☆ Applications of Regular Expression

① Regular Expression give picture to a pattern to recognize the pattern. It is the medium of choice of application that search for patterns in text.

② Regular Expressions are then compiled behind into deterministic / Non deterministic automata which are then simulated to produce a program that recognize a pattern.

(i) Regular Expression in UNIX : UNIX regular expression allows us to write character class, to represent large set of characters.

(ii) Lexical Regular Expression in Lexical Analyser.

Application of RE is specifying component of a compiler i.e Lexical Analyser

(iii) Finding Patterns in Text : Regular expression has been found useful in description of classes of patterns in text.

☆ Properties of Regular Languages :

i) Proving Languages not to be Regular :

We have established that the class of languages known as the regular languages has at least four different descriptions.

They are the languages accepted by DFA's by NFA's and by ϵ-NFA's they are also the

languages defined by regular expressions. Not every language is a regular language.

Eg: $L = \{ w \in (0,1) \mid w$ contains equal number of 0's and 1's $\}$

Non-Regular Languages :-

Suppose there are 'n' no. of objects and 'm' no. of boxes, where the number of objects are greater than the number of boxes. In this case if all 'n' objects are to be placed into m boxes then atleast one box will have more than one object.

Theorem :- (Pumping Lemma for Regular Languages)

Let 'L' be a regular language then there exists 'n' such that for every string 'w' in L such that $|w| \geq n$, we can break w in the form of $w = xy^kz$ such that :

1. $y \neq \epsilon$
2. $|xy| \leq n$
3. For all $k \geq 0$, the string xy^kz is also in L

Repeating y any number of times or deleting it keeps the resulting string in the language 'L'

Proof :-

Suppose 'L' is regular, then $L = L(A)$ for some DFA

Suppose A has n states, Now, Consider string w of length n or more

$w = a_1, a_2, \ldots a_m$

where $m \geq n$ and a_i is an input symbol.

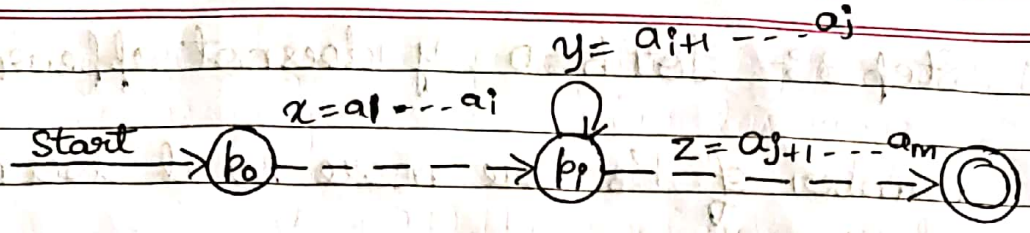
For $i = 0, 1, \ldots n$, define a state p_i with a transition function $\delta(q_0, a_1, a_2 \ldots a_i)$

Since there are n different states we can find two different integers i and j, such that $p_i = p_j$

Now, we can break w in the form of $w = xy^kz$ as follows

1. $x = a_1, a_2 \ldots a_i$
2. $y = a_{i+1}, a_{i+2}, \ldots a_j$
3. $z = a_{j+1}, a_{j+2}, \ldots a_m$

Here x can be empty in case $i = 0$ and and z can be empty with $j = n = m$, i is strictly less than j



Here automata 'A' receives xy^kz for any $k \geq 0$

If $k=0$, then automata goes from state p_0 to p_i on input x. Since p_i is also p_j, automata goes from p_i to accepting state.

If $k>0$, automata goes from p_0 to p_i on x and p_i to p_j, k times and then to the accepting state.

Eg:- 1) Show that $L = \{w/w^R / w \in (0+1)^*\}$ is not regular.

Sol:- Step 1:-

Suppose L is Regular and n is no. of states.

$$\underbrace{1 \cdots 10}_{n} \quad \underbrace{\cdots 00}_{n} \quad \underbrace{\cdots 01}_{n} \quad \underbrace{\cdots 1}_{n}$$

w $\qquad$ w^R

Step 2:- Split the given string in the format $w = xy^kz$

For $k=1$, the given string does not change and the condition is satisfied.

Step 3 :- For $k=0$, y does not appear and number of 1's are more so it does not satisfy the given condition

∴ The given language L is not Regular.

Eg 2) Show that $L = \{a^n b^n / n \geq 0\}$ is not Regular.

classmate
Date
Page
15/12/2021

Equivalence and Minimization of Automata

* For any DFA, we can find an equivalent DFA that has minimum number of states.

Here the goal is to understand when two distinct states p and q can be replaced by a single state that behaves like p and q.

If two states are not equivalent then we say that they are distinguishable. It is a recursive discovery of distinguishable pairs in a DFA.

Eg: Minimise the following Automata:

δ	0	1
→ A	B	F
B	G	C
* C	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

	0	1
AB	BG	FC
AD	BC	FG
AE	BH	FF
AF	BC	FG
AG	BG	FE
AH	BG	FC
BD	GC	CG
BE	GH	CF
BF	GC	CG
BG	GG	CE
BH	GG	CC
DE	CH	GF
DF	CC	GG
DG	CG	GE
DH	CG	GC
EF	HC	FG
EG	HG	FE
EH	HG	EC
FG	CG	GE
FH	CG	GC
GH	GG	EC

X- Not Equivalent

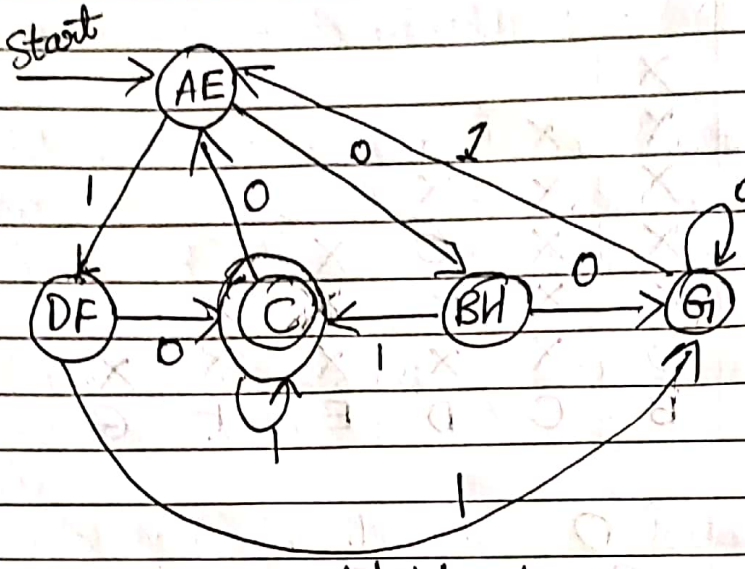
B	X						
C	X	X					
D	X	X	X				
E	O	X	X	X			
F	X	X	X	O	X		
G1	~~X~~	X	X	X	~~X~~	X	
H	X	O	X	X	X	X	X
	A	B	C	D	E	F	G1

	0	1
AE	BH	FF
AG1	BG1	FE
BH	G1G1	CC
DF	CC	G1G1
EG1	HG1	FE

* Mark the pairs AG1 and EG1 with 'X', the pairs AE, BH, DF are not marked and are referred to as indistinguishable states or equivalent states. The other states C and G1 are distinguishable states or Not Equivalent states.

The unmarked states AE, BH & DF are indistinguishable, C & G1 are distinguishable.

	0	1
→ AE	BH	DF
* ~~AG~~	AE	C
BH	G1	C
DF	C	G1
~~E~~G1	G1	AE



Unit - 4

Push Down Automata (PDA) :-

Finite automata is used to recognize regular languages, DFA/NFA is not powerful enough to remember anything.

Automata may require additional amount of storage which will be used to store the data

Eg:- Parenthesis matching problem, if we encounter (it is required to store left paranthesis on stack and as we encounter) pop corresponding right ( ) Here the stack is used to remember scanned symbols

Eg:- Consider a language $L = WCW^R$ where W^R is reverse of W

Here C is middle of strings
Machine will be in the start state q0 in this

state keep on pushing all input symbols onto the stack, Till C is encountered. Immediately after reading C, change the state to q_1 means it has passed middle of string. For each character encountered after mid point there will be corresponding character on the stack. So in state q_1 after scanning input symbol compare it with most recently pushed symbol on the stack. If they are same, discard both the symbol & scan next symbol. Repeat the process and remain in state q_1. Once the last symbol in the input is matched with the symbol on the stack. Finally stack will be empty. Once empty it is evident that string is Palindrome.

Def: A Push down automata is a seven tuple notation

$$M = (Q, \Gamma, \Sigma, \delta, q_0, Z_0, A)$$

where,

Q - Set of Finite States
Σ - Set of input alphabet
F - Set of stack alphabets
δ - Transition function
q_0 - Start State
Z_0 - $Z_0 \in F$ is initial symbol on stack
A - Subset of Q ($A \subseteq Q$) is set of accepting states.

Graphical Representation of PDA

* PDA is graphically represented using set of states of PDA corresponds to Nodes in a graph and are represented using circles.

A start state of PDA is denoted by an arrow labelled start

* Nodes of the graph represented by two concentric circles are the final states of PDA

If there is a transition of the form

$$\delta(P, a, Z_0) = (q, aZ_0)$$

In this transition, on scanning input symbol a & top of stack is Z_0, machine is in state p it transits to the state q and a is pushed onto the stack above Z_0

1) Design a Push-Down Automata (PDA) to accept the language $L(M) = \{WCW^R / W \in (a+b)^*\}$

Where W^R is reverse of W

General Procedure

1) Push all symbols on to the stack till the encounter C, Once we pass middle string, if string is Palindrome for each scanned input symbol there will be corresponding symbol on the stack. Finally there is no input and stack

is empty. It can be said that given string is Palindrome.

Input Symbols are either a or b, q_0 is the initial state Z_0 is initial symbol on the stack. In q_0 when the top of stack is Z_0 whether input symbol is a or b Push it on to the stack and remain in q_0 state

Step 1 :- $\delta(q_0, a, Z_0) = (q_0, a, Z_0)$
$\delta(q_0, b, Z_0) = (q_0, b Z_0)$

$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, a, b) = (q_0, ab)$
$\delta(q_0, b, a) = (q_0, ba)$
$\delta(q_0, b, b) = (q_0, bb)$

Stack may contain a or b irrespective of what is input, keep on pushing all symbols on stack till C is encountered.

Step 2 :- Input symbol is C

If input symbol is C top of the stack may be a or b

Another possibility is in q_0 state first symbol can be C In this case w is null string and Z_0 is on top of the stack, Symbol in the middle of string is C, Change the state to q_1 and do not alter contents of stack

$\delta(q_0, C, Z_0) = (q_1, Z_0)$

$$\delta(q_0, c, a) = (q_1, a)$$

$$\delta(q_0, c, b) = (q_1, b)$$

Step 3 : Input symbol can be either a or b

For each input symbol,
There should be corresponding input symbol same as input symbol on the stack. If input symbol is same as symbol on stack remain in state q_1 and delete that symbol from stack & repeat the process.

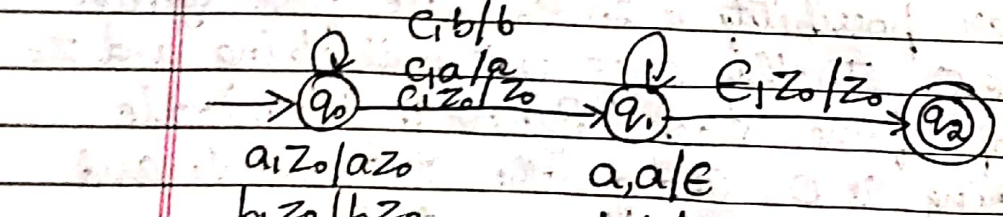
$$\delta(q_1, a, a) = (q_1, \epsilon)$$
$$\delta(q_1, b, b) = (q_1, \epsilon)$$

Finally in state q_1 if string is Palindrome there is NO input symbol to be scanned the stack is empty i.e stack should contain Z_0.

Now, change the state to q_2 and donot alter the contents of stack.

$$\delta(q_1, \epsilon, z_0) = (q_2, z_0)$$

Step 4 : In state q_1



classmate
Date 28/12/2021
Page

Deterministic PDA.

Consider a PDA, $M = (Q, \Gamma, \Sigma, \delta, q_0, Z_0, A)$
The PDA is said to be deterministic if $\delta(q, a, Z_0)$ has only one element and $\delta(q, \in, Z_0)$ is not empty Both the conditions should be satisfied for PDA to be deterministic. If and only one of the condition is false then PDA is non deterministic.

Eg: Is the PDA to accept a language

$L(M) = \{WCW^R \mid W \in (a+b)^*\}$ is deterministic

Sol:

$\delta(q_0, a, Z_0)$ has only one element, the 1st condition is satisfied. If there is an $\in$ transition $\delta(q, \in, Z_0)$, then there should not be any transition from state q_0 when top of the stack is Z_0 ie $\delta(q_0, a, Z_0)$ since it is true the given PDA is deterministic.

Eg: Is the PDA to accept a language

$L(M) = \{W \mid W \in (a+b)^*$ and $n_W(w) = n_b(w)\}$ is deterministic

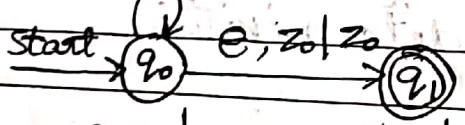
Sol: Language consists of strings of a's and b's of any language.

Step 1: Let q_0 be the start state, and Z_0 be Initial symbol of stack, Scan the Input symbol and push it onto the stack.

Step2:- Once 1st symbol is pushed onto the stack top of the stack may contain either a or b and next input symbol to be scanned may be a or b. If input symbol is same as the symbol on top of the stack, push the current input symbol onto the stack & remain in q_0. Otherwise pop an element from the stack.

For Step1:-

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, a) = (q_0, \epsilon)$$
$$\delta(q_0, b, b) = (q_0, bb)$$
$$\delta(q_0, a, b) = (q_0, \epsilon)$$
$$\delta(q_0, \epsilon, Z_0) = (q_1, Z_0)$$



$a, Z_0 | aZ_0$ $\quad$ $ba | \epsilon$

$b, Z_0 | bZ_0$ $\quad$ $ab | \epsilon$

$aa | aa$

$bb | bb$

Since q_0 is involved in the transition with Z_0

So the given PDA is Non-deterministic.

Turing Machine :-

Turing Machine is a modified version of PDA. Instead of using a stack, turing machine uses a tape to store the symbols. It can recognize all type of languages, Regular languages and context free languages. Here the machine consists of finite control which can be in any finite set of states. The tape is divided into cells and each cell can hold any one of finite number of symbols.

Initialy the input which is of finite length, the symbols choosen from input alphabet are placed on the tape. All other tape cells extending infinitely to the left and right, Initially hold a special symbol called blank.

Formal Notation, $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$.

$\delta(q_2, a_5) = (q_3, b_1, R)$.

Suppose there

Where, Q – is finite Set of states.

Σ – finite set of input symbols.

Γ – complete set of tape symbols, alphabet Σ is a subset of Γ

δ – is the transition function

$\delta(q, x) = (p, y, D)$

where p is next state in Q.

y is the symbol in Γ written in the cell being scanned replacing whatever symbol present

D is direction that can be either L or R standing for Left or Right

q_0 - start state

B - Blank symbol, this symbol is in Γ (Tape) but not in Σ

F - Set of final or accepting states.

Consider a transition

1) $\delta(q_2, a_5) = (q_3, b_1, R)$

It means that if the machine is in state q_2 & next symbol to be scanned is a_5 then machine enters into state q_3 replacing the symbol a_5 by b_1 and R indicates that the control unit is moved from one symbol towards right.

$a_1 q_2 a_3 a_4 q_2 a_5 a_6 a_7 a_8$

$a_1 a_2 a_3 a_4 b_1 q_3 a_6 a_7 a_8$

2) $\delta(q_2, a_5) = (q_1, c_1, L)$

It means that if the machine is in state q_2 and next symbol to be scanned is a_5 then machine enters into state q_1 replacing the symbol a_5 by c_1 and L indicate that the control unit is moved one symbol towards left.

$a_1 a_2 a_3 q_1 a_4 c_1 a_6 a_7 a_8$.

Eg:- Design a Turing Machine to accept a language

$L = \{ 0^n 1^n \mid n \geq 1 \}$

Sol:- The Language accepted by turing machine should have n no. of 0's followed by n no. of 1's.

Let q_0 be the start state & the control unit points to the first symbol of the string to be scanned.

Replace leftmost 0 by X and change the state to q_1 and move the control unit towards right search for left most one and replace it with symbol Y and move towards left.

Repeat the steps

Step 1 :- In state q_0 replace 0 by X change the state to q_1 and move the pointer towards right :

$$\delta(q_0, 0) = (q_1, X, R)$$

Step 2 : In state q_1 we have to obtain leftmost 1 and replace it with Y when it moves towards 1 Symbols encountered may be 0 and Y Irrespective of symbol replace 0 by 0 and Y by Y remain in state q_1 and move towards right.

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

$$\delta(q_1, y) = (q_1, Y, R)$$

Step 3 :- In state q_1 if the input symbol to be scanned is 1 replace 1 by 1, change the state to q_2, and move the pointer towards left

$$\delta(q_1, 1) = (q_2, Y, L)$$

Step 4 :- To obtain left most 0 we need to obtain right most X first. So we can scan for rightmost X. During this process we may encounter Y and 0's. Replace Y by Y and 0 by 0 remain in state q_2 and move unit towards left.

$$\delta(q_2, Y) = (q_2, Y, L)$$

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

Step 5 :- After obtaining right most X, to get leftmost 0 replace X by X change the state to q_0 and move towards right.

$$\delta(q_2, X) = (q_0, X, R)$$

~~Step 6~~

Now Repeat the Steps 1 to 5

Step 6 :-

In state q_0, if scanned symbol is Y it means there are no more 0's and we see that there are no 1's. Further change the state to q_3 replace Y by Y and move pointer towards right.

$$\delta(q_0, Y) = (q_3, Y, R)$$

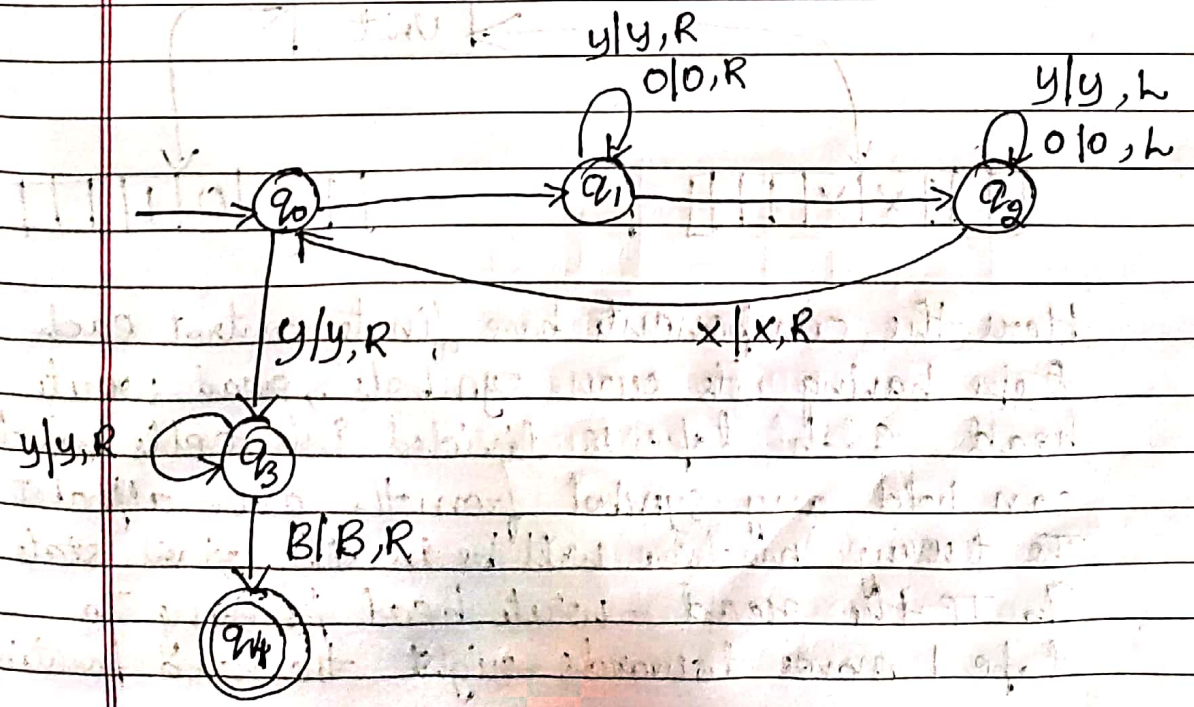
In state q_3 we must see that there are only y's and no more 1's i.e., Scan Y and remain in q_3 state only and move towards right.

$$\delta(q_3, Y) = (q_3, Y, R)$$

If scanned symbol is blank then end of string is encountered.

$$\delta(q_3, B) = (q_4, B, R)$$

Transition Diagram :-



6/1/2022

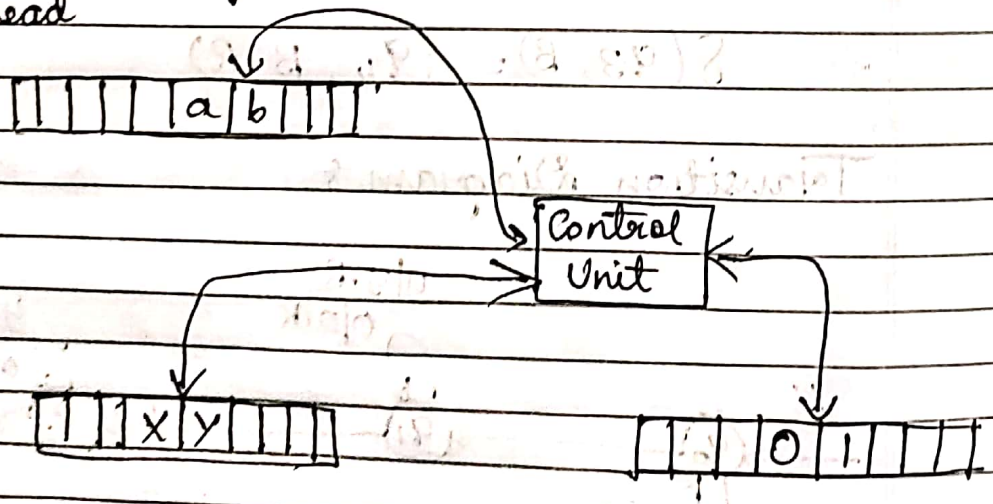
Extension of Turing Machine.

Many variations can be made for standard turing machine with minor modifications it can have following turing machines.

1) Multitape Turing Machine.
2) Non-Deterministic Turing Machine
3)

1) Multi Tape Turing Machine :

It is nothing but a standard turing machine with more no of tapes, Each tape is controlled independently with separate read write head



Here the components have finite control, each tape having its own symbols & read write head Each tape is divided into cells which can hold any symbol from the given alphabet The turing machine will be in the start state q_0 If the read - write head pointing to tape 1 moves towards right the head pointing

to tape 2 and tape 3 may move towards right or left depending on transition. Multitape turing machine is a n-tape machine where

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

Q- is set of finite sets
Σ - set of input alphabets
Γ - set of tape symbols
δ - transition function from $Q \times \Gamma^n$
q_0 - start state.
B - Special Symbol indicating blank character.
F - subset of Q is set of final states.

* The move of the multitape turing machine depends on the current state and scanned symbol by each of the tape heads.

If no of tapes in turing machines is 3 then transitions of

$$\delta(q, a, b, c) = (P, X, Y, Z, L, R, S)$$

2) Non - Deterministic Turing Machine (NDTM)

Difference b/w Non-Deterministic and DTM lies only in the definition of transition function.

NDTM is represented by

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

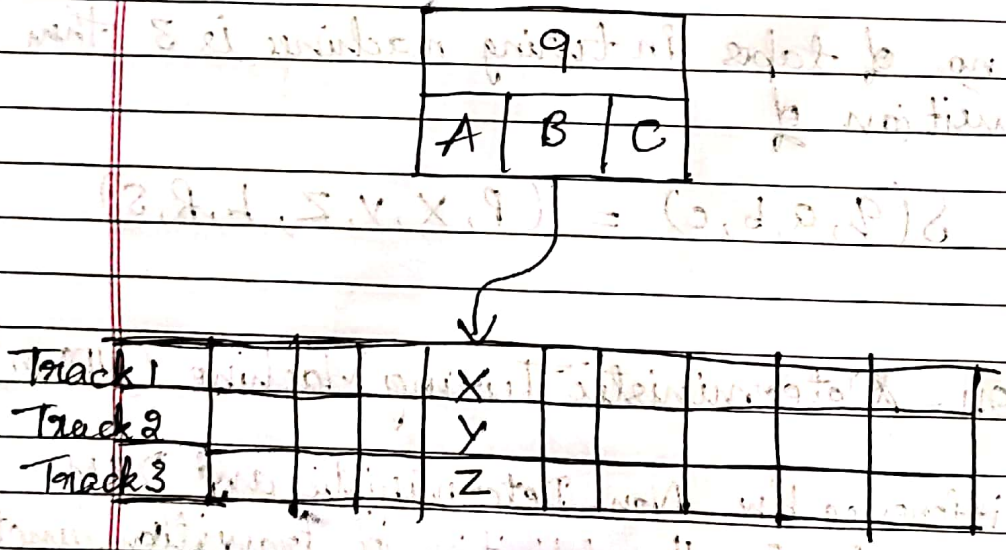
Here the transition function for each state q and tape symbol X i.e the transition of $\delta(q, X)$ is set of triples.

$$\delta(q_1, X) = \{(q_1, X, D)\}$$

★ Programming Techniques for Turing Machine

Turing Machine can be used to compute in a manner similar to that of a conventional computer.

Finite Control can not only represent a position in the program but also to hold a finite amount of data.



Here the turing machine is viewed as having finite control storage and multiple tracks.

In standard turing machine tape is divided into squares where each square holds only one symbol. It can be extended so that

each tape consist of several tracks. This can be done by dividing the tape into no of tracks and each track divided into no of squares.

If there are "n" tracks and read write head points to m^{th} square then all the symbols under different tracks beneath read write head are the symbols to be scanned

Turing Machine can be collection of zero or more Turing machine sub routines where each sub routine is set of states that perform some predefined tasks. The turing machine has a start state & the state without any moves. This state with NO moves has serves as written state. and passes the control to the state which calls the subroutine.

Design a turing Machine to multiply two unary numbers seperated by a delimitter.

$0^m 1 1^n$