Models of Computation

Tuesday, March 16, 2021 11:16 AM

TURING MACHINE:

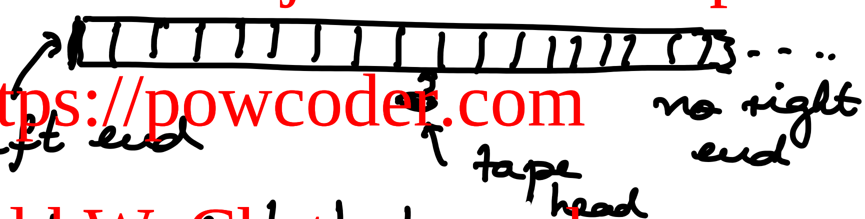
At each step there is a finite amount of "information processing"

9-tuple

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, a, r)$$

Q: finite set of states

A TM has a tape divided into cells



$\Sigma \rightarrow$ input alphabet

$\Gamma \rightarrow$ tape alphabet $\Sigma \subsetneq \Gamma$

$\sqcup \rightarrow$ blank $\in \Gamma \setminus \Sigma$

$\vdash \rightarrow$ turnstile left end marker

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$$

e.g. $\delta(q, a) = (q', b, L)$

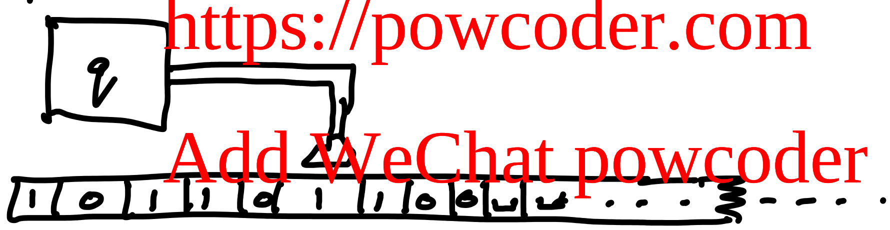
means read "a" in the tape cell where the head is positioned, change state from q to q', erase a and write b, then move one step to the left.

$s \in Q$ start state

$a \in Q$ accept state

$r \in Q$ reject state

A <u>configuration</u> is a description of what is on the tape, where the head is positioned and the current state.



$10\ 110q11\ 00$

This says it is looking at the symbol to its right.

Start : $s\,11011$

always looks like this, obviously with different words.

YIELDS

$u\,a\,q_i\,b\,v \longrightarrow u\,q_j\,a\,c\,v$

if $\underline{\delta(q_i, b) = (q_j, c, L)}$ Instruction

A TM ~~program~~ is a sequence of <u>instructions</u>.

A TM can do one of <u>three</u> things:

(1) it can reach ~~an~~ the accept state and stop

(2) it can reject ~~a~~ the reject state and stop

(3) it can <u>loop forever</u>.

M <u>accepts</u> w if there is a <u>finite</u> sequence of configurations $C_1, C_2 \cdots C_k$ s.t.

1. C_1 is sw
2. Each C_i <u>yields</u> C_{i+1}
3. C_K is an accept config

$u\,a\,v \quad u, v \in \Gamma^*$

NEW PHENOMENON:

looping forever

$L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$

L is Turing recognizable if $\exists$ TM M such that

$$L = L(M)$$

If $w \notin L(M)$: it may be rejected or it may loop forever.

L is Turing decidable if $\exists$ TM M s.t. $L(M) = L$

and $\forall w \in \Sigma^*$ M halts on w.

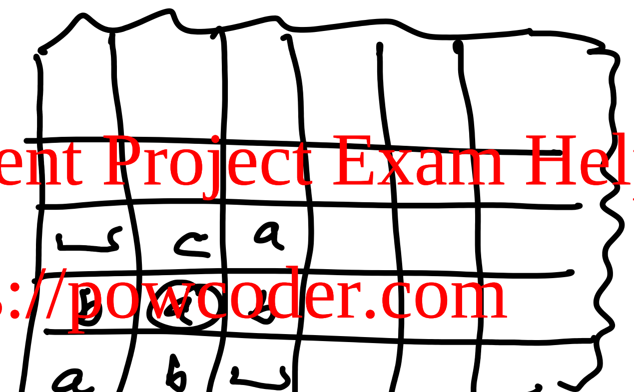
We say L is computably enumerable (CE) if $L = L(M)$

We say L is computable or decidable if $L = L(M)$ for some TM M and M always halts on every word.

Other models of computation:

(I) Variations on the TM model:

(a) 2 tapes with 2 heads
(b) k tapes with k heads
(c) 2-dimensional "tape"



(d) n-dimensional hypertape.

BASIC POINT: All these can be simulated by a primitive TM.

(e) Non-deterministic TM

given a particular tape symbol and state we give it finitely

... to what it can do

many options as to what it can do next. Acceptance means at least one sequence of choices leads to the accept state.

This can be simulated by a deterministic TM.

ND TM = TM = k-tape TM = n-dim TM

The basic TM may be much less efficient.

An algorithm that always terminates in time $O(n^k)$ is called a polynomial-time algorithm.

PTIME

on a basic TM.

An alg that terminates on a Nondet TM in time $O(n^k)$ is called an NP algorithm.

Is $P = NP$?

I can solve this in polynomial time

I can check an answer in polynomial time.

KNOWN $P \subsetneq EXPTIME$

KNOWN $PSPACE = NPSPACE$

RANDOM ACCESS MACHINE : RAM

= TM (ASSEMBLY LANGUAGE)

WHILE PROGAMS:

$x, y, z \cdots$ variables

integer constants

arithmetic expressions $a_1 + a_2 \mid a_1 * a_2 \cdots$

boolean expression

skip | if a then c_1 else c_2 | $x := a$ |

$C_1; C_2$) while a do c

TURING COMPLETE:

Can simulate TM and can be simulated by TM

λ- CALCULUS $\equiv$ TM

↳ core of functional programming

$\equiv$ POST PRODUCTION SYSTEMS

$\equiv$ Phrase - structure grammars

$\equiv$ Markov algorithms

$\equiv$ Combinatory Logic

$\equiv$ 2 stack PDA

$\equiv$ 2 counter machines

DFA + 2 updatable integer vars.
