Church-Turing Thesis

Anything that is "computable" is computable by Turing Machine

If fact, we use this as the *definition of "computable"*.

Something is "computable" if it is computable by a Turing Machine

Implication :

Programs written in modern programming languages

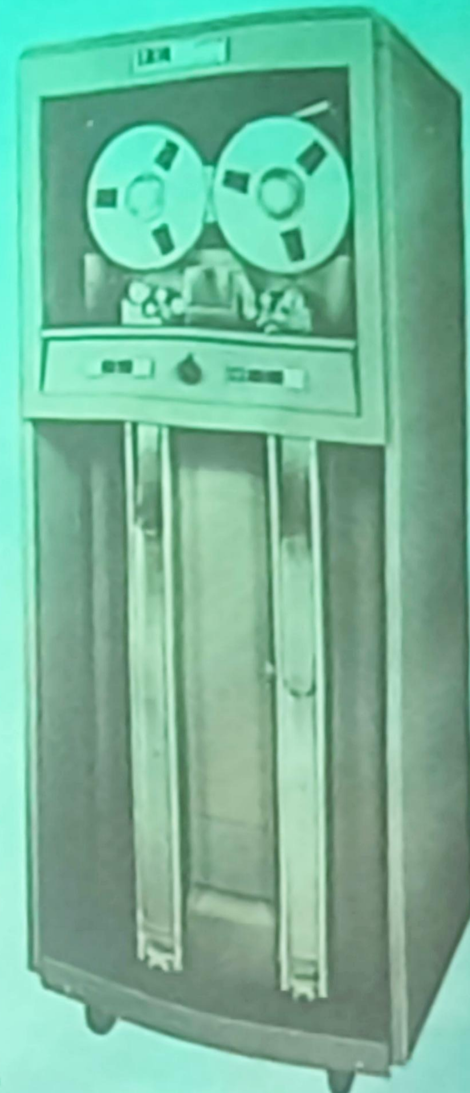
running on modern computers

Equivalent to Turing Machine

Random Access Memory

Random Access Machine (RAM)

Sequential Memory



RAM can be simulated by TM

The RAM can accept a language if and only if a TM can accept the language.

The TM does not halt on some input
if and only if the RAM does not halt on the input

The RAM can decide a language if and only if a TM can decide the language.

If the RAM halts on the input in N many steps
then the TM halts on the input in $O(N^2)$ many steps

Programs written in modern programming languages

Java C# php C++ python Perl

running on modern computers

Equivalent to Turing Machine

A language is r.e./Turing acceptable/Turing recognizable if and only if we can write a `program' s.t.

For all 'yes' instances, the 'program' halts and and produces an 'yes' answer.
For all 'no' instances, the program either halts and produces a 'no' answer
or never halts (falls into an infinite loop/ infinite recursion)

Given the TM we can write our program or given the program we can design our TM in such a way that the TM halts on some input if and only the program halts on that input.

Informally, an algorithm is a program that halts on all input.

Definition : An algorithm is a TM that halts on all input.

variable, function, recursion
branching, jumping, looping
arithmatic/logical operators

There is an algorithm (like one you find in algorithm courses using 'high level primitives')
for a computational problem

if and only if

the corresponding language is decided by a TM that always halts.
(i.e. the language is recursive/decidable/computable)

Definition (r.e. language) : *A language is recursively enumerable (r.e.) if it is accepted by some TM.*
(people also use "Turing acceptable", "Turing Recognisable" or simply "Recognisable" to mean "r.e.")

variable, function, recursion
branching, jumping, looping
arithmatic/logical operators

A language "L is r.e." means

equivalently

There is a TM s.t. on input x	There is a `program' using 'high level primitives' s.t. on input x
If $x \in L$ the TM halts and accepts	If $x \in L$ the program halts and accepts
If $x \notin L$ the TM either does not halt or halts and rejects	If $x \notin L$ the program either does not halt or halts and rejects

The TM halts **if and only if** The `program' halts

The program halts in N steps $\Rightarrow$ The `TM' halts in $O(N^2)$ steps

<u>Without loss of generality, the TM is a '0/1' TM</u>

Definition (recursive) : *A language is recursive if it is accepted by some TM which halts on all input.*
(people also use "Decidable" or "Computable" to mean "recursive")

A language "L is recursive" means

variable, function, recursion
branching, jumping, looping
arithmatic/logical operators

equivalently

There is a TM s.t. on input x	There is a \`program' using 'high level primitives' s.t. on input x
If $x \in L$ the TM halts and accepts	If $x \in L$ the program halts and accepts
If $x \notin L$ the TM halts and rejects	If $x \notin L$ the program halts and rejects

Algorithm

The TM / \`program' always halts

The program halts in N steps ⇒ The \`TM' halts in $O(N^2)$ steps

<u>Without loss of generality, the TM is a '0/1' TM</u>

Computable, Decidable/Undecidable problem

Recursive language
Recursive set

This is a Computational Problem

Given any graph find whether it contains a hamiltonian cycle

This is a Set or a Language

The set of all graphs containing a hamiltonian cycle

problem instances are encoded as string
wlog, the encoding is over alphabet {0,1}

Any TM can be simulated by a '0/1' TM.

Any TM can be simulated by a '0/1' TM.

Problem instances can be encoded as strings over {0,1}.

Uniqueness is not necessary - Suppose we encode integers as binary strings.
We can have arbitray no. of leading zeros.

Wlog we assume that each string encodes some problem instance.

Example : Suppose we want to encode a pair of natural numbers as a string over {0,1}

We *encode* the input pair of natural numbers (m,n) as $0^m 1 0^n 1$

What if input is NOT of the form $0^m 1 0^n 1$*? (like 1101)*

We define the encoding as,

If $x \in \{0,1\}^*$ is of the form $0^m 1 0^n 1$ then x denotes the pair (m,n)

otherwise x denotes the pair (1,1)

This is an arbitrary choice

Encoding a TM

It is convenient to call symbols 0, 1, and B by the synonyms X_1, X_2, X_3, respectively. We also give directions L and R the synonyms D_1 and D_2, respectively. Then a generic move $\delta(q_i, X_j) = (q_k, X_\ell, D_m)$ is encoded by the binary string

$$0^i 1 0^j 1 0^k 1 0^\ell 1 0^m. \tag{8.1}$$

A binary code for Turing machine M is

$$111 \text{ code}_1\ 11 \text{ code}_2\ 11 \cdots 11 \text{ code}_r\ 111, \tag{8.2}$$

The code for TM M is denoted as $\langle M \rangle$

Any string which is not of this format can be interpreted as $\langle \rightarrow\bigcirc \rangle$

Example 8.1 Let $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$ have moves:

$\delta(q_1, 1) = (q_3, 0, R)$,

$\delta(q_3, 0) = (q_1, 1, R)$,

$\delta(q_3, 1) = (q_2, 0, R)$,

$\delta(q_3, B) = (q_3, 1, L)$.

$\langle M, w \rangle$ encodes a pair (M, w) where M is a 0/1 TM and $w \in \{0,1\}^*$

Any string which is not of this format can be interpreted as $\langle \rightarrow\bigcirc, \epsilon \rangle$

Thus one string denoted by $\langle M, 1011 \rangle$ is

1110100100010100110001010100100 11

000100100101001100010001000100101111011

Note that many different strings are also codes for the pair $\langle M, 1011 \rangle$, and any of these may be referred to by the notation $\langle M, 1011 \rangle$.

TM as Computers of Functions

State does not matter

TM starts
content of tape = **input**

TM halts
content of tape = **output**

x ⟶ f(x)

If the TM *does not halt* on input x, then **f(x) is undefined** — *f() is a partial function*

Example : Suppose we want to compute a function $f : \mathbb{N}^2 \rightarrow \mathbb{N}$

We choose *input alphabet* of our TM $\Sigma = \{0, 1\}$

We *encode* the input pair of natural numbers (m,n) as $0^m 1 0^n 1$

Output is supposed to be of the form $0^{f(m,n)} 1$

This *encoding* is just an arbitrary choice.
Any other suitable encoding (like binary) is also fine

What if input is NOT of the form $0^m 1 0^n 1$*? (like 1101)*

Modify original TM M so that,
It first checks if input is in desired format
If 'yes' then proceed normally
else rewrite the tape content as *0101*
and proceed normally

If input is not in correct form, it is treated as (1,1)
(It is just an arbitrary choice)

TM as Computers of Functions

Partial Recursive Function

input → TM may not always halt → output

Total Recursive Function

input → Algorithm (always halts) → output

7.4 A *recursive function* is a function defined by a finite set of rules that for various arguments specify the function in terms of variables, nonnegative integer constants, the *successor* (add one) function, the function itself, or an expression built from these by composition of functions.

7.7 A function is *primitive recursive* if it is a finite number of applications of composition and *primitive recursion*† applied to constant 0, the successor function, or a projection function $P_i(x_1, \ldots, x_n) = x_i$.

† A primitive recursion is a definition of $f(x_1, \ldots, x_n)$ by

$$f(x_1, \ldots, x_n) = \text{if } x_n = 0 \text{ then}$$

$$g(x_1, \ldots, x_{n-1})$$

$$\text{else}$$

$$h(x_1, \ldots, x_n, f(x_1, \ldots, x_{n-1}, x_n - 1))$$

where g and h are primitive recursive functions.

$L_d = \{w \mid w \notin L(M) \text{ where } w \text{ encodes TM } M\}$ is NOT r.e.

$$\{0,1\}^* = \{w_1, w_2, w_3, \ldots\} \ (w_i \text{ s are distinct})$$

For each w_i there is some M_i s.t. w_i encodes M_i
(if $i \neq j$ then $w_i \neq w_j$ but it is possible that $M_i = M_j$; encoding is not unique)

$$\begin{array}{c|cccc}
 & w_1 \text{ encodes } M_1 & w_2 \text{ encodes } M_2 & w_3 \text{ encodes } M_3 & \cdots \\
\hline
w_1 & 0 & 1 & 1 & \cdots \\
w_2 & 1 & 0 & 0 & \cdots \\
w_3 & 1 & 1 & 1 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

$(i,j)^{th}$ entry is 1 iff $w_i \in L(M_j)$

$w_i \in L_d$ iff $(i,i)^{th}$ entry is 0 i.e. $w_i \notin L(M_i)$

$L_d \neq L(M_i)$ for all i

Every TM is encoded by some string
For any TM M, $M = M_i$ for some i
L_d is not accepted by any TM