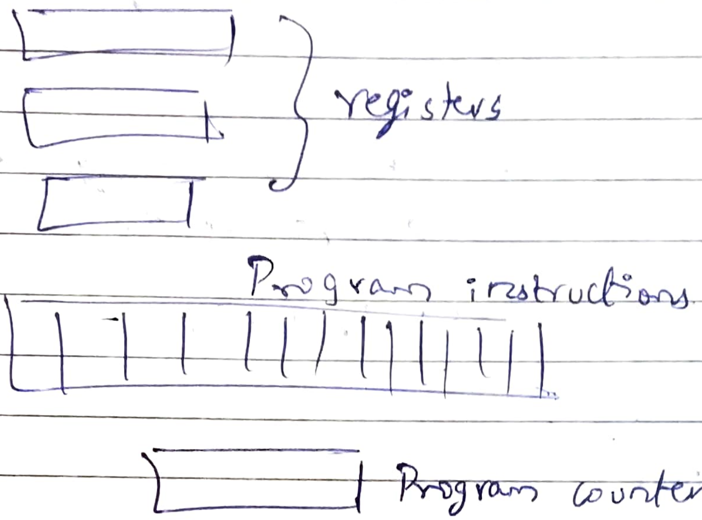
- RAM Model of Computation.

RAM → can access memory randomly

Model (same as taught in CS223)
(just writing keywords)

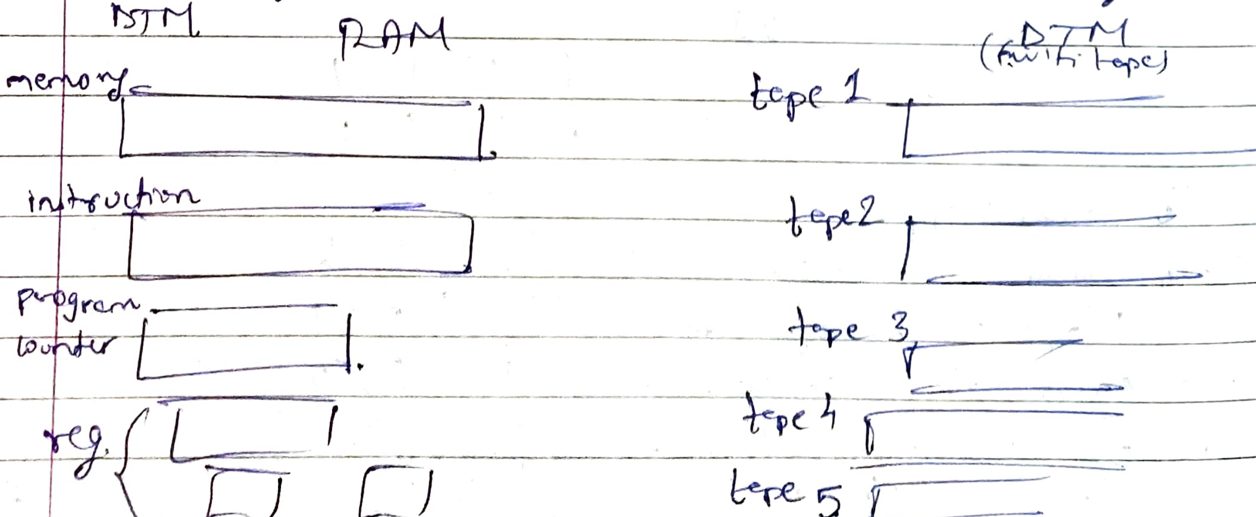


instructions can be like "Load 0X123 reg3"
"save reg 3 0X13.."
"Subtract 0X12 0X45, B"

We will prove RAM has equal power as multi-tape DTM and compute partial recursive functions

"Don't mind the type of tapes we use in DTM, its just to reduce complexity of our proof".

Part1
Simulating RAM Model of computation using multitape DTM



tape 1, 2 store memory, program instruction as.

0 & ← → # 1 & ← → # 2 & ← → # ...

format #(address) & ← → #.

delimiter   content.

tape 3 is counter similar to program counter. adds one after each instruction implementation (can be simulated)

tape 4 is tape for storing register content.

← reg 1 → # ← reg 2 → # ← reg 3 → # ...

tape 5 is used for storing memory address.

So now how to simulate.
Let program instruction is LOAD 0 2
(Assume 0, 1, 2... are tape 1 addresses).

So 0th address content is loaded in reg 2 space
how can it be done [copy 0 in tape 5]
(start from starting of tape 1, compare the address and search for asked content)
then in ~~ta~~ some other tape 6 store 2.
traverse in tape 4 and in reg 2 position
~~load~~ copy contents from memory to reg 2.

Then finally incrementing Program counter can also be done.

So one iteration of an instruction can be performed

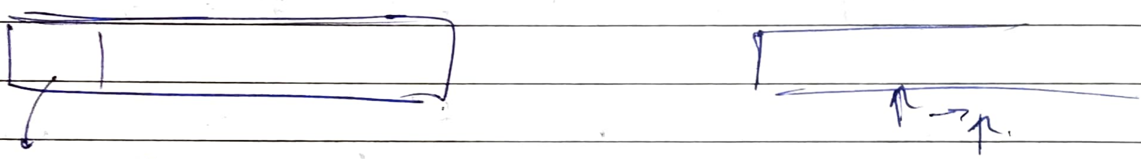
And other instructions like subtract, add can also be implemented by having another tape and computing there.
~~\$~~ We can use any number of tapes, as per our convenience.

Now Part 2

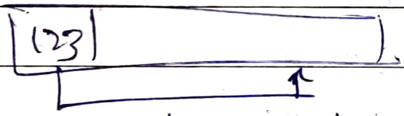
Can we compute single tape DTM by using RAM model

RAM Model

single tape DTM



let its first byte store address.
So like when we are moving tape heads in single tape DTM, say it is at location 123.
So 123 is stored as address.



points somewhere corresponding to 1, 2, 3.

Now we have transition in DTM
Let it be like.

$$\delta(q_0, 0) = (q_2, 3, L)$$

q_0	0	q_2	3	-1

$L \to -1$
$R \to +1$

For each transition we can have a entries in table

So using these values we can change (+, -1) address, read, write in different address on memory.

We can also store ~~a~~ this table in memory.

So DTM and RAM have same computation power and both can compute partial recursive functions.

This also ~~again~~ follows ~~proves~~ church turing thesis.

GRAMMARS

Grammar G is defined as

$G(V, T, P, S) \rightarrow S \in V$ (a special symbol)

V → variables
T → terminals
P → productions/rules

V, T, P are sets

$V \cap T = \phi$

Each production is of the format

$$P : (V \cup T)^{+} \longrightarrow (V \cup T)^{*}$$

Let

$V = \{ v_1, v_2, \ldots, v_k \}$
$T = \{ t_1, t_2, t_3, \ldots, t_j \}$

$+$: 1 or more
$*$: 0 or more

An example to understand better.

$$G(\{S\}, \{a, b\}, \{S \rightarrow aSb, S \rightarrow ab\}, S)$$

S is start symbol.

note: double arrow while deriving. single while production

first production using $S \rightarrow aSb$: $S \Longrightarrow aSb$

$\Downarrow$ 2nd production

$aaSbb$ → sentential form

$\Downarrow$ 3rd production

$aaaSbbb$

$\Downarrow$ 4th production

$aaaabbbb$ → (sentence)

This whole process is called derivation.

Here in example.
In a string/form, if ~~S~~ S is present then it is called sentential form, and if only a, b are there then it is called sentence.

General if form contains $V \cup T \rightarrow$ sentential form
or $T \rightarrow$ sentence

$S \rightarrow aSb.$

~~$(V \cup T)^*$~~

$(V \cup T)^+ \rightarrow (V \cup T)^*$

$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \ldots \Rightarrow \alpha_i \Rightarrow \ldots \Rightarrow \alpha_r$

we can say
" α_i is derived from α_j ($j = 1$ to $j = i-1$) in G "

For every string ~~is~~ derived from G (V, T, P, S)
sums up ~~consists~~ its language (L(G))

$$L(G) = \{ w \mid \overset{\text{sentence}}{w} \in T^* \text{ and } \underbrace{S \overset{*}{\Rightarrow} w}_{\text{(using G)}} \}$$

Language of grammer is L(G)

interpreted as w derived from S using ~~one~~ zero or more productions.

$L = \{ w_1, w_2, \ldots \}$

$L(G_1) = L(G_2) = L$ whenever G_1 and G_2 are equivalent.

~~In ref~~ $L = \{ w \mid w \text{ is } a^i b^i, i \geq 1, i \text{ is int} \}$

or

$L = \{ a^i b^i \mid i \geq 1, i \text{ is int} \}$

In reference to our example and L we just saw (last 3 line previous page)

$L(G) = L$ when $L(G) \subseteq L$
$L \subseteq L(G)$

- If we have certain type of production

$$P: V \to (V \cup T)^*$$

like in eg. $S \to aSb$, $S \to ab$
or $V_1 \to V_2 t_3 V_5$

So such grammers are called context free grammers (CFG)
A language L is equal to L(G) and G is CFG then L is called context free language (CFL)

$$L(G) = \{a^i b^i \mid i \geq 1, i = int\}$$ is CFL

eg. $V = \{S\}$
$T = \{(,)\}$
P: i) $S \to (S)$
ii) $S \to SS$
iii) $S \to \varepsilon$

combinely written as $S \to (S) \mid SS \mid \varepsilon$
(or symbol)

($\varepsilon \to$ empty string)

derivation

$$S \Rightarrow (S) \quad \text{using i}$$
$$\Rightarrow ((S)) \quad \text{using i}$$
$$\Rightarrow (((S))) \quad \text{using i}$$
$$\Rightarrow (((SS))) \quad \text{using ii}$$
$$\Rightarrow ((((S)S))) \quad \text{using i}$$
$$\Rightarrow ((((S)(S)))) \quad \text{using i}$$

$\Rightarrow (((() (S))))$ using iii

$\Rightarrow (((() ())))$ using iii

This is going to generate sentences which are properly parenthesized.

$$S \Rightarrow SS$$
$$\Rightarrow SSS$$
$$\Rightarrow SSSS$$
$$\Rightarrow ()SSS$$
$$\vdots$$
$$\Rightarrow ()()()()$$

another sentence.

e.g.

$S_1 \rightarrow 0S_11 \mid \varepsilon$ (→ or symbol.)

$S_2 \rightarrow 1S_20 \mid \varepsilon$

$S \rightarrow S_1 \mid S_2$

$T = \{1, 0\}$ $V = \{S, S_1, S_2\}$

$$S \Rightarrow S_1$$
$$\Rightarrow \cancel{S_1}\ 0S_11$$
$$\Rightarrow 00S_111$$
$$\Rightarrow 000111$$

so 0^i1^i & $i \geq 0$ format.

$L(G)$ is $\{0^i1^i \mid i \geq 0\}$ or $\{1^i0^i \mid i \geq 0\}$

(You cannot write directly, you need to prove this by induction)

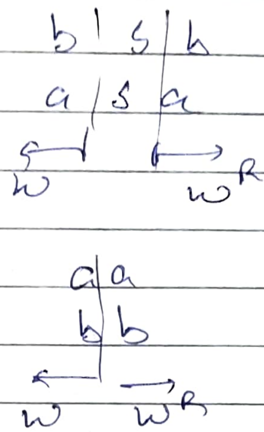
eg $S \to aSa \mid bSb \mid aa \mid bb$
(i) (ii) (iii) (iv)

$V = \{S\}$
$T = \{a, b\}$

$$
\begin{aligned}
S &\Rightarrow aSa \\
&\Rightarrow aaSaa \\
&\Rightarrow aabSbaa \\
&\Rightarrow aabaSabaa \\
&\Rightarrow aababbabaa
\end{aligned}
$$

$L = \{ w w^R \mid w \in T^+ \}$

for



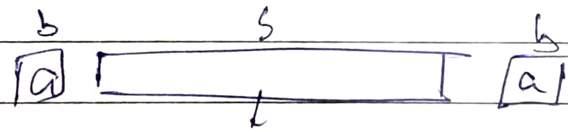
elaborate it for proofs.

so $L(G) \subseteq L$.

Now proving $L \subseteq L(G)$

can we generate any string of L in L(G)

By Induction



in induction hypothesis

if a in block ∃ and big middle block has elements then

$S \Rightarrow aSa$

if b in block and big middle block has elements then

$S \Rightarrow bSb$

if a/b in block, and no middle elements then

$S \Rightarrow aa / bb$

so $L(G) = L$