs?</u>

From now on, we will not focus on machine level or implementation level details.

✗ → At the level of state, transition function etc.

✗ → At the level of head movements, verifying and checking tape contents etc.

✓ → High level. Describe the algorithm in English.

Examples

GRAPH = $\{ w \mid w \in \{0,1\}^*$ is the adjacency matrix of an undirected graph $\}$

$= \{ w \mid |w|$ is a perfect square and $w_{ij} = w_{ji} \}$

CONNECTED-GRAPH = $\{ w \mid w \in \{0,1\}^*$ is the adj. matrix of a connected undirected graph $\}$

For this, we can use BFS or DFS. We reject if the input is not in proper form or if it is not connected.

PRIMES = $\{ w \mid w$ is a prime number $\}$

We can also ask questions about DFA/PDA/TM's etc. The entire description of the DFA or TM has to be encoded, and given as input.

One way to encode:

$\sqcup$ 0

a_1 00

a_2 000

a_3 0000

$q_{acc} = 0$

$q_{rej} = 00$

$q_0 = 000$

$q_1 = 0000$

And use 1's as delimiters

$L = 0$
$R = 00$
$S = 000$

$\delta(q_i, a_j) = (q_p, a_s, R)$

$1 0^{i+3} 1 0^{j+1} 1 0^{p+3} 1 0^{s+1} 1 00 1$

In this manner, we can encode the whole TM, using 0's and 1's. Similarly, we can encode DFA's or PDA's, and then ask questions about them.

→ Given TM M : Does M accept only palindromes?

→ Given TM M : Does M accept any palindromes?

→ Given TM M and w : Does M accept w?

A_{TM}

→ $A_{DFA} = \{ \langle B, w \rangle \mid B$ is a DFA that accepts $w \}$

→ A_{NFA} = Does a given NFA accept a given string?

→ $E_{DFA} = \{ \langle A \rangle \mid A$ is a DFA, $L(A) = \phi \}$

→ $EQ_{DFA} = \{ \langle A, B \rangle \mid A, B$ are DFA's, and $L(A) = L(B) \}$

→ $A_{CFG} = \{ \langle G, w \rangle \mid G$ is a CFG that generates $w \}$.