15/12/19 # Theory of Computation (6-8M) (9491919183)

Dasaradh.R.k

Syllabus:

Text books:

1. Introduction to formal languages & Automata by peter linz
2. Theory of Computation by I.A. Daniel
* 3. Introduction to language, automata & Computation by Ullman; Aho
4. Introduction to automata & compiler design by Dasaradh-R.K.

TOC

Automata: Mathematical sys used to do certain computation

(i) FA
(ii) PDA
(iii) Linear Bounded Automata (LBA)
(iv) TM

Formal Languages

(i) Regular
(ii) Context free
(iii) Context sensitive
(iv) Recursively Enumerable

Fundamentals & Terminology:

Alphabet: An alphabet is finite set of symbols denoted by Σ.

$\Sigma = \{0,1,2\}$ valid

$\Sigma = \{0,1,2 \cdots 9\}$ valid

$\Sigma = \{0,1,2 \cdots\}$ invalid

words / strings / sentences: A finite sequence of symbols over some fixed alphabet called as string, which is denoted by 'w' or 'x'.

Ex: $\Sigma = \{0,1\}$

w = 101110 – valid string

w = 101121 – invalid string

length of the string: The no of symbols composing the string is called length of the string, denoted by $|w|$.

Empty string is denoted by ϵ (or) λ (or) $\wedge$

Eg: $w = 101101 = 1\epsilon011\epsilon01\epsilon$

$|w| = 6$ ↳ length is still 6.

power alphabet: The power alphabet Σ^k denotes set of all strings of length k over Σ.

Eg: $\Sigma = \{0,1\}$

$\Sigma^0 = \{\epsilon\}$

$\Sigma^1 = \{0,1\}$

$\Sigma^2 = \{00,01,10,11\}$

→ Σ^* – All strings of length 0 or more. (i.e., Universal set over the alphabet)

$$\Sigma^* = \Sigma^0 + \Sigma^1 + \Sigma^2 + \cdots = \overset{\infty}{\underset{i=0}{+}} \Sigma^i$$

→ Σ^+ = All strings of length 1 or more.

$$\Sigma^+ = \overset{\infty}{\underset{i=1}{+}} \Sigma^i = \Sigma^* - \epsilon$$

Operations on string:

prefixes: All possible leading symbol substrings

Suffixes: All possible tail end symbol substrings

Eg: $w = toc$

prefixes: ϵ, t, to, toc Proper prefixes: t, to, toc } exclude 'ϵ'

suffixes: ϵ, c, oc, toc proper suffixes: c, oc, toc }

Concatenation: To obtain concatenation of two strings w_1 & w_2. Write w_1 followed by w_2 without any space blw then

$w_1 = cat$
$w_2 = walk$ } $w_1w_2 = catwalk$

palindrome: A string W is said to be a palindrome, it should be identical whether we read forward or backward

Eg: $\Sigma = \{0,1\}$

palindromes: 0, 00100, 1001 ....

Consider the following alphabet and the string w_1 & w_2

$\Sigma = \{0,1\}$ $w_1 = 0011$ $w_2 = 1100$

(a) find no of strings of length 3 begins and end with 1. - 2

(b) $|w_1 w_1| = 8$

(c) $\{w_1 w_2 w_1\}^* = 00111100 0011$

(d) whether $w_1 w_2$ is it palindrome? - Yes

(e) No of prefixes of w_1 - 5(length +1)

(f) No of common suffixes of w_1 & w_2 [ϵ]

(g) How many string of length 5 are palindromes with '1' in 3rd place: 4

Formal Languages: Set of strings over some fixed alphabet is called is language denoted L or L. (finite or infinite)

$\Sigma = \{0,1\}$

$L = \{01, 0011, 000111 \ldots\}$

$L = \{01, 0^2 1^2, 0^3 1^3 \ldots\}$

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

$L = \{w \in \{0,1\}^+ \mid$ every w has equal no of 0's followed by equal no of 1's$\}$

→ Let L_1 be described as set of all strings over set $\{0,1\}$ with equal no of 0's and followed by equal no of 1's.

→ List the members of the language over the alphabet $\{0,1\}$, all strings of length 4.

$\{0000, 0001 \ldots\ldots 1111\}$

$L = \{w \in \{0,1\}^+ \mid |w|=4\}$

Operations on language:

→ $L_1 = \{\} = \phi$ (empty language)

→ $L_2 = \{\in\} = \in$ (empty string language) (Its size is one)

$(L_1 \neq L_2)$

Countably infinite : Eg: $\{1,2,3\ldots\}$

Uncountably infinite : Eg: $\{x \mid x$ is a real number$\}$

$\{12.11, 12.12, 12.111 \ldots\}$

→ All formal languages are either finite or countably infinite but no uncountably infinite.

Previous

Q: Consider the following strings over alphabet $\Sigma = \{a,b,c\}$

$w =$ baacabb

find the no of substring of the w?

Sol:

No of substring of length 0 = 1

" " " " 1 = 7

" " " " 2 = 6

" " " " 3 = 5

⋮

7 = 1

$\frac{7 \times 8}{2} + 1 = 29$

0 length	1 length	2 length	3	4	5	6	7	
ϵ	a	~~aa~~ ba	5	4	3	2	1	$\Rightarrow 25$
	b	aa						
	c	ac						
		ca						
		ab						
		bb						

Q:
Let $\Sigma = \{0,1\}$. How many strings are possible of length n?

a) n b) n+1 c) 2^n ✓ d) 2n

Q: Let $\Sigma = \{a,b,c\}$. No of strings of length ≤ 3.

0	1	2	3	
1	3	9	27	$= 40$

Note:
Let Σ be the alphabet with m symbols, the possible no of strings of length n are: m^n

Language Operations:

Union: $l_1 \cup l_2 = \{w \mid w$ is in either l_1 or $l_2\}$

Intersection: $l_1 \cap l_2 = \{w \mid w$ is in both l_1 & $l_2\}$

Concatenation: $l_1 l_2 = \{xy \mid x \in l_1$ & $y \in l_2\}$

Complementation: l_1' or $\bar{l_1}$ or $\sim l_1 = \{w \mid w \in \Sigma^*$ but not in $l_1\} = \Sigma^* - l_1$

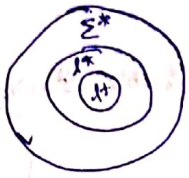
Kleen closure: $l^* = l^0 \cup l^1 \cup l^2 \ldots$ i.e., zero or more concatenations of l.

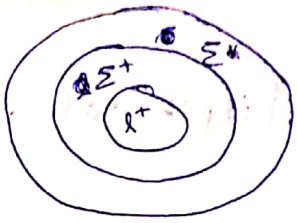
$\Rightarrow l^0 = \{\epsilon\}$

$l^1 = l$

$l^2 = l \cdot l$ (concatenation)

Positive closure: $l^+ = l^* - l^0$ i.e, one or more concatenations of l.





Q: Consider the following language $L = \{a^i b^j \mid i = j > 0\}$

which of the following represents complementation of L

a) $L = \{a^i b^j \mid i \neq j > 0\}$ abab

b) $L = \{a^i b^j \mid i \neq j \geq 0\}$

c) $L = \{w \mid w \in \{a,b\}^+$ and w contains unequal no of a's, b's$\}$

d) None of the above.

Correct answer is d

~~$L = \{w \mid w \in \{a,b\}^*$ and w contains unequal~~

Q: Consider the following lang over the alphabet $\Sigma = \{0,1\}$. which of the following is true

(i) $L_1 = \{w \mid w$ $n_0(w) = n_1(w)\}$,

where n_0 = no of zeroes in w

n_1 = no of ones in w

(ii) $L_2 = \{w \mid w$ contains all strings $\{0,1\}\}$

(iii) $L_3 = \{w \mid w$ contains all palindroms over $\{0,1\}\}$

a) Find the no of strings of length 4 common in all three lang. -2 (1001, 0110)

b) which one of the following relationship true of these lang

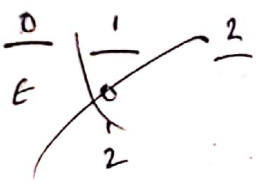
a) 1) $L_1 \subseteq L_2 \subseteq L_3$

2) $L_2 \subseteq L_1 \subseteq L_3$

3) $L_2 \subseteq L_3 \subseteq L_1$

4) None of the above

Q: Consider $L = \{0^i 1^j 2^k \mid i \neq j \neq k > 0\}$. No of possible strings of length 2 or less.



0	1	2
$0^0 1^0 2^0$	$0^1 1^0 2^0$ (not valid)	$0^2 1^0 2^0$ (N.V)
not valid	$0^0 1^1 2^0$ (N.V)	$0^0 1^2 2^0$ (N.V)
	$0^0 1^0 2^1$ (N.V)	$0^0 1^0 2^2$ (N.V)

Ans: 0

Q: Let w be the string of length n and w contains diff symbols. How many non empty substring string are possible with w?

a) 2^n

b) $2n$

c) $n+1$

✓d) $\frac{n(n+1)}{2}$

check counting substring by removing prefix & suffix

Sol:

No of substring of length 'n' = 1
" " " " " n-1 = 2
⋮
2 = n-1
1 = n

} Sum = $\frac{n(n+1)}{2}$

Q: Set of all strings of length n can how many substrings

Ans: $(n+1)$ to $\frac{n(n+1)}{2} + 1$

↓ when all symbol are equals

↓ when all symbols are distinct.

Q: If $L^* = L^0 + L^1 + L^2 + \cdots$ then $L^+ =$ _____

a) $L^* + \varepsilon$

b) $L^* + L$

c) LL^*

d) None

Sol:

$$LL^* = L(\varepsilon + L + L^2 + L^3 + \cdots)$$

$$= L + L^2 + L^3 + \cdots = L^+$$

Q: Let $L = \{w \mid w \in \{0,1\}^* \text{ and } N_0(w) \neq N_1(w)\}$

Find the strings of length 4 or less. in L'.

0	1	2	3	4
1	2	2	8	$16 - {}^4C_2$
				10

$= 2$

Method 1:

for L

0	1	2	3	4
0	2	2	8	$16 - {}^4C_2$
				$= 10$

$= 22$

Total no of string of length 4 or less $= 2^5 - 1 = 31 \quad (2^0 + 2^1 + \cdots + 2^4 = 2^{4+1} - 1)$

$\therefore$ Ans $= 31 - 22 = 9$

Method 2:

Directly compute L'

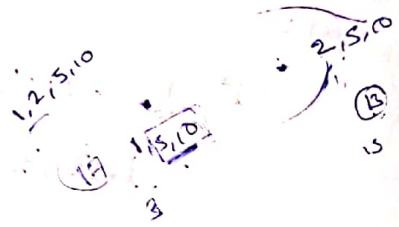
$L' = \{\varepsilon, 00, 11, 0011, 1010, 1100, 0101, 1001, 0110\} \rightarrow 9$

Q: A lang defined over a alphabet $\Sigma=\{0,1\}$ and every w length is 7 such that first place, 4th place and last place are 1's. Find no of string in lang.

Sol:

1 - - 1 - - 1

$2^4 = 16$



Aut

Finite Automata:

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

- Q: set of states
- Σ: input symbols
- δ: Transitions
- q_0: initial state
- F: Set of final states

→ can be represented by (i) state diagram/ transition diagram/ transition system
(ii) state table/ transition table

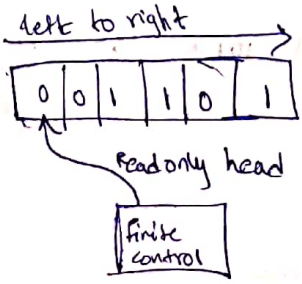
→ FA is a mathematical system consists of 5 tuples

→ →○ ---- initial state

→ ◎ ---- finial state

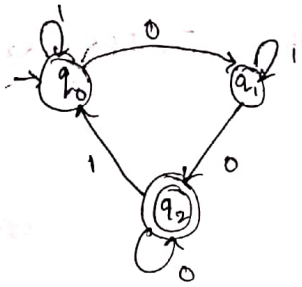
→ Although there are many applications with the F.A, the main obj of the F.A is to perform certain computational operations in the computer science. The following is finite automata to identify whether binary number has odd zeroes or even zeroes

Block diagram:



→ A F.A's basic obj is recogonisation of string. A string w is said to be accepted by F.A if and only if a machine terminated at final state by reading the last symbol in w. In other words a string w said to be accepted by F.A ⟺ $\delta(q_0, w) = p$ for some 'p' in 'F'.

Q: Conside below F.A



101010

0

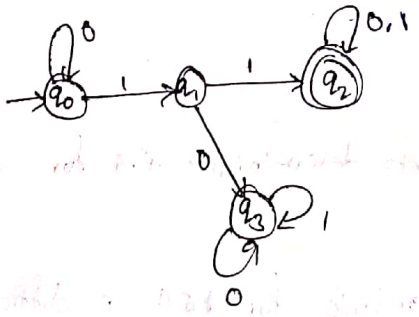
Check whether the following strings are recogonised by the F.A or not

✓a) 101110

✗b) 010101

✗c) 1001

Q: Consider following F.A :

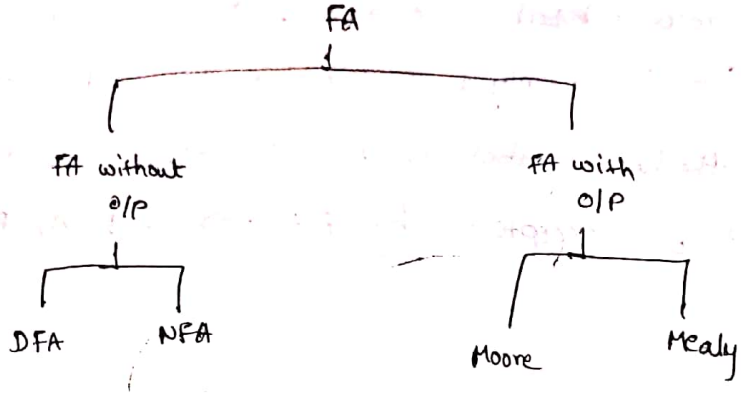


✗a) 010010 ✗b) 1000110

✗c) 0101110 ✗d) 100111001

$L(M) = \{11, 011, 110, 111, 0011, 0110 \ldots\}$

⇒ language of M can be defined as set of all strings over $\{0,1\}$ with any zeroes followed by 2 consecutive 1's followed by any combination of 0's & 1's.



DFA:

→ In DFA for i/p symbol there exist exactly one transition

→ The state table of DFA consists of single entities. The DFA never permits ε transitions.

$$\delta: Q \times \Sigma \to Q$$

NFA:

→ It is a 5 tuple system

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

δ: defined as

$$Q \times \Sigma \to 2^Q$$

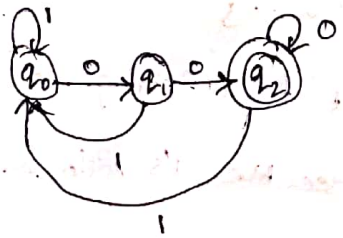
→ For each i/p symbol of NFA there may exist one or more transitions.

→ The state table of NFA consists of multiply defined entries. The NFA may also permits ε-transitions.

→ All NFA are generally translated into deterministic F.A for easy & secured implementation of code

i.e., developing a mechanism or logic for NFA is difficult than that of DFA. Hence we do translate NFAs to DFAs.

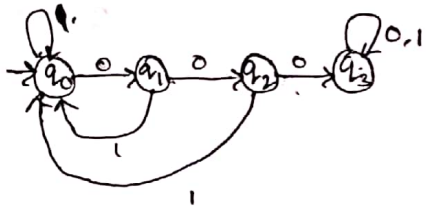
Eg 2: Design a DFA to recognise set of all strings over {0,1} ending with 00



$L = \{00, 000, 100, 0000, 0100, 1000, 1100 \ldots\}$

Q: Design a DFA to recognise set of all strings containing the substring '000'.

$L = \{000, 1000, 0000, 0001, 10001, \ldots\}$

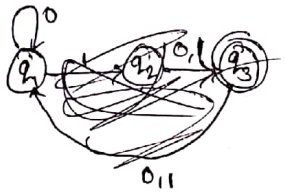


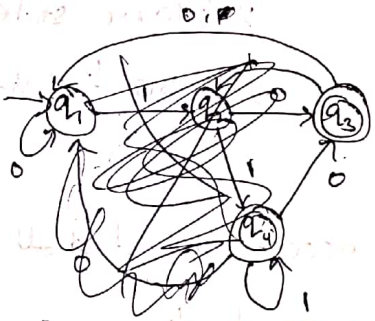
Q: DFA to accept set of strings over $\{0,1\}$ such that 10th symbol from right end is '1'.

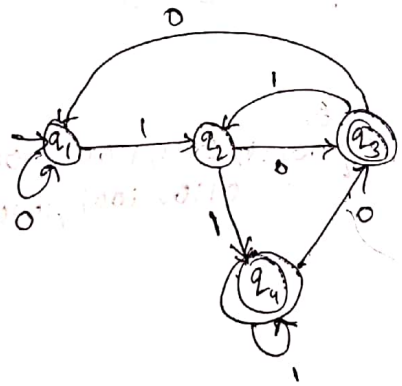
First design DFA such that 2nd symbol from right end is '1'.

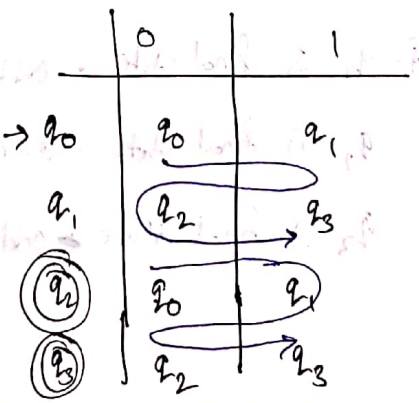
$L = \{10, 11, 110, 111, 010, 011 \ldots\}$

11011









General Method :

2^{2-1} → corresponds to nth digit from right side

⇒ 2 final states

2^2 → states

lets design for 1 at 3rd place from right end.

	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_5
q_3	q_6	q_7
q_4^*	q_0	q_1
q_5^*	q_2	q_3
q_6^*	q_4	q_5
q_7^*	q_6	q_7

$2^{3-1} = 4$ final states

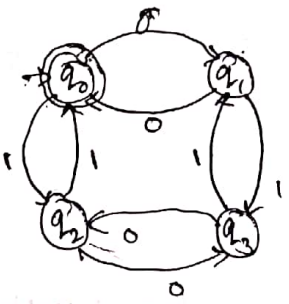
$2^3 = 8$ states

for 1 at 10th place from right end

$2^{10} = 1024$ states

$2^9 = 512$ final state

Q: to accept set of all strings over $\{0,1\}$ with even no of 0's & even no of 1's.



$L = \{\epsilon, 00, 11, 0011, 1100, 110011, 001100, 000011, 0110, 1001, 10100101\}$

In above fig if q_1 is final state - odd 0's even 1's

q_2 is final state - even 0's odd 1's

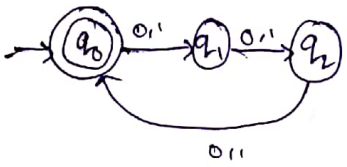
q_3 is final state - odd 0's odd 1's

29/12/19

Theory of Computation :

Q:

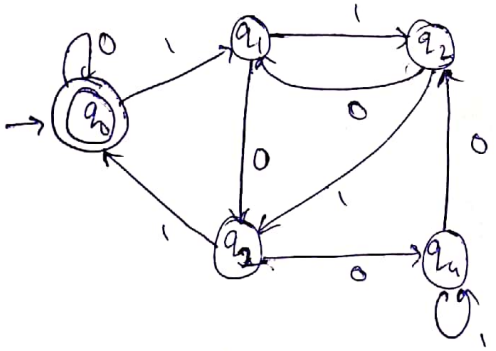
* Design a DFA to recogonite set of all strings whose length is exactly divisible by 3 over alphabet $\Sigma = \{0,1\}$



$L = \{\epsilon, 000, 111, 010, 000111000, 011100, \ldots\}$

Q: Design a DFA to recogonize set of all strings over $\{0,1\}$, here the binary numbers are interpreted as decimal integers. The machine should accept all the strings who decimal value is divisible by 5.

$L = \{0, 101, 1010, 1111, 10100, 11001, 11110, 100011$



q_0 - mods - 0

q_1 - mods - 1

q_2 - mods - 2

q_3 - mods - 3

q_4 - mods - 4

q_1: 10 mods - 2 - q_2

11 mods - 3 - q_3

q_2: 100 mods - 4 - q_4

101 mods - 0 - q_0

q_3: } silly

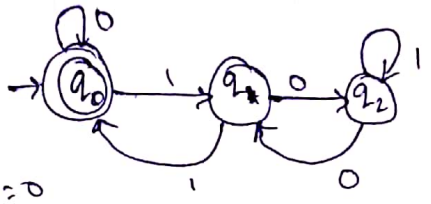
q_4: }

imp

	0	1
$\rightarrow q_0^*$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

Pattern

Silly for divisible by 3



q_0 : 0 mod 3 = 0
2 mod 3 = 2

q_1 : 10 mod 3 = 2 $\Rightarrow q_2$
11 mod 3 = 0 $\Rightarrow q_0$

q_2 : 101 mod 3 = 2 $\Rightarrow q_2$
100 mod 3 = 1 $\Rightarrow q_1$

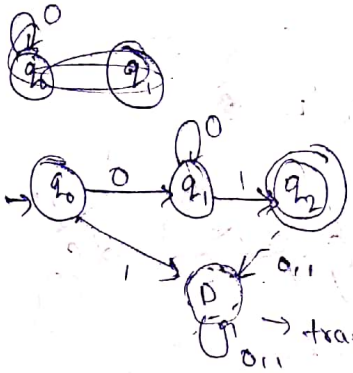
	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

pattern

Q: Construct at DFA for the lang

$$L = \{0^n 1 / n > 0\}$$

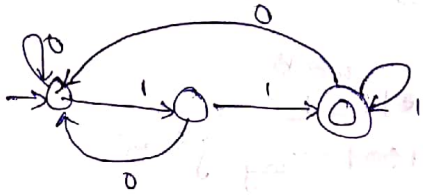
$L = \{01, 001, 0001 \ldots\}$



→ trap state (or) dead state

Previous Questions:

Consider the following DFA



The M accepts all strings over $\{0,1\}$

a) begin with 0 , end with 1
b) ending 1
c) Substring 11
d) ending 11

for this type of questions list possible strings of long given and options (Random verification may go wrong)

given

$L = \{11, 111, 1011, 1011, 011 \ldots\}$

a) $L = \{01 \ldots\}$ ✗

b) $\{1, \ldots\}$ ✗

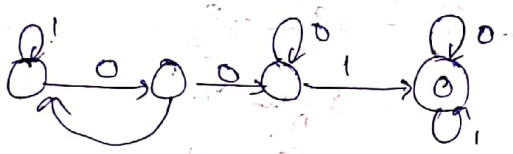
c) $\{11, 011, 110\}$ ✓ ✓ ✗

d) $\{11, 011, 111, 0011, 01011 \ldots\}$ ✓ ✓ ✓ ✓ ✓

∴ Ending with 11.

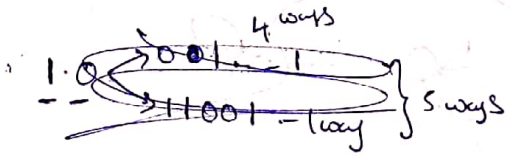
(2003)

Consider the following DFA 'M'



Let S denote set of 7 bit binary strings in which the 1st, 4th and last bits are 1. The no of strings in S that are accepted by M is

a) 1 b) 5 c) 7 d) 8



$L = \{\underline{001}, \underline{1001}, 0\underline{1001}, 0\underline{10010} \ldots\}$

Observing above automata, it accepts strings with substring '001'.

1	2	3	4	5	6	7
	a	a				
	a	a				
	a	a				
	a	a	b			

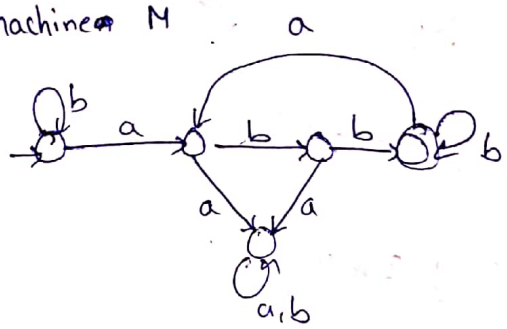
↓ substring — 4 ways

1	2	3	4	5	6	7
	a	b	a	a	b	→ append
	a	b		a	b	
	a	b		b	b	
	a	b			b	

1 way — Similarly 2 more ways

$4+1+1+1=7$

Q: Consider the machine M



The lang recogonized by M is

a) $\{w \in \{a,b\}^* \mid$ every a in w is followed by exactly 2 b's$\}$

b) $\{w \in \{a,b\}^* \mid$ every a in w is followed by atleast 2 b's$\}$

c) $\{w \in \{a,b\}^* \mid w$ contains the substring abb$\}$

d) $\{w \in \{a,b\}^* \mid w$ does not contain aa as substring$\}$

for given

$L = \{$ abb, babb, abbb, babbb, abbabb, $\}$

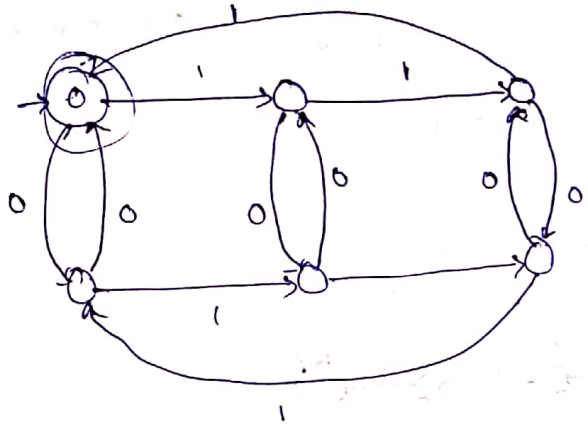
for (a) $L = \{$ abb, abbb ... $\}$ ✗

(b) $L = \{$ abb, babb, bbabb, babbbabb, abb ... $\}$ ✓

(c) $L = \{$ abb, aabb .. $\}$ ✗

(d) $L = \{\epsilon, a, b, ab, aba \ldots\}$ ✗

Q: The following FSM accepts all those binary strings in which no of 1's & 0's are respectively _____



a) divisible by 3 & 2

b) odd & even

c) even & odd

d) divisible by 2 & 3

for given

$L = \{00, 111, 11100, 00111, 10011, 10110, 1001001 \ldots\}$

for (a) $L = \{00, 111, 11100, 11001, 10011, 01110 \ldots\}$ ✓

(b) $L = \{1 \ldots\}$ ✗

(c) $L = \{11 \ldots\}$ ✗

(d) $L = \{11, 000 \ldots\}$ ✗

if no of 1's mod 3 = 1
no of 0's divisible by 2
the in above fig we take

Note:

for 1's divisible by 3 & 0's divisible by 2

remainder combinations are

(0,0) (1,0) (2,0)
(0,1) (1,1) (2,1)

i.e., 6 states

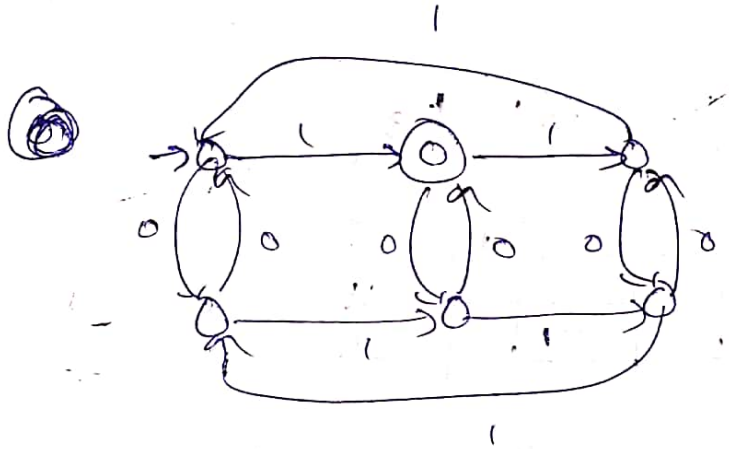
i.e., $3 \times 2 = 6$ states

→ The automata which accepts string containing 1's divisible by 5 and 0's divisible by 3 requires 15 states

i.e., $5 \times 3 = 15$

(0,0) (1,0) (2,0) . . .
(0,1) (1,1) (2,1) . .
(0,2) (1,2) (2,2) . .

→ Design DFA in which no of 1's divisble by 3 gives remainder 1 and no of 0's is divisible by 2



NFA to DFA conversion:

→ An NFA is 5 tuple system

$$M = \{Q, \Sigma, \delta, q_0, F\}$$

where δ: transition function

$$Q \times \Sigma \to 2^Q$$

i.e., for each i/p symbol there may exist more than one transition.

The NFA mechanism is very helpful to describe certain non-deterministic system's behaviour but it is very complex to code.

For every NFA there exists DFA that simulates behaviour of NFA

i.e., for every NFA 'M' there exists equivalent DFA 'M₁' such that

$$L(M_1) = L(M)$$

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA

we construct an equivalent DFA M_1

$$M_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$$

where $Q_1 = 2^Q$ i.e., power set of Q

$F_1 \in Q_1$ but contains element F

δ_1 defined as

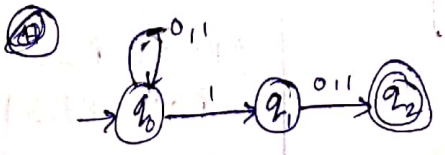
$$\delta_1[\{q_0, q_1, \ldots q_n\}, a] = \delta(q_0, a) \cup \delta(q_1, a) \cup \ldots \delta(q_n, a)$$

$$= P_0 \cup P_1 \cup \ldots \cup P_n$$

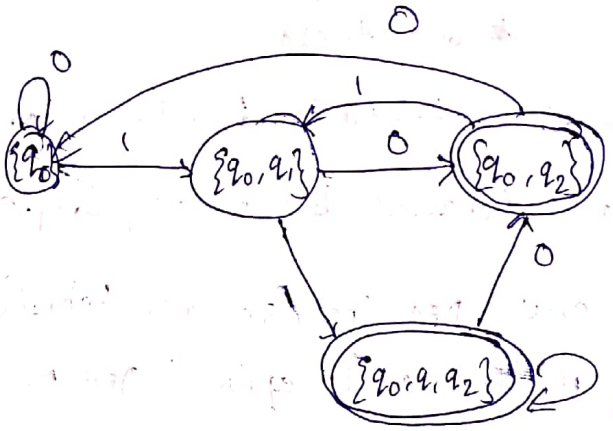
$$= [P_0, P_1 \ldots P_n]$$

Eg: Convert the following NFA to DFA

NFA in which last but one symbol is 1.



	0	1
→ $\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$



→ In the process of conversion ~~for N~~ from NFA to DFA

$$Q_1 = 2^Q$$

which means Q_1 contains 2^n states.

But this may also contain unreachable states.

So we consider only reachable states

→ Convert below NFA to DFA

	0	1
→ P	P, Q	P
Q	R	R
R	-	S
*S	S	S

DFA →

	0	1
→ P	P, Q	P
P, Q	P, Q, R	P, R
P, Q, R	P, Q, R	P, R, S
P, R	P, Q	P, S
P, R, S	P, Q, S	P, S
P, S	P, Q, S	P, S
P, Q, S	P, Q, R, S	P, R, S
P, Q, R, S	P, Q, R, S	P, R, S

• Convert NFA to DFA

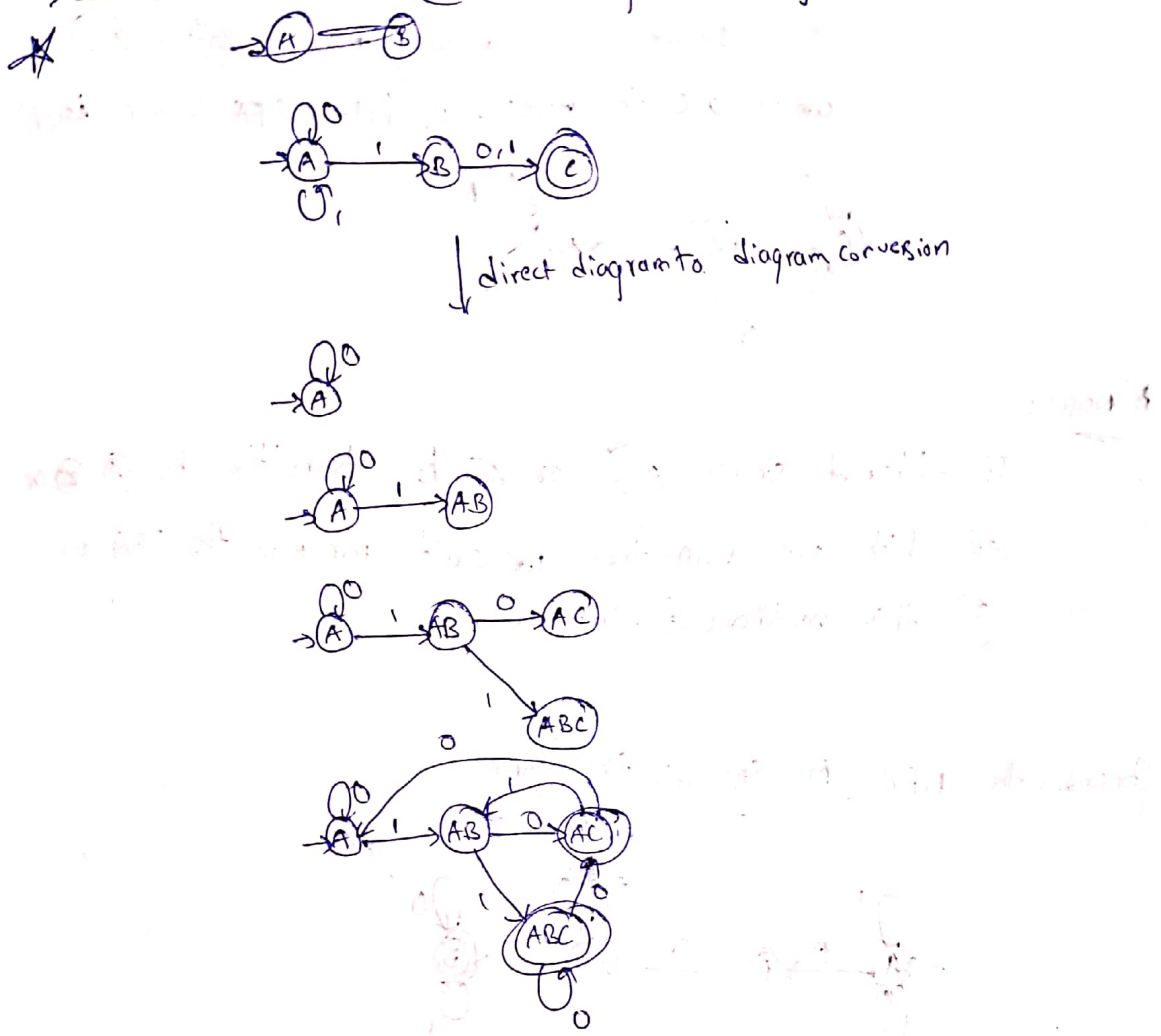
	0	1
→ P	q, r	q
q	r	s
r*	q	s
s*	s	r

to DFA

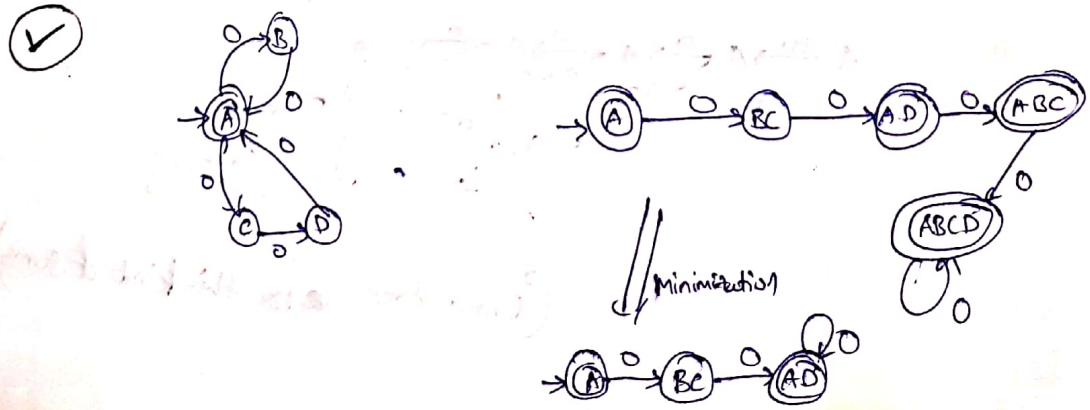
	0	1
p	q, r	q
(q, s)	q, r	s
q	r	s
(s)	s	r
(r)	q	s

~~→ Convert NFA to DFA~~

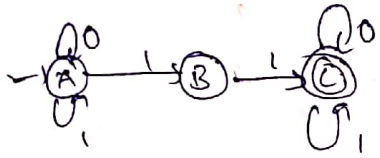
→ Convert NFA to DFA (Direct diagram to diagram conversion)

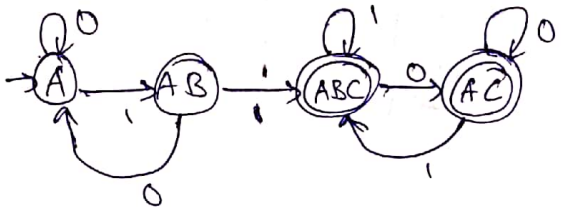


→ Convert NFA to DFA

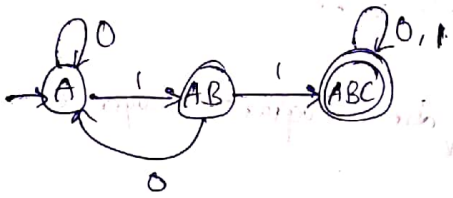


→ Convert NFA to DFA





In above fig once we reach state (ABC) No matter what i/p we read, we only move to (AC) or (ABC) which are final states. So below DFA is minimized

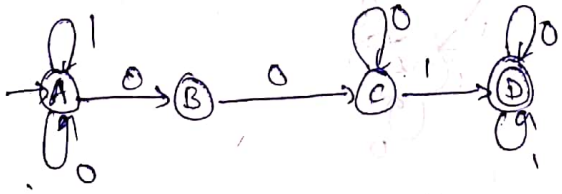


★ NOTE:

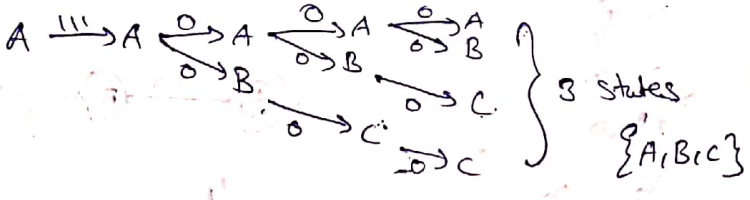
If either of states (ABC) or (AC) has transition to (A) or (AB) which are non-final we can't minimize the DFA in this method of minimization.

Q:

Consider the NFA 'M' shown in the figure

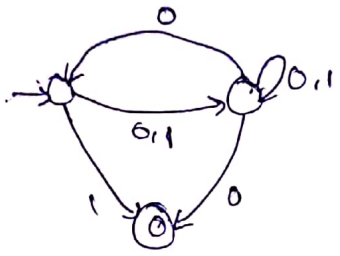


Find the no of reachable states for i/p string w = 11100000



(Draw tree ref for this kind of sums)

Q: Consider below NFA 'M'



Let the lang accepted by M be L

Let L_1 be the lang accepted by NFA 'M_1', obtained by changing the ~~accp~~ accepting state of M to non-accepting state, and by changing non-accepting state of M to accepting state. Which of the following ~~state men~~ statement is true?

a) $L_1 = \{0,1\}^* - L$

b) $L_1 = \{0,1\}^*$

c) $L_1 \subseteq L$

d) $L_1 = L$

This is complementation technique of DFA ~~(or NFA~~ in which changing final to non-final and non-final to final does not work for NFA

given

$L = \{1, 10, 100, 000, 010, 100 \ldots\}$

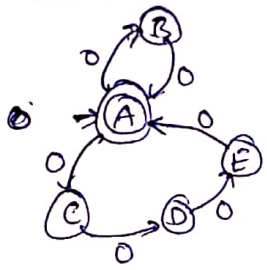
Now

If we observe the above NFA accepts any string of any length

$L = \{\epsilon, 0, 1, 00, 01, 10, 11 \ldots\}$

$\therefore L = \{0,1\}^*$

→ fin min no of states when below NFA is converted to DFA.

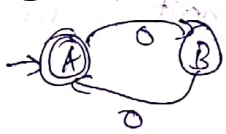


Method:I:

Observing above DFA it is clear that it accepts string containing $2x+4y$ number of zeroes, $x, y \geq 0$

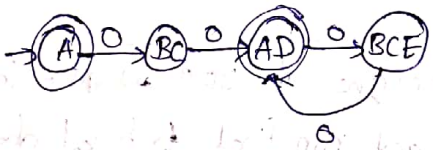
So we can say no of zeroes is divisible by 2

Hence DFA is



Method 2:

direct conversion to DFA



Here direct minimization is not possible as it was in previous cases we can follow long minimization process.

So we follow (Method I)

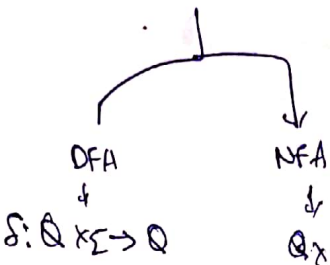
ϵ-NFA

→ permits ϵ-transitions

→ ϵ-NFA 5tuples $M = (Q, \Sigma, \delta, q_0, F)$

$\delta : Q \times \Sigma \cup \epsilon \rightarrow 2^Q$

	0	1	2	ϵ
A	A	-	-	B
B	-	B	-	C
C	-	-	C	-



ϵ-NFA

$Q * \Sigma \cup \epsilon \to 2^Q$

↳ ϵ can't be in Σ so we represent it seperately.

→ lang accepted by NFA T can be described as any no of zeroes followed by any no of 1's follow by any no of 2's shown above

Eg:

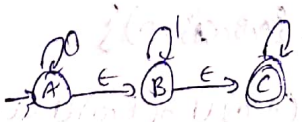
$L = \{\epsilon, 1, 11, 111, 1111, \ldots\}$

i.e, any no of 1's.

ϵ-Closure:

→ ϵ-closure of state q (ϵ-closure(q)) is the set of states P such that every state in the p is reachable from q through ϵ.

Consider below ϵNFA



ϵ-closure (A) = {A}

= {A} ∪ {B}

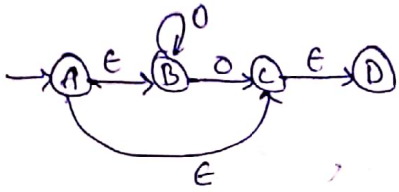
= {A} ∪ {B} ∪ {C}

ϵ-closure(A) = {A, B, C}

ϵ-closure(B) = {B, C}

ϵ-closure(C) = {C}

Eg:



ε-closure(A) = {A, B, C, D}

ε-closure(B) = {B}

ε-closure(C) = {C, D}

ε-closure(D) = {D}

$\hat{\delta}^*(A, \epsilon) = \epsilon\text{-closure}(A)$.

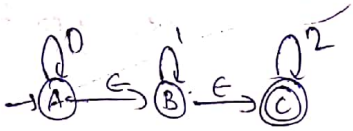
$\hat{\delta}^*$ (or) $\hat{\delta}$ is known as extended transition function

$\hat{\delta}(A, \epsilon) = \{A, B, C, D\}$

Similarly $\hat{\delta}(B, \epsilon) = \{B\}$

$\hat{\delta}(C, \epsilon) = \{C, D\}$

$\hat{\delta}(D, \epsilon) = \{D\}$



$\hat{\delta}(q, a) = \epsilon\text{-closure}(\delta(\hat{\delta}(q, \epsilon), a))$

Eg:

$\hat{\delta}(A, 0) = \epsilon\text{-closure}(\delta(\hat{\delta}(A, \epsilon), 0))$

$= \epsilon\text{-closure}(\delta(\{A, B, C, D\}, 0))$

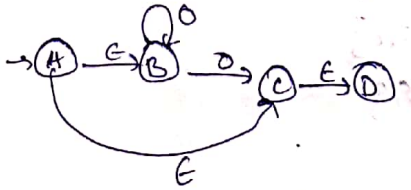
$= \epsilon\text{-closure}(\delta(A, 0) \cup \delta(B, 0) \cup \delta(C, 0) \cup \delta(D, 0))$

$= \epsilon\text{-closure}(A \cup \phi \cup \phi)$

$= \epsilon\text{-closure}(A)$

$= \{A, B, C\}$

Consider



$\hat{\delta}(A,0) =$ ~~$\delta(\hat{\delta}(A,0),0$~~

$= \epsilon\text{-closure}(\delta(\hat{\delta}(A,\epsilon),0))$

$= \epsilon\text{-closure}(\delta(\{A,B,C,D\},0)\}$

$= \epsilon\text{-closure}(\delta(A,0) \cup \delta(B,0) \cup \delta(C,0) \cup \delta(D,0))$

$= \epsilon\text{-closure}(\phi \cup \{B,C\} \cup \phi \cup \phi)$

$= \epsilon\text{-closure}(\{B,C\})$

$= \epsilon\text{-closure}(\{B\}) \cup \epsilon\text{-closure}(C)$

$= \{B\} \cup \{C,D\}$

$= \{B,C,D\}$

→ Let 'w' be a string.

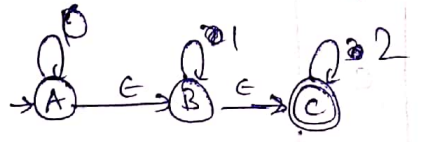
Let 'a' be last symbol of w and x be starting part

$w = xa$

~~$\hat{\delta}(q_0,w) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0,x),a))$~~

$$\hat{\delta}(q_0,w) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0,x),a))$$

⇒ Consider



$\hat{\delta}(B,12)$

$\hat{\delta}(B,1) = \epsilon\text{-closure}(\delta(\hat{\delta}(B,\epsilon),1))$

$= \epsilon\text{-clo}(\delta(\{B,C\},1))$

$= \epsilon\text{-clo}(B)$

$= \{B,C\}$

$\hat{\delta}(B,12) = \epsilon\text{-closure}(\delta(\hat{\delta}(B,1),2))$

$= \epsilon\text{-close}(\delta(\{B,C\},2)$

$= \epsilon\text{-clo}(C) = \{C\}$

ϵ-NFA to NFA :

In notes theorem, a method to convert ϵ-NFA to DFA directly

For every NFA with ϵ, we can construct an NFA without ϵ such that both will perform same job. In other words if L is accepted by NFA with ϵ, then L is also accepted by NFA without ϵ.

Proof :

Let $M = (Q, \Sigma, \delta, q_0, F)$ be ϵ-NFA

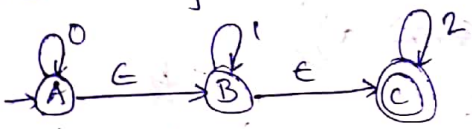
- we can construct an equivalent NFA M_1 to M as

$M_1 = (Q, \Sigma, \delta_1, q_0, F_1)$

δ : $\hat{\delta}(q, a)$ $\forall q \in Q$ & $a \in \Sigma$

$F_1 = \begin{cases} F \cup \epsilon\text{-closure}(q_0) & \text{if } \epsilon\text{-closure}(q_0) \text{ contains element } F \\ F, & \text{otherwise} \end{cases}$

→ Convert the following ϵ-NFA to NFA



	0	1	2	ϵ
→A	A	-	-	B
B	-	B	-	C
C	-	-	C	-

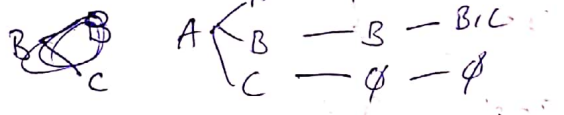
↓ ϵ-NFA to NFA

	0	1	2
→ A*	{A, B, C}	{B, C}	{C}
B*	ϕ	{B, C}	{C}
C*	ϕ	ϕ	{C}

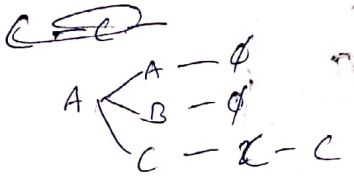
For each entry [q, a] find $\hat{\delta}(q,a)$

$$\hat{\delta}(A,0) = \epsilon\text{-clo}\left(\delta\left(\hat{\delta}(A,\epsilon),0\right)\right)$$
$$= \epsilon\text{-clo}\left(\delta(A,0) \cup \delta(B,0) \cup \delta(C,0)\right)$$
$$= \epsilon\text{-clo}(A)$$
$$= \{A,B,C\}$$

$\hat{\delta}(A,1)$ =



$\hat{\delta}(A,2)$ =



$\hat{\delta}(B,0)$ = B { B — ϕ — ϕ ; C — ϕ — ϕ

$\hat{\delta}(B,1)$ = B { B — B — B,C ; C — ϕ — ϕ

Computing final states

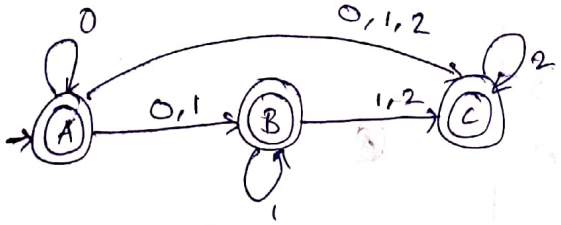
Initial state : A

ϵ-closure (A) = {A, B, C}

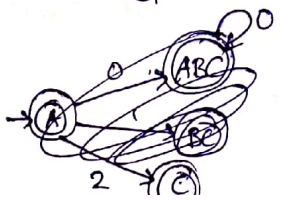
∵ final state exists in ϵ-closure(A)

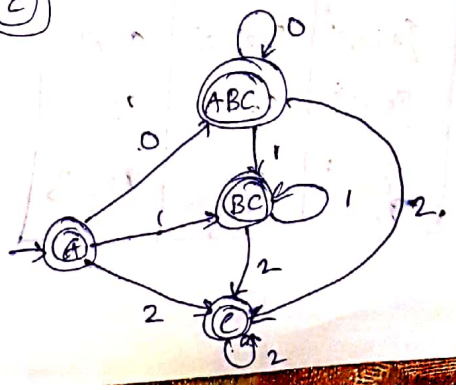
∴ $F_1 = F \cup \{A,B,C\}$

$= \{A,B,C\}$

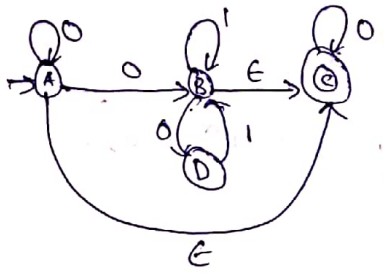


↓ to DFA





→ Consider the following ε-NFA



	0	1
→A*	{A,B,C}	ϕ
B	{C,D}	{B,C}
C*	{C}	ϕ
D	ϕ	{B,C}

$\hat{\delta}(A,0) = A \begin{cases} A - A,B - A,B,C \\ C - C - C \end{cases}$

$\hat{\delta}(A,1) = A \begin{cases} A - \phi \\ C - \phi \end{cases}$

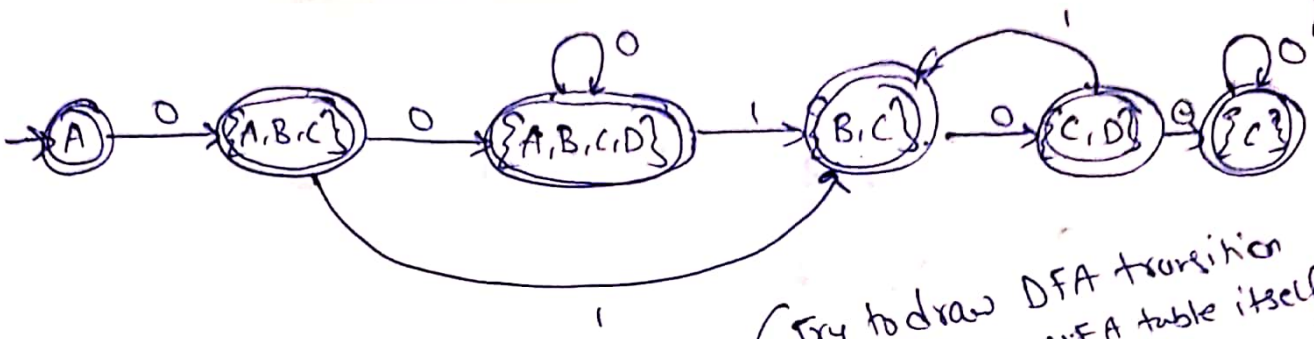
$\hat{\delta}(B,0) = B \begin{cases} B - D - D \\ C - C - C \end{cases}$

$\hat{\delta}(B,1) = B \begin{cases} B - B - B,C \\ C - \phi \end{cases}$

ε-closure(A) = {A,C} includes final state ∴ $F_1 = F \cup \{A,C\}$
$= \{A,C\}$

To DFA

	0	1
→A*	{A,B,C}	ϕ
{A,B,C}*	{A,B,C,D}	{B,C}
{A,B,C,D}*	{A,B,C,D}	{B,C}
{B,C}*	{C,D}	{B,C}
{C,D}*	{C}	{B,C}
{C}*	{C}	ϕ



(Try to draw DFA transition diagram from NFA table itself)

Q: Let 'δ' denote the transition function, δ̂ denote the extended transition function of ϵ-NFA below

	ϵ	a	b
q_0	q_2	q_1	q_0
q_1	q_2	q_2	q_3
q_2	q_0	ϕ	ϕ
q_3	ϕ	ϕ	q_2

In this type of questions Calculate closures first so that we make no mistakes

$\hat{\delta}(q_2, a)$

q_2 → $q_0 - q_1 - \{q_0, q_1, q_2\}$

$q_2 - \phi$

$\hat{\delta}(q_2, ab)$

q_0 → $q_0 - q_0 - \{q_0, q_2\}$

$q_2 - \phi$

q_1 → $q_0 - q_0 - \{q_0, q_2\}$

$q_1 - q_3 - \phi$

$q_2 - \phi - \phi$

q_2 → $q_0 - q_0 - \{q_0, q_2\}$

$q_2 - \phi$

$\Rightarrow \{q_0, q_2\}$

$\hat{\delta}(q_2, aba)$

q_0 → $q_0 - q_1 - \{q_0, q_1, q_2\}$

$q_2 - \phi$

q_2 → $q_0 - q_1 - \{q_0, q_1, q_2\}$

$q_2 - \phi$

$\{q_0, q_1, q_2\}$

$\therefore \hat{\delta}(q_2, aba) = \{q_0, q_1, q_2\}$

(ii) find $\hat{\delta}(q_1, abb)$

$\hat{\delta}(q_1, a)$

q_1 → $q_0 - q_1 - \{q_0, q_1, q_2\}$

$q_1 - q_2 - \{q_1, q_2\}$

$q_2 - \phi$

$\{q_0, q_1, q_2\}$

$\hat{\delta}(q_1, ab)$

$q_0 \leftarrow q_0 - \{q_0, q_2\}$

$q_1 \leftarrow q_0 - \phi$

$q_2 - \phi$

$q_1 \leftarrow q_0 - q_0 - \{q_0, q_2\}$

$q_1 - q_3 - \phi$

$q_2 - \phi$

$q_2 \leftarrow q_0 - q_0 - \{q_0, q_2\}$

$q_2 - \phi$

$\hat{\delta}(q_1, abb)$

$q_0 \leftarrow q_0 - q_0$

$q_2 - \phi$

$q_0 \leftarrow q_0 - q_0 - \{q_0, q_2\}$

$q_2 - \phi$

$q_2 \leftarrow q_0 - q_0 - \{q_0, q_2\}$

$q_2 - \phi$

$\therefore \hat{\delta}(q_1, abb) = \{q_0, q_2\}$

28/10/20

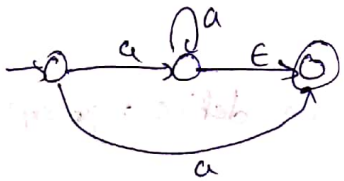
Q: What complementation of lang accepted by below NFA

a) ϕ b) $\{\epsilon\}$ c) a^* d) $\{a, \epsilon\}$

Observing above NFA we can say it accepts strings containing one or more no of a's. $L = \{a, aa, aaa \ldots\}$

$\therefore$ complementation = $a\{\epsilon\}$

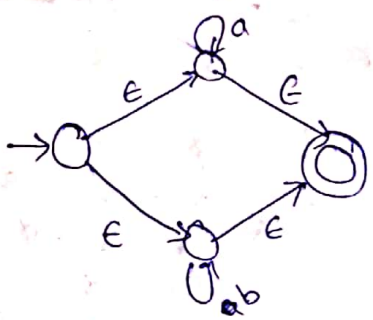
Q: Find complementation of below ϵ-NFA



$L = \{a, aa, aaa \ldots\}$

$\bar{L} = \{\epsilon\}$

Q: find lang accepted by ε-NFA below



a) $\{a,b\}^*$ b) a^* (or) b^* c) $\{a,b\}^+$ d) None

↓

$a^* + b^*$

05/01/20

Regular Sets & Regular Expressions:

→ The languages which are accepted by FA are called regular sets or regular languages.

Eg: Set of all strings over $\{0,1\}$ containing 000, ending with 00, even no of 0's etc.

Regular Expressions:

→ It is possible to express all regular languages in mathematical expressions using the operations

(i) union (+ or |)

(ii) Concatenation (.)

(iii) Kleen closure (*)

→ Let Σ be the input alphabet. We can define regex exp over Σ recursively as follows

(i) $\emptyset$ is regex denotes empty language

(ii) ϵ is regex denotes empty string.

(iii) 'a' is a regex such that $\forall a \in \Sigma$

(iv) Let r_1, r_2 be two regex, then

r_1+r_2 , $r_1 r_2$, r_1^* are also regex.
(union) (concate) (kleen closure)

(v) The expressions which are obtained by using the rules 1 to 4 will also form regular expressions

Identities for regular expressions :

1. $\epsilon R = R\epsilon = R$
2. $\phi R = R\phi = \phi$
3. $\phi + R = R + \phi = R$
4. $R+R = R$
5. $R \cdot R = R^2$
6. $(R^*)^* = R^*$
7. $RR^* = R^*R = R^+$
8. $\epsilon + RR^* = RR^* + \epsilon = R^*$
9. $\epsilon + R^+ = R^*$
10. $P(Q+R) = PQ + PR$
11. $(Q+R)P = QP + RP$
12. $(P+Q)^* = (P^*Q^*)^* = (P^*+Q^*)^*$
13. $(\epsilon+R)^* = (R+\epsilon)^* = R^*$

Union is commutative but concatenation is not
i.e., $P+Q = Q+P$, $PQ \neq QP$

Both Union & Concatenation are associative
i.e., $(P+Q)+R = P+(Q+R)$
$(PQ)R = P(QR)$

Regular set	Regular Exp
$L = \{\}$	$r = \phi$
$L = \{\epsilon\}$	$r = \epsilon$
$L = \{a\}$	$r = a$
$L = \{ab\} \cup \{ba\}$	$r = ab+ba$ (or) $ab \mid ba$
$L = \{aa, ba, bb\}$	$r = aa+ba+bb$

→ $L = \{aa, ab, ba, bb\}$	$R = (a+b)(a+b) = (a+b)^2$
→ $L = \{a,b\}^* \cup \{aa, ba\}$	$(a+b) + (aa+ba)$
→ $L = \{a\}^*$	a^*
→ $L = \{a,b\}^*$	$(a+b)^*$
→ $L = \{a,b\}^* ab$	$(a+b)^* ab$

→ Give regexp over $\{0,1\}$ for following languages

(i) All strings ending with 00 → $(0+1)^* 00$

(ii) All string containing 000 → $(0+1)^* 000 (0+1)^*$

(iii) All strings begin with '1' and end with '0' → $1(0+1)^* 0$

(iv) Contains exactly 2 0's → $1^* 0 1^* 0 1^*$

(v) Contains atmost 2 0's → $1^* + 1^* 0 1^* + 1^* 0 1^* 0 1^*$

~~$\simeq 1^* (0+1) 1^* (0+1) 1^*$~~

(or)

$1^* (\epsilon + 0 + 1) 1^* (\epsilon + 0 + 1) 1^*$

(vi) containing atleast 2 zeroes : $(0+1)^* 0 (0+1)^* 0 (0+1)^*$

(or)

$1^* 0 1^* 0 (0+1)^*$

(or)

$(0+1)^* 0 1^* 0 1^*$

(or)

$1^* 0 (0+1)^* 0 1^*$

(vii) strings of length 3 : $(0+1)^3$

(viii) strings of length 3 or more : $(0+1)^3 (0+1)^*$

(ix) length 3 or less : $\epsilon + (0+1) + (0+1)^2 + (0+1)^3 \simeq (\epsilon + 0 + 1)^3$

(x) containing even no of zeroes : $(1^*01^*01^*)^*1^*$

$\sim (1^*01^*0)^*1^*$

$\sim 1^*(01^*01^*)^*$

(xi) containing odd no of zeroes : ~~$(1^*01^*0)^*0$~~

$1^*0(1^*01^*0)^*1^*$

$1^*(01^*01^*)01^*$

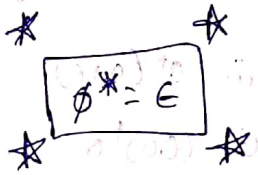
(xii) even 0's & even 1's

$$[00+11+(01+10)(00+11)^*(01+10)]^*$$

→ Consider the following regular expressions

$l_1 = a$

$l_2 = \{\}$



find $l_1 l_2^* + l_1$

$l_2^* = \epsilon$

$\Rightarrow l_1 l_2^* + l_1 = a \quad l_1 = a$

→ $1^* + 00^*1^* = $?

a) 0^*1^* b) 1^*0^* c) $(0+1)^*$ d) None

$1^* + 00^*1^* = (\epsilon + 00^*)1^* = (\epsilon + 0^+)1^* = 0^*1^*$

→ Let $l_1 = a$ $l_2 = \phi$ find $l_1^* l_2^* + l_2^*$

$(l_1^* + \epsilon) l_2^* = l_1^* l_2^* = a^* \phi^* = a^* \epsilon = a^*$

→ Consider the following languages, give regular expressions.

1. $\{a^{2i} \mid i>0\}$ ⟶ $(aa)^+$

2. $\{a^{2i+1} \mid i \geq 0\}$ ⟶ $a(aa)^*$

3. $\{a^n b^m \mid n,m \geq 0\}$ ⟶ $a^* b^*$

4. $\{a^{n+1} b^{m+2} \mid n,m>0\}$ ⟶ ~~$aa^* bbb^*$~~ $aaa^* bbbb^*$

5. $\{a^{2l} b^{3m} c^{4k} \mid l,m,k \geq 0\}$ ⟶ $(aa)^*(bbb)^*(cccc)^*$

6. $\{a^{2n+1} b^{3m+2} c^{4k+4} \mid n,m,k \geq 0\}$ → $a(aa)^* bb(bbb)^* cccc(cccc)^*$

7. $\{a^n b^m \mid n=m \geq 0\}$

This is not a regular language and hence we can't write regex

→ Consider the following regular expressions

(i) $0^*(0+\epsilon)$

(ii) $(00)^* 0$

(iii) $(00)^*$

(iv) 0^*

which of the above are equal

a) i & iii b) ii & iii c) i & iv d) None

$0^*(0+\epsilon) = 0^*0 + 0^*$

$= 0^+ + 0^* = 0^*$

$(00)^*(0+\epsilon) = \underbrace{(00)^* 0}_{(\text{odd})} + \underbrace{(00)^*}_{(\text{even})} = 0^*$

(2013)

→ Consider the following regexp

$$a^*b^*(ba)^*a^*$$

Find min length of string which do not present in above lang.

0	1	2	3
ϵ	a	aa	aaa
	b	ab	aab
		ba	aba
		bb	abb
			baa
			babX

∴ bab ⇒ 3

→ ~~The~~ Q: In the above language how many strings of length 4 can't be produced?

abab
bbab
baab
babb

Ans

→ The regexp denoting set of all strings not containing two consecutive 0's is given by

a) $(1+01)^*(\epsilon+0)$

b) $(0+10)^*(\epsilon+1)$

c) $(1+01)^*$

d) $(\epsilon+0)(101)^*(\epsilon+0)$

$L = \{\epsilon, 0, 101, 010, 110, 011, 111, 1111, \ldots\}$

a ⇒ $\{\epsilon, 0, 1, 101, 011, 110 \ldots\}$

b ⇒ 100 belongs X

c ⇒ 0 does not belong X

d ⇒ 00 belong X

Q:
Which one of the following regexp over {0,1} denotes the set of all strings not containing 100 as substring

a) $0^*(1+0)^*$

b) 0^*1010^*

c) $0^*1^*01^*$

d) $0^*(10+1)^*$

req $l = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \ldots\}$

a => $\{\epsilon, 0, 1, 0100\}$ ✗

b => $\{\underset{\times}{\epsilon} \ldots\}$

c => $\{\underset{\times}{\epsilon}, \ldots\}$

d => $\{\epsilon, 0, 1, 00, 01, 10, 11, 000 \ldots\}$ ✓
100 ✗

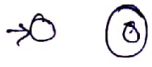
∴ $0^*(10+1)^*$

Regular Exp to Finite Automata :

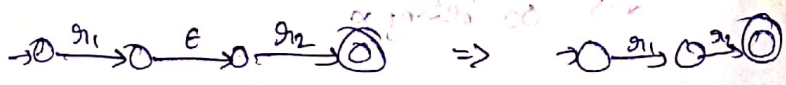
- For every regex r, we can construct an equivalent FA M such that $l(M) = l(r)$ i.e., lang accepted by M is equivalent to lang generated by r.

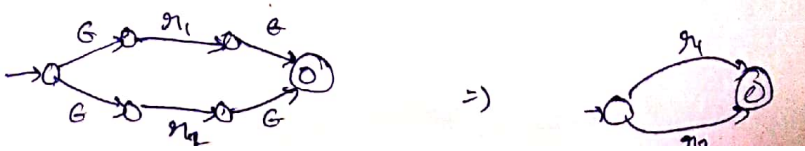
Synthesis Method :

Decomposition :

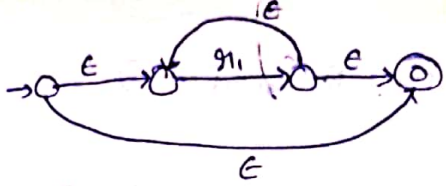
$l = \phi$ => 

$l = \epsilon$ => 

$l = r_1 r_2$ => 

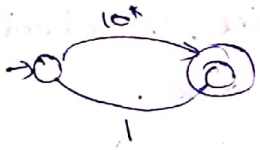
$l = r_1 + r_2$ => 

$L = r_1^*$ ⇒

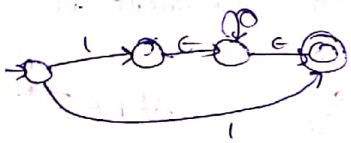


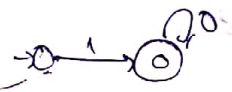
⇒ 

→ let $r_1 = 10^* + 1$



⇒ 

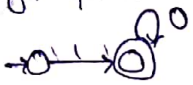
⇒ 

⇒ 

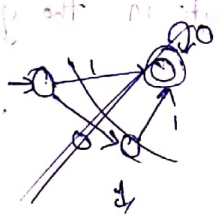
$r_1 = 10^* + 1$

$= 1(0^* + \epsilon)$

$= 10^*$

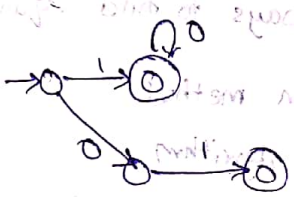


→ $r_1 = 10^* + 01$



↓

(This is wrong)



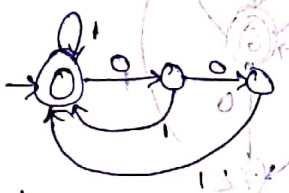
→ $r_1 = (1+01+001)^*(\epsilon+0+00)$

The lang description for above lang is

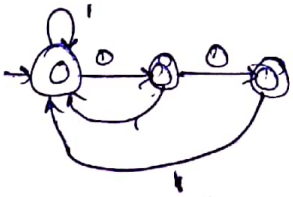
all strings over $\{0,1\}$ which do not more than 2 consecutive 0's.

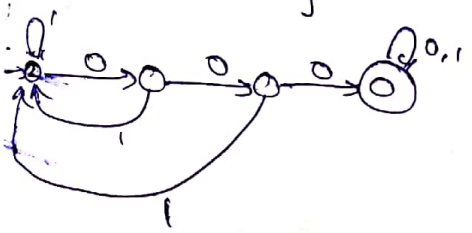
So we can draw complementation of FA of substring 000

for $(1+0(+001)^*$

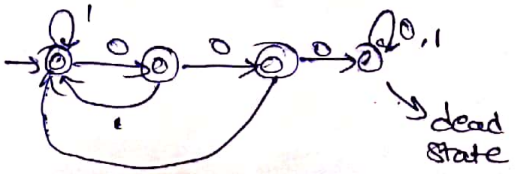


for $(1+01+001)^*(\epsilon+0+00)$





↓ complements



Minimum State DFA (Reduced FA)

→ For every FA 'M' we can construct an equivalent minimum state DFA that performs same task as M does i.e., the main objective of minimization of DFA is reducing complexity of actual machine. The reduced one will also perform the same task with min no of states.

→ Two states P & Q are said to be equivalent iff

$$\left(\delta(P,x), \delta(Q,x)\right) = (R,S) \quad , \quad \forall x \in \Sigma^*$$

where R & S are two final states

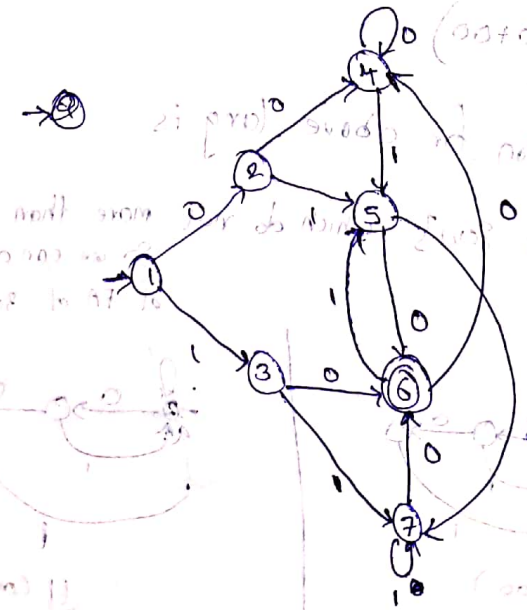
(or)

R & S are two non-final states

There are multiple ways to find equivalent state in the given DFA

(i) Tabulation method
(ii) partition algorithm
(iii) knowledge based

Eg: Find the min state DFA corresponding to machine 'M' shown below



	0	1
1	2	3
2	4	5
3	6	7
4	4	5
5	6	7
6*	4	5
7	6	7

Partition Alg:

$P_0 = \{1,2,3,4,5,6,7\}$ (All states)

Partition $P_1 = \{\{NF\}, \{F\}\}$ (Divide into final & non-final)

$P_1 = \{\{1,2,3,4,5,7\}\{6\}\}$

Partition $P_2 = \{\{1,2,4\}, \{3,5,7\}, \{6\}\}$

Partion $P_3 = \{\{1,2,4\}, \{3,5,7\}, \{6\}\}$

(Partition should continued until we obtain a partition similar to previous one)

① ③ ⑤

while partition take two states from one set and checked

Let these states be q_1, q_2

Now find

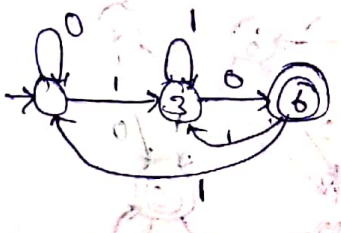
$\delta(q_1, a)$ $\delta(q_2, a)$

$\forall a \in \Sigma$

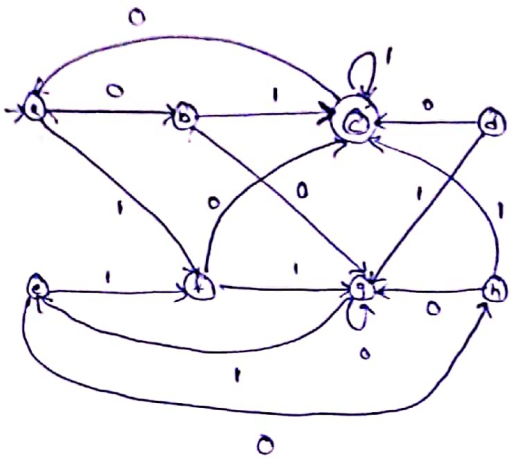
Now if both $\delta(q_1,a)$ & $\delta(q_2,a)$ produces results from same set then put q_1, q_2 in same set.

Otherwise divide them.

	0	1
→1	1	3
3	6	3
6*	1	3



→ Minimize the following DFA



	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

Tabulation method:

Check for same entries of 0 & 1 in two rows, Now merge them and draw a new table. Now repeat the process until there are no two same entries

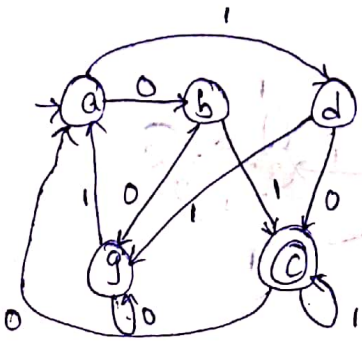
$P_1 = \{\{a,b,d,e,f,g,h\}, \{c\}\}$

$P_2 = \{\{a,e,g\}, \{b,h\}, \{d,f\}, \{c\}\}$

$P_2 = \{\{a,e\}, \{g\}, \{b,h\}, \{d,f\}, \{c\}\}$

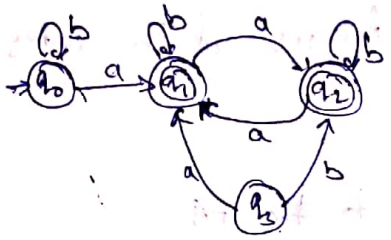
$P_3 = \{\{a,e\}, \{g\}, \{b,h\}, \{d,f\}, \{c\}\}$

a g b d c



	0	1
→ a	b	d
b	g	c
c*	a	c
d	c	g
g	g	a

→ Minimize



(Since q_3 is unreachable remove it)

	a	b
→q_0	q_1	q_0
q_1*	q_2	q_1
q_2*	q_1	q_2
q_3	q_1	q_2

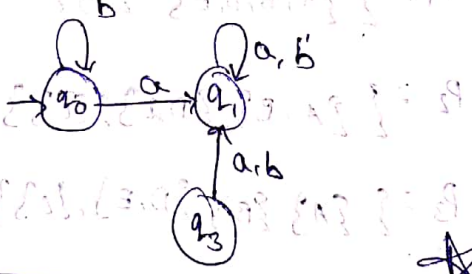
In this problem we can't find any identical entries and hence ~~we can~~ using tabulation method is ~~difficult~~ difficult. So we prefer partition method

$P_1 = \{\{q_0, q_3\}, \{q_1, q_2\}\}$

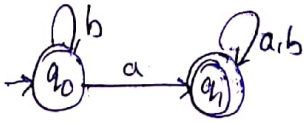
$P_2 = \{\{q_0\}, \{q_3\}, \{q_1, q_2\}\}$

$P_3 = \{\{q_0\}, \{q_3\}, \{q_1, q_2\}\}$

	a	b
q_0	q_1	q_0
q_1	q_1	q_1
q_3	q_1	q_1



Here q_3 is unreachable states, so remove it too



↳ This step can be performed even before applying actual partition algorithm

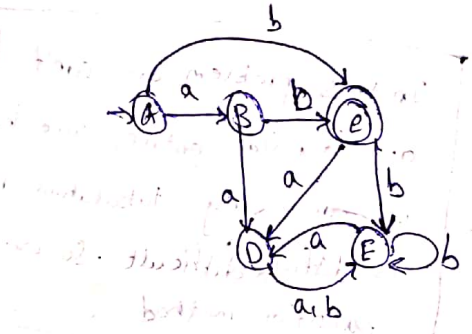
→ The minimization of FA procedures applicable for DFA only

i.e., if given machine is NFA, it should be transformed into DFA before application of any minimization procedures.

→ We have to ensure that the DFA machine is free from ~~one~~ unreachable states.

→ If there is any unreachable state in DFA, it should be ignored and procedure should be applied for rest of the states

Eg: Find min state DFA corresponding to the FA



	a	b
→A	B	C
B	D	C
C*	D	E
D	E	E
E	D	E

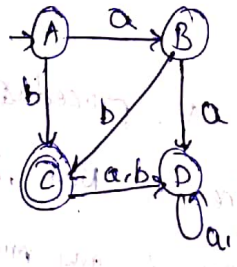
$P_1 = \{ \{A, B, D, E\}, \{C\} \}$

$P_2 = \{ \{A, B\}, \{D, E\}, \{C\} \}$

$P_3 = \{ \{A\} \{B\} \{D, E\}, \{C\} \}$

$P_4 = \{ \{A\} \{B\} \{D, E\} \{C\} \}$

	a	b
→A	B	C
B	D	C
C*	D	D
D	D	D

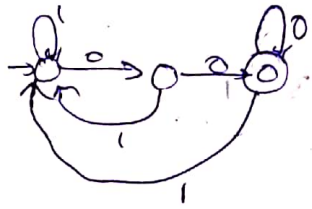


→ Finite language always contains a dead state.

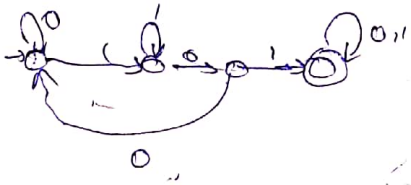
→ Infinite language may or may not have a dead state

Q: The min state DFA to recogonize set of all strings over $\{0,1\}$ such that every string end with 00. has ___3___ no of states.

Sol:-

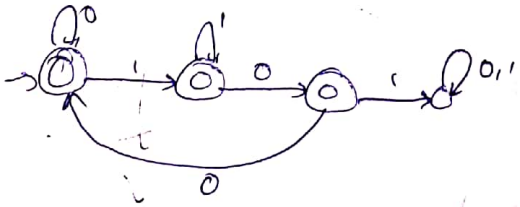


Q: Determine minstate DFA to recogonize set of all strings containing 101 substring.



∴ 4 states

For do not contain 101



∴ 4 states (including dead state too)

→ The min state DFA to recogonize set of all strings over Σ, ending with some certain string of length 'n' has n+1 states

→ The no of states in DFA to recogonize the language set of all strings over Σ contain a substring x such that $|x|=n$ has n+1 states

→ The min state DFA to accept language L has n states, the DFA for L' also have same no of states

→ The min state DFA to recognise set of all strings over {0,1} such that second symbol from right end is one.

Sol:

2nd symbol from right end is 1

n = 2

no of states = $2^n = 2^2 = 4$

no of final state = $2^{n-1} = 2$

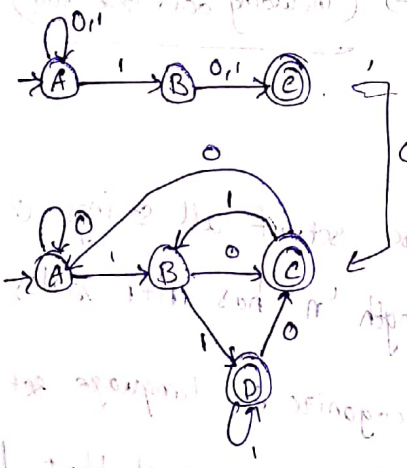
Transition table:

	0	1
→ q_0	q_0	q_1
q_1	q_2	q_3
q_2*	q_0	q_1
q_3*	q_2	q_3

∴ 4 states

Method 2:

Regex: $(0+1)^* 1 (0+1)$

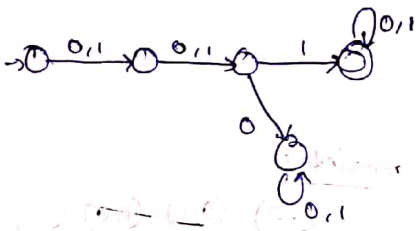


→ Min no of states in DFA containing nth symbol from right end is contains 2^n states defined over {0,1}

→ Set of all strings over {0,1} has 3rd symbol from left end as 1 has ____ states.

Sol :

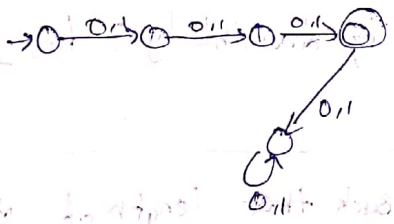
Regex : $(0+1)(0+1)1(0+1)^*$



∴ 5 states

→ The min state DFA to recognize over {0,1} such that nth symbol from left end is 1 is n+2

→ Set of all string of length 3. Min state DFA = 5 states



Reg ex : $(0+1)(0+1)(0+1)$

Remember that finite language always contains a dead state

→ The min state DFA to recognize set of all strings of length n over alphabet {0,1} has n+2 states

→ The min state DFA to recognize set of all strings over {0,1} (or any Σ) of length n or less has n+2 states

→ The min state DFA to recognize set of all strings over $\{0,1\}$ such that length of string is n or more has $n+1$ states

→ The min state DFA to recognize set of all strings over $\{0,1\}$ such that no of 0's is divisible by 2 and no of 1's is divisible by 2 has 4 no of states

Sol:

$2 \times 2 = 4$

remainders

(0,0) (0,1) (1,0) (1,1)

∴ 4 states

0's by 2 < 0, 1

1's by 2 < 0, 1

} 4 combinations

→ The min state DFA to recognize set of all strings over $\{0,1\}$ such that no of 0's is divisible by m & no of 1's is divisible by n has $m \times n$ no of states.

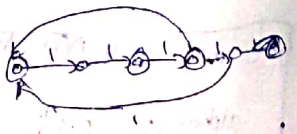
→ The min state DFA to recognize set of all strings over $\{0,1\}$ such that length of the string divisible by 3 has 3 states

→ Set of all strings over Σ such that length of string is divisible by n. This lang's min DFA has n states

Problems:

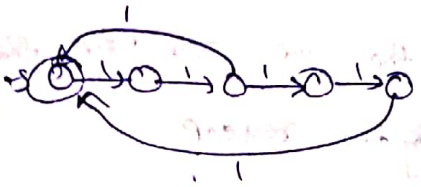
(2006) Consider the regular lang

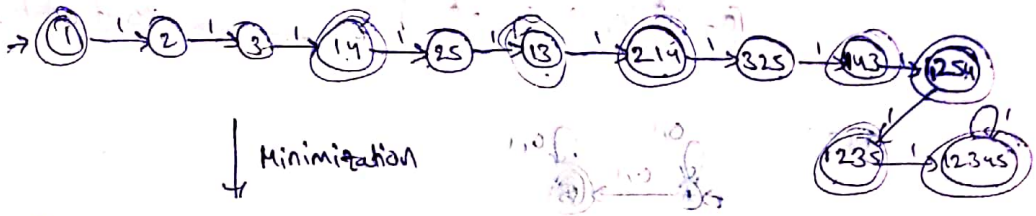
$$L = (111 + 1111)^*$$



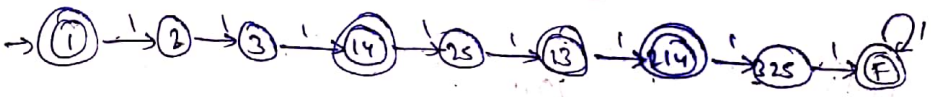
The min no of states in a DFA accepting this lang has ___ states

a) 3 b) 5 c) 8 d) 9





Minimization



∴ 9 states

Method 2:

1 — X

11 — X

111 — 3

1111 — X

11111 — 5

1^6 — 3+3

1^7 — X

1^8 — 3+5

1^9 — 3+3+3

1^{10} — 5+5

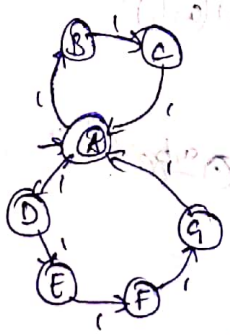
1^{11} — 3+3+5

i.e., More than 8 ones (or) 8 ones

DFA accepts any length

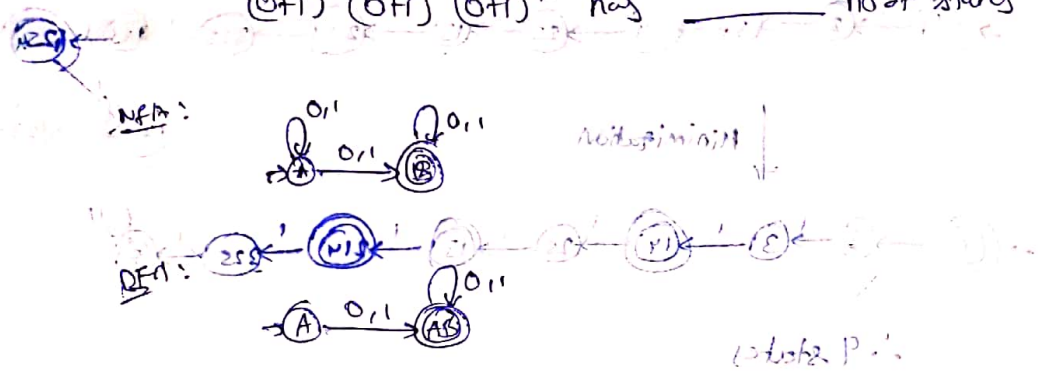
∴ we can draw it with 9 states

DFA can also be drawn as



(2016) Consider the no of states in the min sized DFA that accepts the language defined by the regexp

$(0+1)^*(0+1)(0+1)^*$ has _____ no of states



(2017) 2M Consider the lang L given by the regexp $(a+b)^*b(a+b)^*$ over $\{a,b\}$

The smallest no of states needed in DFA accepting L.

Sol:

This DFA which accepts strings whose 2nd symbol from right end is b

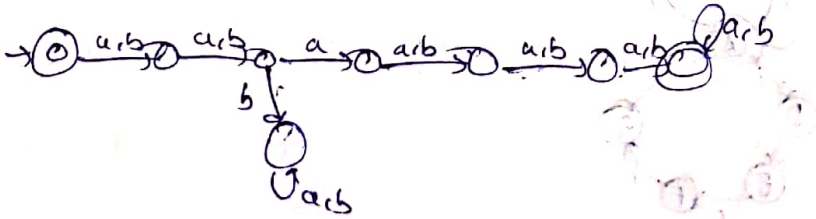
here $n=2$

$2^2 = 4$ states.

(2017) The min possible state of DFA that accept regular lang

$L = \{w_1 a w_2 \mid w_1, w_2 \in \{a,b\}^*$, length of w_1 is 2 and length of w_2 is greater than or equal to 3$\}$

$(a+b)(a+b)\,a\,(a+b)(a+b)(a+b)(a+b)^*$

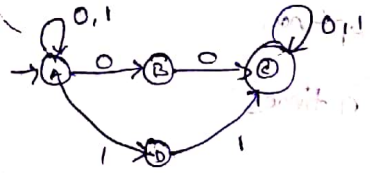


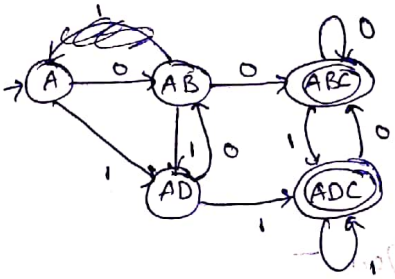
$\therefore$ 8 states

12/01/20

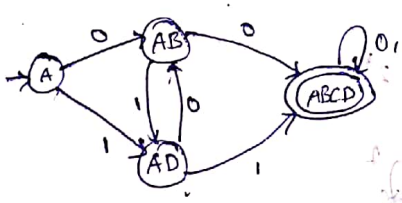
Q: The min state DFA to recogonize set of all strings over $\{0,1\}$ containing the substring 00 (or) 11 has ____ no of states

$$(0+1)^*(00+11)(0+1)^*$$





⇓ Minimization



∴ 4 states

(2011) (N) Definition of the language L with alphabet $\{a\}$ is given as follows

$$L = \{a^{nk} \mid k>0\}$$ and n is positive integer constant.

what is min no of states needed in DFA to recogonise L?

a) $k+1$ b) $n+1$ c) 2^{n+1} d) 2^{k+1}

Put $n=2$

$$\{a^{2k} \mid k>0\}$$

$$L = \{aa, aaaa \ldots\}$$

i.e., n+1 states

→ Let L be lang representing $\Sigma^* 0011 \Sigma^*$ where $\Sigma = \{0,1\}$

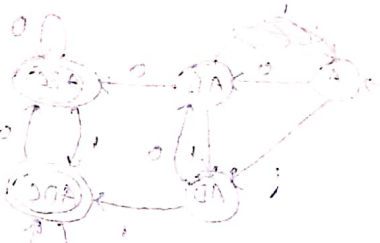
Find min no of states in DFA representing $\bar{L}$.

5 states

→ No of states in min DFA accepting

$(0+1)(0+1) \ldots$ n times

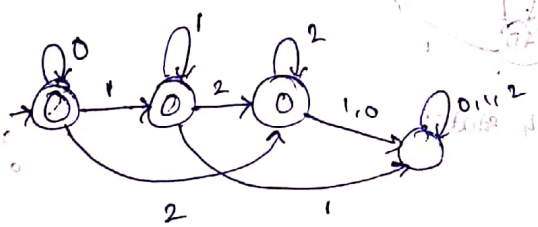
Ans: $n+2$ states.



→ Min no of states for $0^* 1^*$

3 states

→ Min no of states for $0^* 1^* 2^*$



∴ 4 states

(3)

→ Min no of states in $a^* b^* c^* \ldots z^*$

(4)

26+1 = 27 states

$a_1^* a_2^* a_3^* \ldots a_n^*$

n+1 states

→ Which one of the following regexp represent the lang set of all binary strings having 2 consecutive 0's and 2 consecutive 1's

a) $(0+1)^*(00+11)(0+1)^* + (0+1)^*1100(0+1)^*$

✓b) $(0+1)^*(00(0+1)^*11 + 11(0+1)^*00)(0+1)^*$

c) $(0+1)^* 00 (0+1)^* + (0+1)^* 11 (0+1)^*$

d) $00(0+1)^* 11 + 11(0+1)^* 00$

req lang

$(0+1)^* 00 (0+1)^* 11 (0+1)^* + (0+1)^* 11 (0+1)^* 00 (0+1)^*$

$(0+1)^* [00(0+1)^* 11 (0+1)^* + 11 (0+1)^* 00 (0+1)^*]$

$(0+1)^* [00 (0+1)^* 11 + 11 (0+1)^* 00] (0+1)^*$

→ Let $L = \{ w \in \{0,1\}^* / w$ has even no of 1's $\}$

a) $(0^* 1 0^* 1)^*$

b) $0^* (10^* 10^*)^*$

c) $0^* (10^* 1)^* 0^*$

d) $0^* 1 (10^* 1)^* 10^*$

$L = \{\epsilon, 0, 00, 000, 11, 101, 011, 110, \ldots\}$

a => $\{\epsilon, 11 \ldots\}$ 00 ∉

b => $\{\epsilon, 0, 00, 000, 11, 101, 011, 110 \ldots\}$ ✓ 11011 ∈

c => $\{\epsilon, 0, 00, 000, 11, 101, 011, 110, \ldots\}$ 11011 ∉

d => $\{\epsilon, \chi \ldots\}$

(N) → Let (N) $r = 1(1+0)^*$

$s = 11^* 0$

$t = 1^* 0$

be 3 regular expressions. which of the following is true?

(i) $L(s) \subseteq L(r)$ and $L(s) \not\subseteq L(t)$

(ii) $L(r) \subseteq L(s)$ and $L(s) \subseteq L(t)$

(iii) $L(t) \subseteq L(s)$ and $L(s) \subseteq L(r)$

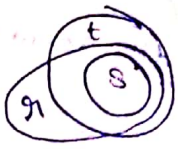
(iv) $L(t) \subseteq L(s)$ and $L(s) \not\subseteq L(r)$

$r = 1(1+0)^*$: { 1, 10, 110 . . . }

$s = 11^*0$: { 10, 110, 1110 . . . }

$t = 1^*0$

= {0, 10, 110, 1110 . . . }



$\therefore L(s) \subseteq L(t)$, $L(s) \subseteq L(r)$ but $L(t) \notin L(r)$

→ Consider the following reg exp

$$r = (a+b^*)^* b (a+b)^*$$

How many strings are there in $l(r)$ of length 3 or less.

Sol:

The lang is containing substring b

i.e., containing atleast one 'b'.

length 0	length 1	length 2	length 3
ϵ	~~0~~ 1	4 - 1 (aa)	8 - 1 (aaa)
0	1	3	7

= 11

→ The reg exp $0^*(10^*)^*$ denote the same set as

a) $(1^*0)^*1^*$

b) $0 + (0+10)^*$

c) $(0+1)^*10(0+1)^*$

d) no of the above

$l = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, \ldots\}$

$a = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, \ldots\}$

$b = \{\epsilon, 0, 1, \ldots\}$

$c = \{\epsilon, 11\}$

Ans: a

→ Let S & T be languages over {a,b} represented by the reg exp $(a+b^*)^*$ and $(a+b)^*$ respectively which of the following is true

a) $S \subset T$

b) $T \subset S$

✓ c) $S = T$

d) $S \cap T = \phi$

$$(a+b)^* = (a^*+b^*)^* = (a^*b^*)^* = (a^*+b)^* = (a+b^*)^*$$

Transformation of Finite Automata to Regular Expression

Methods:

(i) Arden's theorem

(ii) kleen's theorem

(iii) State Elimination method

Arden's Theorem:

Let P, Q & R be 3 reg exp and $Q \neq \phi$. The expression

$R = Q + RP$ have one and only solution

$R = Qp^*$

and p does not contain ϵ

Consider the following FA, M, Determine the reg exp corresponding to it

$0^*11^*0(0+1)^*$

Using Arden's theorem

$A = A0 + \epsilon$

$B = A1 + B1$

$C = B0 + C0 + C1$

> Write all incoming edges of a state including state from which edge is coming and label. For initial state include ϵ.

Now we need to obtain expression at C.

$A = A0 + \epsilon$

$A = \epsilon + A0$

i.e., $R = Q + RP$

$R = QP^*$

$\Rightarrow A = \epsilon 0^* = 0^*$

Now

$B = A1 + B1$

$B = 0^*1 + B1$

i.e., $R = Q + RP$

$R = QP^*$

$B = 0^*11^*$

Now

$C = B0 + C0 + C1$

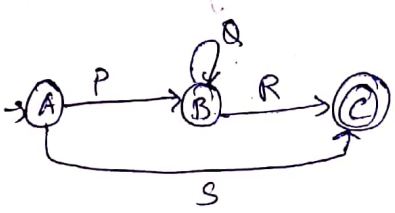
$C = \underset{R}{0^*11^*}\underset{Q}{0} + \underset{R}{C}\underset{P}{(0+1)}$

$C = 0^*11^*0(0+1)^*$

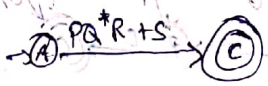
> Solving equations we find the expression at a final state which is the required reg exp. If there are more than one final states then union of expressions at those final states will be the required reg exp

State Elimination Method:

Rule 1:

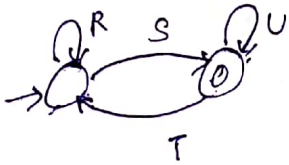


eliminating B



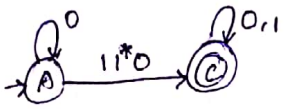
$\therefore$ $R = PQ^*R + S$

Rule 2:



Final expression: $(R + SU^*T)^*SU^*$ (or) $R^*S(U + TR^*S)^*$ } Analyze

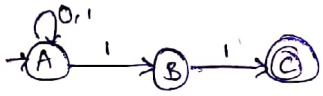
Eg:

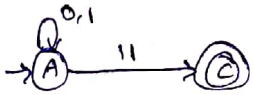


$$r = (0 + \underbrace{11^*0}_{S}\,\underbrace{(0+1)^*}_{R}\,\underbrace{\phi}_{T})^* \; 11^*0\,(0+1)^* \quad \text{[from rule 2]}$$

$$r = 0^*11^*0(0+1)^*$$

Eg: Find the reg exp of FA 'M' shown in fig





$(0+1)^*11$

$A = A0 + A1 + \epsilon$

$B = A1$

$C = B1$

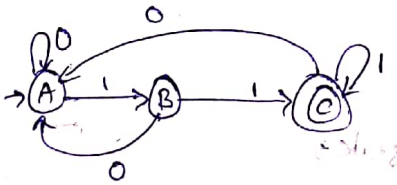
Now $A = \epsilon + A(0+1)$

$A = \epsilon(0+1)^* = (0+1)^*$

$B = A1 = (0+1)^*1$

$C = B1 = (0+1)^*11$

Eg: Find reg Exp



~~A = A0 +~~

$A = A0 + C0 + B0 + \epsilon$ ↳ (1)

$B = A1$ — (2)

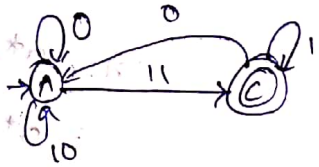
$C = B1 + C1$ — (3)

$C = B1 + C1$

$C = A11 + C1$ (from (2))

$C = A111^*$ — (4)

~~Subs~~



$((0+10) + 111^*0)^*111^*$

$= (0+10+111^*0)^*111^*$

Substitute ④ & ② in ①

$$A = A0 + A10 + A111^*0 + \epsilon$$

$$A = \epsilon + A(0 + 10 + 111^*0)$$

$$A = \epsilon(0 + 10 + 111^*0)^*$$

$$A = (0 + 10 + 111^*0)^*$$

$$C = A111^*$$

$$C = (0 + 10 + 111^*0)^* 111^*$$

✱

5:- ~~firing~~ Find reg exp

$$A = A0 + \epsilon$$

$$B = A1 + B1$$

$$C = B0 + C0 + C1$$

$$A = \epsilon + A0$$

$$A = \epsilon 0^* = 0^*$$

$$B = 0^*1 + B1$$

$$B = 0^*11^*$$

Since Both A & B are final states

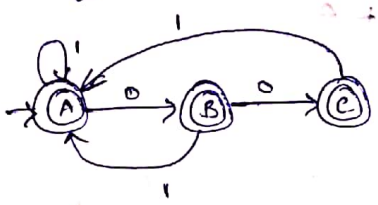
$$R = 0^* + 0^*11^*$$

$$= 0^*(\epsilon + 11^*)$$

$$= 0^*(\epsilon + 1^+)$$

$$= 0^*1^*$$

Eg:- Find the reg exp for DFA



$A = A1 + B1 + C1 + \epsilon$

$B = A0$

$C = B0$

$C = B0 = A00$

$\Rightarrow A = A1 + B1 + C1 + \epsilon$

$A = \epsilon + A1 + A01 + A001$

$A = \epsilon + A(1 + 01 + 001)$

$A = \epsilon(1 + 01 + 001)^*$

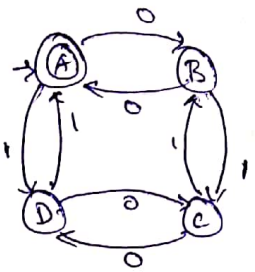
$C = A00$

$= (1 + 01 + 001)^*00$

$B = A0 = (1 + 01 + 001)^*0$

$A + B + C = (1 + 01 + 001)^*(\epsilon + 0 + 00)$

Eg:



$A = B0 + D1 + \epsilon$

$B = A0 + C1$

$C = B1 + D0$

$D = A1 + C0 \Rightarrow D = (B0 + D1)1 + (B1 + D0)0$

$D = (B01 + D11) + (B10 + D00)$

$D = (B01 + B10)^* (11 + 00)^*$

$A = B0 + (B01 + B10 + 1)(00+11)^* 1 + \epsilon$

$A = B[0 + (01+10+1)(00+11)^* 1] + \epsilon$

$C = B1 + D0$

$= B1 + (B01+B10+1)(00+11)^* 00$

$= B[1 + (01+10+1)(00+11)^* 0]$

$B = A0 + C1$

$= (B[0 + (01+10+1)(00+11)^* 1] + \epsilon)0 + B[1 + (01+10+1)(00+11)^* 0]1$

$= B00 + B[01 + 1 + (01+10+1)(00+11)^* (1+01)]$

We need the expression for A. So we change all the expression into terms of A.

$C = B1 + D0$

$= A01 + C11 + A10 + C00$

$C = A(01+10) + C(00+11)$

$C = A(01+10)(00+11)^*$

Now $B = A0 + C1$

$= A0 + A(01+10)(00+11)^* 1$

$= A[0 + (01+10)(00+11)^* 1]$

Now $D = A1 + C0$

$= A1 + A(01+10)(00+11)^* 0$

$= A[1 + (01+10)(00+11)^* 0]$

$A = \epsilon + B0 + D1$

$= \epsilon + A[0 + (01+10)(00+11)^* 1]0 + A[1 + (01+10)(00+11)^* 0]1$

$= \epsilon + A[00 + (01+10)(00+11)^* 10] + A[11 + (01+10)(00+11)^* 01]$

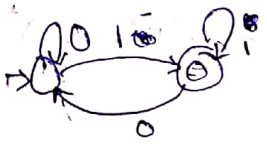
$$A = \epsilon + A[00+11+(01+10)(00+11)^*(10+01)]$$

$$A = \epsilon[00+11+(01+10)(00+11)^*(10+01)]^*$$

$$A = [00+11+(01+10)(00+11)^*(10+01)]^*$$

Problems:

(2014) which of the reg exp given below represent the following DFA



I. $0^*1(1+00^*1)^*$

II. $0^*1^*1 + 11^*0^*1$

III. $(0+1)^*1$

a) I & II b) I & III c) II & III d) I, II, III

$R_1 = (0+10^*0)^*10^*$

$L = \{1, 01, 11, 111, 001, 011, 101 \ldots\}$ ending with 1

I = $0^*1(1+00^*1)^*$ clearly directly represents DFA

II = $(0^*1^* + 11^*0^*)1$

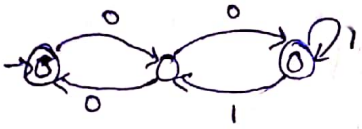
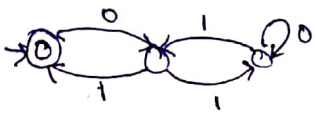
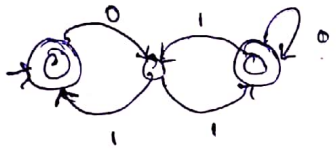
1011 is not accepted by II

III = $(0+1)^*1$ this is ending 1 clearly

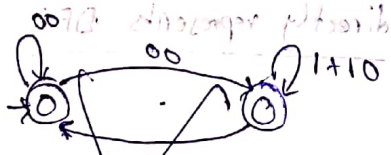
$\therefore$ I & III

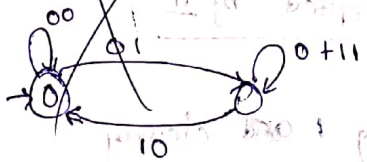
(2008) Match the following DFAs with the reg exp.

(2)

NFA	Reg Exp
P) 	1. $\epsilon + 0(01^*1 + 00)^*01^*$
Q) 	2. $\epsilon + 0(10^*1 + 00)^*0$
R) 	3. $\epsilon + 0(10^*1 + 10)^*1$
S) 	4. $\epsilon + 0(10^*1 + 10)^*10^*$

	P	Q	R	S
a)	2	1	3	4
b)	1	3	2	4
✓c)	1	2	3	4
d)	3	2	1	4

P) 

Q) 

$\epsilon + [00 + 01(0+11)^*10]^*$

$\epsilon + [0(\epsilon + 1(0+11)^*10)]^*$

P → 001

- in (2) it always ends with 0

∴ eliminate opt (a)

P → 00 in (3) ends with 1

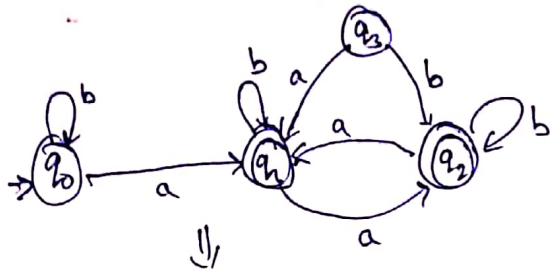
∴ eliminate opt (d) ⇒ P = 1

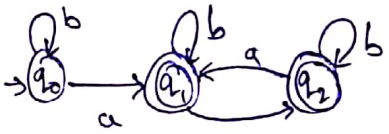
now Q ends with 0 or ε

now R ends with 0

∴ C ✓

Q: Consider the following FA





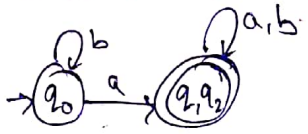
a) $b^*ab^*ab^*ab^*$

b) $(a+b)^*$

✓ c) $b^*a(a+b)^*$

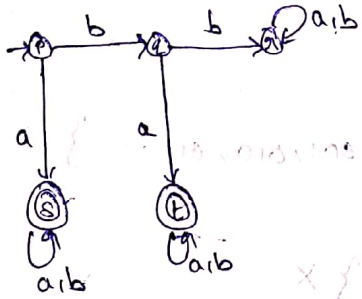
d) $b^*ab^*ab^*$

Minimize FA



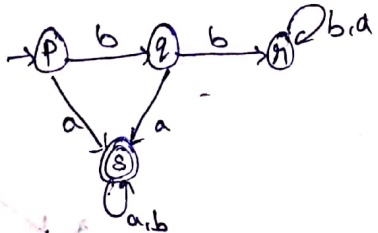
$\Rightarrow b^*a(a+b)^*$

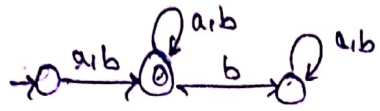
Q: Deterministic FA over $\Sigma=\{a,b\}$

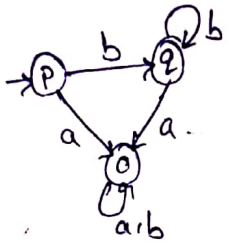


Which of the following FSM is valid min DFA $(a+ba)(a+b)^*$

a)



b) 

c) 

d) None;

from fig $r_1 = a(a+b)^* + ba(a+b)^* = (a+ba)(a+b)^*$

b) => accepts string starting with bb

c) also accepts few string staring with bb

∴ a

Q: which of the following reg exp accept set of all strings not containing 100 as substring

a) $0^*(1+0)^*$

b) 0^*1010^*

c) $0^*1^*01^*$

d) $0^*(10+1)^*$

$L = \{\epsilon, 0, 1, 00, 01, 10, 11, 001, 010, 011 \dots\}$

a) => $\{\epsilon, 0, 1, 00, 01 \dots\}$ ✗

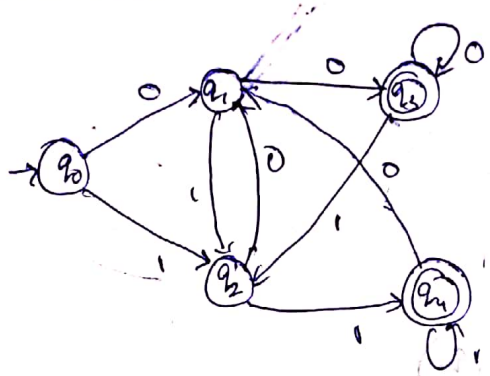
b) => $\{\epsilon \dots\}$ ✗

c) => $\{\epsilon \dots\}$ ✗

d) => $\{\epsilon, 0, 1, 00, 01, 10, 11, 001, 010, 011 \dots\}$ ✓

Material problems

(23)



(24) question means that number divisible by 29099

$\therefore$ 29099 no of states

(25)

~~(a+b)*ab~~

FSM with Outputs:

An FSM is 6 tuple system

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$

where Q = finite set of states

Σ = i/p alphabet

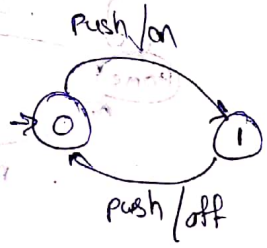
Δ = o/p alphabet

δ = transition function

λ = o/p function

q_0 = initial state

Behavior of digital computer switch



$Q = \{0, 1\}$

$\Sigma = \{push\}$

$\Delta = \{on, off\}$

δ: $\delta(0, push) = 1$

$\delta(1, push) = 0$

λ: $\lambda(0, push) = on$

$\lambda(1, push) = off.$

$q_0 : 0$

This FSM with o/p is classified into two types

1. Mealy machine
2. Moore machine

Mealy machine:

It is a 6 tuple system

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

where the o/p function $\lambda : Q \times \Sigma \rightarrow \Delta$

i.e., o/p Mealy machine depends on both presents state and i/p

Moore Machine:

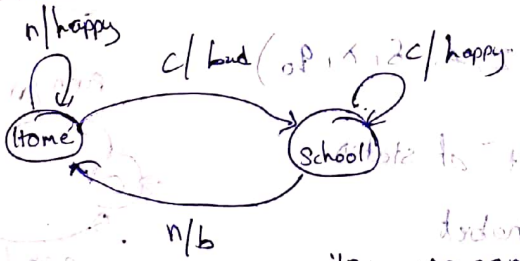
→ 6 tuple system

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

o/p function $\lambda : Q \rightarrow \Delta$

i.e., o/p of the machine depends only on current state.

Eg of Mealy:



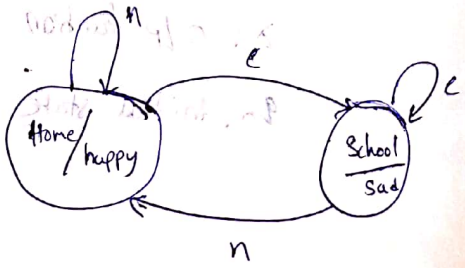
	n	c
H	H,h	S,b
S	H,b	S,h

i/p : w = ccnn

b h b h

Eg of Moore

o/p	PS	n	c
h	→H	H	S
b	S	H	S



i/p:

w = n c c n

H → H → S → S → H

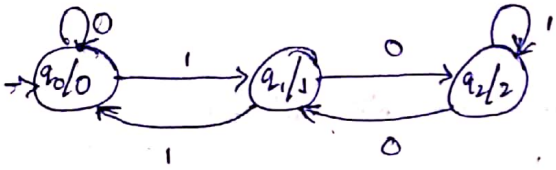
h h b b h

→ For mealy machine if length of i/p is n then length of o/p is n
→ For moore machine if length of i/p is n then length of o/p is n+1

→ Design Moore & Mealy machines to recognize residue of mode 3 numbers

Eg: for the binary i/p integer i/p

(N)

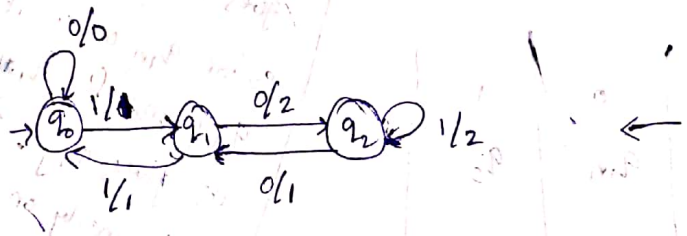


o/p	PS	0	1
0	→q_0	q_0	q_1
1	q_1	q_2	q_0
2	q_2	q_1	q_2

1010 → q_2 i.e., 2 (remainder)

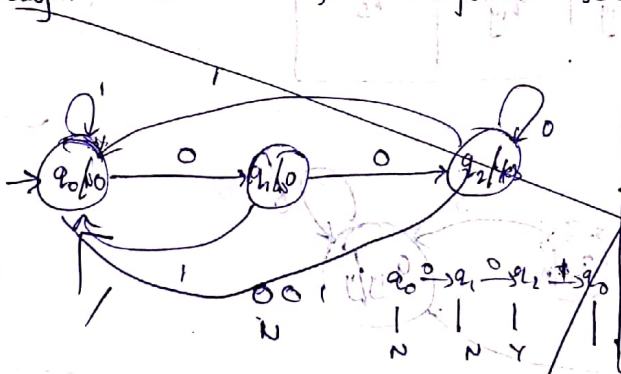
~~Mealy to Moore~~ Moore to Mealy:

Moore to Mealy



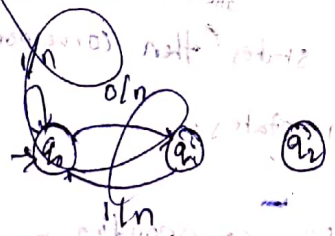
PS	0	1
→q_0	q_0,0	q_1,1
q_1	q_2,2	q_0,0
q_2	q_1,1	q_2,2

Eg: Design Mealy & Moore machines to recognize set of all strings ending with 00



o/p	P.S	0	1
NO	→q_0	q_1	q_0
NO	q_1	q_2	q_0
YES	q_2	q_2	q_0

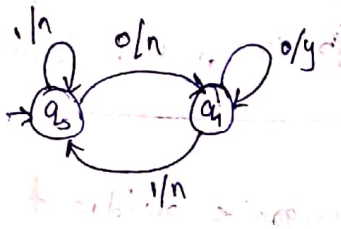
Mealy to Moore



Eg: Design Mealy & Moore for strings ending with 00.

→

(N)



	0	1
$\rightarrow q_s$	q_1, n	q_s, n
q_1	q_1, y	q_s, n

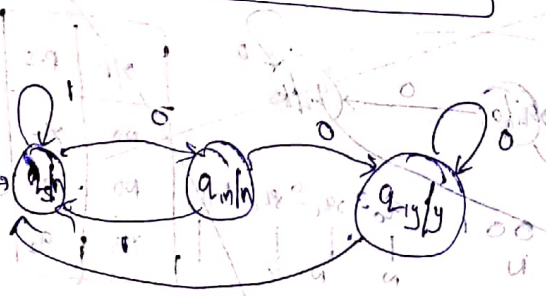
✱ Mealy to Moore:

Step 1: Identify o/p associated with each state

$q_s - n$	~~Step~~ $q_s - n$
$q_1 \overset{n}{\underset{y}{<}}$	$q_{1n} - n$ $q_{1y} - y$

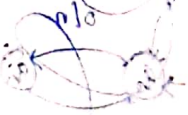
Here for q_1 in mealy on 0 it goes to q_1 giving o/p y so for q_{1n} on '0' it gives q_{1y} and for q_{1y} on 0 its gives q_{1y}. for q_1 in mealy on 1 it goes to q_s giving o/p n. So it goes to q_s

o/p	P.S	0	1
n	$\rightarrow q_s$	q_{1n}	q_s
n	q_{1n}	q_{1y}	q_s
y	q_{1y}	q_{1y}	q_s



Note:

→ If Mealy machine has n no of states and each state associated with m o/p then corresponding moore machine can have atmost n x m states.



→ If moore machine has n states then corresponding mealy machine will have exactly n states

Eg: Design a ~~more~~ moore machine to recogonize residue of mod 5 number over the ~~ter~~ ternary i/p alphabet

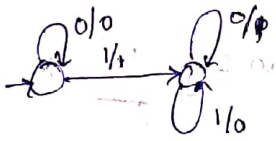
o/p	P.S	0	1	2
0	→ q_0	q_0	q_1	q_2
1	q_1	q_3	q_4	q_0
2	q_2	q_1	q_2	q_3
3	q_3	q_4	q_0	q_1
4	q_4	q_2	q_3	q_4

Same pattern can be use in this case too

Problems:

Q: Consider the following FSM shown in the figure

(N)



It accepts the binary string from least significant bit positions and o/p the result in same order. which of the following is true

a) It computes 1's complement of

b) It increaments i/p pattern by 1.

c) It decreaments i/p patter by 1

d) It computes 2's complements.

001
010
100
0010
0011

Consider ←

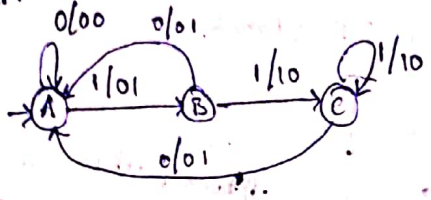
i/p: 0 1 1 0

o/p: 1 0 1 0

0110 $\xrightarrow{1's}$ 1001 $\xrightarrow{2's}$ 1010

∴ 2's complement.

Q: The FSM



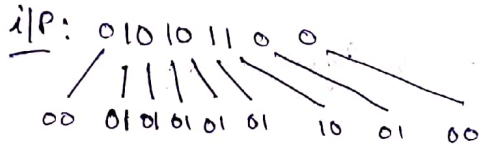
Each edge label is x/y, x stands for 1 bit i/p and y stands for 2 bit o/p

a) outputs the sum of the present and previous bits of the i/p

b) outputs 01 whenever i/p sequence is 11

c) outputs 00 whenever i/p sequence contains 10

d) none



From above i/p & o/p we can say 'a' is correct answer

Q: The FSM in following state table has single i/p 'x' & single o/p 'z'

	x=1	x=0
A	D,0	B,0
B	B,1	C,1
C	B,0	D,1
D	B,1	C,0

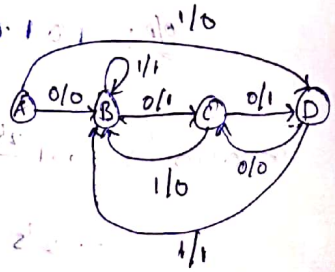
If the initial state is unknown, then shortest i/p sequence to reach final state C is ——→

a) 01 b) 10 c) 101 d) 110

for A

A —1→ D —0→ C, 01

∴ 10

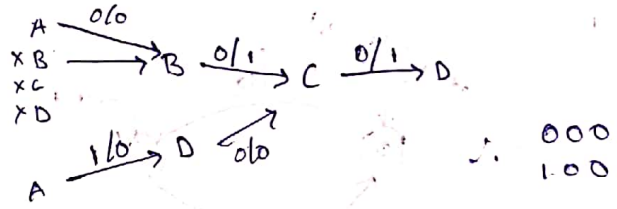


Q: Determine min length of the string to reach the state D with o/p '1' from the initial state A: (from previous question)

Sol:

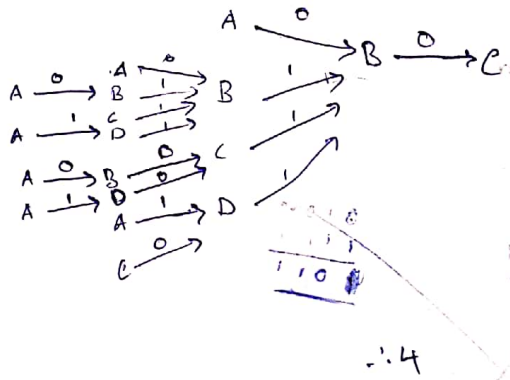
Ans: 000

Q: No of strings of length 3 to reach state D with o/p '1' starting at A.



∴ 000
100

Q: No of strings of length 4 to reach state C with o/p '1' from A



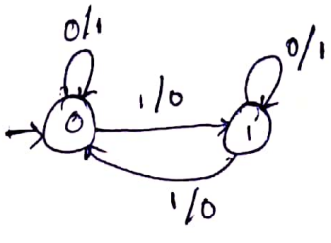
∴ 0110
1110
0010
1010

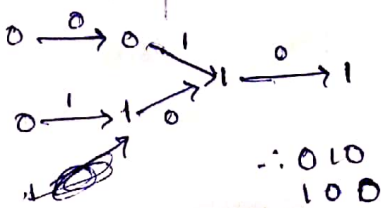
∴ 4

Q: Consider the following

PS	i/p	NS	o/p
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

If initial state of the machine is 0 How many strings of length 3 are existed to reach 1 with o/p '1'





∴ 010
100

Grammar:

A grammar is a mathematical system used to define sentences of the formal languages

A grammar G is a 4 tuple system

$$G = (V, T, P, S)$$

where

V → variables or Non-terminals

T → set of terminals

P → set of productions or re-writing rules

~~Production is~~

S → start symbols.

Eg: Consider

$$G = (\{S, A, B\}, \{a, b\}, P, S)$$

P is given by

$S \to AB$
$S \to BA$
$A \to a$
$A \to aS$
$B \to b$
$B \to bS$

(or)

$S \to AB \mid BA$
$A \to a \mid aS$
$B \to b \mid bS$

} equal no of a's & b's

$\Rightarrow L(G) = \{ab, baba, ba \ldots\}$

Check if below strings belong to above grammar G

(i) baab

$S \to BA$
$\to bA$
$\to baS$
$\to baAB$
$\to baaB$
$\to baab$

(ii) bbaa

$S \to BA$
$\to bSA$
$\to bBAA$
$\to bbAA$
$\to bbaA$
$\to bbaa$

(iii) baaabb

$S \to BA$
$\to bA$
$\to baS$
$\to baAB$
$\to baaSB$
$\to baaABB$
$\to baaaBB$
$\to baaabB$
$\to baaabb$

→ If we check we can't derive baaba using above grammar. So we can conclude the grammar is the lang producing strings with equal no of a's & b's

19/01/20

Eg Consider below grammar

$S \to aB \mid bA$

$A \to a \mid as \mid bAA$

$B \to b \mid bS \mid aBB$

check if below strings belong to the grammar

a) baaabb

$S \to bA$
$\to baS$
$\to baaB$
$\to baaaBB$
$\to baaabB$
$\to baaabb$

b) bbaaab

$S \to bA$
$\to bbAAA$
$\to bbaA$
$\to bbaaS$
$\to bbaaaB$
$\to bbaaab$

c) bbbabaaa

$S \to bA$
$\to bbAA$
$\to bbbAAA$
$\to bbbaAA$
$\to bbbabAAA$
$\to bbbabaAA$
$\to bbbabaaA$
$\to bbbabaaa$

d) bbbabaaa

This string doesn't belong to the grammar

→ Thus the above grammar also represents language consisting of equal no of a's & b's.

Q: The below grammar represents

$S \to aSb \mid \epsilon$

a) $\{a^n b^n \mid n \geq 0\}$

b) $\{a^n b^n \mid n \geq 1\}$

c) $\{a^n b^m \mid m \geq 1, n \geq 0\}$

d) $\{a^n b^m \mid m \geq 0, n \geq 1\}$

$L = \{\epsilon, ab, aabb \ldots\}$

Q: The below grammar represent the language

$S \to aS \mid bS \mid a \mid b$

a) $(a+b)^*$ ✓ b) $(a+b)(a+b)^*$ ✓ c) $(a+b)^*(a+b)$ d) all the above

(2005) Consider below grammar G

(N)

$S \to aA \mid bS \mid b$

$A \to bA \mid aB$

$B \to bB \mid aS \mid a$

Let $N_A(w)$ & $N_B(w)$ denote the no of a's & b's in the string w

The language $L(G) \subseteq (a+b)^*$ generated by G is _____

a) $\{w / N_a(w) > 3N_b(w)\}$

b) $\{w / N_b(w) > 3N_a(w)\}$

c) $\{w / N_a(w) = 3k, k \in \{0,1,2\cdots\}\}$

d) $\{w / N_b(w) = 3k, k \in \{0,1,2,\cdots\}\}$

Seeing options its either b or c

$L = \{b, aaa, abaa, aabba \cdots\}$

↓

Eliminate option (b)

(or)

observing grammar we can see S,A,B as ~~three~~ three states in which S is final state. A every state reading 'b' it remains in same state. Getting 'a' it moves to next state. So we can conclude that above language accepts strings contains no of a's as multiples of 3.

Classification of Grammar (Chomsky Hierarchy)

Type	Alternative name	condition	Language	Eg.	Automata
type 0	unrestricted grammar (UG)	$\alpha \to \beta$ & $\alpha \neq \in$ α, β are arbitrarily long strings	Recursively Enumerable language	$S \to aBbA$ $aA \to bb$ $aB \to b$	Turing Machine (TM)
type 1	Context Sensitive (CSG) Grammar	$\alpha \to \beta$ & $\lvert\beta\rvert \geq \lvert\alpha\rvert$ α, β are arbitrarily long string	Context sensitive Language	$S \to aBbA$ $aA \to bb$ $aB \to bA$	Linear Bounded (LBA) Automata
type 2	Context Free Grammar	$A \to \alpha$ $A \in V$ $\alpha = (V \cup T)^*$	context free Language	$S \to AB \mid BA$ $A \to a \mid as$ $B \to b \mid bS$	Push Down Automata (PDA)
type 3	Regular Grammar	$A \to w \mid wB$ (or) $A \to w \mid Bw$ $A, B \in V$ $w \in T^*$	Regular language	$S \to aS \mid bS \mid a \mid b$	Finite (FA) Automata

★ type 3 ⊆ type 2 ⊆ type 1 ⊆ type 0 ★

→ In type 3 grammar

it is of form

$A \to w \mid wB$ (Right linear grammar)

$A \to w \mid Bw$ (Left - linear grammar)

Eg: what is the type of below grammar

$S \to aA \; bB$

$aA \to bbAb$

$Ba \to bb$

$A \to a$

a) Type 0 ✓ b) Type 1 c) Type 2 d) Type 3

→ For every automata M, we can construct an equivalent grammar G, such that L(M) = L(G)

FA to Reg. Grammar

Consider below FA

V = Node labels = {A, B, C}

T = edge labels = {a, b}

S = initial state = A

finding P:

δ(A, a) = B

A → aB

δ(B, b) = A δ(B, b) = C

B → bA | bC | b

δ(C, a) = C δ(C, b) = b

C → aC | bC | a | b

i.e., A → aB

B → bA | bC | b

C → aC | bC | a | b

for every

δ(A, a) = B

add production

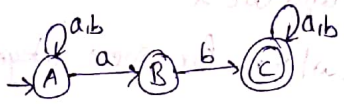
A → aB, if B is non-final

else add

A → aB | a

Eg: Find regular grammar corresponding to the regular Exp

$(a+b)^* ab (a+b)^*$



S → aA | aB | bA

B → bC | b

C → aC | bC | a | b

→ The answer of above eg is a right linear grammar.

Writing it same in left linear form

$A \to Aa|Ba|b$

$B \to Cb|b$

$C \to Ca|cb|a|b$

i.e., language is reversed.

> i.e., If G is right-linear and G' is left linear to G then G is nothing but reverse of G'

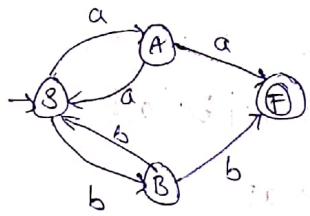
→ Also for every left-linear grammar we can construct an equivalent right linear grammar and vice versa.

Regular Grammar to FA:

Eg: $S \to aA|bB$

(2) $A \to a|aS$

$B \to b|bS$



Pumping lemma for Regular sets:

→ It is a tool used to prove certain languages are not regular

Let L be regular language and $w \in L$

with condition $|w| \geq n$ for some integer n

then $w = xyz$ such that

(i) $y \neq \epsilon$

(ii) $|xy| \leq n$

(iii) $xy^iz \in L \ \forall \ i \geq 0$

> Here n is constant and n denotes no of state in the corresponding FA

Eg: P.T $L=\{a^n b^n / n>0\}$ is not regular

Let $w = a^m b^n$

$xy = a^n = a^m a^{n-m}$, $n \neq m$

$y = a^{n-m}$

$xy^i z$ for $i=0$

$xy^0 z = a^m b^n$

since $m \neq n$

$xy^0 z \notin L$

$\therefore \{a^n b^n / n>0\}$ is not regular language

Note:

we generally apply the following principle to identify whether a given language is regular or not.

(i) All finite language are regular

(ii) All infinite languages, a regular language must have the following

a) A finite Automata

b) Regular expression

Note:

Consider

$L_1 = \{a^{2i} / i \geq 0\}$

$L_1 = \{\epsilon, aa, aaaa, \ldots\}$

* Here diff blw length of successive strings is const.

∴ Regular

$L_2 = \{a^{i^2} / i>0\}$

$L_2 = \{a, aaaa, aaaaaaaaa \ldots\}$

* Here diff blw length of successive strings is not const

∴ Not regular

It is always possible to write regular expressions for single symbol alphabet languages iff there exist const diff blw successive strings of language

Eg:

Identify regularity whether below languages are regular or not

→ $\{a^{2i} \mid i \geq 0\}$ ⟶ regular

→ $\{a^{2^i} \mid i \geq 0\}$ ⟶ not regular

→ $\{a^p \mid p \text{ is prime}\}$ → not regular

→ $\{a^p \mid p \text{ is non-prime}\}$ → not regular

→ $\{a^{n!} \mid n > 0\}$ ⟶ not regular

☆ → $\{a^{i^3} \mid i < 2\}$ ⟶ Regular (∵ finite)

☆ → $\{a^m b^n \mid m, n \geq 0\}$ → Regular

☆ → $\{a^n b^n \mid n \geq 0\}$

here $\Sigma = \{a, b\}$

we have a relationship b/w $a \neq b$ ($i.e., n_a(w) = n_b(w)$) which are members of Σ

∴ It is not regular

→ $\{a^i b^{2j} / i = j > 0\}$ → not regular

→ $\{a^i b^j / i \neq j\}$ → not regular

→ $\{ww^R \mid w \in \{a, b\}^*\}$ → not regular

→ $\{wcw^R \mid w \in \{a, b\}^*\}$ →

c is some terminal

It is non-regular

$\{a^i b^{2j} c^{3k} \mid i, j, k > 0\}$ ⟶ regular

$a^+(bb)^+ (ccc)^+ \equiv aa^* bb(bb)^* ccc(ccc)^*$

$L = \{a^{2i+1} b^{3j+1} c^{4k+1} d^{5m+1} / i, j, k, m > 0\}$ ⟶ regular

Q: Consider following languages

$S_1 = \{0^{2n} \mid n \geq 1\}$ is regular language

$S_2 = \{0^m 1^n 0^{m+1} \mid m \geq 1 \text{ and } n \geq 1\}$ is regular

which of the following statement is correct

a) only S_1 ✓

b) only S_2

c) both S_1 & S_2

d) none

Q: Find similar languages to $L = \{x^n y^n \mid n \geq 1\}$

(i) $E \to xEy \mid xy$

(ii) $xy \mid x^+xyy^+$

(iii) x^+y^+

a) I only ✓ b) I & II c) II, III d) II only

Q: which of the following is regular

(N) a) $\{ww^R \mid w \in \{0,1\}^+\}$

b) $\{ww^Rx \mid x, w \in \{0,1\}^+\} \to L = \{000, 111, 011000 \ldots\}$

c) $\{wxw^R \mid x, w \in \{0,1\}^+\} \to$ reg exp: $[0(0+1)^+0] + [1(0+1)^+1]$

d) $\{xww^R \mid x, w \in \{0,1\}^+\}$

(think) using this reg exp. all possible strings of lang in opt (c) can be written.

Q: which of the following are regular

$L_1 = \{ww \mid w \in \{a,b\}^*\}$

$L_2 = \{ww^R \mid w \in \{a,b\}^*\}$

$L_3 = \{0^{2i} \mid i \text{ is integer}\}$ $L_4 = \{0^{i^2} \mid i \text{ is an integer}\}$

a) L_1, L_2 b) L_2, L_3, L_4 c) L_3, L_4 d) L_3 ✓

Closure Properties of Regular sets:

Let l_1, l_2 be 2 regular language
Regular languages are closed under below operations.

~~1. Regular Union~~

1. Union

$l_1 \cup l_2$ is regular

2. Intersection

$l_1 \cap l_2 = \overline{\bar{l}_1 \cup \bar{l}_2}$ is regular

3. Concatenation

$l_1 l_2$ is regular

4. Complementation

$\bar{l}_1$ is regular

5. kleen closure

l_1^* is regular [Also positive closure]

6. Reversal operation

l_1^R is regular

To obtain FA for l_1^R first design FA for l_1. Now reverse the direction of edges of FA of l_1 and make final state as initial state and initial state as final state.
If more than one final state exists the create new state (initial) and put an ϵ-transition to all final states of FA of l_1.

7. Homomorphism

If H is homomorphism, then $H(l_1)$ is also regular

8. Inverse Homomorphism

If H is homomorphism, then $H^{-1}(l_1)$ is also regular

9. Substitution

If S is a substitution, then $S(l)$ is regular

10. Right Quotient

l_1/l_2 is regular

12. Difference

$L_1 - L_2 = L_1 \cap \bar{L}_2$ is also regular

13. Prefix (INIT operator)

To obtain FA for prefix of FA of L_1, make all states of the FA (except dead state) as final states.

Note:

Union, intersection etc. operation on two regular languages gives a regular language. But atleast if one of the language is non-regular then application of these operations may result in either regular or non-regular language.

for example,

$L_1 = \{a^n b^n \mid n>0\}$ ..... not regular

$L_2 = \{a^n b^m \mid n,m>0\}$ ..... regular

now $L_1 \cup L_2 = L_2$ i.e., regular

also $L_1 \cap L_2 = L_1$ i.e., non-regular

Homomorphism:

It refers to substitution of a string, which is defined on another alphabet

$\Sigma_1 \longrightarrow \Sigma_2$

Eg: (N)

$\Sigma_1 = \{a,b\}$ $\Sigma_2 = \{0,1,2\}$

Let h be homomorphism

$L = \{ab, ba\}$

$h(a) = 011$ $h(b) = 122$

Now $h(L) = \{011122, 122011\}$

Inverse Homomorphism

Eg: (N)

$\Sigma_1 = \{a,b,c\}$ $\Sigma_2 = \{0,1\}$

$h(a) = 01$ $h(b) = 10$ $h(c) = 0$

$h^{-1}(1001) = \{ba\}$ $h^{-1}(10001) = \{bca\}$ $h^{-1}(100100) = \{bacc, bcbc\}$

while calculating h^{-1} we'll find all possibilities

Right Quotient:

Right quotient $l_1/l_2 = \{x / xy \in l_1 \text{ & } y \in l_2\}$

Let $l_1 = \{rot, carrot, parrot\}$

$l_2 = \{rot\}$

$l_1/l_2 = \{\epsilon, par, car\}$

Eg: ① $l_1 = 010^*$ $l_2 = 0^*$

~~$l_1/l_2 = \{01, \epsilon\}$~~

$l_1 = \{01, 010, 0100, 01000 \ldots\}$

$l_2 = \{\epsilon, 0, 00, 000, \ldots\}$

$l_1/l_2 = 010^*$

Eg: ② $l_1 = 0^*10^*$ $l_2 = 0^*1$

$l_1 = \{1, 01, 10, 010, 0010, 00010, 000100 \ldots\}$

$l_2 = \{1, 01, 001, 0001, 00001 \ldots\ldots\}$

$l_1/l_2 = \{\epsilon, 0, 00, 000 \ldots\} = 0^*$

Eg: ③ $l_1 = 0^*10^*$ $l_2 = 10^*1$

$l_1/l_2 = \phi = \{\}$

Because y in l_2 has two 1's but the whole xy itself in l_1 has only one 1.

Q: l_1, l_2 are two languages. Find a true statement.

a) $l_1 \cup l_2$ is regular

b) $l_1 \cap l_2$ is regular

c) $\bar{l_1}$ is regular

d) can't be determined ✓

Q: l_1, l_2 are two regular languages. which is true

I: $l_1 - l_2$ is regular

II: $\Sigma^* - l_1$ is not regular

a) I, II b) I ✓ c) II d) both are wrong

(2014) $L_1 = \{w \mid w$ has atleast as many occurances of 110's as 011's, $w \in \{0,1\}^*\}$

(2) $L_2 = \{w \in \{0,1\}^* \mid w$ has atleast as many occurances of 000's as 111's$\}$

a) L_1 is regular, but not L_2

b) L_2 is regular, but not L_1

c) both L_1 & L_2 are regular

d) neither L_1 not L_2 are regular

Ans in Gate Overflow

Q: Which of the following are regular

I: $\{a^n b^{2m} \mid n \geq 0, m \geq 0\}$

II: $\{a^n b^m \mid n = 2m\}$

III: $\{a^n b^m \mid n \neq m\}$

IV: $\{xcy \mid x, y \in \{a,b\}^*\}$

a) I, IV b) I, III c) I only d) IV only

I $\rightarrow a^*(bb)^*$

IV $\rightarrow (a+b)^* c (a+b)^*$

Q: $X_0 \rightarrow 1X_1$

$X_1 \rightarrow 0X_1 \mid 1X_2$

$X_2 \rightarrow 0X_1 \mid \epsilon$

which of the following precisely represent the language of the grammar if X_0 is the start symbol.

a) $10(0^* + (10)^*)1$ b) $10(0^* + (10^*)^*)^*1$

c) $1(0+10)^*1$ d) $10(0+10)^*1 + 110(0+10)^*1$

solve

Q: which of the following language is/are regular

$l_1 = \{wxw^R \mid w, x \in \{a,b\}^* \text{ and } |w|, |x| > 0, w^R \text{ is reverse of } w\}$

$l_2 = \{a^n b^m \mid m \neq n \text{ & } m, n \geq 0\}$

$l_3 = \{a^p b^q c^r \mid p, q, r \geq 0\}$

a) l_1, l_3 b) l_2 c) l_2, l_3 d) l_3

Problems:

(10)

II. Infinite unions

Consider $l_1 = \{ab\}$

$l_2 = \{aabb\}$

$l_3 = \{aaabbb\}$

Consider $l_1 \cup l_2 \cup l_3 \cup \ldots = \{a^n b^n \mid n > 0\}$

∴ not regular

Silly infinite intersection

Also regular language is not closed under substring operation

(26)

(i) → produces nothing i.e., empty language

(iii) → produces nothing i.e., empty language

Q: which of the following is false

a) Every subset of regular set is regular

b) Intersection of two regular sets is regular

c) Every finite subset of regular set is regular

d) None

Consider $l_1 = \{a^n b^m \mid n, m > 0\}$ ---- regular

$l_2 = \{a^n \mid b^m \mid n, m > 0 \text{ & } n > m\}$ ... non regular

$l_2 \subseteq l_1$ but l_2 is not regular ∴ option (a)

Context Free Language

→ It is language generated by context free grammar.

→ A grammar G is said to be CFG if every production is of the from

$$A \to \alpha$$

where $A \in V$

$\alpha \in (V \cup T)^*$

i.e., every production left side symbol must be single non-terminal but right side one can be combination of variables and terminals.

→ CFGs are very useful to define syntaxes of many programming language such as pascal, FORTRAN hence all these languages are called as CFLs.

→ CFLs as play important role in many fields such as natural language processing, design of programming languages, text editors etc.

→ The following are some examples of CFGs over alphabet $\{a,b\}$

(i) equal no of a's & b's

(ii) unequal no of a's & b's

(iii) palindromes

Q: Design CFG for below languages

(i) $L = \{a^n b^n \mid n>0\}$

$S \to ab \mid aSb$

(ii) $L = \{a^n b^{2n} \mid n \geq 0\}$

$S \to aSbb \mid \epsilon$

(iii) $L = \{ww^R \mid w \in \{a,b\}^+ \text{ & } w^R \text{ is reverse of } w\}$

$S \to aSa \mid bSb \mid \epsilon$ (even length palindromes)

(iv) $L = \{w \mid w \in \{a,b\}^* \text{ and } w \text{ is odd length palindrome}\}$

$S \to a \mid b \mid aSa \mid bSb$

(v) $L = \{a^n b^m \mid n > m \geq 0\}$

$S \to a \mid aSb \mid aS$

(vi) $L = \{w \mid w \in \{a,b\}^+, N_a(w) = N_b(w)\}$

$L = \{ab, ba, aabb, abab, baab, bbaa, abba \cdots\}$

$S \to ab \mid ba \mid aSb \mid bSa \mid SS$

(or)

$S \to AB \mid BA$

$A \to aS \mid a$

$B \to bS \mid b$

(or)

$S \to aB \mid bA$

$A \to a \mid aS \mid bAA$

$B \to b \mid bS \mid aBB$

(vii) $L = \{a^n b^n \mid n \neq 2\}$

$L = \{\epsilon, ab, aaabbb, aaaabbbb \cdots\}$

$S \to \epsilon \mid ab \mid aAb$

$A \to aAb \mid aabb$

(or)

$S \to \epsilon \mid ab \mid aaaA bbb$

$A \to \epsilon \mid aAb$

(viii) $L = \{a^i b^j \mid i \neq j \geq 0\}$

$L = \{a, b, aa, bb, aab, abb, aaa, bbb \cdots\}$

$S \to AC \mid CB$

$A \to aA \mid a$

$B \to Bb \mid b$

$C \to aCb \mid \epsilon$

Derivations:

A string 'w' can be derived from the grammar in multiple ways but in practice we use 3 methods:

(i) Left most derivation
(ii) Right most derivation
(iii) Parse trees (or) Derivation trees

Left most Derivation: (LMD)

A derivation is said to be LMD iff in each step of the derivation left most variable is replaced.

Right most Derivation (RMD)

A derivation is said to be RMD iff if a right most variable is replaced.

Parse Tree:

When a derivation is represented in the form of a tree, it is called parse tree or derivation tree.

To obtain yield of the parse tree concentrate all leaf node labels from left to right order without repetition of labels.

Eg: $S \to SaS \mid SbS \mid c$

$w = cacbc$

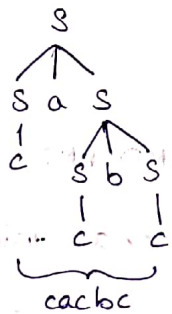
LMD:

$S \to SaS$
$\to caS$
$\to casbS$
$\to cacbS$
$\to cacbc$

RMD:

$S \to SaS$
$\to SasbS$
$\to SasbC$
$\to SacbC$
$\to cacbc$

Parse tree:



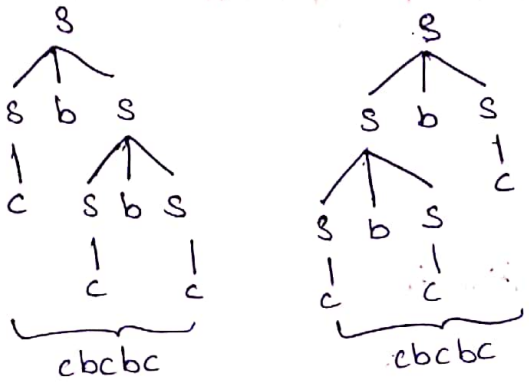
- Parse tree is same for both LMD & RMD

~~Eg: abcbc~~

Ambiguous Grammar:

A grammar is said to be ambiguous if its language has atleast one string with more than one LMD or RMD.

Eg: Consider string cbcbc for above grammar

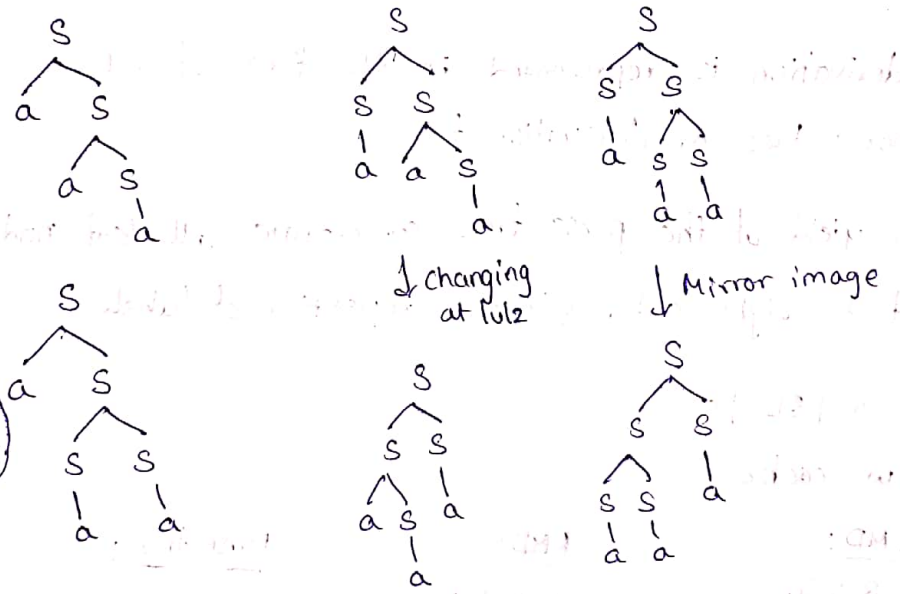


∴ It is ambiguous grammar

Q: Consider grammar

$S \rightarrow aS|a|SS$

How many parse trees can be drawn for w = aaa



26/01/20

Q: Construct LMD, RMD & parse trees for the string

i) w = abbbaa using grammar

$S \rightarrow aB|bA$

$A \rightarrow a|aS|bAA$

$B \rightarrow b|bS|aBB$

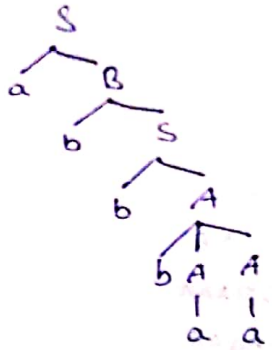
LMD

$S \to aB$
$\to abS$
$\to abbA$
$\to abbbAA$
$\to abbbaA$
$\to abbbaa$

RMD

$S \to aB$
$\to abS$
$\to abbA$
$\to abbbAA$
$\to abbbAa$
$\to abbbaa$

Parse Tree:



(ii) $w = aabbab$

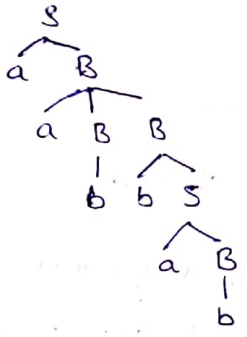
LMD

$S \to aB$
$\to aaBB$
$\to aabB$
$\to aabbS$
$\to aabbaB$
$\to aabbab$

RMD

$S \to aB$
$\to aaBB$
$\to aaBbS$
$\to aaBbaB$
$\to aaBbab$
$\to aabbab$

P.T:



Q: Construct LMD, RMD & parse tree for

$S \to AB \mid BA$
$A \to a \mid aS$
$B \to b \mid bS$

$w = abbaba$

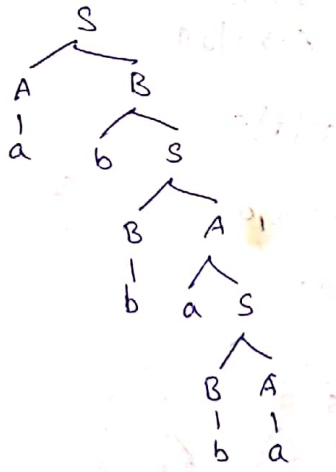
LMD:

$S \to AB$
$\to aB$
$\to abS$
$\to abBA$
$\to abbA$
$\to abbaS$
$\to abbaBA$
$\to abbabA$
$\to abbaba$

RMD:

$S \to AB$
$\to AbS$
$\to AbBA$
$\to AbBaS$
$\to AbBaBA$
$\to AbBaBa$
$\to AbBabA$
$\to AbBaba$
$\to abbaba$

P.T:

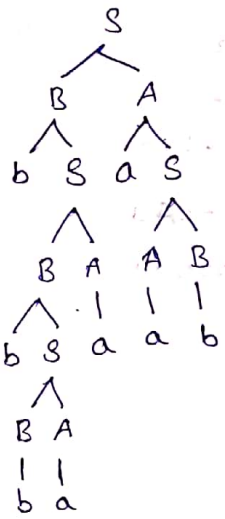


(ii) $w = bbbaaaab$

LMD:

$S \to BA$
$\to bSA$
$\to bBAA$
$\to bbSAA$
$\to bbBAAA$
$\to bbbaAA$
$\to bbbaaA$
$\to bbbaaaS$
$\to bbbaaaAB$
$\to bbbaaaaB$
$\to bbbaaaab$

P.T:



Ambiguous Grammar:

Eg: $S \to Sas \mid Sb S \mid c$

Finding out if a grammar is ambiguous or not is based on trail & error method. This is undecidable problem.

Eg: Check whether below grammar is ambigous or not

(i) $S \to SS \mid a \mid \epsilon$

(ii) $E \to E+E \mid E-E \mid id$

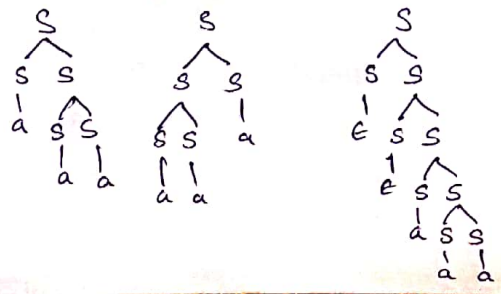
(iii) $S \to Sas \mid b$

(iv) $S \to aaB \mid Ab$
$B \to b$
$A \to a \mid aA$

(i) $S \to SS \mid a \mid \epsilon$

$L = \{\epsilon, a, aa, aaa \cdots\}$

for aaa

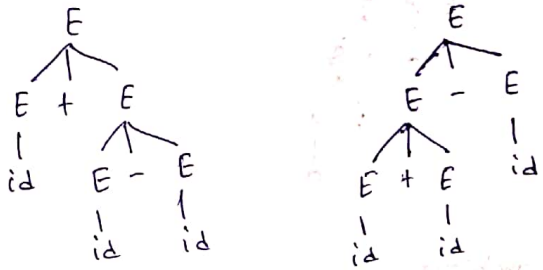


Here we can draw infinite no of parse tree for one string because $S \to \epsilon$ is present

(ii) $E \to E+E \mid E-E \mid id$

$L = \{ id,\ id+id,\ id-id,\ id+id-id \cdots \}$

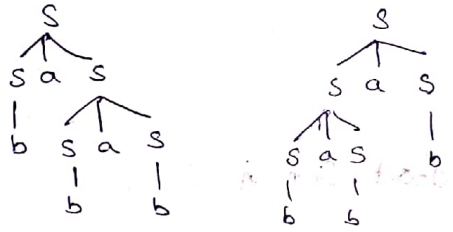
for id+id-id



(iii) $S \to SaS \mid b$

$L = \{ b, bab, bababab, bababbab \cdots \}$

for bababab



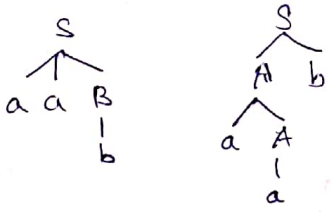
(iv) $S \to aaB \mid Ab$

$B \to b$

$A \to a \mid aA$

$L = \{ ab, aab, aaab, aaaab \cdots \}$

for aab



Q: Check if below grammar is ambigous

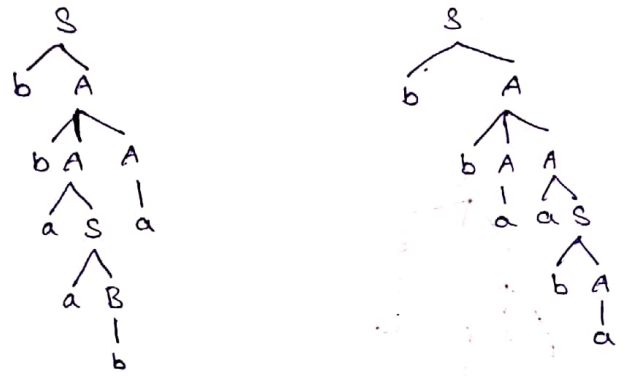
(2)

$S \to aB \mid bA$

$A \to a \mid aS \mid bAA$

$B \to b \mid bS \mid aBB$

Consider string ~~bbaaaabb~~ bbaaba



∴ The grammar is ambiguous

★ Note:

In most of the cases (not all) if there is a production in which two non-terminals appear consecutively, then the grammar is most probably an ambiguous grammar

Eg: $T \rightarrow ABB$

Q: Consider following statements about CFG, G

$S \rightarrow SS \mid ab \mid ba \mid \epsilon$

I: G is ambiguous

II: G produces all strings with equal no of a's & b's

✓

a) I only b) II only c) I & II d) none

Consider string ababab

$S \rightarrow SS$
$\rightarrow abS$
$\rightarrow abSS$
$\rightarrow ababab$

$S \rightarrow SS$
$\rightarrow Sab$
$\rightarrow SSab$
$\rightarrow abSab$
$\rightarrow ababab$

All strings produced by G contains equal no of a's & b's

But G does not produce all strings which contain equal no of a's & b's

i.e, aabb is not produced by G

∴ I only

Q: Which of the following grammar generates

$L = \{a^i b^j \mid i \neq j\}$

a) $S \to AC \mid CB$
$A \to aA \mid \epsilon$
$B \to Bb \mid \epsilon$
$C \to aCb \mid a \mid b$

b) $S \to aS \mid Sb \mid a \mid b$

c) $S \to AC \mid CB$
$A \to aA \mid \epsilon$
$B \to Bb \mid \epsilon$
$C \to aCb \mid \epsilon$

d) $S \to AC \mid CB$
$A \to aA \mid a$
$B \to Bb \mid b$
$C \to aCb \mid \epsilon$

a) generates aabb

$S \to AC \to aaCb \to aabb$

b) generates ab

$S \to aS \to ab$

c) generates ab

$S \to AC \to \epsilon C \to aCb \to ab$

d) because A & B does not produce ϵ

Q: In the correct grammar of above question what is the length of derivation (no of steps starting forom S) to generate the string $a^l b^m$ with $l \neq m$.

a) $\max(l,m)+2$

b) $l+m+2$

c) $l+m+3$

d) $\max(l,m)+3$

Consider aabbbb

$S \to CB$
$\to aCbB$
$\to aaCbbB$
$\to aabbB$
$\to aabbb$

5 steps i.e., $\max(2,3)+2$
$3+2=5$

$\therefore$ opt (a)

(2017) Identify the language generated by the grammar where S is the start symbol

$S \to XY$
$X \to aX \mid a$
$Y \to aYb \mid \epsilon$

a) $\{a^m b^n \mid m \geq n, n > 0\}$

b) $\{a^m b^n \mid m \geq n, n \geq 0\}$

c) $\{a^m b^n \mid m > n, n \geq 0\}$

d) $\{a^m b^n \mid m > n, n > 0\}$

XY
↑ atleast one a — $a^n b^n$ $n \geq 0$

$\Rightarrow a^m b^n$ $\Rightarrow a^m a^n b^n$, $n \geq 0, m > 0$

$a^x b^y$, $x > y, y \geq 0$

★

Q: Consider the following CFG

G_1: $S \to aS \mid B$
$B \to b \mid bB$

G_2: $S \to aA \mid bB$
$A \to aA \mid B \mid \epsilon$
$B \to bB \mid \epsilon$

which of the following languages generated by G_1 & G_2 respectively

a) $\{a^m b^n \mid m > 0$ (or) $n > 0\}$
$\{a^m b^n \mid m > 0$ & $n > 0\}$

b) $\{a^m b^n \mid m > 0$ & $n > 0\}$
$\{a^m b^n \mid m > 0$ (or) $n \geq 0\}$

c) $\{a^m b^n \mid m \geq 0$ (or) $n > 0\}$
$\{a^m b^n \mid m > 0$ and $n > 0\}$

d) $\{a^m b^n \mid m \geq 0$ and $n > 0\}$
$\{a^m b^n \mid m > 0$ (or) $n > 0\}$

G_1: $l = \{ab, b, aab, aabb\} \Rightarrow m \geq 0$ & $n > 0$

G_2: $l = \{a, b, ab, aab, abb \cdots\} \Rightarrow m > 0$ (or) $n > 0$
aaa, bbb

Q: Find the language of the grammar

$S \to AB$

$A \to 0A \mid \epsilon$

$B \to 1B \mid \epsilon$

a) $L = \{0^m 1^n \mid n, m \geq 0\}$

b) $L = \{0^m 1^n \mid n, m \geq 1\}$

c) $L = \{0^m 1^n \mid m > 0 \text{ and } n \geq 0\}$

d) None

Elimination of useless symbols & productions:

G: $S \to AC \mid BA$

$B \to BdD$ (useless cuz it does not produce terminating string)

$A \to a$

$C \to b$

$D \to d$ (useless since B is useless)

↓ elimination of useless symbols & productions

G: $S \to AC$

$A \to a$

$C \to b$

$L(G) = \{ab\}$ in both grammars

So in the above grammar we have 3 useless symbols

i.e, B, D, d

Steps in elimination:

① Every production in P must produce terminal strings either directly or indirectly (repeated substitution). Otherwise productions are useless

Eg: $S \to BA$, $B \to BdD$ in above grammar

② Thus G is $S \to AC$

$A \to a$

$C \to b$

$D \to d$

② Every production in the result of rule 1 must produce terminal string by participating in atleast one sentence derivation. Otherwise they are useless

i.e., $D \to d$ in above grammar

∴ The reduced grammar is

R: $S \to AC$
$A \to a$
$C \to b$

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

List of useless symbols are B, D, d

★

Q: A Rank is defined as no of useless symbols in the grammar. Find the rank of the following grammar.

$S \to AB \mid BD$
$B \to BD \mid BC$
$D \to f \mid BD$
$A \to a$
$C \to c$
$D \to DD$

Step 1: Identify productions producing direct terminals

$A \to a$
$C \to c$
$D \to f$

Step 2: Identify productions producing terminals ~~indirectly~~ indirectly

$D \to DD$
$A \to a$
$C \to c$
$D \to f$

Step 3: (rule 2)

$B \to BD \mid BC$ does not produce terminals and thus productions $S \to AB \mid BD$ $B \to BD \mid BC$ $D \to BD$ are useless production

useless symbols are S, A, B, C, D, a, c, f ⇒ rank = 8

Q: Find rank of below grammar

$S \to AC \mid BD$

$B \to BD \mid BC$

$D \to f \mid BD$

$A \to a$

$C \to c$

$D \to DD$

Step 1:

$A \to a$

$C \to c$

$D \to f$

Step2:

$A \to a$

$C \to c$

$D \to f$

$D \to DD$

$S \to AC$

Step3:

useless productions are

$S \to BD$

$B \to BD \mid BC$

$D \to BD$

$D \to f \mid DD$

Thus useless symbols are B, D, f

reduced grammar

$S \to AC$

$A \to a$ rank = 3

$C \to c$

Q: for the grammar given below what is the equivalent CFG without useless symbols

$S \to AB \mid a$

$A \to a$ ✓

a) $S \to a$, $A \to a$ b) $S \to a$ c) $A \to a$ d) $S \to A \mid a$, $A \to a$

All the above options are equivalent to given grammar but removing useless symbols we get (b)

Elimination of Unit & Null Productions:

A production P is said to be null if it is of the form $A \to \epsilon$. For every null production grammar G we can construct an equivalent grammar G_1 without null productions such that

$$L(G_1) = L(G) - \epsilon$$

Eg: $S \to AB$
$A \to a/\epsilon$
$B \to bS/\epsilon$

$\Rightarrow L(G) = \{\epsilon, a, b, ab\}$

Removing null productions

$S \to AB/A/B$
$A \to a$
$B \to bS/b$

$L_1 = \{a, b, ab\}$

i.e., $L_1 = L - \epsilon$

Eg: $S \to AB$
$A \to a/\epsilon$
$B \to bS/\epsilon$

Step1: Identify direct null productions.

$A \to \epsilon$
$B \to \epsilon$

Step2: Find productions which would become null by substitution of of above production

$S \to AB$
$\to \epsilon B$
$\to \epsilon\epsilon$
$\to \epsilon\epsilon$

} $S \to AB \to \epsilon$

$\therefore$ Final null variables are $\{S, A, B\}$

Step3 : For each production substitute null values

$S \to AB$

$\Rightarrow S \to AB|A|B$

for $A \to a$

$A \to a$

$B \to bS$

$b \to bS|b$

> find all combination and write it unless it is ϵ

final result is

$S \to AB|A|B$

$A \to a$

$B \to bS|b$

Q: Construct an equivalent grammar without null productions

(N)

$S \to ABa|BA$

$A \to aBa|b|\epsilon$

$B \to AS|b|\epsilon$

Step1 :

$A \to \epsilon$

$B \to \epsilon$

Step2 :

$S \to \epsilon$

$A \to \epsilon$

$B \to \epsilon$

Step3 :

$S \to ABa|Ba|Aa|a|A|B|ABA$

$A \to aBa|aa|b$

$B \to AS|S|A|b$

Q: Eliminate ϵ-productions from the following grammar

$S \to XY$

$X \to Zb$

$Y \to bW$

$Z \to AB$

$W \to Z$

$A \to aA|bA|\epsilon$

$B \to Ba|Bb|\epsilon$

Step1 :

$A \to \epsilon$

$B \to \epsilon$

Step2 :

$Z \to \epsilon$

$W \to \epsilon$

$A \to \epsilon$

$B \to \epsilon$

Step3 :

$S \to XY$

$X \to Zb \mid b$

$Y \to bW \mid b$

$Z \to AB \mid A \mid B$

$W \to Z$

$A \to aA \mid a \mid bA \mid b$

$B \to Ba \mid a \mid Bb \mid b$

Elimination of unit productions:

A production P is said to be unit ~~iff~~ production is of form

$A \to B$ where $A, B \in V$

Eg: $S \to A$, $A \to B$, $B \to C$, $C \to D$, $D \to d$ } $L(G) = \{d\}$

On removing all unit productions we get $S \to d$

Step1 : $S \to A$, $A \to B$, $B \to C$, $C \to d$

Step2 : $S \to A$, $A \to B$, $B \to d$

Step3 : $S \to A$, $A \to d$

Step4 : $S \to d$

→ For every grammar with unit productions we can produce an equivalent grammar without ~~equivalent~~ unit productions

Q: Eliminate unit & null productions from following grammar

(2)

$S \to AB$

$A \to aB \mid \epsilon$

$B \to b \mid \epsilon$

Eliminate null productions

null variables { S, A, B }

$S \to AB \mid A \mid B$

$A \to aB \mid a$

$B \to b$

> while eliminating null & unit productions first remove null production, then remove unit

Eliminating unit productions

$S \to AB \mid \underbrace{aB}_{A} \mid a \mid \underbrace{b}_{B}$

$A \to aB \mid a$

$B \to b$

→ To construct a reduced grammar for given G apply the following in sequence

(i) Eliminate null productions from G to build a new grammar G_1,

(ii) Eliminate unit productions in G_1, if any to build G_2.

(iii) Eliminate useless symbols in G_2 to build G_3.

(iv) Eliminate useless productions of given G.

Q: Consider grammar G

$S \to AB$

$A \to aAA \mid \epsilon$

$B \to bBB \mid \epsilon$

① Find null variables in the grammar

a) A b) B c) A, B, S ✓ d) A & B

② If G_1 is constructed from G after elimination of ϵ productions

G_1 is given by

$A \to \epsilon$

$B \to \epsilon$

$S \to \epsilon$

$S \to AB \mid A \mid B$

$A \to aAA \mid aA \mid a$

$B \to bBB \mid bB \mid b$

⑤ Eliminate unit productions from ②'s answer

$S \to AB \mid aAA \mid aA \mid a \mid bBB \mid bB \mid b$

$A \to aAA \mid aA \mid a$

$B \to bBB \mid bB \mid b$

☆

Q: Which of the following statements are true?.

(i) Every left recursive grammar can be converted into right recursive grammar and vice versa

(ii) Every ϵ-production can be removed from any CFG by suitable transformation

(iii) The language generated by CFG $X \to w$, $X \to wY$ where w is string of terminals and Y is a non-terminal is always regular.

(iv) The derivation trees of string generated by the CFG is called as parse tree.

✓ a) i, ii, iii, iv

b) ii, iii, iv

c) i, iii, iv

d) i, ii, iv

Q: Remove all null productions from following grammar

Find no of productions

$S \to ABa \mid bz$

$z \to AB$

$w \to xz$

$X \to zB$

$Y \to aZ \mid bX$

$A \to aAb \mid \epsilon$

$B \to bBb \mid \epsilon$

$A \to \epsilon$
$B \to \epsilon$
$Z \to \epsilon$
$X \to \epsilon$
$W \to \epsilon$

null variables : A, B, Z, X, W

Removing ϵ-productions

$S \to ABa \mid Aa \mid Ba \mid a \mid bZ \mid b$
$Z \to AB \mid A \mid B$
$W \to XZ \mid X \mid Z$
$X \to ZB \mid Z \mid B$
$Y \to aZ \mid a \mid bX \mid b$
$A \to aAb \mid ab$
$B \to bBb \mid bb$

(23)

Instead of this long process find null variables and find no of productions directly.

Normal Forms:

Two normal forms are:

(i) Chomsky Normal Form (CNF)
(ii) Griebach Normal Form (GNF)

Chomsky Normal Form:

A Grammar G is said to be in CNF iff every production P is either of the form

$A \to BC$ (or) $A \to a$ where $A, B, C \in V$ and $a \in T$

i.e., right side of every production must have either 2 variables or single terminal

Eg: $S \to AB \mid BA$
$A \to a$
$B \to b$

Eg: Convert the below grammar into its CNF

$S \to Ab \mid Ba$
$A \to a$
$B \to b$

$S \to AX \mid BY$
$X \to b$
$Y \to a$
$A \to a$
$B \to b$

Eg: Convert the below grammar into CNF

(N)

$S \to aAB \mid BAb$
$A \to aS \mid b$
$B \to bS \mid a$

Step1: Eliminate unit and null productions if there is any

Step2: Consider the production containing terminal on RHS with variables

$S \to aAB \Rightarrow S \to XAB$, $X \to a$
$S \to BAb \Rightarrow S \to BAY$, $Y \to b$
$A \to aS \Rightarrow A \to XS$
$A \to b \Rightarrow A \to b$
$B \to bS \Rightarrow B \to YS$
$B \to a \Rightarrow B \to a$

$S \to XAB$
$X \to a$
$S \to BAY$
$Y \to b$
$A \to XS$
$A \to b$
$B \to YS$
$B \to a$

Step3: Restrict right side length

$S \to XAB \Rightarrow S \to XZ$, $Z \to AB$, $X \to a$

$S \to BAY \Rightarrow S \to BC$, $C \to AY$, $Y \to b$, $A \to XS$, $A \to b$, $B \to YS \mid a$

The final CNF form is

$S \to XZ \mid BC$
$A \to XS \mid b$
$B \to YS \mid a$
$C \to AY$
$X \to a$
$Z \to AB$

Now consider

$w = aaabaa \Rightarrow |w| = 6$

$S \to XZ$
$\to aZ$
$\to aAB$
$\to aXSB$
$\to aaSB$
$\to aaXZB$
$\to aaaZB$
$\to aaaABB$
$\to aaabBB$
$\to aaabaB$
$\to aaabaa$

Here no of steps = 11

Consider

$w = aba$

$|w| = 3$

$S \to XZ$
$\to aZ$
$\to aAB$
$\to abB$
$\to abB$
$\to aba$

Here no of steps = 5

we can conclude that if length of string is n, and the grammar is in CNF then length of derivation is $2n-1$

If the grammar is in CNF, then parse tree of its every string is a binary tree

Note:

full binary tree: 2 (or) 0 children

Complete binary tree: node are filled from left to right level by level.

perfect binary tree: full + complete

The time complexity for derivation of string is in the order of n if G is in CNF

→ Convert the following grammar into CNF

$S \to aBAb \mid bb$

$A \to as \mid b$

$B \to b$

Step1 : Remove unit & null productions

Step2 : $S \to XBAY \mid YY$

$A \to XS \mid b$

$B \to b$

$X \to a$

$Y \to b$

Step3 : $S \to XBAY \Rightarrow S \to XM$

$M \to BN$

$N \to AY$

∴ CNF is $S \to XM \mid YY$

$A \to XS \mid b$

$B \to b$

$M \to BN$

$N \to AY$

$X \to a$

$Y \to b$

Griebach Normal form (GNF):

A grammar G is said to be in GNF iff every production in P of G is of the form

$A \to a\alpha$

where

$a \in T$

$\alpha \in V^*$

i.e., Right side of every production must have terminal followed by any no of variables

Eg:

$S \to aAB$

$A \to as \mid b$

$B \to bS \mid a$

Consider

$w = aba \Rightarrow |w| = 3$

$S \to aAB$
$\to abB$
$\to aba$

no of steps = 3

Consider

$w = aaabaa \Rightarrow |w| = 6$

$S \to aAB$
$\to aaSB$
$\to aaaABB$
$\to aaabBB$
$\to aaabaB$
$\to aaabaa$

no of steps = 6

i.e., If length of the string is 'n', no of derivation steps is n if grammar is in GNF
All derivations with GNF are LMD
Time complexity for derivation is of the order of n

Eg: Convert below grammar into GNF

$S \to ABC$
$A \to aBS \mid bB \mid a$
$B \to a$
$C \to d$

Substitute A in S productions

$S \to aBSBC \mid bBBC \mid aBC$
$A \to aBS \mid bB \mid a$
$B \to a$
$C \to d$

Q: Match the following list1 with list2

List-I	List2
E: Checking that identifiers are declared before their use	P: $L = \{a^n b^m c^n d^m \mid n \geq 1;\ m \geq 1\}$
F: No of formal parameters in the declaration of a function agree with no of actual parameter in use of the function	Q: $x \to xbx \mid xcx \mid dxf \mid g$

G: Arithematic expression with matched pairs of paralysis.

H: palindromes

R: $L = \{wcw \mid w \in \{a,b\}^*\}$

S: $X \to bXb \mid cXc \mid \epsilon$

	E	F	G	H
a)	P	R	Q	S
b)	R	P	S	Q
c) ✓	R	P	Q	S
d)	P	R	S	Q

Q: $X \to \overset{E+E}{XbX} \mid \overset{E-E}{XcX} \mid \overset{(E)}{dXf} \mid \overset{id}{g}$

∴ Arithematic expression i.e., G

S: $X \to bXb \mid cXc \mid \epsilon$

palindrome i.e., H

R: $L = \{wcw \mid w \in \{a,b\}^*\}$

Here a constraint is imposed such that the language consists of exactly one 'c' and w being $\{a,b\}^*$

∴ It can be thought as a name of variable

i.e., E

P: $L = \{a^n b^m c^n d^m \mid n \geq 1; m \geq 1\}$

i.e., F

It is designed using CSG; not CFG

(2002) C language is

a) Context Free language

b) ✓ Context Sensitive Language

c) Regular Language

d) Recursively Enumerable Language

☆ → CYK algo is used to ~~recogino~~ ~~recognize~~ tell where language is finite or infinite

if there is cycle → infinite, No cycle → finite, No terminal → empty

G: Arithematic expression with matched pairs of paralysis

R: $L = \{wcw \mid w \in \{a,b\}^*\}$

S: $X \to bxb \mid cxc \mid \epsilon$

H: palindromes

	E	F	G	H
a)	P	R	Q	S
b)	R	P	S	Q
c)	R	P	Q	S
d)	P	R	S	Q

Q: $X \to \overset{E+E}{xbx} \mid \overset{E-E}{xcx} \mid \overset{(E)}{dxf} \mid \overset{id}{g}$

∴ Arithematic expression i.e., G

S: $X \to bxb \mid cxc \mid \epsilon$

palindrome i.e., H

R: $L = \{wcw \mid w \in \{a,b\}^*\}$

Here a constraint is imposed such that the language consists of exactly one 'c' and w being $\{a,b\}^*$

∴ It can be thought as a name of variable

i.e., E

P: $L = \{a^n b^m c^n d^m \mid n \geq 1; m \geq 1\}$

i.e., F

It is designed using CSG; not CFG

(2022) C language is

a) Context Free language

b) Context Sensitive Language ✓

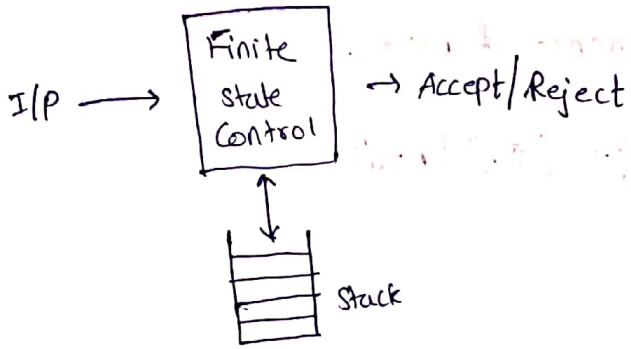
c) Regular Language

d) Recursively Enumerable Language

→ CYK algo is used to ~~recognise~~ ~~recognise~~ tell whether language is finite or infinite.

if there is cycle → infinite, no cycle → finite, No terminal → empty

Push Down Automata



→ PDA accepts CFG.

→ The extra stack used helps recogonize CFG.

A PDA can be defined as a 7-tuple

$$P = \{Q, \Sigma, \tau, \delta, q_0, z_0, F\}$$

Q → set of state

Σ → set of i/p symbols

τ → set of stack symbols

q_0 → start state

z_0 → start symbol of stack

F → set of accepting states

δ → Transition function.

PUSH, POP operations can be performed on stack. At one time, any no of symbols can be pushed onto stack. But at one time only a single symbol can be popped

It takes a triple $\delta(q, a, X)$ as argument

$q \in Q$, $a \in \Sigma \cup \epsilon$, X is stack symbol i.e., $X \in \tau$

$\delta: Q \times (\Sigma \cup \epsilon) \times \tau \rightarrow Q \times \tau^*$

Instantaneous Description (ID):

→ In FA the current state defines status of FA's computation.

→ But in PDA both state and contents of stack are considered

Thus we represent configuration of PDA by a triple (q, w, γ)

1. q is the current state
2. w is remaining i/p
3. γ is the stack contents

This triple is called ID

For γ, we generally show top of stack at left end and bottom at right end

Turnstile notation:

$\vdash$ sign is called turnstile notation and represents one move

$\vdash^*$ sign represents sequence of moves.

Eg: Define PDA for language $\{a^n b^n / n>0\}$

Sol:

$Q = \{q_0, q_1\}$

$\Sigma = \{a, b\}$

$\tau = \{A, Z\}$

$\delta(q_0, a, Z) = \{(q_0, AZ)\}$

$\delta(q_0, a, A) = \{(q_0, A, A)\}$

$\delta(q_0, b, A) = \{(q_1, \epsilon)\}$

$\delta(q_1, b, A) = \{(q_1, \epsilon)\}$

$\delta(q_1, \epsilon, Z) = \{(q_1, \epsilon)\}$

Eg: Consider $w = aaabbb$

The below shows sequence of ids while computation of w

$\delta(q_0, aabb, Z)$

$\downarrow$

$\delta(q_0, abb, AZ)$

$\downarrow$

$\delta(q_0, bb, AAZ)$

$\downarrow$

$\delta(q_1, b, AZ)$

$\downarrow$

$\delta(q_1, \epsilon, Z)$

$\downarrow$

$\delta(q_1, \epsilon, \epsilon)$.

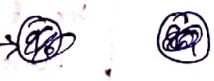
(This acceptance by empty stack)

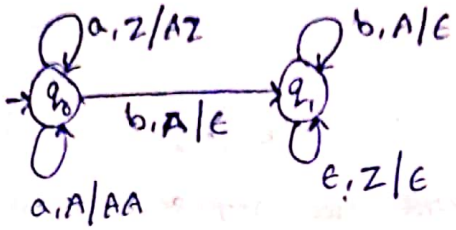
Graphical Notation for PDA

An edge label marked "$a, X/\alpha$" from state q to P mean

$$(q, a, X) \vdash (P, \alpha)$$

The graphical representation for above example is.





Acceptance of PDA:

A PDA can accept strings in two ways

i) Acceptance by Final state

$$L(P) = \{ w \mid (q_0, w, z_0) \vdash^* (q_f, \epsilon, \alpha) \}$$

$q_f \in F$, $\alpha = \Gamma^*$

ii) Acceptance by Empty Stack:

$$L(P) = \{ w \mid (q_0, w, z_0) \vdash^* (q, \epsilon, \epsilon) \}$$

where $q \in Q$

These two methods are equivalent i.e., A language L has a PDA that accepts it by final state, if and only if L has a PDA that accepts it by empty stack.

Note:

→ if $(q, x, \alpha) \vdash_P^* (p, y, \beta)$ then

i) $(q, xw, \alpha) \vdash_P^* (p, yw, \beta)$

ii) $(q, x, \alpha r) \vdash_P^* (p, y, \beta r)$

iii) $(q, xw, \alpha r) \vdash_P^* (p, yw, \beta r)$

→ if $(q, xw, \alpha) \vdash_P^* (p, yw, \beta)$ then

$(q, x, \alpha) \vdash_P^* (p, y, \beta)$.

Note:

→ The example PDA designed for $\{a^n b^n / n \geq 0\}$ is deterministic PDA i.e., there is only one move from any state for every combination of i/p symbol and stack symbol.

→ The Non-deterministic PDA can have more than one move.

→ It is not always possible to convert NPDA to DPDA.

→ Thus expressive power of NPDA is greater than that of DPDA

→ DPDA can express only a subset of CFGs.

Eg: Design a PDA for

$$L = \{ww^R / w \in \{0,1\}^*\}$$

Here we design NPDA

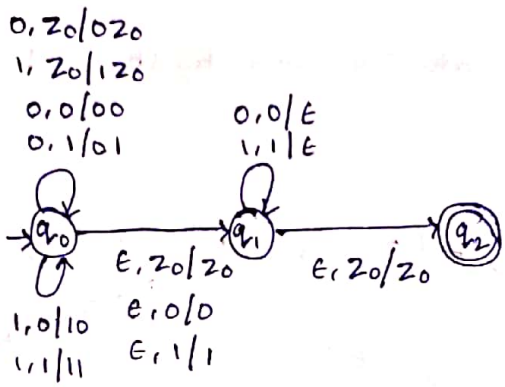
$\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$
$\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$
} These rules act for the first i/p read

$\delta(q_0, 0, 0) = \{(q_0, 00)\}$
$\delta(q_0, 0, 1) = \{(q_0, 01)\}$
$\delta(q_0, 1, 0) = \{(q_0, 10)\}$
$\delta(q_0, 1, 1) = \{(q_0, 11)\}$
} Here for every combination of 0,1 symbols of i/p and stack we stay at q_0 assuming we haven't reached the middle of i/p yet.

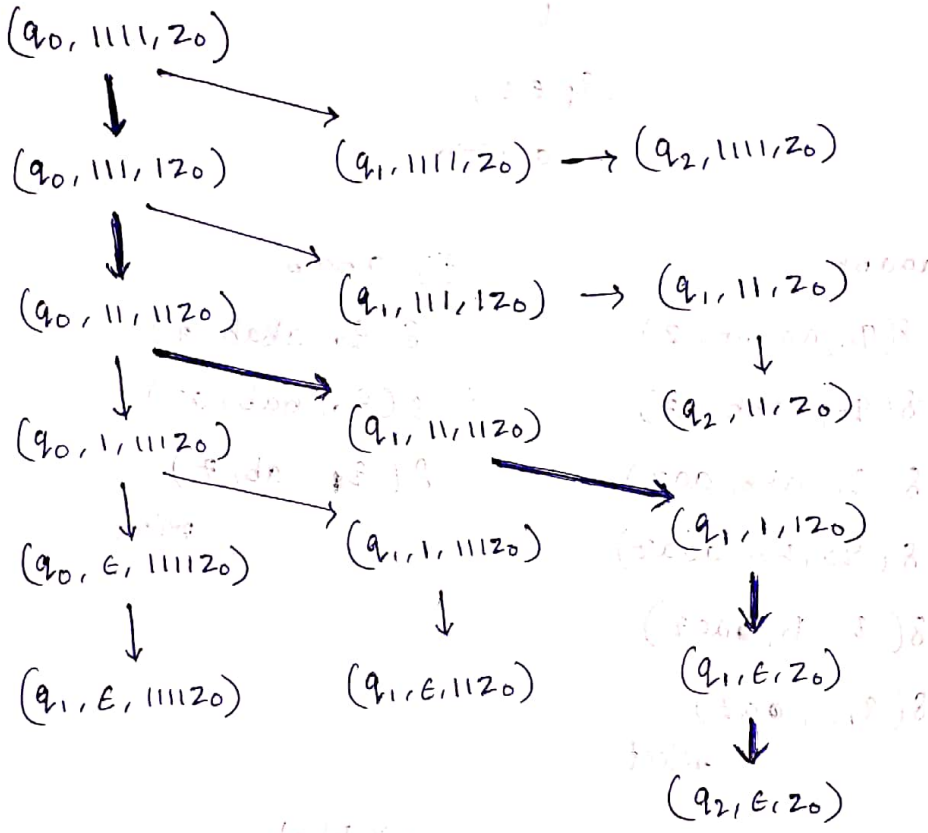
$\delta(q_0, \epsilon, Z_0) = \{(q_1, Z_0)\}$
$\delta(q_0, \epsilon, 0) = \{(q_1, 0)\}$
$\delta(q_0, \epsilon, 1) = \{(q_1, 1)\}$
} These rules allows to go for q_0 to q_1 spontaneously reading nothing. Every time we go to q_1 assuming it might the middle of the i/p.

$\delta(q_1, 0, 0) = \{(q_1, \epsilon)\}$
$\delta(q_1, 1, 1) = \{(q_1, \epsilon)\}$
} Here i/p symbols are matched to stack top and popped symbols of stack.

$\delta(q_1, \epsilon, Z_0) = \{(q_2, Z_0)\}$ Finally if we expose the bottom-of-stack(Z_0) with complete string being read.. It means we found i/p of form ww^R.



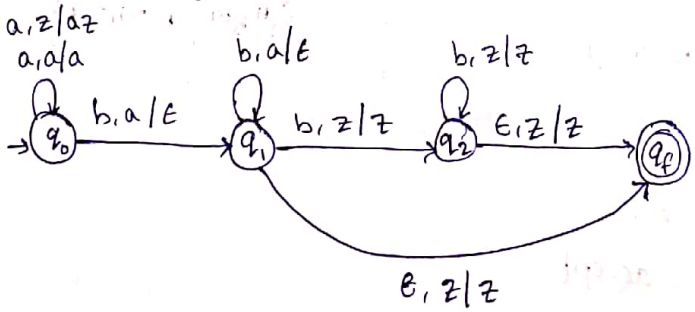
Now consider string $\omega = 1111$. Find all ID's of PDA that can be reached using ~~ω~~ ω.



In the above thick arrows represent accepting path in which middle of string is guessed perfectly.

02/02/2020

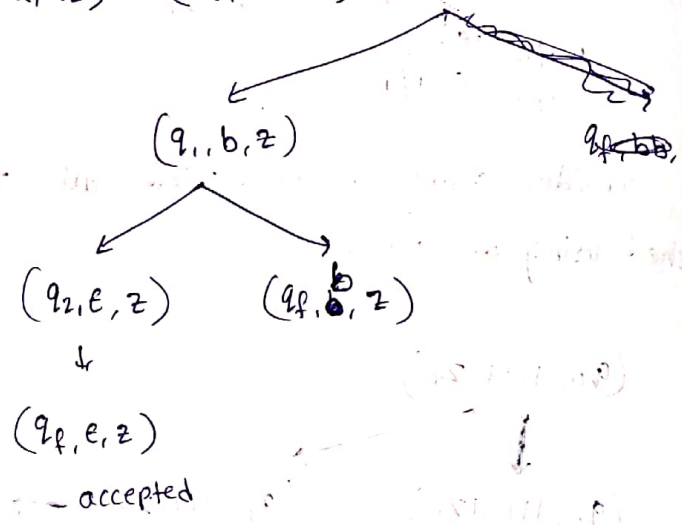
Q. Consider the following PDA



Check whether following strings are accepted by the PDA

(i) aabbb

$(q_0, a, z) \vdash (q_0, a, az) \vdash (q_0, b, aaz) \vdash (q_1, b, az)$



- accepted

(ii) aaabb

$\delta(q_0, aaabb, z)$

$\delta(q_0, aabb, az)$

$\delta(q_0, abb, aaz)$

$\delta(q_0, bb, aaaz)$

$\delta(q_1, b, aaz)$

$\delta(q_1, \epsilon, az)$

reject

(iii) abab

$\delta(q_0, abab, z)$

$\delta(q_0, bab, az)$

$\delta(q_1, ab, z)$

reject

(iv) aaabbb

$\delta(q_0, aaabbb, z)$

$\delta(q_0, aabbb, az)$

$\vdash^*$

$\delta(q_0, bbb, aaaz)$

$\delta(q_1, bb, aaz)$

$\delta(q_1, b, az)$

$\delta(q_1, \epsilon, z)$

$\delta(q_f, \epsilon, z)$

accept.

(v) bbbb

$\delta(q_0, bbbb, z)$

reject

Thus the language of PDA is

$\{a^n b^m \mid m \geq n \geq 1\}$

Q: Determine the no of string of length ≤ 3 accepted by previous question's PDA.

ab
abb ∴ 2

Design of PDA:

→ To design a PDA for regular language, we first design FA.

For every transition

$\delta(q, a) = p$ in FA.

write a transition

$\delta(q, a, z) = (p, z)$ in PDA.

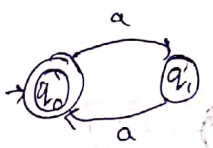
→ To design a PDA for any language we don't have a standard algorithm. But we can use the following guidelines to design a PDA

① If given language is regular design FA and don't disturb the stack

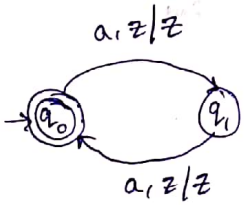
Eg: Design a PDA to recognize the language

$L = \{a^{2n} \mid n \geq 0\}$

FA:



PDA:



② For the non-regular but CFL languages try to design the machine with minimum no of states. This can be obtained by

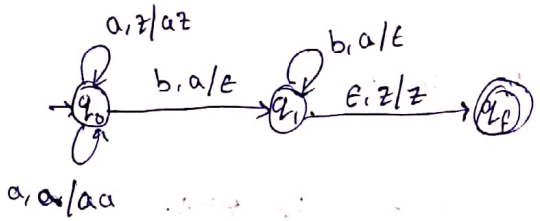
a) Change the state when there is a change in operation

i.e., when there is a change in operation from push to pop or skip to push or skip to pop or pop to skip

The main objective while designing PDA is balancing PUSH & POP operations. In this process first part of the w, we put it on stack using push operation, the second part of the w will be in the tape for comparision.

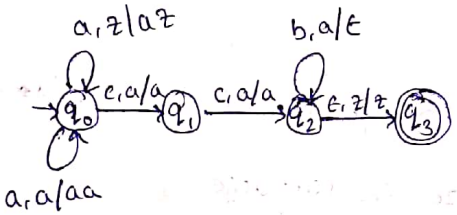
Eg: Design a PDA to recoginize the language

$$L=\{a^n b^n / n>0\} \Rightarrow L=\{ab, aabb, aaabbb\cdots\}$$

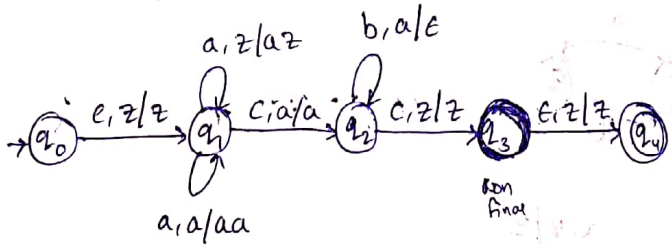


Eg: $L=\{a^n cc b^n / n>0\}$

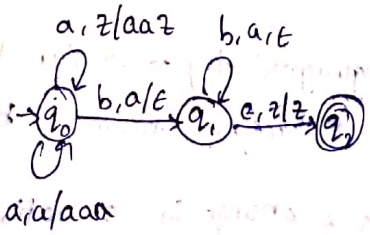
$L=\{accb, aaccbb, aaaccbbb\cdots\}$



Eg: $L=\{ca^n cb^n c / n>0\}$

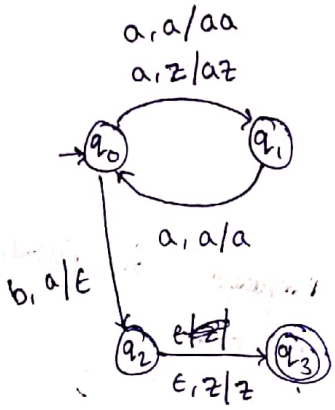


Eg: $L=\{a^n b^{2n} / n>0\}$



Eg: $L = \{a^{2n} b^n \mid n > 0\}$

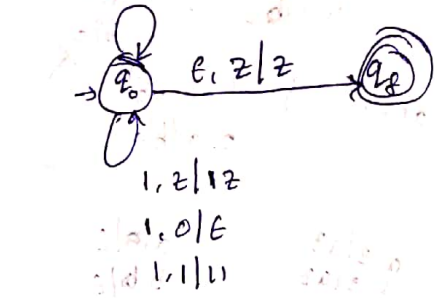
(N)



Eg: $L = \{w \mid w \in \{a,b\}^*, 2N_a(w) = N_b(w)\}$

i.e., no of b's is twice the no of a's

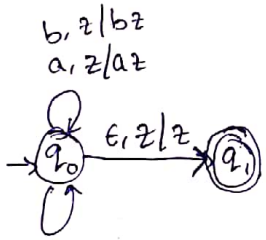
(N)



Eg: $L = \{w \mid w \in \{a,b\}^* \; N_a(w) = N_b(w)\}$

(N)

$L = \{\varepsilon, ab, ba, aabb, baab \ldots\}$



a, b/ε
b, a/ε
a, a/aa
b, b/bb

for ~~abab~~ abba

$\delta(q_0, abba, z)$

$\delta(q_0, bba, az)$

$\delta(q_0, ba, \bullet z)$

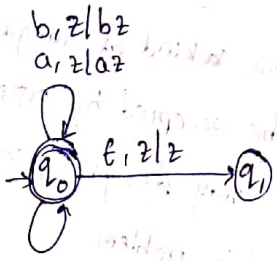
$\delta(q_0, a, bz)$

$\delta(q_0, \varepsilon, z)$

$\delta(q_1, \varepsilon, z)$

Eg: $L = \{w \mid w \in \{a,b\}^* \; N_a(w) \neq N_b(w)\}$

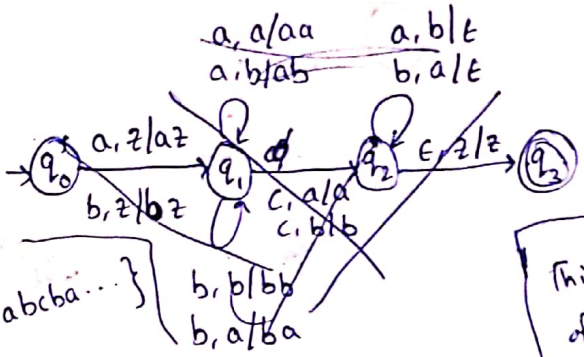
(N)



a, b/ε
b, a/ε
a, a/aa
b, b/bb

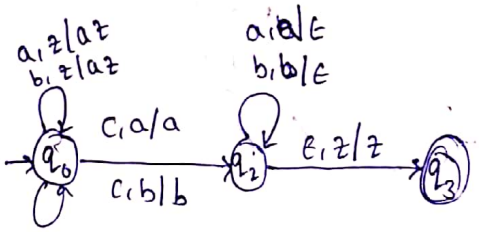
This technique of making final as non-final and non-final to final does not work for all the problems. It works only for certain problems

Eg: $L = \{wcw^R \mid w \in \{a,b\}^+$ and c is terminal$\}$



$L = \{aca, bcb, abcba \ldots\}$

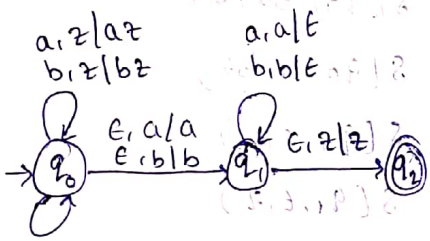
This language is for palindromes of odd length



a, a/aa
b, b/bb
a, b/ab
b, a/ba

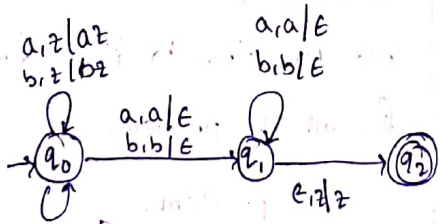
Eg: $L = \{ww^R \mid w \in \{a,b\}^+\}$

$L = \{aa, bb, abba, aaaa, baab \ldots.\}$



a, a/aa
b, b/bb
a, b/ab
b, a/ba

(or)



a, a/aa
b, b/bb
a, b/ab
b, a/ba

This kind of language can't be accepted by DPDA. we can only design NPDA for this problem

The transitions of above PDA are

$\delta(q_0, a, z) = \{(q_0, az)\}$

$\delta(q_0, b, z) = \{(q_0, bz)\}$

$\delta(q_0, a, a) = \{(q_0, aa), (q_1, \epsilon)\}$

$\delta(q_0, b, b) = \{(q_0, bb), (q_1, \epsilon)\}$

$\delta(q_0, a, b) = \{(q_0, ab)\}$

$\delta(q_0, b, a) = \{(q_0, ba)\}$

$\delta(q_1, a, a) = \{(q_1, \epsilon)\}$

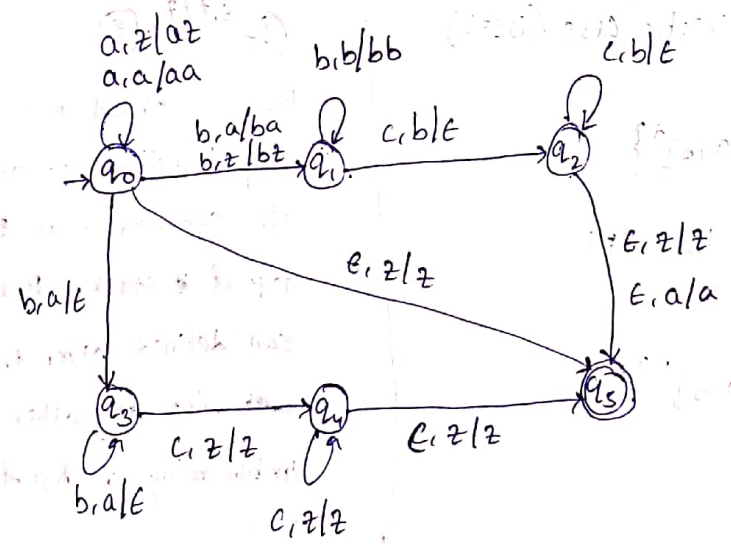
$\delta(q_1, b, b) = \{(q_1, \epsilon)\}$

$\delta(q_1, \epsilon, z) = \{(q_2, z)\}$

→ Languages accepted by deterministic PDA are called DCFL

Eg: $L = \{a^n b^m c^k \mid n=m \text{ (or) } m=k\}$

$L = \{ab, bc, aabb, abc, aabbc, abbbcc, \ldots\}$



Types of PDA:

A PDA can be classified into two types

(i) Deterministic PDA (DPDA)

(ii) Non-Deterministic PDA (NPDA) (or) (NDPDA)

Deterministic PDA:

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is deterministic if

(i) for no q in Q, Z in Γ and a in Σ or ϵ does $\delta(q, a, Z)$ contains more than one element

(ii) for each q in Q, Z in Γ whenever $\delta(q, \epsilon, Z)$ is non-empty then $\delta(q, a, Z)$ is should be empty $\forall a \in \Sigma$

~~Explanation~~

→ The languages accepted by DPDA are called deterministic CFLs (DCFL)

Eg: $L = \{ww^R \mid w \in \{a,b\}^+\}$

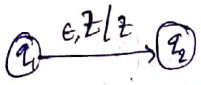
is not DCFL but CFL

$L = \{wcw^R \mid w \in \{a,b\}^+\}$

is DCFL

explation of (ii)

Consider below situation



then q_1 should not defined any transition of other i/p symbols with Z being top of stack. But q_1 can defined other transition with any other stack symbol being on top of the stack

Q: Check whether following are DCFL or not

(N)

1. $L=\{a^n b^n c^m \mid n,m \geq 0\}$ ⟶ DCFL

2. $L=\{a^m b^n c^n \mid n,m \geq 0\}$ ⟶ DCFL

3. $L=\{a^n b^m c^{n+m} \mid n,m \geq 0\}$ ⟶ DCFL

4. $L=\{a^n b^{n+2} c^k \mid n,k \geq 0\}$ ⟶ DCFL

5. $L=\{a^{2n+2} b^n \mid n \geq 0\}$ ⟶ DCFL

6. $L=\{a^n b^m c^k \mid n=m \text{ (or) } m=k\}$ ⟶ CFL but not DCFL

7. $L=\{a^i b^j c^k \mid i \neq j \text{ (or) } j \neq k\}$ ⟶ CFL but not DCFL

8. $L=\{w \mid w \in \{a,b\}^+ \& N_a(w)=N_b(w)\}$ ⟶ DCFL

★ 9. $L=\{a^i b^j \mid i=j^2\}$ ⟶ Not CFL

★ 10. $L=\{a^n b^n c^n \mid n>0\}$ ⟶ Not CFL

★ 11. $L=\{ww \mid w \in \{a,b\}^+\}$ ⟶ Not CFL

★ 12. $L=\{a^n b^n c^{2n} \mid n>0\}$ ⟶ Not CFL

13. $L=\{a^n b^m c^m d^n \mid n,m>0\}$ ⟶ DCFL

14. $L=\{a^n b^n c^m d^m \mid n,m>0\}$ ⟶ DCFL

15. $L=\{a^n b^m c^n d^m \mid n,m>0\}$ ⟶ Not CFL

16. $L=\{a^i b^j c^{i+j} d^m \mid i,j,m>0\}$ ⟶ DCFL

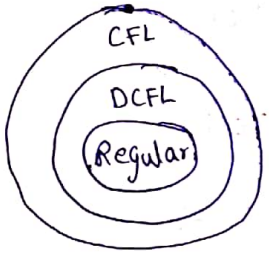
17. The language defined on single symbol alphabet is CFL iff it is regular.

Eg: $L=\{a^{2i} \mid i>0\}$ is regular & DCFL

$L=\{a^{i^2} \mid i>0\}$ is not regular & not CFL

18. $L = \{a^p / p \text{ is prime}\}$ ⟶ Not CFL (∵ not regular).

Note:



CFG to PDA:

→ For every context free grammar G we can construct an equivalent PDA M that accepts every string of G by null stack

Eg: Convert the following CFG into PDA

$S \to aSb$

$S \to ab$

Let $M = (Q, \Sigma, \tau, \delta, q_0, z_0, F)$

$G = (V, T, P, S)$

$V = \{S\}$

$T = \{a, b\}$

$\delta = \{\delta\}$

Now

$Q = \{q\}$

$\Sigma = \{T\} = \{a, b\}$

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

$q_0 = q$

$z_0 = S$

$F = \phi$

finding δ:

1. for every

$A \to \alpha \Rightarrow 1.\delta(q, \epsilon, A) = (q, \alpha)$

2 ~~for every a in Σ~~ 2. for every a in Σ

$\delta(q, a, a) = (q, \epsilon)$

Now

① $S \to aSb$

$$\delta(q, \epsilon, S) = (q, aSb)$$

$S \to ab$

$$\delta(q, \epsilon, S) = (q, ab)$$

② $\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

Now for aabb

$\delta(q, \underset{\epsilon}{aabb}, S)$

$\vdash \delta(q, aabb, aSb)$

$\vdash \delta(q, \underset{\epsilon}{abb}, Sb)$

$\vdash \delta(q, abb, aSbb)$

$\vdash (q_0, \overset{a}{bb}, abb)$

$\vdash (q_0, bb, bb)$

$\vdash (q_0, b, b)$

$\vdash (q_0, \epsilon, \epsilon)$

Q: Design PDA for

$L = \{ ww^R \mid w \in \{a,b\}^+ \}$ from grammar

$S \to aSa \mid bSb \mid aa \mid bb$

① $S \to aSa \Rightarrow \delta(q, \epsilon, S) = (q, aSa)$

$S \to bSb \Rightarrow \delta(q, \epsilon, S) = (q, bSb)$

$S \to aa \Rightarrow \delta(q, \epsilon, S) = (q, aa)$

$S \to bb \Rightarrow \delta(q, \epsilon, S) = (q, bb)$

② $\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

for $w = abba$

$\delta(q, abba, s)$

$\vdash \delta(q, abba, asa)$

$\vdash \delta(q, bba, sa)$

$\vdash \delta(q, bba, bba)$

$\vdash \delta(q, ba, ba)$

$\vdash \delta(q, a, a)$

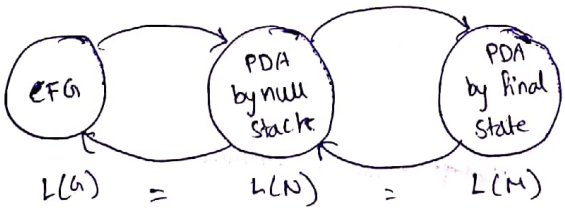
$\vdash (q, \epsilon, \epsilon)$ accepted

PDA to CFG:

The procedure for conversion of PDA to CFG is complicated but possible.

Note:

The following inter-conversions are possible



* → The is no conversion proceduce from NPDA to DPDA

$L(NPDA) \neq L(DPDA)$

$L(DPDA) \subseteq L(NPDA)$

Closure Properties of CFL:

→ CFLs are closed under union

→ " " " " Concatenation

→ " " " " kleen closure

→ Reversal

→ positive closure

→ Homomorphism

→ Inverse homomorphism

→ Substitution

* CFLs are not closed under

(i) intersection

i.e., if l_1 & l_2 are two CFLs then $l_1 \cap l_2$ need not to be a CFL

(ii) Complementation

i.e., if l_1 is a CFL then $\bar{l_1}$ need not to be a CFL.

Some CFLs are designed only with NPDA. Here complementation technique doesn't work. Thus CFL is not close under Complementation. But DCFL is closed under complemention.

→ when CFLs are intersected with regular sets the result is CFL

if l is CFL & r is regular

$l \cap r$ is CFL

Eg: $l_1 = \{a^n b^m \mid n, m > 0\}$ → regular

$l_2 = \{a^n b^n \mid n > 0\}$ → CFL

$l_1 \cap l_2 \rightarrow l_2$ i.e., CFL

DCFL Properties

→ DCFL's are closed under complementation, but not CFLs.

→ DCFL's are not closed under intersection.

→ when DCFL is intersected with regular set, the result is DCFL.

→ Every language accepted by DPDA is unambiguous, but every unambiguous grammar need not to be accepted by DPDA

Eg: $S \rightarrow 0S0 | 1S1 | \epsilon$ is unambiguous and accepted by NPDA only

Q: Consider below languages

$L_1 = \{0^p 1^q 0^r \mid p,q,r \geq 0\}$

$L_2 = \{0^p 1^q 0^r \mid p,q,r \geq 0, p \neq r\}$

which of the below statements is false

a) L_2 is CF

b) $L_1 \cap L_2$ is CF

c) L_1 is CF

d) Complement of L_1 is context-free but not regular

$L_1 \rightarrow$ regular

$L_2 \rightarrow$ CFG

$L_1 \cap L_2$ i.e., regular $\cap$ CFG = CFG

L_1 is CF cuz it is regular

d) $L_1 \rightarrow$ regular $\bar{L_1} \rightarrow$ regular

$\therefore$ d

(2010)

Q: Consider the languages

$L_1 = \{0^i 1^j \mid i \neq j\}$

$L_2 = \{0^i 1^j \mid i = j\}$

$L_3 = \{0^i 1^j \mid i = 2j+1\}$

$L_4 = \{0^i 1^j \mid i \neq 2j\}$

which one of the following stmt is true?

a) only L_2 is CF

b) only L_2, L_3 are CF

c) only L_1 & L_3 are CF

d) All are CF

(2014) Consider the following language over the alphabet $\Sigma = \{0, 1, c\}$

(N)

$$L_1 = \{0^n 1^n \mid n > 0\}$$

$$L_2 = \{w c w^R \mid w \in \{0,1\}^*\}$$

$$L_3 = \{w w^R \mid w \in \{0,1\}^*\}$$

Here w^R is the reverse of w

Which of these languages are deterministic CFLs

a) none b) L_1 c) L_1, L_2 d) All

(2015) which of the following languages are CFLs?

(N)

$$L_1 = \{a^m b^n a^n b^m \mid m, n \geq 1\}$$

$$L_2 = \{a^m b^n a^m b^n \mid m, n \geq 1\}$$

$$L_3 = \{a^m b^n \mid m = 2n+1\}$$

a) L_1, L_2 b) L_1, L_3 c) L_2, L_3 d) L_3

(2016) Consider the following languages

$$L_1 = \{a^n b^m c^{n+m} \mid m, n \geq 1\}$$

$$L_2 = \{a^n b^n c^{2n} \mid n \geq 1\}$$

which is true?

a) L_1 & L_2 are CF

b) L_1 is CF, but L_2 is not

c) L_2 is CF, but L_1 is not

d) none are CF

SET-I

(2017) Consider the following languages over the alphabet $\Sigma = \{a, b, c\}$

(N)

$$L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$$

$$L_2 = \{a^m b^n c^n \mid n, m \geq 0\}$$

Which of the following are CFs

(i) $L_1 \cup L_2$ (ii) $L_1 \cap L_2$

a) i b) ii c) i, ii d) none

$L_1 \cap L_2 = \{a^n b^n c^n / n \geq 0\}$ i.e., not CFG but CSG.

set-II

(2017) Consider the languages given below

$$L_1 = \{a^p / p \text{ is a prime}\}$$

$$L_2 = \{a^n b^m c^{2m} \mid n, m \geq 0\}$$

$$L_3 = \{a^n b^n c^{2n} \mid n \geq 0\}$$

$$L_4 = \{a^n b^n \mid n \geq 1\}$$

Identify correct statement

(i) L_1 is CF but not regular

(ii) L_2 is not CF

(iii) L_3 is not CF but recursive

(iv) L_4 is a deterministic CFL

a) i, ii, iv b) ii, iii c) i, iv d) iii, iv

(2018) Consider the following languages

(2)

I. $\{a^m b^n c^p d^q \mid m+p = n+q, m, n, p, q \geq 0\}$

II. $\{a^m b^n c^p d^q \mid m=n \text{ and } p=q \text{ & } m, n, p, q \geq 0\}$

III. $\{a^m b^n c^p d^q \mid n=n=p \ \& \ p \neq q, \ m,n,p,q \geq 0\}$

IV. $\{a^m b^n c^p d^q \mid mn = p+q\}$

which of the languages above are context free

a) I, IV b) I, II c) II, III d) II, IV

for I: perform m pushes for a^m

now pop for b^n

if $m > n$

m-n no of a's will be left on stack

else if $m < n$

perform pop for m times and push 'b' for n-m times

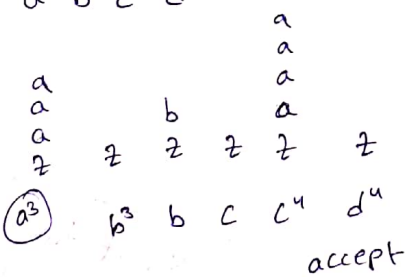
now push a's for p times

now pop the stack for d^q

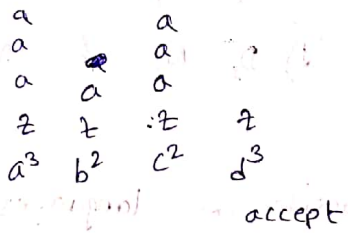
pop n-m times for (n-m) in c^p and push p-(n-m) times for c^p

now pop out for d^q

Eg: $a^3 b^4 c^5 d^4$



Eg: $a^3 b^2 c^2 d^3$



(2019) which of the following languages over {a,b} is not CF

a) $\{ww^R \mid w \in \{a,b\}^*\}$ b) $\{a^n b^i \mid i \in \{n, 3n, 5n\}, n \geq 0\}$

c) $\{wa^n b^n w^R \mid w \in \{a,b\}^*, n \geq 0\}$ d) $\{wa^n w^R b^n \mid w \in \{a,b\}^*, n \geq 0\}$

a => CFL

b => $\{a^n b^n\} \cup \{a^n b^{3n}\} \cup \{a^n b^{5n}\}$ => CFL .not DCFL

c => $w a^n b^n w^R$

↑ ↑ ↑ ↑

Push Push pop pop

$(a^n b^m c^m d^n)$

d => $w a^n w^R b^n$

It is kind of in form

$a^n b^m c^n d^n$

2005 Let N_f & N_p denote the classes of languages accepted by NFA & NPDA respectively. Let D_f & D_p denote the classes of languages accepted by deterministic FA and DPDA respectively. which one of the following is true

a) $D_f \subset N_f$ & $D_p \subset N_p$

b) $D_f \subset N_f$ & $D_p = N_p$

c) $D_f = N_f$ & $D_p = N_p$

d) $D_f = N_f$ & $D_p \subset N_p$

Q: Consider the languages

N

$l_1 = \{a^n b^n c^m \mid n, m > 0\}$

$l_2 = \{a^n b^m c^m \mid n, m > 0\}$

here intersection of two DCFLs need not to be CFL. So check it based on the context

which one of the following statements is false?

a) $l_1 \cap l_2$ is CF

b) $l_1 \cup l_2$ is CF

c) l_1, l_2 are CF

d) $l_1 \cap l_2$ is context sensitive language

* $l_1 \cap l_2 = \{a^n b^m c^n \mid n > 0\}$ is CSG

Q: The language

$L: \{0^i 2 1^i \mid i \geq 0\}$ over the alphabet $\{0,1,2\}$ ar ____

a) not recursive

b) is regular language

c) is recursive and deterministic CFL

d) is not a DCFL, but CFL

(2009) Let $L = L_1 \cap L_2$ where L_1 & L_2 are the languages defined as

$$L_1 = \{a^m b^m c a^n b^n \mid m,n \geq 0\}$$

$$L_2 = \{a^i b^j c^k \mid i,j,k \geq 0\}$$

then L is ____ ?

a) not recursive

b) regular

c) CFL but not regular

d) recursively enumerable but not CFL

Q: Consider following statements about context free grammar

(N)

$$G = \{S \to SS,\ S \to ab,\ S \to ba,\ S \to \epsilon\}$$

~~a) G is~~

(i) G is ambiguous

(ii) G produces all strings with equal no of a's & b's

(iii) G can be accepted by DPDA

Which combination below expresses all the true statements about G.

a) i b) i, iii c) ii & iii d) i, ii, iii

$S \to abS \mid baS \mid \epsilon$

(i) → Since $S \to \epsilon$ is present in it is ambiguous

(ii) → aabb is not produced. So (ii) is false

(iii) → DPDA can be designed (Check previous example) (we can convert G into unambiguous)

Q: Consider the languages

$L_1 = \{ww^R \mid w \in \{0,1\}^*\}$

$L_2 = \{w\#w^R \mid w \in \{0,1\}^*, \# \text{ is special symbol}\}$

$L_3 = \{ww \mid w \in \{0,1\}^*\}$

which one of the following is true

a) L_1 is DCFL b) L_2 is DCFL

c) L_3 is CFL, but not DCFL d) L_3 is DCFL

Q: Let

$L_1 = \{0^{n+m}1^n0^m \mid n,m \geq 0\}$

$L_2 = \{0^{n+m}1^{n+m}0^m \mid n,m \geq 0\}$

$L_3 = \{0^{n+m}1^{n+m}0^{n+m} \mid n,m \geq 0\}$

which of the languages are not context-free

a) L_1 only b) L_3 only c) L_1 & L_2 d) L_2 & L_3

$L_2 \rightarrow 0^{n+m}1^{n+m}0^m$

for 0's we push n+m times

for 1's we pop n+m times

we can't skip 0's because, no of zeroes must be less than no of 0, 1's because $m \leq n+m$

$L_3 \rightarrow$ & $0^{n+m}1^{n+m}0^{n+n} \cong 0^n1^n0^n$

set-II

(2017) Let L_1 & L_2 be any two context free languages and R be any regular language. Then which one of the following is/are correct

I. $L_1 \cup L_2$ is CF

II. $\bar{L}_1$ is CF

III. $L_1 - L_2$ is CF

IV. $L_1 \cap L_2$ is CF

a) I, II, IV b) I, III ✓ c) II, IV d) I only

I. $L_1 \cup L_2$ is CF ✓

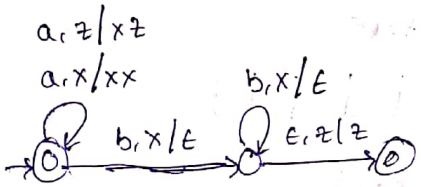
II. $\bar{L}_1$ need not to be CF

III. $L_1 - L_2 = L_1 \cap \bar{L}_2$ (CF ∩ regular) $\Rightarrow$ CF ✓

IV $L_1 \cap L_2$ need not to be CF

Q: Consider the following PDA

(N)



which one of the following is true?

a) $L = \{a^n b^n / n \geq 0\}$ and is not accepted by any DPDA

b) $L = \{a^n / n \geq 0\} \cup \{a^n b^n / n \geq 0\}$ and is not accepted by any DPDA

c) $L = \{a^n / n \geq 0\} \cup \{a^n b^n / n \geq 0\}$ and is DCFL ✓

d) none of the above

a) $\Rightarrow L = \{a^n b^n / n \geq 0\}$ is ~~correct~~ wrong and also it can be accepted by DPDA

~~$L = \{a^n b^n / n \geq 0\}$ is correct representation and is accepted by some DPDA~~

c) $\Rightarrow \{a^n / n \geq 0\} \cup \{a^n b^n / n \geq 0\}$ is correct and it is DCFL

Problems:

Pumping Lemma for CFL's

Let L be any infinite CFL. Then there is a constant 'n' depending on L such that if z is in L and $|z| \geq n$ then we may write

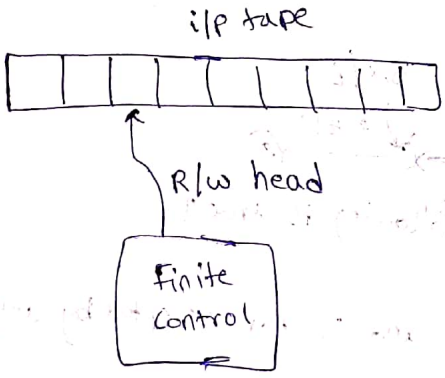
$z = uvwxy$ such that

(i) $|vx| \geq 1$

(ii) $|vwx| \leq n$ and

(iii) for $\forall i \geq 0$, uv^iwx^iy is in L

Turing Machines



→ A turing machine (T.M) is a mathematical model of digital computer. A T.M is used to recogonize Recursively Enumerable Languages.

→ According to Turing if any problem solvable by digital computer can also be solved by T.M

→ The T.M block diagram as shown in the figure consists of i/p tape with infinite length. The head can move in two directions (left or right) with read / write capabilities.

→ A T.M is seven tuple system

$M: (Q, \Sigma, \tau, \delta, q_0, B, F)$ where

Q = finite set of states

Σ = i/p alphabet

τ = tape alphabet

δ = transition function mapping

$$Q \times \Sigma \rightarrow Q \times \tau \times \{L, R\}$$

q_0 = initial state

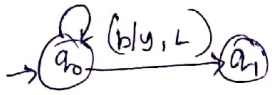
B = blank symbol

F = set of final states

A T.M can be represented in multiple ways

1. State diagram
2. State table

1. State diagram



a/x, R

a is i/p symbol

x is replacement symbol

R is direct of movement of read/write head

2. State table

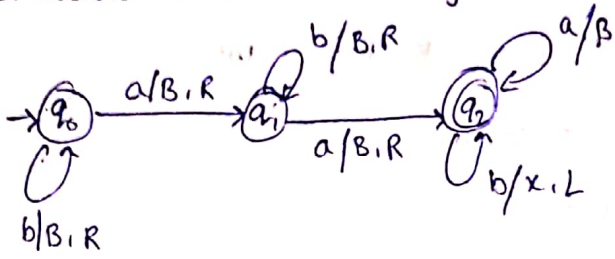
	a
→ q_0	x, q_0, R

→ Transition function is written as

$\delta(q_0, a) = (q_0, x, R)$

$\delta(q_0, b) = (q_1, y, L)$

Eg: Consider below T.M's state diagram



$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Gamma = \{x, B\}$

$q_0 = q_0$

$F = q_2$

$\delta(q_0, a) = (q_1, B, R)$

$\delta(q_0, b) = (q_0, B, R)$

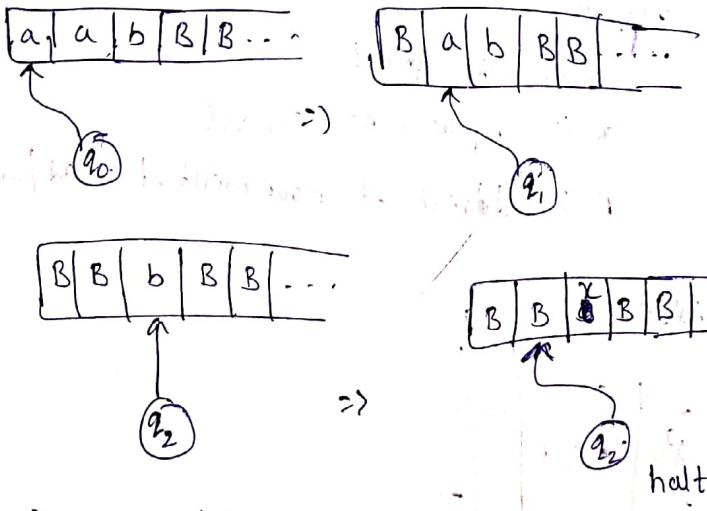
$\delta(q_1, a) = (q_2, B, R)$

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

$\delta(q_2, a) = (q_2, B, R)$

$\delta(q_2, b) = (q_2, x, L)$

Now consider $w = aab$



This is represented as

$\vdash q_0 aab$

$\vdash Bq_0 ab$

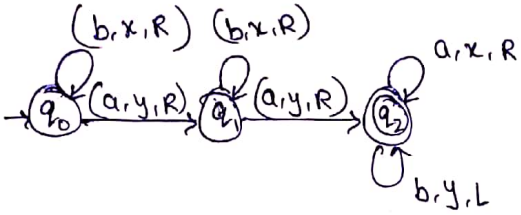
$\vdash BBq_2 b$ ~~$\vdash$~~ $\vdash Bq_2 Bx$

~~$\vdash BBxq_2 B$ ∴ halt~~

09/02/20

Consider the following Turing Machine

(N)



Here $Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$\tau = \{x, y, B\}$

$q_0 = q_0$

$F = \{q_2\}$

$B = B$

δ: $\delta(q_0, a) = (q_1, y, R)$

$\delta(q_0, b) = (q_0, x, R)$

⋮

a) $\omega = abab$

$\vdash q_0abab$

$\vdash yq_1bab$

$\vdash yxq_1ab$

$\vdash yxyq_2b$

~~$\vdash yxyyB$~~

$\vdash yxq_2yy$

Halt

accept

b) $\omega = baab$

$\vdash q_0baab$

$\vdash xq_0aab$

$\vdash xyq_1ab$

$\vdash xyyq_2b$

$\vdash xyq_2yy$

Halt

accept

c) $\omega = bbbaa$

q_0bbbaa

$\vdash xq_0bbaa$

$\vdash xxq_0baa$

$\vdash xxxq_0aa$

$\vdash xxxyq_1a$

$\vdash xxxyyq_2B$

Halt

accept

Acceptance of a string 'w' by Turing Machine.

A string w is said to be accepted by TM iff $q_0w \vdash^* \alpha_1 q_f \alpha_2$ for some $q_f \in F$ and $\alpha_1, \alpha_2 \in \tau^*$ (i.e., w is completely processed)

* → Even If current state is final state and string is not completly processed then it is not accepted yet

Eg: for previous TM check acceptance of baabb

q_0baabb

$\vdash xq_0aabb$

$\vdash xyq_1abb$

$\vdash xyyq_2bb$

$\vdash xyq_2yyb$

halt & reject (because string is not completely processed)

w = aaaab

q_0aaaab

xq_1aaab

yyq_2aab

$yyxq_2ab$

$yyxxq_2b$

$yyxq_2xy$

halt & accept

Eg: Consider the following TM

	0	1	B
$\rightarrow q_0$	$(q_1, 1, R)$	$(q_1, 1, R)$	Halt
q_1^*	$(q_1, 1, R)$	$(q_0, 1, L)$	(q_0, B, L)

w = 001

q_0001

$\vdash 1q_101$

$\vdash 11q_11$

$\vdash 1q_011$

$\vdash 11q_111$

$\vdash 1q_0111$

$\vdash 11q_1111$

:

doesn't Halt and hence it is considered rejected

A string w is generally rejected by TM iff

(i) TM halts at non-final state

(ii) TM halts at final state without procession w completly

(iii) TM does not halt

b) w = 10

$q_0 10$
$1 q_1 0 B$
$11 q_2 B$
$1 q_0 1 B$
$11 q_1 B$
$1 q_0 1 B$
:

doesn't halt and hence rejected

c) w = 0

$q_0 0$
$\vdash 1 q_1 B$
$\vdash q_0 1 B$
$\vdash 1 q_1 B$
:

doesn't halt and hence rejected

Design of Turing Machine:

A TM must have min no of state. This can be obtained by following guidelines

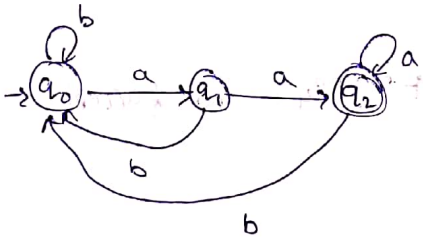
(i) Change the state if there is a change in replacement symbol.

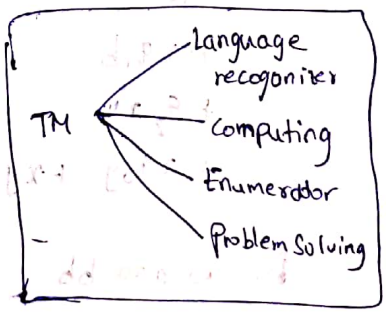
(ii) Change the state if there is a change in direction of the head

(iii) A TM for any particular problem can have n number of solutions but try achieve it with minimum no of states

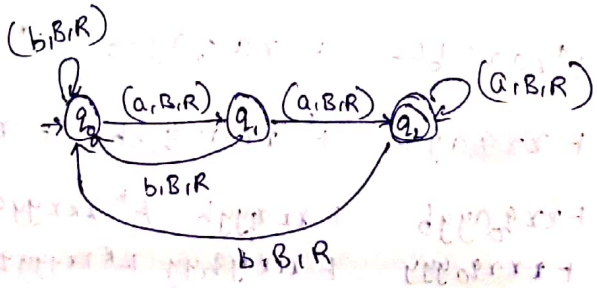
Eg: Design a TM to recogonize the language $L = (a+b)^* aa$

DFA:





TM:



Eg: Design TM for $L = \{a^n b^n \mid n>0\}$

$L = \{ab, aabb, aaabbb, \ldots\}$

let $w = aabb$

$q_0 aabb$

$\vdash x q_1 abb$

~~$\vdash x x q_1$~~

$\vdash x a q_1 bb$

$\vdash x q_2 a y b$

$\vdash q_2 x a y b$

$\vdash x q_0 a y b$

$\vdash x x q_1 y b$

$\vdash x x y q_1 b$

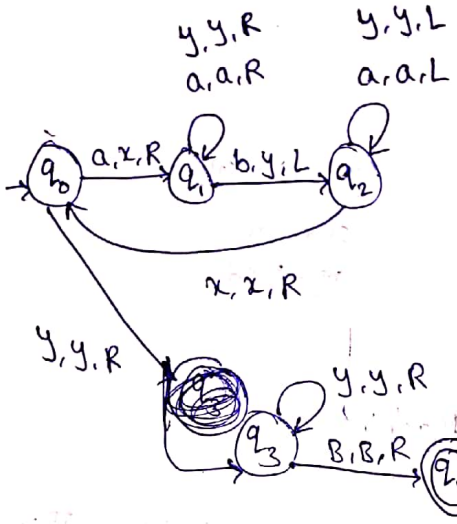
$\vdash x x q_2 y y$

$\vdash x q_2 x y y$

$\vdash x x q_0 y y$

$\vdash x x y q_3 y \vdash x x y y q_3 B \vdash x x y y B q_4$

halt & accept



Here we can write q_3 as final state also

for $w = ab$

$q_0 ab$

$\vdash x q_1 b$

$\vdash q_2 x y$

$\vdash x q_0 y \vdash x y q_3 B \vdash x y B q_4$ halt & accept

for $w = aaabbb$

$q_0 aaabbb$

$\vdash x q_1 aabbb \vdash^* x a a q_1 bbb \vdash x a q_2 a y bb \vdash^* q_2 x a a y bb$

$\vdash x q_0 a a y bb \vdash x x q_1 a y bb \vdash^* x x a y q_1 bb \vdash x x a q_2 y y b$

$\vdash^* x q_2 x a y y b \vdash x x q_0 a y y b \vdash x x x q_1 y y b \vdash^* x x x y y q_1 b \vdash x x x y q_2 y y$

$\vdash^* x x q_2 x y y y \vdash x x x q_0 y y y \vdash x x x y q_3 y y \vdash^* x x y y y q_3 B \vdash x x x y y y B q_4$

Eg: Design TM for $L=\{a^n b^{n+1} \mid n>0\}$ [Think how to design TM]

Eg: Design TM for $L=\{a^n b^n c^n \mid n>0\}$

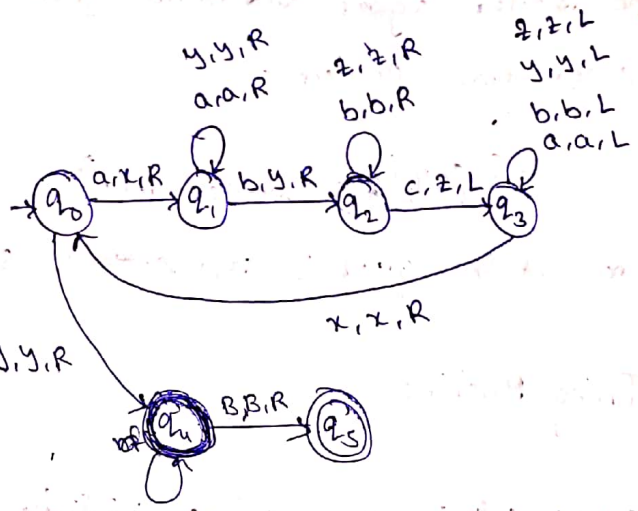
$L=\{abc, aabbcc, aaabbbccc \ldots\}$

$w = aabbcc$

$q_0 aabbcc$

$\vdash^*$

xxq_0yyzz



Here q_4 can be written as final state in below way

TM as Computing Machine:

Eg: Design a TM to compute

$f(n) = n+1$ where n is represented in unary form

i/p	o/p
4	5
1111	11111

$q_0 1111$

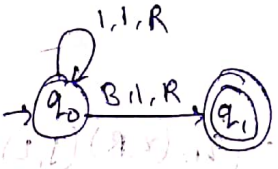
$\vdash 1q_0 111$

$\vdash 11q_0 11$

$\vdash 111q_0 1$

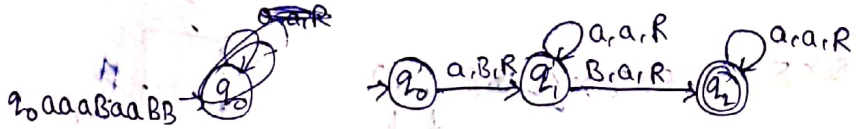
$\vdash 1111q_0 B$

$\vdash 11111q_1$



TM as Enumerator

i/p: aaaBaaBB... o/p: aaaaaBB...



$q_0 aaaBaaBB$

$\vdash Bq_1 aaBaaBB$

$\vdash Baaq_1 BaaBB \vdash Baaaq_2 aaB \vdash^* Baaaaaq_2 B$

TM as Problem Solver

Any arbitrary problem can be expressed as language. Any instance of the problem is encoded into string.

For example whether a given undirected graph is connected or not we can encode above problem as

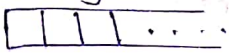
$$L = \{\langle G \rangle \mid G \text{ is connected graph}\}$$

Every instance of the graph is represented as string

Types of Turing Machines

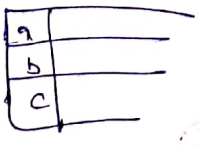
1. ~~Two way infinite TM~~ (or) Semi-infinite tape TM

i.e If L recognized with two way infinite TM $\Leftrightarrow$ L is recognized with one way infinite tape



2. Multitape Turing Machine

L is accepted by Multitape TM $\Leftrightarrow$ L is accepted by some single tape TM



$$\delta(q_0, a, b, c) = (q_1, (x, R)(y, L)(z, R))$$

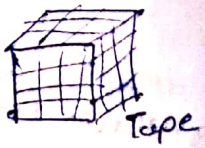
3. Non-deterministic TM:

L is accepted by NDTM $\Leftrightarrow$ L is accepted by DTM

4. Multidimensional Tape TM:

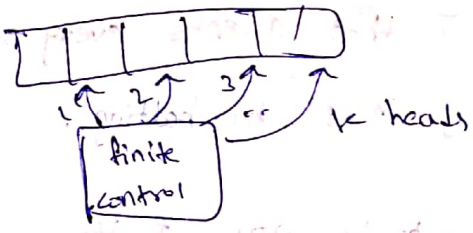
$$\delta: Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R, U, D\}$$

U: UP, D: Down



5. Multihead TM:

L is accepted by k head TM <=> L is accepted by one head TM.



6. Universal TM (UTM):

UTM can that can simulate the behaviour of the digital computer i.e., The UTM is a TM that reads description of some other TM along with its i/p. Then it validates whether the other TM can recogonize the given i/p or not.

Turing Theses:

→ Any that can be done on any digital computer can also be done by Turing Machine.

→ No one has an algorithm that can't be solved by a TM.

→ Alternative models have been proposed but none of them is more powerful or less powerful than actual TM.

→ The set of all turing machines is although infinite, but it is countable. Similarly set of all regular sets, set of CFGs although infinite they are countable. i.e., Infinitely countable.

(v) Positive closure
(vi) ~~Homomorphism~~
(vii) Intersection
(viii) Reversal
(ix) Inverse Homomorphism
(x) Quotient with Regular set

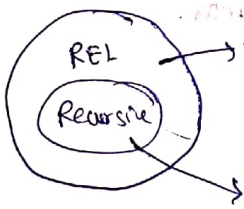
→ Recursively enumerable sets are closed under:

(i) Concatenation
(ii) Union
(iii) Intersection
(iv) Kleen closure
(v) Homomorphism
(vi) Substitution
(vii) Inverse homomorphism
(viii) Reversal
(ix) Quotient with regular set

→ All recursive languages are closed under complementation
i.e., if L is recursive then L' is also recursive

→ Recursively Enumerable languages are not closed under complementation
i.e., If L is recursively Enumerable then L' may or may not be a REL

i.e., If it is given that both L & L' are recursively Enumerable then it is possible only where L is Recursive



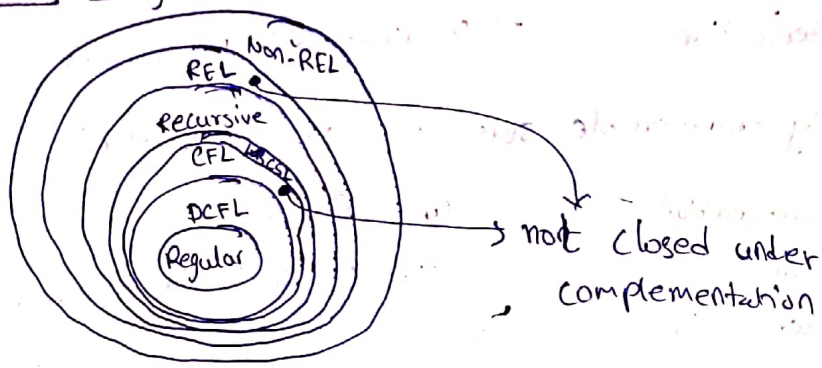
→ not closed under complement

→ ~~closure not closed under~~ closed under complement

It means REL which are recursive are only closed under complementation

→ REL is not closed under difference

Relationship among languages:



* → If L & L' both are REL then L & L' are necessarily Recursive.
* → Complementation of 'CFL but not DCFL' is recursive.
* → The grammar of Turing Machine is Unrestricted Grammar (UG)

Every TM 'T' can be converted in UG 'G' and vice versa such that L(T) = L(G)

Note: ~~If given~~

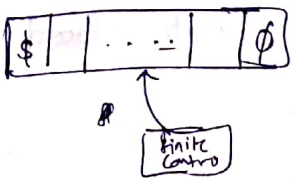
→ If given grammar 'G' is unrestricted grammar there is no specific algorithm to decide whether L(G) is finite (or) infinite (or) empty

→ There is no specific algorithm to decide whether w belongs to L(G) if G unrestricted grammar.

Linear Bounded Automata (LBA):

It is a Restricted model of Turing machine that contains two special symbols $ (at leftmost of tape) and ¢ (at rightmost of tape) such that

$$\delta(q, ¢) = (q_i, ¢, L)$$
$$\delta(q, \$) = (q_i, \$, R)$$
$$\forall\, q \in Q$$



i.e., head of finite control can't move any right of ¢ and any left of $.

→ Below languages are few examples of ~~recursive languages~~ CSLs

- $L = \{a^p \mid p \text{ is prime}\}$

  $L = \{a^{i!} \mid i > 0\}$

  $L = \{a^{i^2} \mid i > 0\}$

  $L = \{a^{\frac{n(n+1)}{2}} \mid n > 0\}$

  $L = \{0^{n!} \mid n > 0\}$

Set-I

(2015) (P) For any two languages L_1 & L_2 such that L_1 is CFG and L_2 is recursively enumerable but not recursive. Which of the following is/are necessarily true?

I. $\bar{L}_1$ is recursive

II. $\bar{L}_2$ is recursive

III. $\bar{L}_1$ is CF

IV. $\bar{L}_1 \cup L_2$ recursively enumerable

a) I only b) III only c) III, IV d) I, IV

$\bar{L}_1$ is recursive $\Rightarrow$ $\bar{L}_1$ is recursively enumerable too

L_2 is REL

$\therefore \bar{L}_1 \cup L_2$ is REL ~~cuz~~ REL are closed under union

(2016) (P) Consider

L_1: Regular

L_2: CF

L_3: Recursive

L_4: REL

which of the following is/are true

I. $\bar{L}_3 \cup L_4$ is REL

II. $\bar{L}_2 \cup L_3$ is Recursive

III. $\bar{L}_1 \cap L_2$ is CF

IV $L_1 \cup \bar{L}_2$ is CF

a) I b) I & IV c) I, II, III d) I, IV

I. $\bar{L}_3 \rightarrow$ recursive $\rightarrow$ REL

$\bar{L}_3 \cup L_4 \rightarrow$ REL

II. $\bar{L}_2$ is Recursive

$\bar{L}_2 \cup L_3$ is recursive

III. $\bar{L}_1 \rightarrow$ Regular

$\bar{L}_1 \cap L_2$ is CF

IV $L_1 \rightarrow$ regular

$\bar{L}_2 \rightarrow$ recursive

$L_1 \cup \bar{L}_2 \rightarrow$ recursive

Q: Define the languages L_0 & L_1 as follows

(N)

$L_0 = \{ \langle M, w, 0 \rangle \mid M \text{ halts on } w \}$

$L_1 = \{ \langle M, w, 1 \rangle \mid M$ doesn't halt on $w \}$.

Here $\langle M, w, i \rangle$ is a triple whose first component M is encoding of Turing machine

w is string

i is a bit

Let $L = L_0 \cup L_1$

which of the following is true

a) L is REL but $\bar{L}$ is not

b) $\bar{L}$ is REL but L is not

c) both L & $\bar{L}$ are recursive

d) neither L, nor $\bar{L}$ is recursively Enumerable

Sol:

L_0 → recursive → REL

L_1 → REL

$L = L_0 \cup L_1$

i.e., L = REL

REL is not closed under complementation

Thus $\bar{L}$ is not REL

Undecidability:

A problem P is said to be decidable if there exist an alogorithm (TM) for infinite instances of P then P is called decidable problem

Post Correspondance Problem:

It is an undecidable problem

	List I	List II
1	01	0
2	111	111
3	01	101

In PCP problem the answer is stating stating sequence of number such that they produce same string from List-I & List-II

Eg: 123

List 1 : 01 111 01 = 0111101

List 2 : 0 111 101 = 0111101

Silly 123123, 2, 22.....

Decidability overview on Regular Languages:

→ Every question about FA or regular language is decidable

Eg: whether given NFA and DFA accept same language or not

decidable.

Eg: deciding whether union of two regular languages is CFL or not.

It CFL & decidable

Decidability Overview on CFL & PDA:

Some of the questions on CFLs are decidable, but some are not.

→ Union of two CFLs is CFLs — decidable

→ Intersection of two CFL is CFL.

Intersection of two CFL may or may not be CFL

∴ undecidable

→ Let G be a CFG.

whether $L(G) =$ finite, — decidable (two)

$L(G) =$ infinite — decidable

$L(G) = \phi$ — decidable

→ Let G is CFG and w is string

membership: whether $w \in L(G)$ — decidable

→ where G is ambiguous — ~~decidable~~ undecidable

→ whether two PDAs accepts same language — undecidable

*It undecidable for arbitrary CFG G_1 & G_2 and R is regular set whether

→ $L(G_1) \cap L(G_2)$ - is empty

→ $L(G) = \Sigma^*$ (it similar to checking if G is equal to some other G)

→ G is ambiguous

→ G_1 & G_2 accept same language i.e., $L(G_1) = L(G_2)$

→ $L(G_2) \subseteq L(G_1)$

→ $L(G_1) = R$

→ ~~$R \subseteq L(G)$~~ $L(G_1) \subseteq R$ (whether given CFG is regular)

→ $\overline{L(G_1)}$ is CFL

→ $L(G_1) \cap L(G_2)$ is CFL

* Let G is be a arbitrary CFG it undecidable to say whether $L(G)$ is regular or not.

Decidability overview on Turing Machines and RELs:

→ Halting problem of TM is undecidable

i.e., finding out whether TM halts for every string or not

→ All recursive languages are decidable

→ ~~It is undecidable~~ Let G be a UG, then the problem of determining whether or not

$L(G) = \phi$ is undecidable

$L(G)$ = finite is undecidable

$L(G)$ = infinite, is undecidable

→ Some language accepted by TM is regular or not — undecidable

→ whether complementation of REL is REL is undecidable

Reducibility:

Reducibility is concept used to solve the harder problems using easier problem with reduction concept.

i.e., when a harder problem doesn't have direct solution, then it is reduced to an easier problem and solutions found.

→ If reduced problem has a solution then actual problem also has a solution

→ If reduced problem has no solution then actual problem will not have a solution.

→ If there is a reduction from problem P_1 to P_2 then

I. If P_1 is undecidable then so is P_2

II. If P_1 is not RFL then so is P_2

(2008) which of the following are decidable

(N)

I. whether intersection two regular languages is infinite

II. whether a given CFG is regular

III. whether two PDAs accept same language

IV. whether given grammar is CFG

a) I, II b) I, IV ✓ c) II, III d) II, IV

Q: which of the following is/are undecidable

(N)

I. G is CFG. is $L(G) = \phi$?

II. G is CFG. is $L(G) = \Sigma^*$

III. M is TM. is $L(M)$ = regular

IV. A is a DFA & N is an NFA. is $L(A) = L(N)$?

a) III b) III, IV c) I, II, III d) II, III ✓

Note:

A language accepted by a TM is regular or not — undecidable

A language accepted by a TM is CFG or not — undecidable

A string w, G is unrestricted Grammar, whether $w \in L(G)$ — undecidable

membership of w for TM — undecidable

Q: (2)
Find whether the following are decidable:

1. does a given program ever produce an o/p?
2. If L is a CF, then is $\bar{L}$ a CF?
3. If L is regular, then is $\bar{L}$ a regular?
4. If L is recursive, then is $\bar{L}$ a recursive?

a) 1,2,3,4 b) 1,2 c) 2,3,4 d) 3,4 ✓

(2014)
Q: Let <M> be the encoding of the TM over the alphabet $\Sigma = \{0,1\}$

(2) Let $L = \{ \langle M \rangle \mid M$ is a turing machine that accepts all strings of length 2020$\}$ then L is ——

a) decidable & REL
b) undecidable & REL ✓
c) undecidable & but not REL
d) decidable & but not REL