Intro...!!

By
-Manam Ramu

std out
std in
std err
} → pointers to files

Monitor is physical device connected to output file

⇒ In array mathetical formula involved is finite Mapping

Domain set (finite) Indices

Range set type of element

* Rules used for formation of words from alphabet are called Lexical Rules.

* Rules used for formation of sentences from words are called Syntactic Rules.

* In UNIX, we have automated tools for lexical analysis and syntax analysis

Syntax Analysis :- checks if every statement is defined or written following syntactic rules of language.

Semantics :- Meaningful

Operative semantics

Applicative semantics

Axiomatic Semantic

* FORTRAN, PASCAL, COBOL, C are called Imperative language

Theory of Computation

Induction:

prove f_1

Assume f_i

prove f_{i+1}

Strong Mathematical Induction

prove f_1

Assume f_i $(i \leq k)$ (all cases are true)

prove f_{k+1}

pigeon holes principle: If there are n+1 pigeons and n holes then atleast one hole will have more than 1 pigeon.

* proof by contradiction

Theory of Computation.

Automata Theory
→ Finite state Machines & Acceptance

Formal Languages
↳ Grammar (set of sentences generated by language)
↳ Types of formal languages and their hierarchy
↳ Membership problem

computation.
↳ computable
→ Not computable
→ decidable
↳ undecidable.
↳ turing machine
↳ Bn-P problem
↳ Non determinism

Automata :-

Automaton :- Machine
(singular)

Finite state Machine :-

→ Internal configuration of Machine at a particular time.

Eg:- Moore, Mealy

(i) state
(ii) Transition : How change of state is happening
(iii) Input String : These are triggers for changing configuration

- Finite state Machines are generally Acceptor Machines
- If the final state reached is acceptable, then the Input is acceptable.
- Language accepted by FSM is set of words
- For writing Lex we use FSM.

Membership problem :-

Given a Grammar, Grammar generates Language if a word is given and asked to check if it belongs to the Language?

Sol :- CYK Algorithm
↳ an elementary kind of parsing.

Ackerman Function (m, n) ↳ if $m \geq 3$ it takes many years to compute

Fastest growing Mathematic function

log*n :- its value varies from 0 to 5

Inverse Ackerman (slowest growing) $\log^*(2^{65,576}) = 5$.

* problems which have polynomial time complexity and have non-deterministic problems are called np problems

FORMAL LANGUAGES

* Language is a set of words following some rules of that Language.

⊙ Regular Languages (or) Linear Languages → Finite Automata
→ Linear Grammar — Left Linear, Right Linear

Deterministic Contex Free Languge → DPDA
↳ DCFG

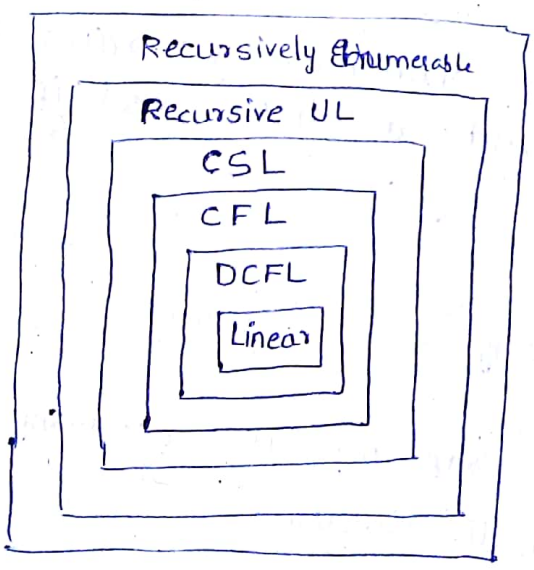
Context Free Languages → Push Down Automata.
↳ context Free Grammar (CFG)

Context Sensitive Languages → Linear Bounded Automata.
↳ context sensitive Grammar (CSG)

Unrestricted (or) Type 0 Language → Turing Machine
↳ unrestricted Grammar

Chomsky's Hierarchy

*



Finite Automata gives us the way how to define certain regular language. (to recognize the language)

Its not unique for every regular language

regular language	Automata.
L_1	A_1
L_2	A_2

Finite Automata
- DFA (Deterministic Finite Automat)
- NFA (Non " " ")
- ε NFA (Recognises NULL to change state)
- 2 DFA (Deterministic FA which can move both Left & Right)

Unrestricted Language
- Recursive Languages → Turing Machine that is guaranteed to halt
- Recursively Enumarable Languages → TM

A state symbol is a representation of of Internal configuration of the system.

FINITE AUTOMATA

① Deterministic Finite Automata

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

- A: Automata
- Q: Finite set of states
- Σ: Input Alphabet (set of Input symbols)
- δ: Transition Function
- q_0: Initial state
- F: F is a subset of Q (Set of final states (set of Acceptable states))

* Transition function uses current state, current input and decides next state (unique) → in case of deterministic

0, 1, a, b, c ..... input symbols

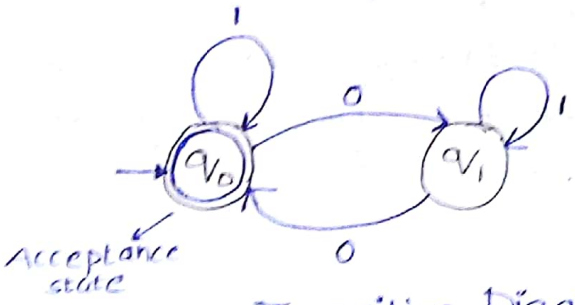
w, x, y, z ..... strings of input symbols

q_0, q_1 .. P_0, P_1 ..... states

Initial state must be unique always

Design Finite Automata which contains even numbers of 0's over strings of 0,1

① State diagram to recognise even zero numbered.



100010 | 00100

Transition Diagram

$$A = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_0\})$$

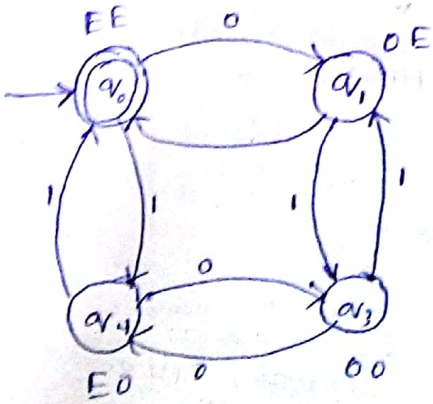
Transition Table

δ	0	1
→ * q_0	q_1	q_0
* q_1	q_0	q_1

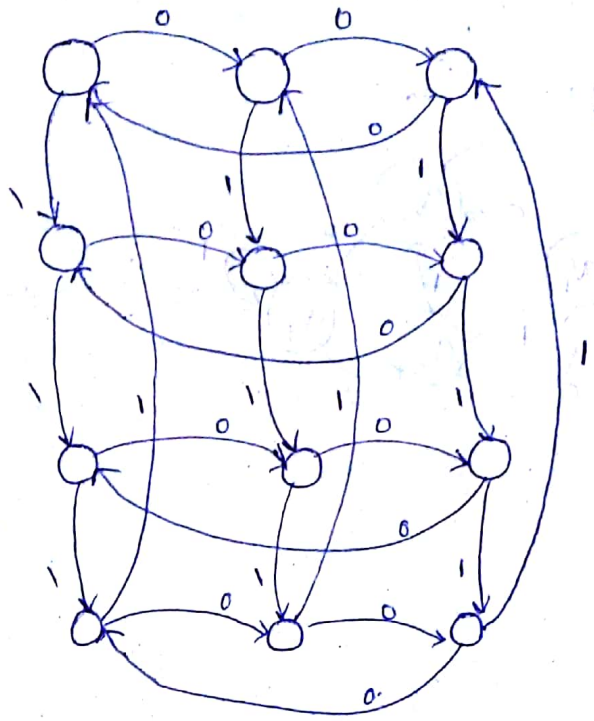
→ ... Initial states

* ... Accepted states

② For both even numbers of 1's & 0's



③ 0's divisible by 3
1's divisible by 4



④ int atoi (char *s)
{
int n=0;
while (*s != '\0')
{
n=n*10+ (*s-48);
s++;
}
return n;
}

'\0'

Variable → unknown & fixed value (In Mathematics)

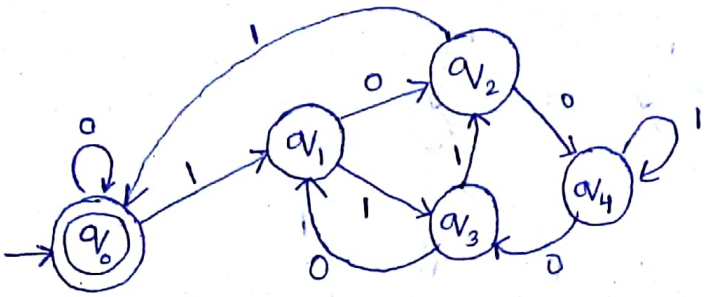
Known at any time & not fixed value (In programming)

'Identifier' → identifies a memory location

Q: Binary Input strings beginning with '1' such that its equivalent decimal number is divisible by 5.

eg:- $(1010)_2 = (10)_{10}$ ✓

$(10101)_2 = (21)_{10}$ ×



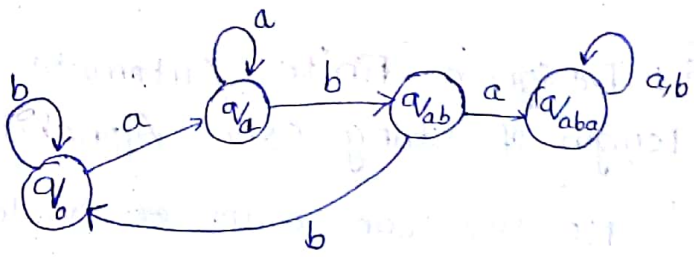
Q:- What is the number of prefix sequences for a given string of length n.

n+1

ε
a
ab
abc

abc → 4

Q:- Design a DFA to recognize 'aba' all strings over a,b which contains the substring 'aba'.



'aba' must be found atleast once.

Now,

'aba' must be found exactly once

$\Rightarrow$ $A = (Q, \Sigma, \delta, q_0, F)$ accepts L

can we design Finite Automata that accepts $\bar{L}$ (complete)

$$A' = (Q, \Sigma, \delta, q_0, Q-F)$$

$$L = L(A)$$

Q :- Design a Finite Automata for finding out the length of string over Alphabet a, b

NO DFA can count the length of arbitrary string.

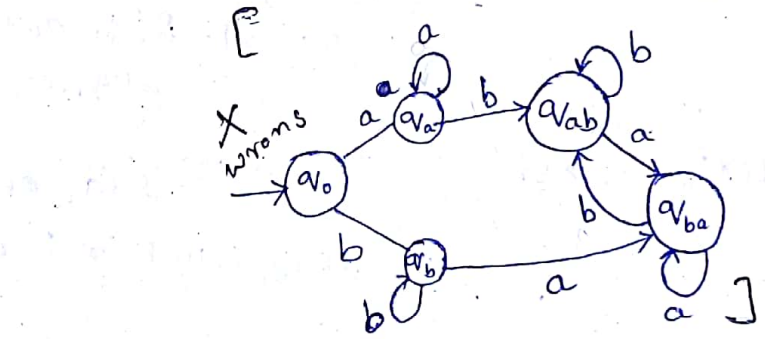
Q :- Design a DFA to recognize

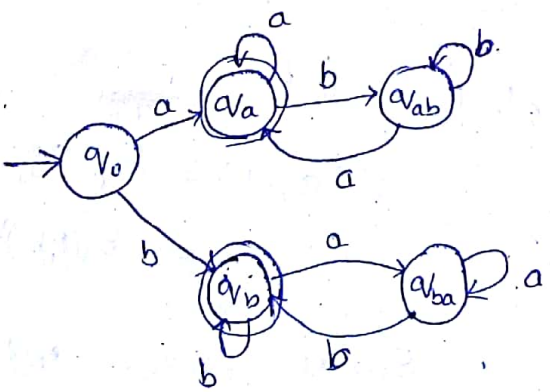
$L = \{x / x \text{ in } (a+b)^*, |x| = 5, x \text{ has atleast two a's}\}$

Q:- consider a DFA to check whether third symbol from right end is 1 in any binary string.

Q:- Design a DFA consider all strings over Alphabet $\{a, b\}$

number of occurrences of substrings "ab" = number of occurences of substring "ba"





$L = \{ab, aba, ba, bb\}$

$L.L = \{xy \mid x, y \text{ in } L\} = L^2$

$L^* = \{t \mid L^0 \cup L^1 \cup L^2 \cup \cdots\}$

Q. $L = \{a^n \mid n \geq 0\}$

No. of states must be Finite, Length can be Infinite also.

Q. $L = \{a^n b^n \mid n \geq 0\}$

No DFA can be designed because states are Infinite

In DFA

$\delta(q_0, x) = \delta(q_0, a_0 a_1 \cdots a_k)$

$= \delta(q_1, a_1 \cdots a_k)$

$\vdots$

$= \delta(q_k, a_k) = q_k$

If q_k is in F, x is accepted

DFA

$\delta : Q \times \Sigma \longrightarrow Q$

NFA

$\delta : \delta(q, a) = P \subseteq Q$

$\delta(q, a) = \{P_1, P_2, \cdots P_n\}$

$\delta(q, ab) = \delta(\{P_1, P_2, \cdots P_n\}, b)$

$= \delta((P_1, b)) \cup \delta(P_2, b) \cup \delta(P_3, b) \cdots$

$\delta(q, x) = \{r_1, r_2 \cdots r_k\}$

if anyone r_i is in F

then x is accepted

Non-Deterministic Finite Automata (NFA)

The next state is one of the possible states.

$$\delta(q, a) = P \subseteq Q$$

- q: input state
- P: set of possible states
- Q: all the states

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

$$\delta(q, a_1 a_2 a_3) = \delta(\delta(q, a_1), a_2 a_3)$$

$$= \delta(\{P_1, P_2, \ldots P_k\}, a_2 a_3)$$

$$= \delta(P_1, a_2 a_3) \cup \delta(P_2, a_2 a_3) \cup \cdots \cup \delta(P_k, a_2 a_3)$$

$$= \{r_1, r_2, \ldots r_s\}$$

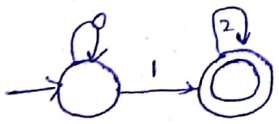
Acceptance

$$\delta(q_0, w) = \{r_1, r_2, \ldots r_k\}$$

and atleast one of $r_1, r_2, \ldots r_k$ is in F

then w is acceptable.

* proper set includes NULL set.

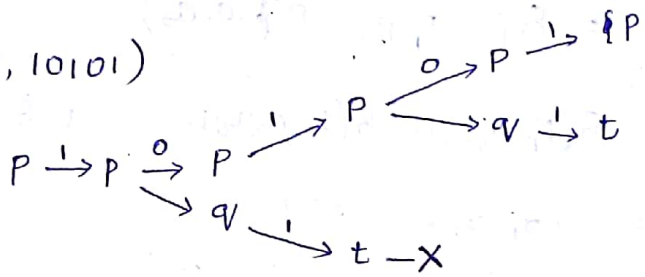


$1+2^+$

Q:- Transition table

	δ	0	1
→	p	{p,q}	{p}
	q	{r, s}	{t}
	r	{p,r}	{t}
*	s	∅	∅
*	t	∅	∅

(i) $\delta(p, 10101)$



$\delta(\{p,t\}, 0) = \{p, q\}$

$\delta(p, 10101) = \{p, t\}$

∵ t is one of the final state

∴ Input string is acceptable.

** For a given NFA construct an equivalent DFA.

A_1 is an NFA and $L = L(A_1)$ (L → language)

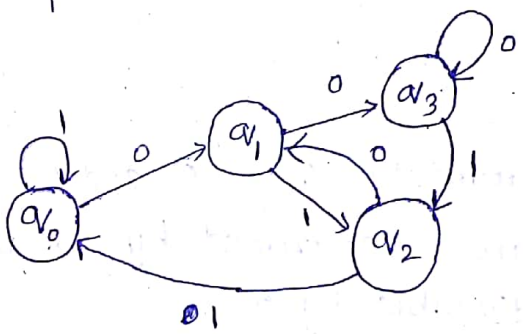
→ construct a DFA A_2 such that $L(A_2) = L$

→ For every state a unique next state has to be defined for every input string.

DFA $\delta([pq],$

NFA $\delta(\{pq\},$

		0	1
q_0	→ [p]	[pq]	[p]
q_1	[pq]	[pqrs]	[pt]
q_2	* [pt]	[pq]	[p]
q_3	* [pqrs]	[pqrs]	[pt]

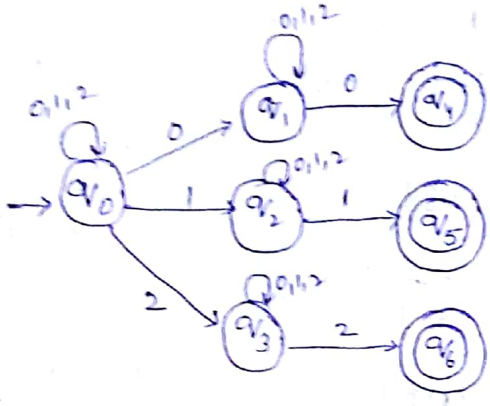


Language accepted by this DFA : $(0+1)^*0(0+1)$

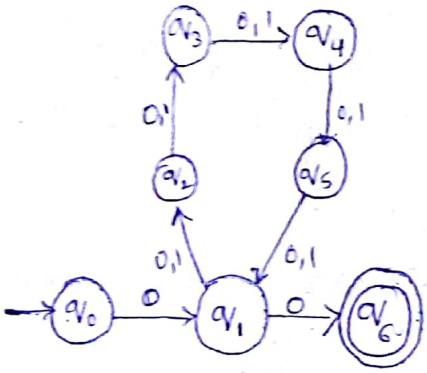
Q:- Design a Finite Automata over 0, 1, 2 such that the last symbol has appeared earlier.

$\{wa \mid w \text{ contains } a\}$ w in $(0+1+2)^*$

where w contains a a is 0 (or) 1 (or) 2



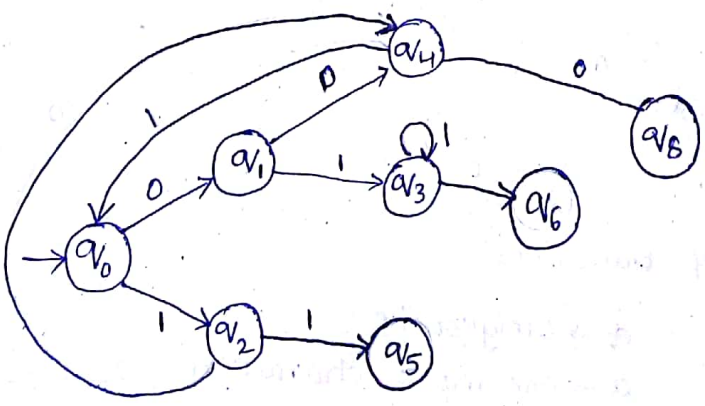
Q:- Design a Finite Automata over 0 and 1 such that some two 0's are separated by a string whose length is divisible by 5.



Q:- convert into DFA

	0	1
→ P	{q, s}	{q}
* q	{r}	{q, r}
r	{s}	{P}
* s	ϕ	{P}

	0	1
q_0 → [P]	[qs]	[q]
q_1 * [qs]	[r]	[pqr]
q_2 * [q]	[r]	[qr]
q_3 * [pqr]	[qrs]	[pqr]
q_4 [r]	[s]	[P]
q_5 * [qr]	[rs]	[pqr]
q_6 * [qrs]	[rs]	[pqr]
q_7 * [rs]	[s]	[P]
q_8 * [s]	[s]	[P]



Q:- construct a Finite Automata for recognising identifiers in c language.

letter (letter | digit)*

letter a ... z A ... Z
digit 0 ... 9

Note:- Denotions of transitions

$\delta(q, a)$
$\delta(q, w)$
$\delta(p, a)$
$\delta(p, w)$

$q \rightarrow$ singlestate
$a \rightarrow$ one input characters
$p \rightarrow$ set of states
$w \rightarrow$ set of input characters

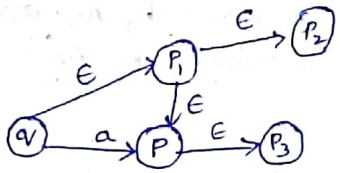
ϵ-NFA

(ϵpslon transition)

Finite Automata can undergo transition without any input symbol.

$$\delta'(q, \epsilon) = \{ p / p \text{ is reachable from } q \text{ on } \epsilon \}$$

Eg.



$$\delta(q, a) = p$$

** ϵ-closure of any state 'q' is the set of all states which can be reached from q on ϵ

ϵ-closure(q) =

Q:- Find the ϵ-closures of all the states

	ϵ	a	b	c
→ p	∅	{p}	{q}	{r}
q	{p}	{q}	{r}	∅
*r	{q}	{r}	∅	{p}

ϵ-closure (p) = {p}

ϵ-closure (q) = {pq}

ϵ-closure(r) = {pqr}

strings of length 1 accepted are 'c'

" " length 2 " " bb, ac, ca, cb....

Given an ε-NFA can we find Equivalent Finite Automata with no ε transition..??

Given E-NFA $A_E = (Q_E, \Sigma, \delta_\epsilon, q_{0_1}, F_E)$) ε-closure

NFA $A_N = (Q_N, \Sigma, \delta_N, q_0', F_N)$

DFA $A_D = (Q_D, \Sigma, \delta_E, P_0, F_D)$

such that $L(A_D) = L(A_E)$ [i.e., all the strings accepted by A_D are also accepted by A_E]

① Replace transitions with ε-closure.

for previous problem

	a	b	c
→ p	{p}	{pq}	{pqr}
q	{pq}	{pqr}	ϕ
r	{pqr}	ϕ	{p}

ϕ → null
not defined
transition is undefined

Regular Expressions :-

These Expressions are used to indicate a set of strings.

reg. exp a corresponding symbol a denotes $\{a\}$

r - represents the set of strings $L\{r\}$

a in Σ denotes a represents $\{a\}$

ϕ represents ϕ (null set)

ϵ represents $\{\epsilon\}$ (set consisting of null element)

** If r_1 and r_2 are two regular expression representing L_1 and L_2 then $r_1 + r_2$ is a regular expression representing r. $L = L_1 \cup L_2$

** $r_1 \cdot r_2$ represents $L = L_1 \cdot L_2 = \{xy / x \text{ in } L_1 \text{ \& } y \text{ in } L_2\}$

Kleen's closure

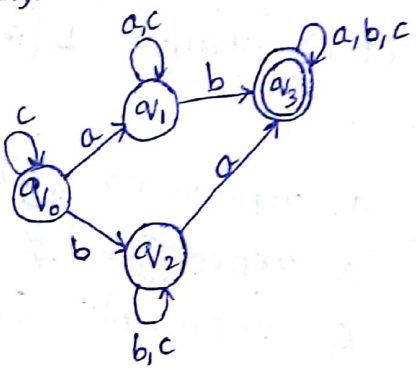
* r_1^* representing $L = L_1^* = \bigcup_{n=0}^{\infty} L_1^n$

L_1^n is L_1 concatenated with itself n times

Q:- Set of strings over {a b c} which contains atleast one a and atleast one b. also design a DFA

Language : ~~a*b*c*~~ (order not important)

(~~Regular Expression~~) $c^*(a(a+c)^*b + b(b+c)^*a)(a+b+c)^*$



Q:- Give a regular expression to recognize strings over {1,0} such that it should contains

(i) exactly one pair of consecutive 1's

$(10+0)^*\,11\,(01+0)^*$

↓ check start conditions with 1, 0

↓ check end conditions with 0, 1

(ii) atmost one pair of consecutive 1's

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

(iii) atleast one pair

$\left((10+0)^*\,11\,(01+0)^*\right)^+$

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

Q: Regular Expression for integers in C language.

Let dig = 0+1+2+3+4+5+6+7+8+9

dig^+

Note** :-

$[a? = (\epsilon + a)]$

↳ optional

~~$(\epsilon + -) =$~~

Ans: (sign) ? dig^+

Lex is an automated tool to recognise regular expression.
it recognises metacharacter
↳ character that is used to represent other character.

. → matches with every character except new line

- → (hyphen) range of characters

+ → operator.

decimal :-

$(dig)^+ \backslash . (digit)^+$

↳ match with '.' but not any character.

Regular Expression used to indicate Language used by Finite Automata.

$\epsilon + R = R$?

$+$ Union
$\cdot$ concatenation
$*$ closure

construct

ϵ-NFA to accept a language generated by a regular expression 'R'

Given r, construct ϵ-NFA A_E such that

$$L(A_E) = L(r)$$

(i) with zero operators

a $\{a\}$ → ○ —a→ ◎

ϵ $\{\epsilon\}$ → ◎

ϕ $\{\}$ → ○ ◎

r_1 & r_2 have less than 'n' operators

$R_1 = L(r_1)$ ∃ A_1 such that $L(A_1) = R_1$

$R_2 = L(r_2)$ ∃ A_2 such that $L(A_2) = R_2$

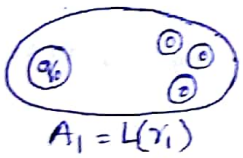
$r_1 + r_2$

$r_1 + r_2$

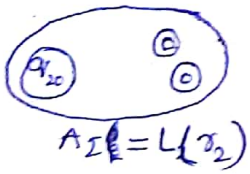
$L(r_1 + r_2)$

$L(r_1) \cup L(r_2)$

$R_1 \cup R_2$

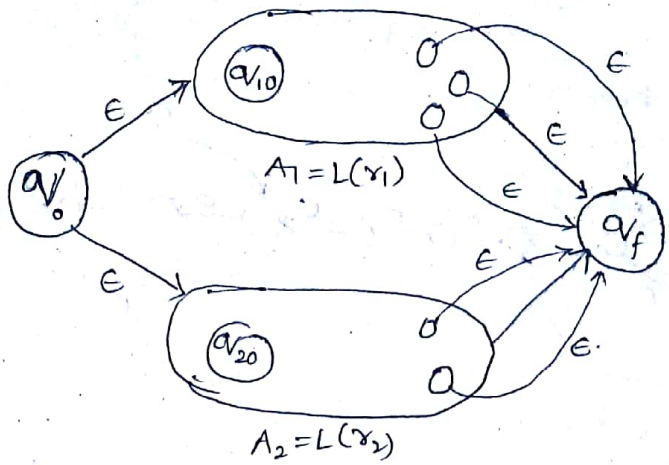


$A_1 = (Q_1, \Sigma, \delta_1, q_{10}, F_1)$

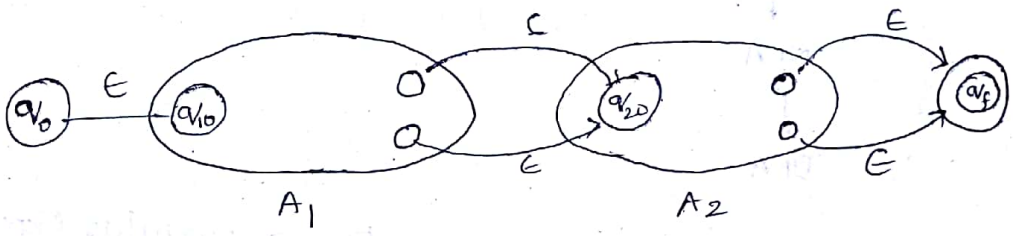


$A_2 = (Q_2, \Sigma, \delta_2, q_{20}, F_2)$

$$A = (Q_1 \cup Q_2 \cup \{q_0, q_f\}, \Sigma, \delta_1 \cup \delta_2 \cup \{\delta(q_0, \epsilon) = q_{10}, \delta(q_0, \epsilon) = q_{20}\}, q_f)$$



ii) $L(r_1 \cdot r_2) = L_1 \cdot L_2 = \{xy / x \in L_1 \text{ and } y \in L_2\}$



$A = (Q_1 \cup Q_2 \cup \{q_0, q_f\}, \Sigma, \delta_1 \cup \delta_2 \cup \{\delta(q_0, \epsilon) = q_{10}, \delta(q_{1i}, \epsilon) = q_{20}$ for all q_{1i} in F_1

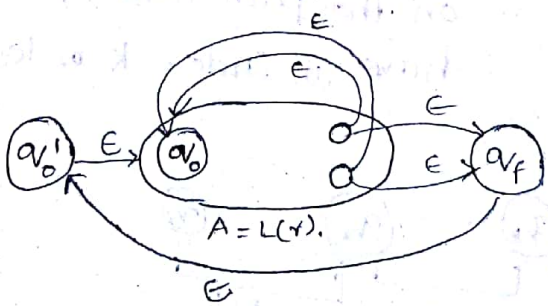
$\delta(q_{2j}, \epsilon) = q_f$ for all q_j in F_2

$F = \{q_f\}$).

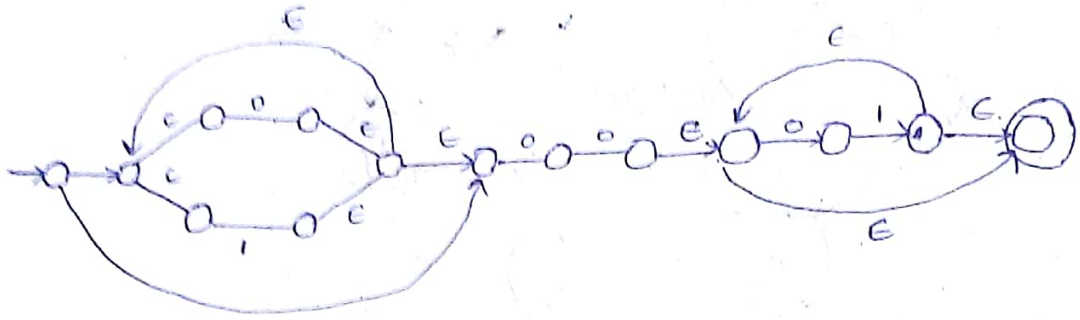
iii) $L(r^*)$.

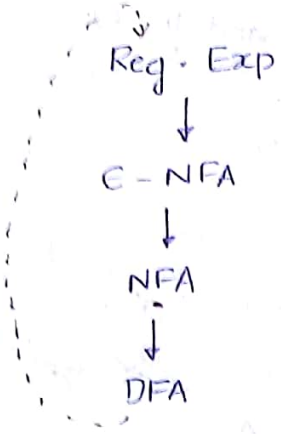
$L = L(r)$ $A = (Q, \Sigma, \delta, q_0, F)$

$L(A) = L$ and r has less than 'n' operators.



Q :- $(0+1)^* 00\,(01)^*$, construct ϵ NFA.



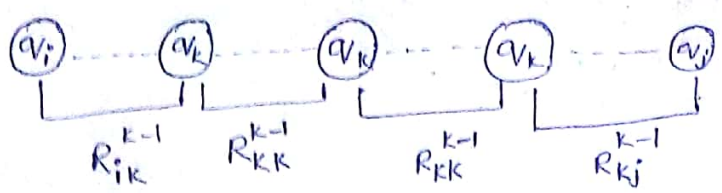


Given a DFA, can we generate a regular Expression

Q: Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$,
find a reg. expression such that $L(r) = L(A)$

consider $Q = \{q_1, q_2, \ldots q_n\}$

$R_{ij}^{k} = \{ x / \delta(q_i, x) = q_j$, all the intermediate states on the path from q_i to q_j have an index k or less $\}$

$R_{ij}^{k} =$ 

$= (R_{ik})^{k-1} (R_{KK}^{k-1})^* R_{kj}^{k-1} \quad \cup \quad R_{ij}^{k-1}$

$$R_{ij}^{k} = L\left(r_{ik}^{k-1}\left(r_{kk}^{k-1}\right)^{*} r_{kj}^{k-1} + r_{ij}^{k-1}\right)$$

$F = \{q_m, q_s\}$

$L(A) = \{x / \delta(q_1, x) = q_m \text{ or } q_s\}$

$L(A) = R_{1m}^{n} \cup R_{1s}^{n}$

$= L\left(r_{1m}^{n} + r_{1s}^{n}\right)$

$r_{ij}^{0} = a$ if $\delta(q_i, a) = q_j$ and $i \neq j$

$= a + \epsilon$ if $\delta(q_i, a) = q_j$ and $i = j$

$= \epsilon$ if $i = j$ & no transition

$= \phi$ if $i \neq j$ & there is no transition.

$r_{ij}^{1} = r_{i1}^{0}\left(r_{11}^{0}\right)^{*} r_{1j}^{0} + r_{ij}^{0}$

$\vdots$

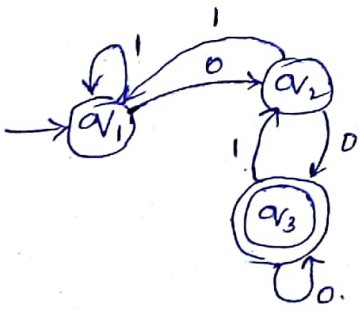
$r_{ij}^{n-1} =$

Q:- consider the DFA. Find r_{ij}^{k}

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

Sol:-

ij	k=0	k=1
11	1+ϵ	1*
12	0	1*0
13	ϕ	
21	1	
22	ϵ	
23	0	
31	ϕ	
32	1	
33	0+ϵ	



$r_{11}^{0} = 1+\epsilon$

$r_{12}^{0} = 0$

$r_{13}^{0} = \phi$

$r_{21}^{0} = 1$

$r_{22}^{0} = \epsilon$

$r_{23}^{0} = 0$

$r_{31}^{0} = \phi$

$r_{32}^{0} = 1$

$r_{33}^{0} = 0+\epsilon$

$$r_{11}^1 = r_{11}^0 (r_{11}^0)^* r_{11}^0 + r_{11}^0$$

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

$$= 1^* \qquad \text{/* Note:- it includes } \epsilon \text{ */}$$

$$r_{12}^1 = r_{11}^0 (r_{11}^0)^* (r_{12}^0) + r_{12}^0$$

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

$$= 1^* 0 + 0$$

$$= 1^* 0$$

$$r_{13}^1 = (r_{11}^0)(r_{11}^0)^* (r_{13}^0) + r_{13}^0$$

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

$$= 1^* \phi + \phi = 1^* \phi$$

$$r_{21}^1 = (r_{21}^0)(r_{11}^0)^* (r_{11}^0) + r_{21}^0$$

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

$$= 1^* + 1$$

$$r_{22}^1 = (r_{21}^0)(r_{11}^0)^* (r_{12}^0) + r_{22}^0$$

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

$$= 1^* + \epsilon$$

$$= 1^*$$

$$r_{23}^1 = (r_{21}^0)(r_{11}^0)^* (r_{13}^0) + r_{23}^0$$

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

$$= 1^* \phi + 0$$

$$r_{31}^1 = (r_{31}^0)(r_{11}^0)^* (r_{11}^0) + r_{31}^0$$

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

$$= 1^* \phi + \phi = 1^* \phi$$

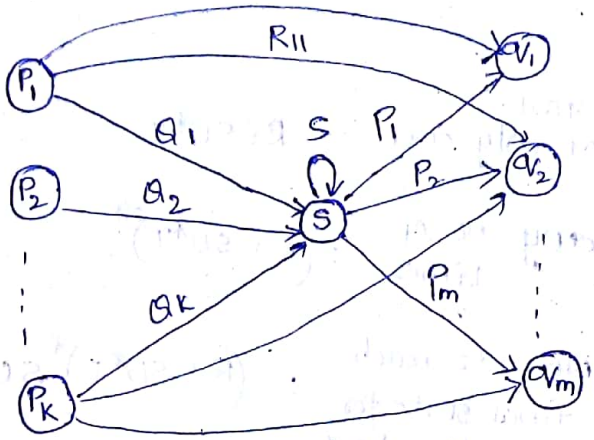
$$r_{32}^1 = (r_{31}^0)(r_{11}^0)^* (r_{12}^0) + (r_{32}^0)$$

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

$$= \phi 1^* 0 + 1$$

State elimination Method :-

For Finding reg. expres for Language accepted by a Finite Automata. by eliminating intermediate states.



* P_i, q_i may not be distinct

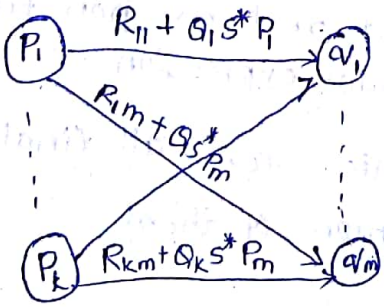
$i = 1$ to k

$$\delta(P_i, R_{ij}) = q_j$$

R_{ij} is a set of strings that take the finite Automata from P_i to q_j.

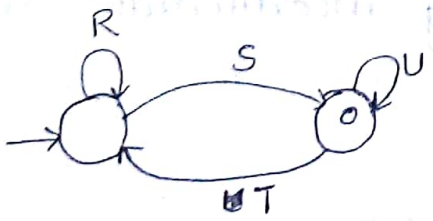
$$\delta(P_i, Q_i) = S$$
$$\delta(S, P_j) = q_j$$
$$\delta(S, S) = S$$

Now, if we eliminate S



** $$\therefore \delta(P_i, R_{ij} + Q_i S^* P_j) = q_j$$

(i) one initial state, one final state

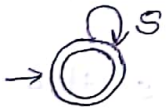


if it has to come to start
start only once : $R+SU^*T$

any no. of times : $(R+SU^*T)^*$

Now to reach final state for the last time : $(R+SU^*T)^*SU^*$

(ii) initial and final states are same



S^*

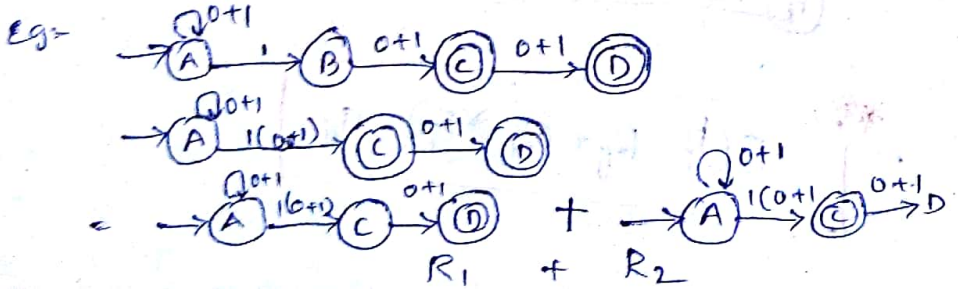
(iii) More than one final state.

Except one final state, remove all & take union of all.

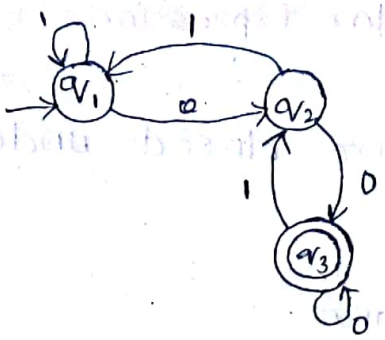
i.e, Assume at a time only final state is there and rest as non final find regular expression

Repeat this for all final states take union of them

Eg:-



Q :-



Closure properties of regular expression.

* Regular expressions are closed under

① + union

② . concatenation

③ * Kleen's closure

④ Complementation ($\bar{R}^*$)

↳ change all final states → non final states
non final states → final states

⑤ Intersection.

⑥ Reversal (L^R)

** For any given Regular set (R) then subsets of R need not necessarily be regular.

Prove that

If L_1 and L_2 are regular, then * $L_1 \cup L_2$, $L_1 - L_2$, L_1^*, $\bar{L}_1$, $L_1 \cap L_2$ are regular

Using Demorgan's Law

$$L_1 \cap L_2 = \overline{\bar{L}_1 \cup \bar{L}_2}$$

∴ if L_1, L_2 are regular then $L_1 \cap L_2$ are regular

Q:- Given a binary string over {0,1} reading from right to left, can we design DFA to recognize if its decimal equivalent divisible by 5.

Ans:- 11110

/* Note :- ---------- if L is regular, then L^R (L reverse) is also regular

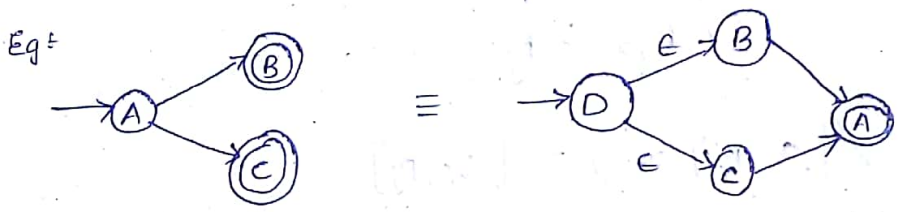
$L^R = \{x / x^R \text{ is in } L\}$

(i) interchange the states

(ii) Reverse the transitions

if there are multiple final state */

Eg:-



---------- */

Sol:-

Q:- Prove that if L_1 and L_2 are regular then $L_1 \cap L_2$ is regular without using Demorgans law.

$$A_1 = (Q, \Sigma, \delta_1, q_0, F_1)$$

$$A_2 = (P, \Sigma, \delta_2, p_0, F_2)$$

$$L(A) = L(A_1) \cap L(A_2)$$

$$A = (Q \times P, \Sigma, \delta[q_0, p_0], F_1 \times F_2)$$

$$\delta(q_0, x) = q_i$$

$$\delta(p_0, x) = p_j$$

$$\delta([q_0, p_0], x) = [q_i, p_j]$$

Pumping lemma

L is regular language

'n' is any constant

z is in L, $|z| > n$

split z as uvw such that $|uv| \leq n$, $|v| > 0$

$uv^i z$ for all $i \geq 0$, is in L

z → string

statement:-

Let L be a regular language. There exists a constant 'n' such that every string z in L, $(|z| \geq n)$ can be split as uvw suchthat $|uv| \leq n$, $|v| \geq 0$

Then for all $i \geq 0$

$uv^i w$ is also in L

Note:- Pumping lemma is used to show/prove some irregular language by proof by contradiction.

To show that L is not regular

Assume L is regular

Adversary : n is any constant

You : z is in L, $|z| \geq n$

Adversary : split z as uvw s.t $|uv| \leq n$, $|v| > 0$

chose i uv^iw for all $i \geq 0$

is not in L

Q:- Prove that $\{0^p \mid p \text{ is prime}\}$ is not regular.

Assume 'L' is regular

Let n be constant

Let $z = 0^p$ ($p \geq n+2$, p is prime)

$z = uvw$, $|uv| \leq n$, $|v| > 0$

$|uvw| = p$ Let $|v| = m$

$1 \leq |v| \leq n$

$|uw| = p-m$

Consider

$$|uv^{p-m}w| = |uw| + (p-m)|v|$$

$$= p-m + (p-m)(m)$$

$$|uv^{p-m}w| = (p-m) * (m+1)$$

↳ This is not prime, when $p-m \geq 2$ & $m+1 > 1$

As $p \geq n+2$, $m \leq n$

$\therefore p-m \geq 2$

and $m > 0$, $m+1 \geq 2$

$\therefore |uv^{p-m}w|$ is not prime

$\therefore$ L is not regular.

Q $L = \{0^k 1^k / k \geq 0\}$ is irregular.

'n' is constant

$z = 0^n 1^n$

$z = uvw$, $|uv| \leq n$, $|v| > 0$

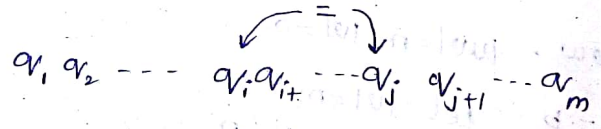
$z' = uw = 0^{n-m} 1^n$

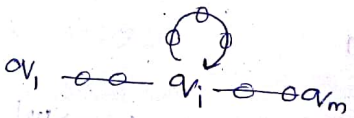
* L is regular

$(Q, \Sigma, \delta, q_1, F)$ accepts L

$Q = \{q_1, q_2, \ldots q_n\}$

$x = a_1 a_2 \ldots a_m$





** Two states in Finite Automata are distinguishable if one is in F and other is not in F.

q_1 & q_2 are distinguishable

if $q_1 \xrightarrow{x} p_1$

$q_2 \xrightarrow{x} p_2$ and

p_1, p_2 are distinguishable

States

→	A
	B
*	C
	D
	E
*	F
	G
	H
*	I

	B
	C
	D
	E
	F
	G
	H
	I

First

(A,B) $\xrightarrow{0}$

(A,D) $\xrightarrow{0}$ $\xrightarrow{1}$

(A,E) $\xrightarrow{0}$

(A,G) $\xrightarrow{0}$ $\xrightarrow{1}$

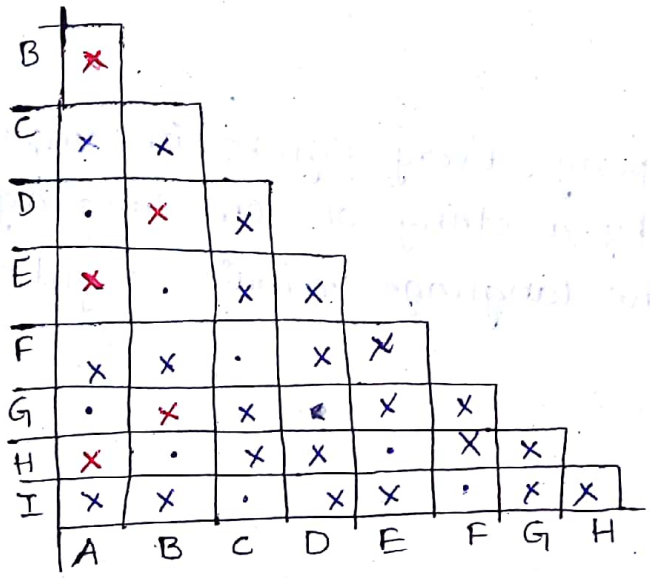
(A,H) $\xrightarrow{0}$

(B,D) $\xrightarrow{0}$

(B,E) $\xrightarrow{0}$ $\xrightarrow{1}$

States Minimization Algorithm

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



First mark clearly distinguishable pairs

(A,B) $\xrightarrow{0}$ (B,C) ✓

? (A,D) $\xrightarrow{0}$ (B,E)
$\xrightarrow{1}$ (E,H)

(A,E) $\xrightarrow{0}$ (B,F) ✓

(A,G) $\xrightarrow{0}$ (B,H)
$\xrightarrow{1}$ (E,B)

(A,H) $\xrightarrow{0}$ (B,I) ✓

(B,D) $\xrightarrow{0}$ (C,E) ✓

(B,E) $\xrightarrow{0}$ (C,F)
$\xrightarrow{1}$ (F,I)

(B,G) $\xrightarrow{0}$ (C,H)

(B,H) $\xrightarrow{0}$ (C,I)
$\xrightarrow{1}$ (F,C)

(C,F) $\xrightarrow{0}$ (D,G)
$\xrightarrow{1}$ (H,B)

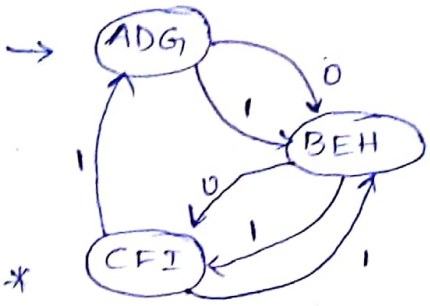
(C,I) $\xrightarrow{0}$ (D,A)
$\xrightarrow{1}$ (H,E)

(D,G) $\xrightarrow{0}$ (E,H)
$\xrightarrow{1}$ (H,B)

(E,H) $\xrightarrow{0}$ (F,I)
$\xrightarrow{1}$ (I,C)

(F,I) $\xrightarrow{0}$ (G,A)
$\xrightarrow{1}$ (B,E)

A,D	B,E	C,F
A,G	B,H	C,I
D,G	E,H	F,I



* substitution property : Every symbol in input stream is replaced by a string of some language so, that still the language remains regular.

* Homomorphism

* If there is not transition from initial state to final state then the Language is said to be empty.

Q: $L_1 = \{x / 00$ is not a substring of $x\}$ $\Sigma = \{0, 1\}$

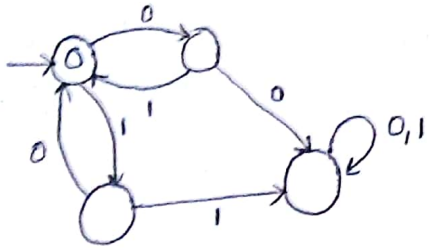
$L_2 = \{x / x$ ends with $01\}$ L_1, L_2 are two regular language.

(i) DFA for $L_1 - L_2$

Q: $0^*1(0^*10^*1)^*0^*$ What is the Language accepted by this.

Strings containing odd number of 1's

Q. What is the language accepted



$(01+10)^*$

Strings in which there are no three consecutive zeroes.

*** Note:-

$\phi + L = L + \phi = L$

$\epsilon \cdot L = L \cdot \epsilon = L$

$\phi \cdot L = L \cdot \phi = \phi$

$L(M+N) = LM + LN$

$L + L = L$

$L^* = \epsilon + L^+$

$L? = \epsilon + L$

$L^+ = L \cdot L^*$

CONTEXT FREE Grammar (CFG)

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

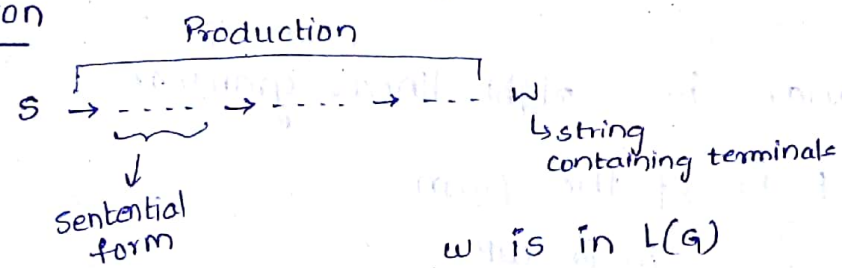
V - variable symbol (i.e., can be replaced with string)

T - Terminal symbol (i.e., cannot be replaced with any symbol/string)

P - Production (i.e., rules) — set of

S - start symbol

* **Derivation**



w is in L(G)

a, b, c — Terminals

S, A, B, C — variable

x, y, z — strings of terminal

X, Y, Z — single symbol either Variable or Terminal

α, β, γ — strings of grammar symbols

i.e., α is in $(V \cup T)^*$

Γ (gamma) — string consisting of only variables

Γ is in V^*

'$\rightarrow$' derives.

$A \rightarrow \alpha$ production

General form of production

$\alpha \rightarrow \beta$

* If the left side of productions are single variable

$\alpha \rightarrow \beta$

i.e; If all α's are single variables

(If all the productions are of the form $A \rightarrow \beta$)

G_1 is either regular or CFG

* Grammar is right linear grammar (↳ also regular)

if β is of the form

w or wB

* Grammar is left linear grammar

if β is of the form

w or Bw

* $\alpha \rightarrow \beta$

if there are no restrictions on 'α' & 'β' then grammar is unrestricted grammar.

* If the restriction is on only length of α, β then the grammar is context sensitive.

Q: $G = (\{S\}, \{0,1\}, P, S)$

$P: S \to 0S1$

$S \to \epsilon$

Sol: $S \to 0S1 \to 00S11 \to 0011$

$\epsilon, 01, 0011, 000111, \ldots$

$\Rightarrow 0^n 1^n$

Note:-

1. $A \to \beta_1$

   $A \to \beta_2$

   $A \to \beta_3$

   $A \to \beta_1 | \beta_2 | \beta_3$

Q:- $S \to aSa | bSb | a | b | \epsilon$

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

What is the Language accepted by this.

Sol:- All palindromes over $\{a,b\}$

$= \{w / w = w^R\}$

Q: $E \to E+E \mid E*E \mid (E) \mid a$

What are the strings that can be generated

All binary expressions with + and *

() → is used to disambiguate mul & add.

Eg: $a + a * a$

$E \to E+E$

$\to E + E*E$

$\to a + E*E$

$\to a + a*E$

$\to a + a*a$

Left most derivation sequence

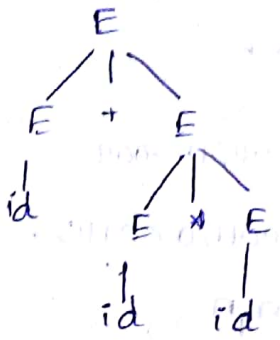
$E \longrightarrow E+E$

$\rightarrow id+E$

$\xrightarrow{lm} id+E*E$

$\rightarrow id+id*E$

$\rightarrow id+id*a$



Left most derivation tree

~~Right~~ Left most derivation Sequence

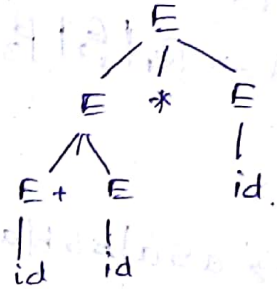
$E \longrightarrow E*E$

$\rightarrow E+E*E$

$\rightarrow id+E*E$

$\rightarrow id+id*E$

$\rightarrow id+id*id$



** Given string 'w' has two or more left most derivations (or) two or more right most derivations then the grammar corresponding is said to Ambiguous.

* Ambiguity is due to precedence of operators

Q:- $E \rightarrow E+T$

$T \rightarrow T*F$

$F \rightarrow (E)|i$

This is

but here

* is given

i.e., only

any

* exponentiation

it must

E

T

F

G

Flaw is

left

Q. $E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

This is similar to previous question but here there is no ambiguity i.e; * is given higher preference over '+' i.e, only 1 left most derivation tree for any particular sequence.

* exponentiation is right associative

it must be given higher precedence operator.

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow F \wedge G \mid G$

$G \rightarrow (E) \mid id$

Flaw is, it assumes exponentiation as left associative

↓ To avoid this

$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow G \wedge F \mid G$

$G \rightarrow (E) \mid id$

} Grammar for expression with '+', '-', '*'

12/09

$S \to \alpha AB \to \alpha\gamma\beta$

↓ sentential form

$S \overset{*}{\underset{G}{\Rightarrow}} w$ → number of productions (0 or more)

G → Grammar which is involved in production

Q:- can we generate CFG generated over {a, b} with strings containing equal number of a's and b's

Sol:- Yes

$S \to asbs \mid bsas \mid \epsilon$

$S \to asbs \to abs \to abasbs \to ababs \to abab$

it is ambiguous [∵ two (or) more leftmost derivation trees]

Q:-

Q: Generate a CFG for the language containing strings

(i) $x = \{ a^i b^j c^k \mid i, j, k \geq 0 \}$

$S \rightarrow AB$

$A \rightarrow aAb \mid \epsilon$

$B \rightarrow cB \mid \epsilon$

(A for a, b
B for c)

(ii) $x = \{ a^i b^j c^k \mid i, j, k \geq 0 , i \neq j \}$

$S \rightarrow AB$

$A \rightarrow aAb \mid aC_1 \mid bC_2$

$C_1 \rightarrow aC_1 \mid \epsilon$

$C_2 \rightarrow bC_2 \mid \epsilon$

$B \rightarrow cB \mid \epsilon$

Q:- What is the Language generated by the Grammar

$$E \to E+E \mid E*E \mid (E) \mid I$$
$$I \to Ia \mid Ib \mid I0 \mid I1 \mid a \mid b$$

Sol:- Names of the identifiers

↓

alpha (alpha/digit)*

↓ ↓

{a,b} {0,1}

Q:- Consider the Language $L = \{a^n b^n c^m d^m, n,m \geq 1\}$ & Generate grammars.

$$S_1 \to A_1 B_1$$
$$A_1 \to aA_1b \mid ab$$
$$B_1 \to cB_1d \mid cd$$

$$L_2 = \{a^n b^m c^m d^n, n,m \geq 1\}$$

$$S_2 \to aA_2d \mid aS_2d$$
$$A_2 \to bA_2c \mid bc$$

$L_1 \cup L_2 = \{a^+b^+c^+d^+$

$L_1 \cap L_2 = \{a^n b^n c^n d^n, n \geq 1\}$

CFG's are closed under union but not on intersection.

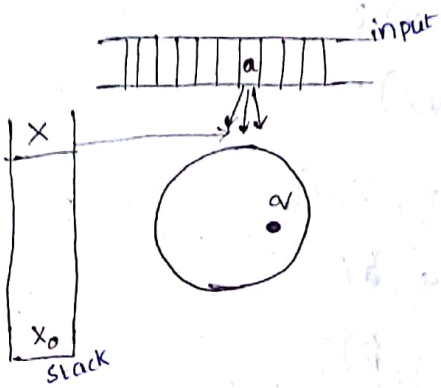
PUSH-DOWN AUTOMATA (PDA)

- Finite state machine
- Additional stack (to store symbols)
- Non determinstic

$P = (Q, \Sigma, \Gamma, \delta, q_0, X_0, F)$

Symbol for special, bottom of stack

- PDA
- set of finite states
- input alphabet
- stack alphabet
- start state



$\delta(q, a, X) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots\}$

↓

pops out X and replaces γ

γ → symbol containing both variable or terminals

$\hat{\delta}(q_0, w, X_0)$

(i) acceptance by final state $L = L(P)$ → L is accepted PDA on final state

(ii) acceptance by empty stack : i.e., if after reading entire input if stack is empty then it is accepted. $L = N(P)$

Q.

$$P = (\{q_0, q_1, q_2\}, \{0,1\}, \{X, Y, Z_0\}, \delta, q_0, Z_0, \{q_2\})$$

$\delta(q_0, 0, Z_0) = (q_0, XZ_0)$

X is being pushed
(i.e, Z_0 is poped & XZ_0 is pushed)

$\delta(q_0, 1, Z_0) = \{(q_0, YZ_0)\}$

$\delta(q_0, 0, X) = \{(q_0, XX)\}$

$\delta(q_0, 1, Y) = \{(q_0, YY)\}$

$\delta(q_0, 0, Y) = \{(q_0, XY)\}$

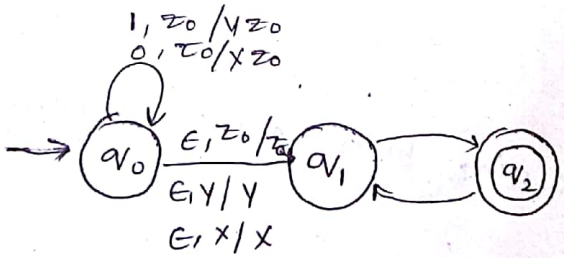
$\delta(q_0, 1, X) = \{(q_0, YX)\}$

$\delta(q_0, \epsilon, Z_0) = \{(q_0, Z_0)\}$

$\delta(q_0, \epsilon, Y) = \{(q_0, Y)\}$

$\delta(q_0, \epsilon, X) = \{(q_0, X)\}$

$\delta(q_1, \epsilon, Z_0) = \{(q_2, Z_0)\}$



Instantaneous Description of the PDA

$(q, aw, x\gamma)$

- q → current state
- aw → unexplored input beginning with correct input symbol
- $x\gamma$ → stack contents with top of stack as left most

$(q, aw, x\alpha) \vdash (p, w, \beta\alpha)$ if $\delta(q, a, x)$ contains (p, β)

Initial Instantaneous description of any PDA

$$(q_0, w, x_0) \overset{*}{\underset{P}{\vdash}} (p, \epsilon, \alpha), \; p \text{ in } F$$

$$L(P) = \{ w / (q_0, w, x_0) \overset{*}{\underset{P}{\vdash}} (p, \epsilon, \alpha), \; p \text{ in } F \}$$

If p is in F, w is in $L(P)$

$$N(P) = \{ w / (q_0, w, x_0) \overset{*}{\underset{P}{\vdash}} (p, \epsilon, \epsilon) \}$$

**