T.O.C [Note] - its a new syllabus notes of TOC

Theory of Computation (Pokhara University)

Proof techniques

Date
Page

1) Mathematical Techniques

The principle of mathematical induction states that only set of natural number containing zero, and with the property that it contains n+1 whenever ~~natural~~ n contains all numbers up to end including n, must in fact be the set of all natural number.

It has three steps

(a) Basic step. First we show that property i is true for 0

(b) Induction Hypothesis Assume that i holds for 0, 1, 2, ... n.

(c) Induction step: using the induction hypothesis show that i is true for n+1.

d Then by the induction principle, i is true for all natural numbers

Example

Prove that the sum of first n natural numbers is given by

$$1+2+3+\cdots+n = \frac{n(n+1)}{2}, n \geq 1$$ by mathematical induction

Date
Page

Proof :- We can use mathematical induction as follow

(a) Basic step for $n=1$, LHS $=1$ & RHS $= \frac{1(1+1)}{2}$

LHS = RHS $= 1$ $= 1$

This implies that, basic step is true for $n=1$

(b) Inductive hypothesis

for $n=k$ given statement becomes

$$1+2+3+\cdots+k = \frac{k(k+1)}{2}$$

(c) Induction step

for $n=k+1$, given statement becomes

$$1+2+3+\cdots+k+k+1$$

$$= \frac{k(k+1)}{2} + k+1$$

$$= \frac{k(k+1)+2(k+1)}{2} = \frac{(k+1)(k+1)}{2}$$

This shows that the given statement is true for $n=k+1$

Hence, $1+2+3\cdots+n = \frac{n(n+1)}{2}$ proved #

Date
Page

Example:- prove that sum of first n positive odd number $1+3+5+\cdots+2n-1=n^2$

(2) The pigeonhole principle

If S_1 & S_2 are non-empty finite sets and $|S_1| > |S_2|$, then there is no one-to-one function from S_1 & S_2 [i.e there is no bijection)

let A and B be two finite sets with $|A| > |B|$ Then one to one function can not exists from A to B

(3) The Diaginalization principle

Consider set S_i with numbers $k \in N$ as initialized below.

$S_i \backslash k$	0	1	2	3
S_0	•		•	•
S_1				
S_2	•	•		
S_3				•

Now, we define the diagonal principle as follows:-

let R be a binary relation on a set S, let D the diagonal set of R, be

$\{x : x \in b$ and $(x,x) \notin R\}$ for each $x \in S$; let $Rx = \{b : be\ S$ and $(x,y) \in R\}$, then D is distinct from each Rx

$R_0 = \{0, 2, 3\}$

$R_1 = \{\}$

$R_2 = \{0, 1, 3\}$

$R_3 = \{3\}$

(4) Proof By contradiction

Prove that sum of rational and irrational number is irrational by using ~~part by~~ proof by contradiction technique

Proof:-

We have to show that " sum of rational and irrational number is irrational.

Let 'r' and 'x' be two rational and irrational numbers respectively.

We have to prove $(r+x)$ is irrational
Assume that $(r+x)$ is rational. we know that if r is rational, then $-r$ is also rational. And sum of two rational number is always rational.

Then we can write :-

$(r+x) + (-r) \Rightarrow$ rational

$r + x - r \Rightarrow$ rational

$x \rightarrow$ rational

Which contradicts that our orginal assumption

that x is irrational Hence, sum of rational and irrational number is irrational proved.

5. Doretailing

We can show that union of finite number of countably infinite set is countably infinite.

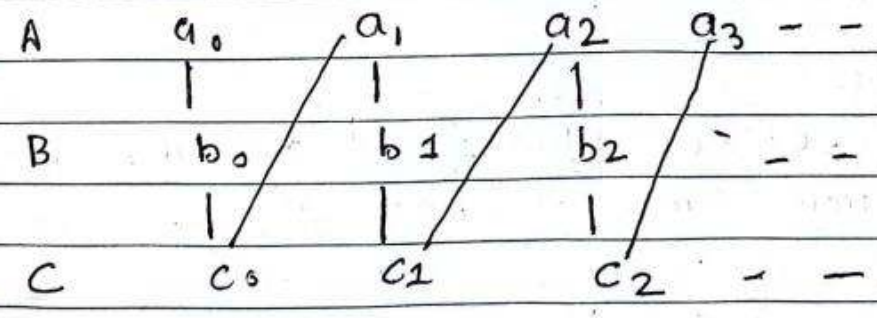
Let us take ~~three~~ three countably infinite sets A, B and C as:-

$$A = \{a_0, a_1, a_2, \ldots\}$$
$$B = \{b_0, b_1, b_2, \ldots\}$$

Now $A \cup B \cup C = \{a_0, b_0, c_0, a_1, b_1, c_1, \ldots\}$

In doretailing, we visit first elements of first set first element of second set, first element of third set and so on.

It can be visualize as



Hence, doretailing can be defined as technique of interleaving the enumeration of several sets

Date
Page

Concatenation of string

Concatenation of two strings x and y is denoted by $x.y$ or xy and is given by:

If $x = 010$, $y = 101$ then, $x.y = 010101$

* Substring

A string v is said to be substring of a string as if $w = xvy$

- Both x & y called be ϵ
- A string is substring of itself.

* Reversal of string

let w be any string then it reversed is denoted by w^R let $w = baa$ then $w^R = aab$.

Language:

A set of strings all of which are taken from some alphabet Σ^* is called language.

$$L \subseteq \Sigma^*$$

Eg:- $L = \{w \in \{a, b\}^* :$ we has odd number of $a\}$

$L = \{a, ab, abaa, aaa, abaab, \ldots\}$

Date
Page

Eg:- The language of all strings consisting of n 0's followed by n 1's, for some $n \geq 0$:

$L = \{w \in \{0,1\}^* : w$ has n 0s followed by n 1's and $n \geq 0\}$

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

Note: Σ^* is a language for any alphabet Σ.

- ϕ, the empty language, is a language over any alphabet.
- $\{\epsilon\}$, the language consisting of only the empty string, is also a language over alphabet.
- $\phi \neq \{\epsilon\}$, the former has no strings and later has, one string.

The function L is defined as follows

(i) $L(\phi) = \phi$ and $L(a) = \{a\}$ for each $a \in \Sigma$

(ii) If α and β are regular expressions, then $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$

~~(iii) If α~~

(iii) If α and β are regular expressions, the $L(\alpha\beta) = L(\alpha)\, L(\beta)$

(iv) If α is a regular expression, then $L(\alpha^*) = L(\alpha)^*$ kleen closure

What language is represented by

RE · $L((a \cup b)^* a)$

Date
Page

$= L((a \cup b)^* \{a\})$
$= (L(a) \cup L(b))^* \{a\}$
$= (\{a\} \cup \{b\})^* \{a\}$
$= \{a,b\}^* \{a\} = \{w \in \{a,b\}^* \mid w \text{ ends with a}\}$

Regular Expression

Regular expression is a formula representing language in terms of a form that uses three operations: Concatenation with and kleen closure. The expression is regular if:

ϕ and each members of Σ are regular expression.

If α and β are regular expression then so is $(\alpha\beta)$

If α and β are regular expression then so is $(\alpha \cup \beta)$

If α is regular expression then so is α^*

Nothing is else regular expression and the language they represented is established by the use of the function L.

$L(\phi) = \phi$ and $L(a) = \{a\}$ for each a in Σ

$L((\alpha \cup \beta)) = L(\alpha) \cup L(\beta)$

$L((\alpha\beta)) = L(\alpha) \cdot L(\beta)$

$L(\alpha)^* = L(\alpha)^*$

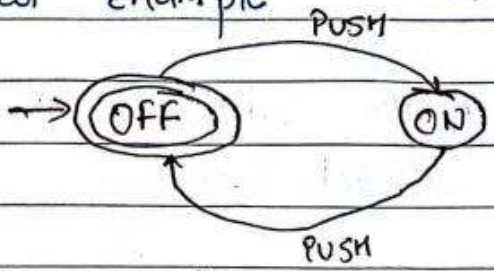
Date
Page

Finite Automation and Register language

Finite Automation

A finite automation has finite set of states / edges lead from one state to another and each edge is labeled with a symbol

Practical Example



switch

some uses:-

- software for designing and checking behavior of digital circuits
- ~~Used~~ lexical analyzer of typical compiler
- for scanning large batteries of texts such as collections of webpages.
- Software for verifying systems.

Block Diagram

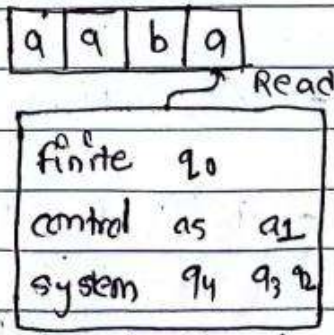


figure: A finite automation q_0, q_1, q_2, q_3, q_4 q_5 are

Date
Page

states in finite control system and a & b we input symbols.

Deterministic Finite Automation (DFA)

Formal Definition

A Deterministic finite automate is a quintuple

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

where, Q: is a non-empty finite set of states present in the finite control system.

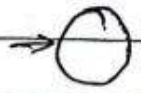
Σ: is an input alphabet

q_0: is an ~~input~~ initial state

F: is a set of final states $F \subseteq Q$

δ: is a transition function defined from $Q \times \Sigma$ to Q

Transition Graph

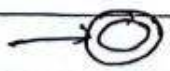


initial state

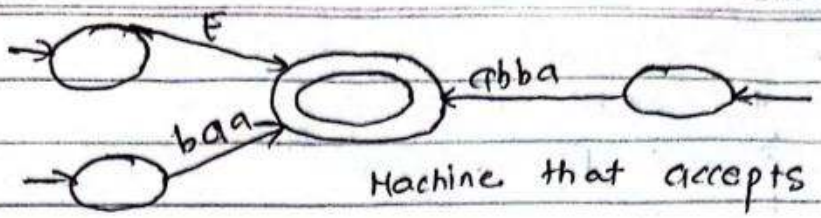
final state / accepting state

Examples

a)  Machine that accepts nothing

b)  Machine that accepts NULL string.

c)



Machine that accepts ε, bba & abba

Design Example

Design a deterministic finite automate (DPA) is that accepts the language given by L = {w ∈ {a,b}* : w has even number of b}

Soln let M = (Q, Σ, δ, S, F) be

where, Q = {q0, q1}

Σ = {a,b}

P = q0

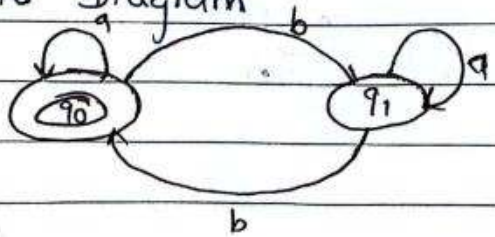
F = {q0}

And the transition function δ is as follow:

Transition table

State \ input	a	b
q0		
q1		

State Diagram



let us test for a string

(q0, abab) → (q1)

Date
Page

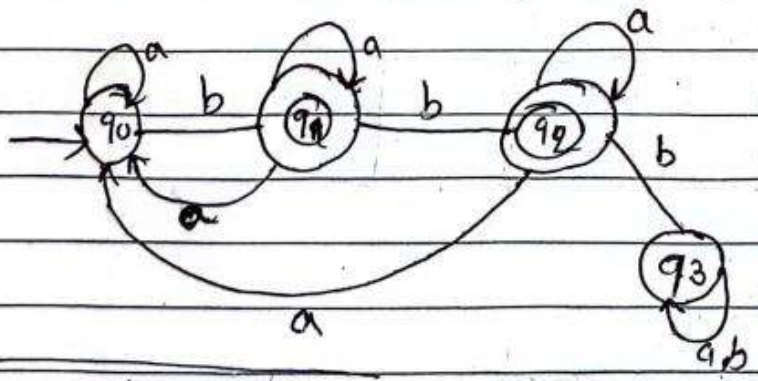
$(q_0, abab) \vdash^{a} (q, bab)$
$\vdash^{b} (q, ab)$
$\vdash^{a} (q_1, b)$ ~~Accepted~~
$\vdash^{b} (q_0, f)$ Accepted ||

Design a deterministic finite automate (DFA) is that accepts a language given by

$L = \{ w \in \{a, b\}^* : w$ done does not contain three consecutive b's $\}$

Soln let $M = (Q, \Sigma, \delta, s, F)$ be required DFA

where, $Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$s = q_0$
$F = \{q_0, q_1, q_2\}$



state	diagram a	b
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_3
q_3	q_3	q_3

Date
Page

3. Design a DFA that accepts a language given by $w = \{ w \in \{0,1\}, w \text{ begins with } 01 \}$

4. Design a DFA that accepts a language given by $L = \{ w \in \{0,1\}^* : w \text{ begins with } 01 \text{ and have even length} \}$

5. Design a DFA that accepts a language given by $L = \{ w \in \{0,1\}^* : w \text{ has neither, } 00 \text{ nor } 11 \text{ as substrings} \}$

6. Design a DFA that accepts a language given by $L = \{ w \in \{a,b\}^* : w \text{ has number of 'a' multiple of 3} \}$

7. Design DFA ~~for~~ the following language ~~given~~ are $\{0,1\}$

a. All strings with even no of 0s and even no of 1s
b. All strings of length at most 4.

Solutions :-

(3) DFA that accepts a language
w begins with 01

Date
Page

Non- Deterministic Finite Automata (NFA OR NDFA

NDFA has ability to change states in a way that is only partially determined by the current state and input symbol. There will be several possible "next states" for a given combination current state and input symbol.

NDFA needs the input string and choose at each step to go into any of the legal next states, the choice is not determined by anything in the model and is therefore said to be non- deterministic.

Formal Definition:

A non- deterministic automation (NDFA) is a quntiple.

$$M = (Q, \Sigma, \Delta, S, F)$$

where,

Q: is a finite set of states
Σ: is a set of input symbol
$s = s \in Q$ is an initial state
$F = F \subseteq Q$ is a final state
Δ = is transition function mapping from $Q \times \Sigma \rightarrow 2^Q$ which is power set of Q.

Example 1: Design a non-deterministic finite Automat (NFA) that accepts the set of stings containing

an occurance of the pattern bb or of the pattern bab.

Sol^n

Let $M = (Q, \Sigma, \Delta, S, F)$ be the required NFA

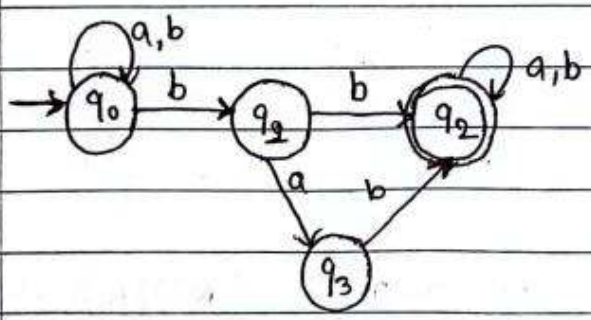
where, $Q = \{q_0, q_1, q_2, q_3\}$

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

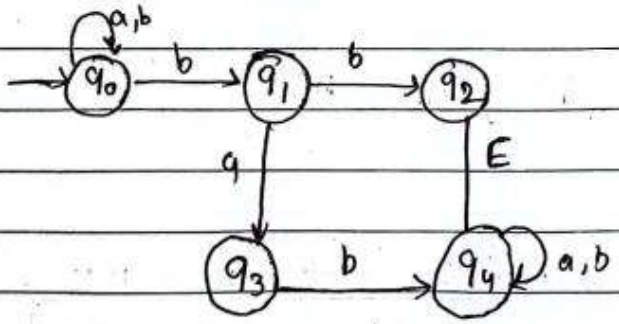
$S = q_0$

$F = \{q_2\}$

And transition is given by



Q \ Σ	a	b	E
q_0	q_0	q_0, q_1	–
q_1	q_3	q_2	–
q_2	ϕ	ϕ	q_4
q_3	ϕ	q_4	.
q_4	q_4	q_4	



where, $Q = \{q_0, q_1, q_2, q_3, q_4\}$

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

$S = q_0$

$F = \{q_2\}$

Date
Page

#2 Design a NFA for the language L = all strings over {0,1}, that have at least to consecutive 0's or 1's.

Soln

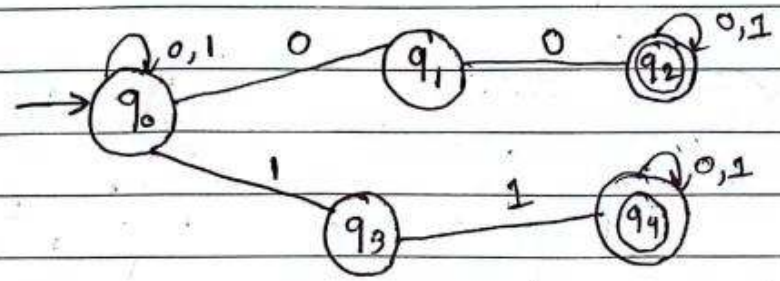
Let M = (Q, Σ, Δ, S, F) be the required NFA

where, Q = {q0, q1, q2, q3, q4}

Σ = {0, 1}

S = q0

F = {q2, q4}

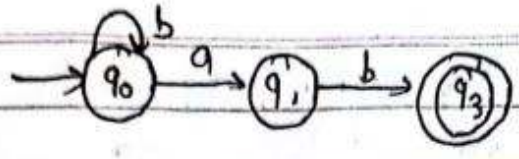


Q \ Σ	0	1
q0	q0, q1	q0, q3
q1	q2	ϕ
q2	q2	q2
q3	ϕ	q4
q4	q4	q4

#3 NDFA for R = b* ab | bb* ab

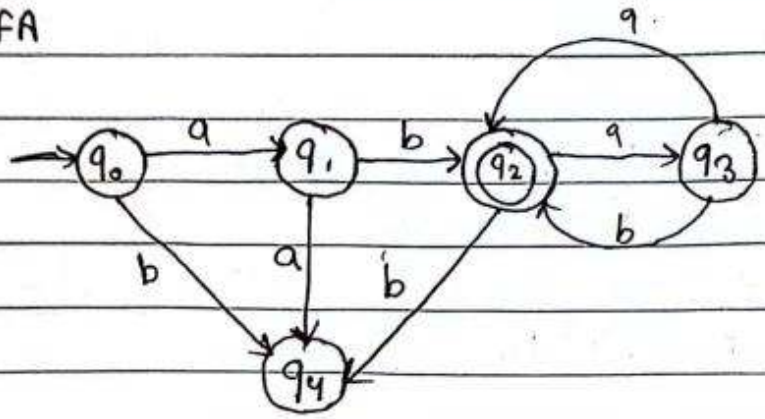
Soln

let M = (Q, Σ, Δ, S, F) be the required NDFA

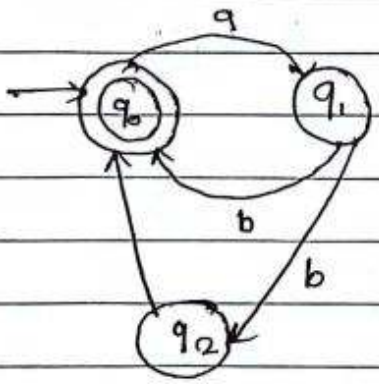


#4- Design a DFA for a regular expression (abUaba)*. And also design NFA for same regular expression.

DFA

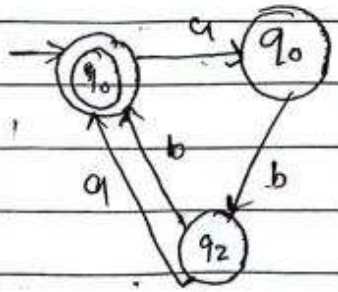


NFA



NFA for (abUaba)*

or,



NFA for (abUaba)*

#5 Find the NFA with four state for the language $L = \{(a^n : n \geq 0) \cup (b^*a : n \geq 1)\}$

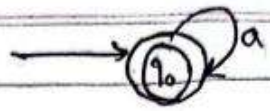
Soln

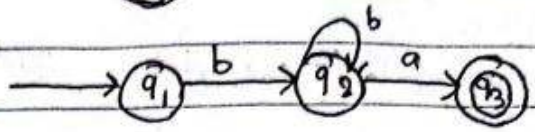
$L = L_1 \cup L_2$

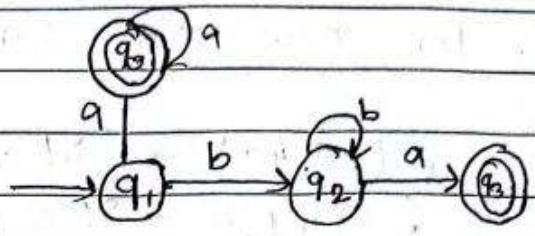
$L_1 = a^n : n \geq 0$

$L_2 = b^n a : n \geq 1$

Date
Page

NFA for L1 is 

NFA for L2 is 

NFA for L1UL2 is 

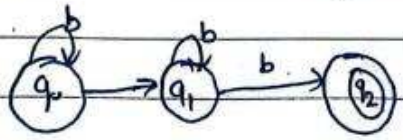
Equivalence of DFA and NFA

" For each non-deterministic finite automation there is an equivalent deterministic finite automation.

Steps :-

1. Observe all the transitions from starting state q_0 for every symbols in Σ.
2. For new states appeared in step1, check all the transition for every Σ
3. Repeat step1, till new states are appeared.

Example :-
convert the following NFA into equivalent DFA



Solution :-

step1: Here initial state is q_0, so we write (q_0) for new DFA

Then, $\delta(\{q_0\}, 2) = \delta(q_0, q) = \{q_1\}$ new state.

$\delta(\{q_0\}, b) = \delta(q_0, b) = \{q_0\}$ old state.

Step2: For new state $\{q_1\}$, we have

$\delta(\{q_1\}, a) = \delta(q_1, q) = \phi$

$\delta(\{q_1\}, b) = \delta(q_1, b) = \{q_1, q_2\}$ new state.

Step3: For new state $\{q_1, q_2\}$ where,

$\delta(\{q_1, q_2\}, a) = \delta(q_1, a) \cup \delta(q_2, q)$

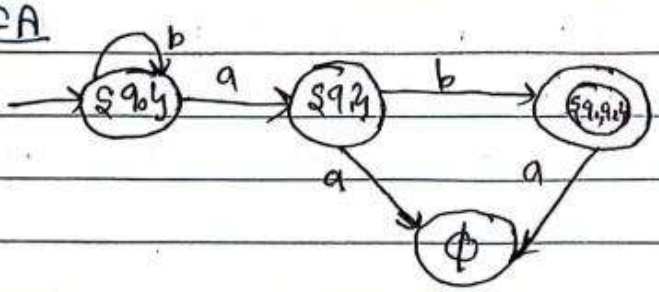
$= \phi \cup \phi = \phi$

$\delta(\{q_1, q_2\}, b) = \delta(q_1, b) \cup \delta(q_2, b)$

$= \{q_1, q_2\} \cup \phi$

$= \{q_1, q_2\}$ old states.

DFA



State	a	b
$\{q_0\}$	$\{q_1\}$	$\{q_0\}$
$\{q_1\}$	ϕ	$\{q_1, q_0\}$
$\{q_1, q_2\}$	ϕ	$\{q_1, q_0\}$

Date
Page

Conversion of ε-NFA to DFA

ε- Transitions are the transition made without symbols

steps:

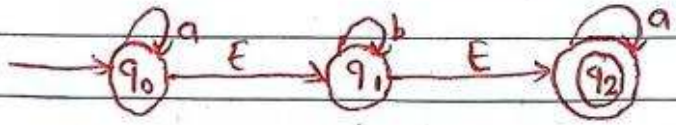
1. Find out the ε- closure of starting state and calculate the union of ε- closure and transitions of every symbol of Σ.

Note:- ε- closure is a set of the states in ENFA which can be reached without consuming symbols.

2. Repeat the same process till new states are appeared.

Example:

convert the following NFA to DFA



sol^n

ε- closure of q_0 is ε- close (q_0) or

$E(q_0) = \{q_0, q_1, q_2\}$ similarly,

$E(q_1) = \{q_1, q_2\}$

$E(q_2) = \{q_2\}$

Date
Page

Now,

$$\delta'(\{q_0, q_1, q_2\}, a) = \delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)$$

$$= E(q_0 \cup \phi \; q_2)$$
$$E(q_0) \cup E(q_2)$$
$$(q_0, q_1, q_2) \cup \{q_2\}$$
$$= \{q_0, q_1, q_2\}$$

$$\delta'(\{q_0, q_1, q_2\}, b) = E(\delta(q_0, b) \cup \delta(q_2, b) \cup \delta(q_1, b))$$

$$= E(\phi \cup q_1 \cup \phi)$$
$$= E(q_1) = \{q_1, q_2\} \text{ new state.}$$

Now, using new state,

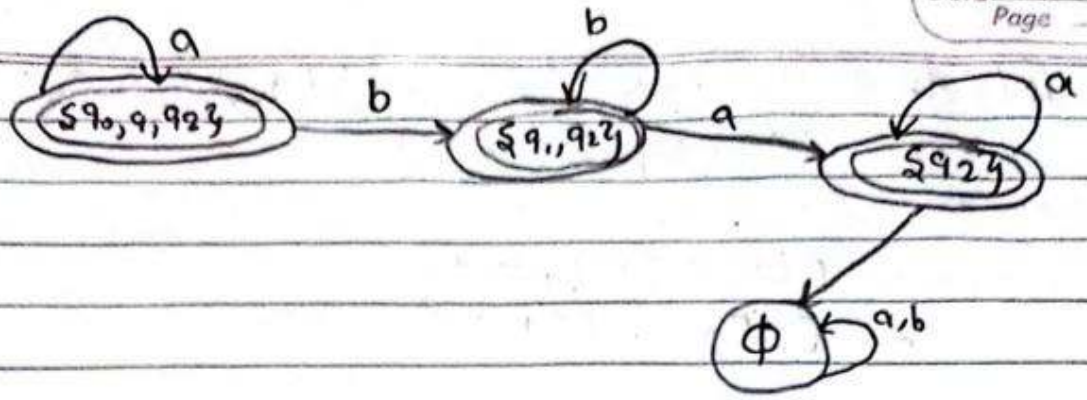
$$\delta'(\{q_1, q_2\}, a) = E(\delta(q_1, a) \cup \delta(q_2, a))$$
$$= E(\phi \cup q_2)$$
$$= E(q_2) = \{q_2\} \text{ new state.}$$

$$\delta'(\{q_1, q_2\}, b) = E(\delta(q_1, b) \cup \delta(q_2, b)$$
$$= E(q_1 \cup \phi) = E(q_1) = \{q_1, q_2\} \text{ old state.}$$

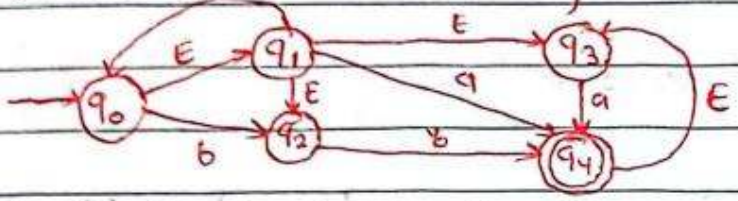
using new state $\{q_2\}$

$$\delta'(\{q_2\}, a) = \cancel{E(q_2, a) = E\{q_2\}}.$$
$$= E(\delta(q_2, a)) = E(q_2) = \{q_2\} \text{ old state}$$

$$\delta(\{q_2\}, b)' = E(\delta(q_2, b)) = E(\phi) = \phi$$



Convert the following NFA to DFA



ε- closure of q_0 is ε-close (q_0) or

$E(q_0) = \{q_0, q_1, q_2, q_3\}$
$E(q_1) = \{q_1, q_2, q_3\}$
$E(q_2) = \{q_2\}$
$E(q_3) = \{q_3\}$
$E(q_4) = \{q_3, q_4\}$

Now, $\delta'(\{q_0, q_1, q_2, q_3\}, a) = E(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a))$

$= E(\phi_0 \cup \phi)$

$E(\phi \cup q_0 \cup \phi \, q_3)$
$E(q_0) \cup E(q_3)$
$(q_1, q_2, q_3) \cup (q_3)$ $\{q_0, q_1, q_2, q_3) \cup \{q_3, q_4\}$
$= (q_1, q_2 q_3)$ $= \{q_0, q_1, q_2, q_3, q_4\}$ new st.

δ' (Now using

Date
Page

$\delta'(\{q_0, q_1, q_2, q_3\}, b) = E(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b), \delta(q_3, b)$

$= E(q_2 \cup \phi \cup q_4 \cup \phi)$
$= E(q_2) \cup E(q_4)$

$= (q_2) \cup (q_3, q_4)$
$= (q_2, q_3, q_4)$ new state.

Now, using new state,

$\delta'(\{q_0, q_1, q_2, q_3, q_4\}, a) = E(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta($
$\cup \delta(q_3, a) \cup \delta(q_4, a))$

$E(\phi \cup q_4 \cup \phi\ q_4 \cup \phi)$
$E(q_4) \cup E(q_4)$
$= (q_3, q_4)$ new state

$\delta'(\{q_0, q_1, q_2, q_3, q_4\}, b) = E(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b)$
$\cup \delta(q_3, b) \cup \delta(q_4, b))$

$= E(q_2 \cup \phi\ q_4 \cup \phi)$
$= E(q_2) \cup E(q_4)$
$= (q_2, q_3, q_4)$ new state.

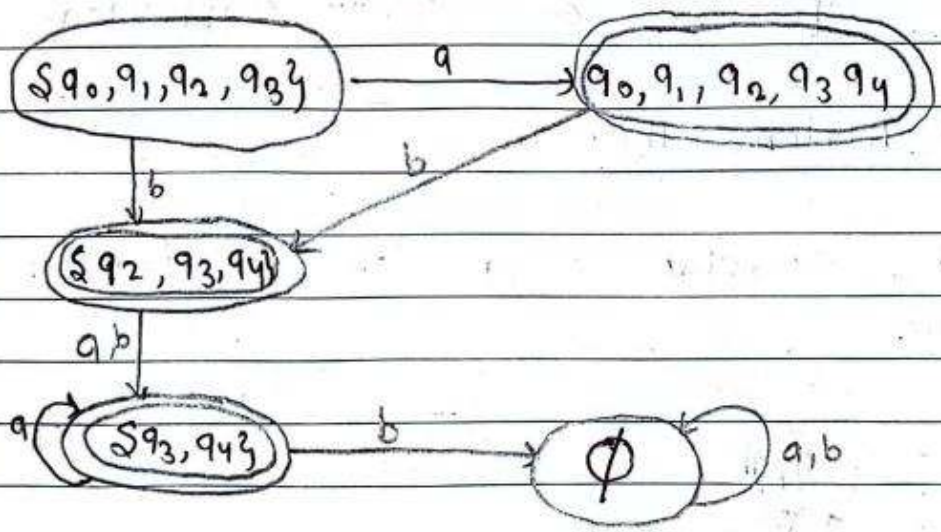
$\delta'((q_2, q_3\ q_4), a) = E(\delta(q_2, a), \delta(q_3, a), \delta(q_4, a))$
$= E(\phi \cup q_4 \cup \phi)$
$= E(q_4)$
$= (q_3, q_4)$ new state.

Date
Page

$\delta'(q_2, q_3, q_4, b) = E(\delta(q_2, b), \delta(q_3, b), \delta(q_4, b))$
$= E(q_4 \phi \cup \phi)$
$= E(q_4)$
$= (q_3, q_4)$ new state.

$\delta'(\{q_3, q_4\}, a) = E(\delta(q_3, a), \delta(q_4, a))$
$= E(q_4 \cup \phi)$
$= (q_3, q_4)$ old state.

$\delta'((q_3, q_4), b) = E[\delta(q_3, b), \delta(q_4, b)]$
$= E(\phi \cup \phi)$
$= E(\phi)$
$= \phi$



Arden's Theorem

Let P and Q be two regular expression over alphabet Σ. If P doesnot contain null string E, then

$$R = Q + RP$$

has unique solution that is $R = QP^*$

This can be understood as $R = Q + RP$

Lets put the value of R in RHS.

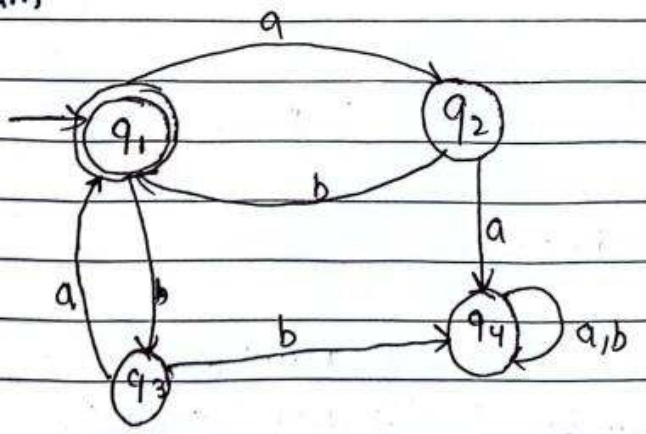
$$R = Q + (Q + RP)P = Q + QP + RP^2$$

When we put the value of R again, we get the following equation.

$$R = Q + QP + QP^2 + QP^3 \cdots$$
$$= Q(1 + P + P^2 + P^3 + \cdots)$$
$$= Q(E + P + P^2 + P^3 + \cdots)$$
$$= QP^*$$ (By the defination of closure operation of regular expression.

Use of Arden's theorem :-

#1) Find the regular expression for transition diagram



Now, let's find the equations:

$$q_1 = E + q_2 b + q_3 a$$
$$q_2 = q_1 a$$
$$q_3 = q_1 b$$
$$q_4 = q_2 b + q_4 a + q_4 b + q_2 a$$

Date
Page

Put the value of q_2 and q_3 in q_1 as,

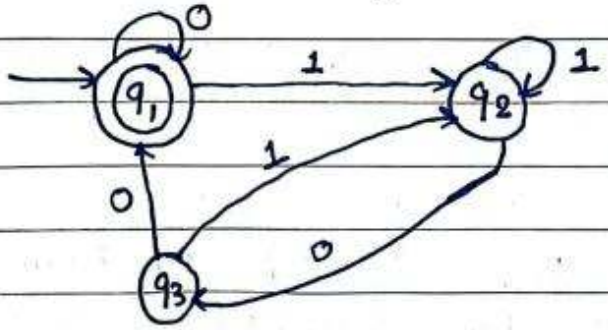
$$q_1 = E + q_2ab + q_1ba$$
$$= E + q_1(ab + ba)$$

Comparing with Arden's theorem, $R = Q + RP$
$\Rightarrow R = QP^*$

$$q_1 = E(ab + ba)^*$$

So, the required regular expression is $(ab + ba)$

Construct a regular expression corresponding to the state diagram.



Now, let's find the equations:

$$q_1 = E + q_10 + q_30$$
$$q_2 = q_21 + q_11 + q_31$$
$$q_3 = q_20$$

$$q_2 = q_11 + q_21 + (q_20)1 = q_11 + q_2(1 + 01)$$
$$q_2 = q_11(1 + 01)^*$$

q_1:-

So, $q_1 = q_10 + q_30 + E = q_10 + q_200 + \epsilon = q_10 + (q_11(1 + 01^*)00 + E$

Date
Page

$q_1 = q_1(0+1(1+01)^*00) + \epsilon$

$q_1 = \epsilon(0+1(1+01)^*00)^*$

So regular expression is $(0+1(1+01)^*00)^*$

Regular language

Regular languages can be defined from the empty set and from some finite number of singulaton sets, by the operations of union, composition (concatenation) and kleen closure. Specially, consider any alphabet Σ, then a regular sets over Σ is defined in the following way.

~~The empty set ϕ, the set $\{\Sigma\}$ is defined to the following~~

(a) The empty set ϕ, the set $\{\Sigma\}$ containing empty string and the set $\{a\}$ for each symbol a in Σ are regular sets.

(b) If L_1 and L_2 are regular sets, then so are the union $L_1 \cup L_2$, the composition $L_1 L_2$ and the kleen closure L_1^*

(c) No other is regular

Date
Page

Theorem:

A set is a regular set if and only if it is accepted by a finite state automation.

→ Regular sets of the form ϕ, $\{\varepsilon\}$, $\{a\}$, $L_\alpha \cup L_\beta$, $L_\alpha L_\beta$ and L_α^* are quite often denoted by ϕ, ε, a, $(\alpha)+(\beta)$, $(\alpha)(\beta)$ and $(\alpha)^*$ respectively where α and β are assumed to be the expressions that denote L_α and L_β in a similar manner respectively and a is assume to be a symbol from the alphabet. Expressions that denote regular set are regular expression

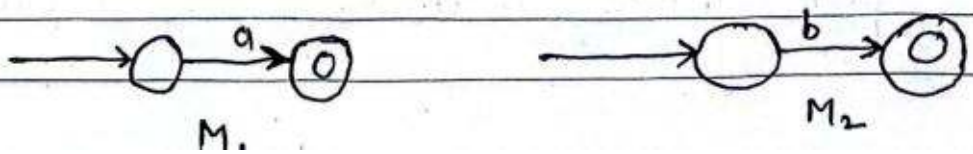
It is evident that the empty set and all singulaton are indeed accepted by the finite automata. The finite automata languages are closed under union, concatenation, and kleen star. hence, every regular set (language) is accepted by some finite automation.

construction of NFA from regular Expression

Given R.E ⟶ $(abUba)^*$

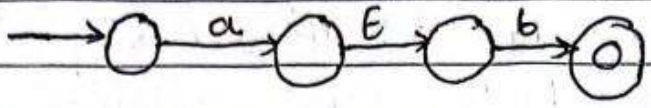
We can construct required NFA in following steps.

Step 1: a:b

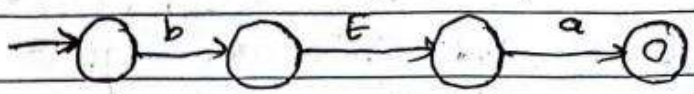


Date
Page

step2: ab : ba

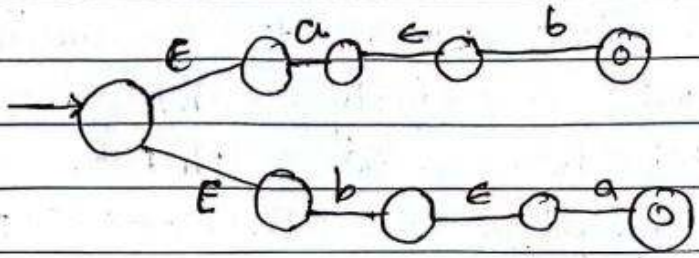


$M_1 = ab$



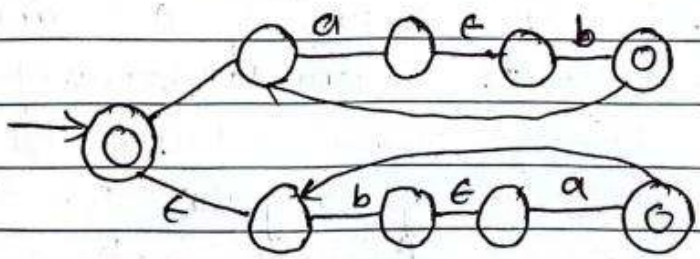
$M_2 = ba$

step!3



$M = ab \cup ba$

step!4



$M = (ab \cup ba)^*$

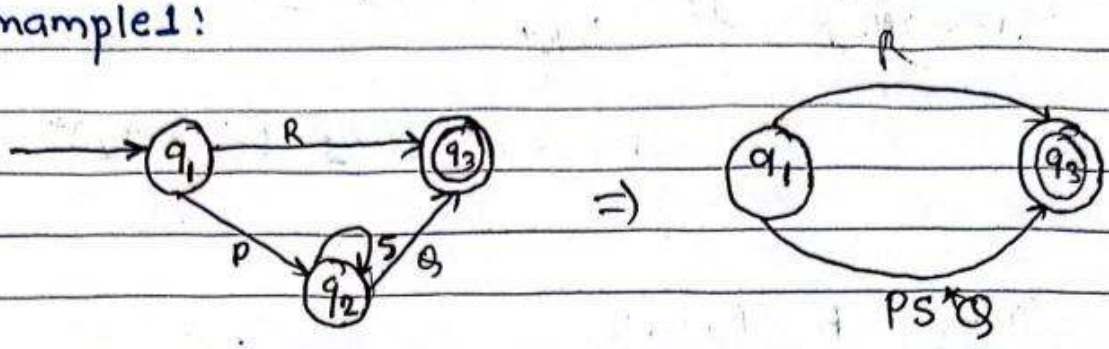
Euquivalence of finite automata and Regular Enpression

We can convert each of the regular enpression into corresponding finite automata and can also find out the regular enpressions from given finite automata as follow.

Date
Page

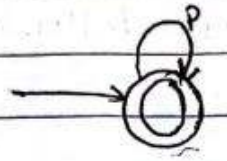
Example 1:

1



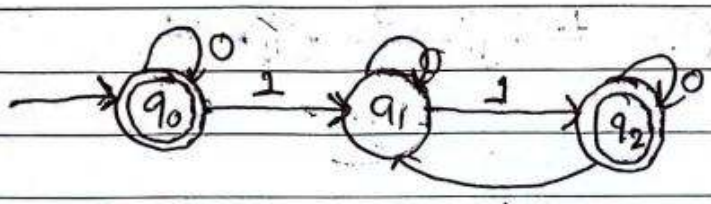
$R \cdot \varepsilon = R \cup PS^*Q$

2



→ RE = P^*

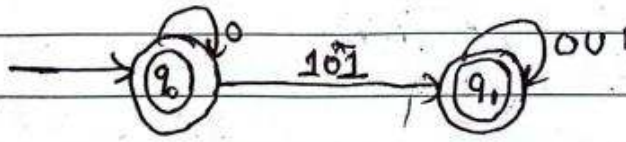
3



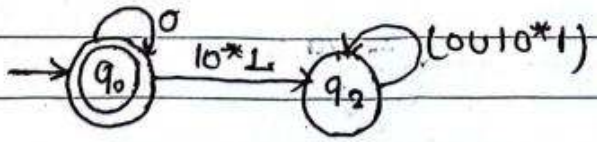
Construction of NFA from regular expression.

Soln

here, q_1 is neither initial state nor final state, so we can eliminate q_1 as follows:

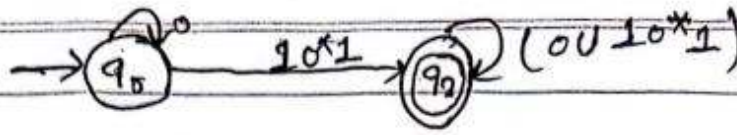


Now, turn off state q_2, we get



here, q_2 acts as dead state, so Regular expression $R_1 = 0^*$

Again Turn Off q_0, we get,

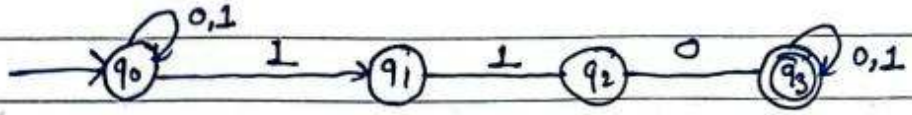


$R_2 = 0^* 10^*1 (0 \cup 10^*1)^*$

Then, $R = R_1 \cup R_2$

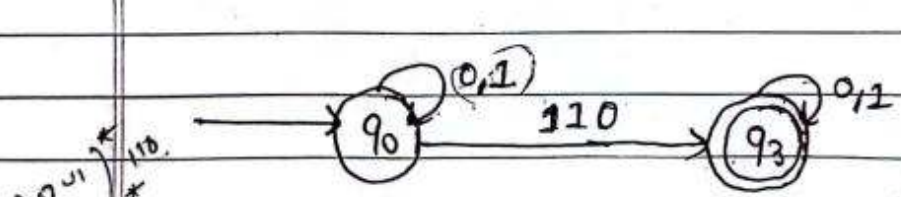
$= 0^* \cup 0^* 10^*1 (0 \cup 10^*1)^*$

Q) Find the regular expression from following NFA

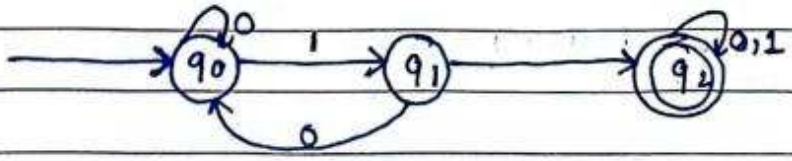
① 

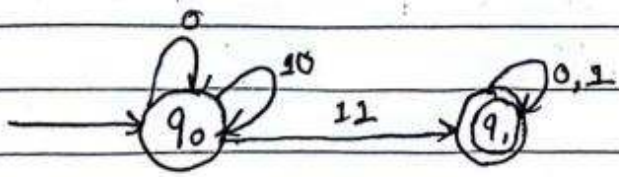
→ Soln

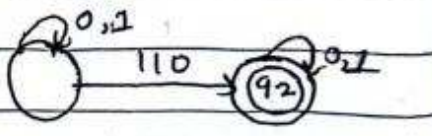
here, q_1 and q_2 neither initial state nor final state, so we can eliminate q_1 & q_2 as follows.

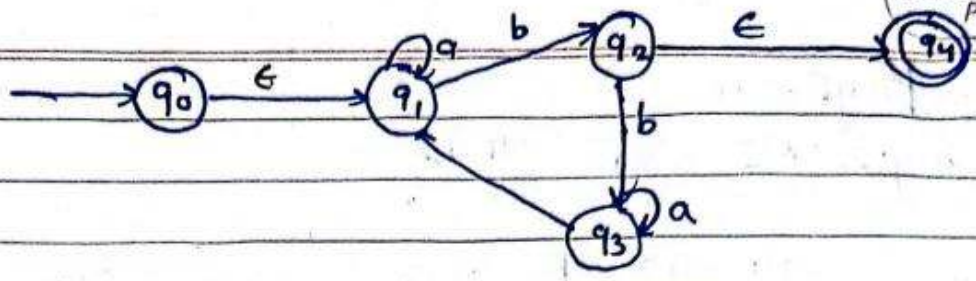


~~Now, Turn off q_3 state, we get,~~

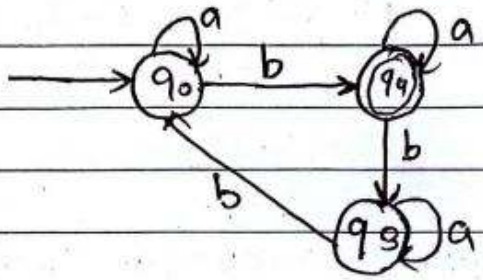
② 

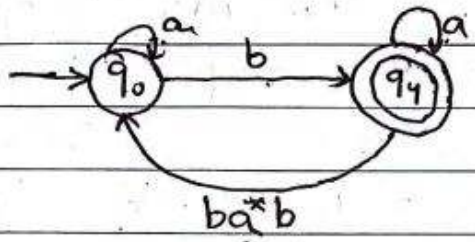






Soln Removing ε transition





a*

$\Rightarrow a^* b\, a^* \cup (ba^* ba^* b)^*$

Pumping lemma for regular set (language)

statement

Let L be a regular language. And w be any string $w \in L$, in be a pumping constant with $n \geq 1$ & $|w| \geq n$.

Then we can break w into three substrings $w = xyz$ such that

i) $y \geq 1$ or $y \neq \epsilon$

(ii) $|xy| \leq n$

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

i.e, we always find a non-empty string y not too far from the beginning of w that can be pumped. It means seperating y in any number of times keeps the ~~resulting~~ resulting string in the language.

Proof:

Suppose L is a regular language Then $L = L(m)$ for some DFA, m. Suppose M has n states. Now, consider any string w of length n or more say $w = a_1 a_2 a_3 \cdots a_n$, where $m \geq n$ and each a_i is an i/p ~~sym~~ symbol, for $i = 0, 1, 2 \cdots m$.

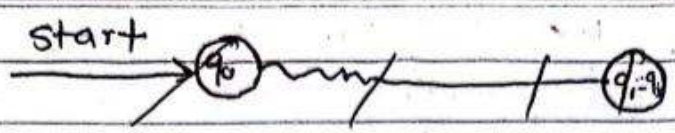
Let $\delta(q_0, q_1, q_2, \cdots q_i) = q_r$, But it is not possible for each of the $n+1$ states $q_0, q_1, q_2 \cdots q_n$ to be ~~direct~~ distinct, since ~~three~~ there only n difference states. Thus we can find ~~the~~ two different integers i and j each $0 \leq i \leq j \leq n$ such that $q_r = q_j$ Now we can break $w = xyz$ as follows:

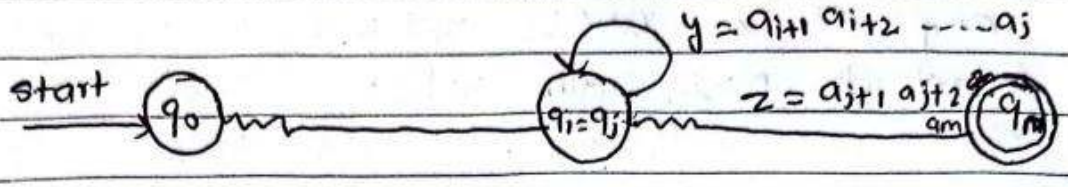
$x = q_1 q_2 \cdots q_i$

$y = a_{i+1} q_{i+2} \cdots q_j$

$z = a_{j+1} a_{j+2} \cdots a_m$

Date
Page





Note :- that x may be empty in case that $i=0$ & also z be empty if $j = n$ I m but y can be empty since $i < j$.

Now, when the automation M receives xy^iz for any $i \geq 0$, if $i=0$, then the automation, M with, i/p x on start state q_0 reaches state $q_i (= q_0)$. Then for i/p z at start $q_i = q_j$, M goes from q_i to the accepting state q_n Thus, it accepts xz.

If $i > 0$, then m goes from q_0 to q_1 on i/p x, circles from q_1 to $q_i (= q_j)$ i times on i/p y, and then goes to the accepting state on i/p z.

Thus for any $i \geq 0$, xy^iz is also accepted by M that is xy^iz is in L.

Date
Page

Pumping lemma for Regular language

Proof! let L be a regular language w be any string $w \in L$ and n be a pumping constant $n \geq 1$ with $|w| \geq n$

Then w can be re-written as $w = xyz$ such that

$|xy| \leq n$

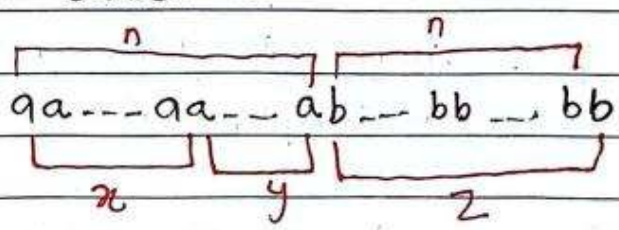
$|y| \geq n$

$xy^i z \in L$ for all $i \geq 0$

From pumping lemma, we can write

$w = xyz = a^n b^n$

which can be written as



here, $y = a^k$, $k \geq 1$

using pumping lemma

for $i = 0$, we get

$xz = a^{n+k} b^n \notin L$

which contradicts the theorem, hence the given language

$L = \{a^n b^n \mid n \geq 1\}$ is not regular.

proved #

Date
Page

Example 2: Prove that the language $L = \{a^m b^{2m} : i \geq 1\}$ is not regular language

Proof: Let L be a regular language w be any $w \in L$ and n be a pumping constant $m \geq 1$ with $|w| \geq m$

Then w can be re-written as $w = xyz$ such that

$|xy| \leq m$

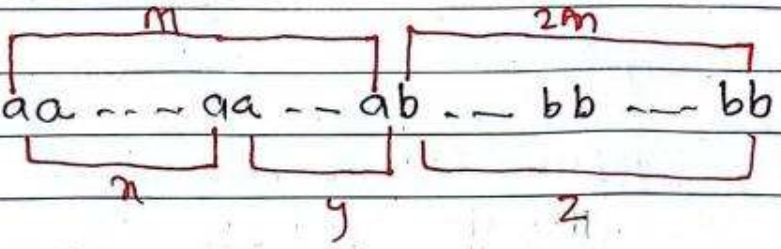
$|y| \geq 1$

$xy^i z \in L$ for all $i \geq 0$

for pumping lemma, we can write

$w = xyz = a^m b^{2m}$

which can be written as



here,

$y = a^k$, $k \geq 1$

using pumping lemma

for $i = 0$, we get

$xz = a^{m+k} b^{2m} \notin L$

Which contradicts the theorem, hence the given language,

$L = \{a^m b^{2m} \quad m \geq 1\}$ is not regular

Date
Page

#3- show that $L = \{ a^{m!} : m \geq 1 \}$ is not regular by using pumping lemma. for regular language.

Proof:

Let L be a regular language w be any string $w \in L$ and n be a pumping constant

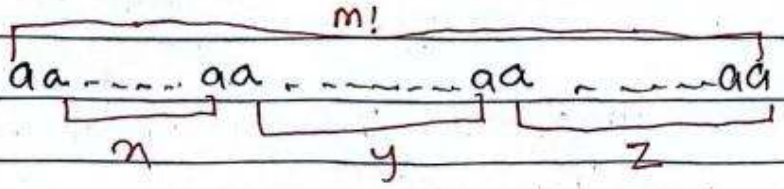
$w = xyz = a^{m!}$

If $y = a^k$, $k \geq 1$

using pumping lemma, $i = 2$

$$xy^2z = xyyz = a^{m!+k}$$

If $m! + k = p!$ (i.e perfect factorial)



here

If $y = a^k$, $k \geq 1$

Using pumping lemma

for $i = 2$

$xy^2z = xyyz = a^{m!+k}$

If $m! + k = p!$ (i.e pefect factorial).

$m! + k \leq m! + m$

$< m! + m!$ for $m > 1$

$< m!m + m!$

$= m!(m+1)$

$= (m+1)!$

$m! + k \neq (m+1)!$ $(m+1)! \neq p!$ hence,

$a^{m!+k} \in L$

Closure properties of languages that are occupied by finite automata.

Chapter – 3

Date
Page

Content. free Grammar (CFG)

Introduction :-

Let us take regular expression $a(a \cup b)^*b$. This expression gives strings consisting of a leading a followed by mid part $(a \cup b)^*$ followed by ~~trailing b~~ tailing b.

If S is a new symbol then we can write

$S \rightarrow aMb$
$M \rightarrow aM$
$M \rightarrow bM$
$M \rightarrow \epsilon$

A content free grammer is a language generator that operates on the rules like above.

Now, let us take generation of aab for this, we have to replace M by aM, whatever be the particular or ~~so~~ surroundings. so, it is called content free.

This can be obtained as,

$S \Rightarrow aMb \quad [\because S \rightarrow aMb]$
$\Rightarrow aaMb \quad [\because S \rightarrow aM]$
$\Rightarrow aab \quad [\because S \rightarrow \epsilon]$

Date
Page

Formal Definition:-

A content free grammar is a finite set of variables (called non-terminals or syntatic categories) each of which represent or language. The languages represented by variables are described recursively in terms of each other and primitive symbols called terminals. The rules relating the variables are called productions.
Mathematically,

CFG is defined as ~~quanti~~ quadrupte

$$G = (V_n, V_t, P, S)$$

where, V_n = finite set of non-terminals, generally represented by capital letters A, B, C, D

V_t = finite set of terminals, generally represented by small letters a, b, c, ---

P = set of rules or productions of CFG

S = starting non-terminals, called start symbol of the grammar $S \in V_n$

G is content free and all productions in p have the form $\alpha \rightarrow \beta$

where, $\alpha \in V_n$ and $\beta \in (V_t \cup V_n)^*$,

Some Terminologies

Derivation:

let $G = (V_n, V_t, P, S)$ be a content free grammer (CFG) If $w_1, w_2, w_3 \dots w_n$ are strings over $(V_n \cup V_t)^*$ such that $w_1 \Rightarrow w_2 \Rightarrow w_3 \dots \Rightarrow w_n$ then we can say w_n is derivable from obtain w_n from w_1 is called derivation.

language of content free grammer L(G):-

If $G = (V_n, V_t, P, S)$ is a CFG, then the language of G denoted by $L(G)$ is the set of terminal strings that have derivations from the start symbol.

$$\text{i.e } L(G) = \{ w \in V_t^* : S \overset{*}{\underset{G}{\Rightarrow}} w \}$$

Sentential Form:

Derivations from the start symbol produce strings that have a special role- This is called sentential form. That is, if $G = (V_n, V_t, P, S)$ is a CFG, then any string α in $(V_n \cup V_t)^*$ such that $S^* \Rightarrow \alpha$ is a sentential form.

Example:-

1. Consider a grammer $G = (V_n, V_t, P, S)$ where,

Date
Page

$V_n = \{S\}$, $V_t = \{a, b\}$ and set of production P is given by

$$P = \{ S \to aSb \\ S \to ab \}$$

here S is the only non-terminal which is the starting symbol of the grammar. 'a' and 'b' are the terminals and there are two productions $S \to aSb$ and $S \to ab$.

Now, we will ~~show~~ show that how the string a^2b^2 can be derived we have,

$$S \Rightarrow aSb$$
$$\Rightarrow aabb \quad [\because S \to ab]$$
$$\Rightarrow a^2b^2$$

here, we need to apply first production and then the second one. By applying first production $(n-1)$ times, followed by second production, we get,

$$S \Rightarrow aSb$$
$$\Rightarrow aaSbb$$
$$\Rightarrow aaaSbbb$$
$$\vdots$$
$$a^{n-1}Sb^{n-1}$$
$$\Rightarrow a^{n-1}abb^{n-1}$$
$$\Rightarrow a^nb^n$$

Thus the language for the above grammer is

$$L(G) = \{a^nb^n : n \geq 1\}$$

#(2) Let $G = (V_n, V_t, P, S)$ where, $V_n = \{S\}$, $V_t = \{a, b\}$,
$P = \{S \to aaS, S \to b\}$

Now, find $L(G)$

So,

$S \Rightarrow aaS$
$\Rightarrow aaaaS$

aa aa aaS

$(aa)^n S$
$(aa)^n b$

hence the language of given grammer G is

$L(G) = \{(aa)^n b : n \geq 0\}$

#3) Let $G = (V_n, V_t, P, S)$, $V_n = \{S, A\}$, $V_t = \{a, b\}$

$P = \{ \cdot\ S \to aA,$
$\cdot\ A \to bS,$
$S \to E \}$

$S \to aA$
$\Rightarrow abS$
$\Rightarrow abaA$
$\Rightarrow ababS$
$\vdots$
$\Rightarrow (ab)^n S$
$\Rightarrow (ab)^n$ //

hence, the language of the given grammer is

$L(G) = \{(ab)^n : n \geq 0\}$ #

{a, aa, aa,

Date
Page

II Writing content free Grammar enamples :-

1. Write a content free Grammar (CFG) for the language given by $L = \{a^n b^n : n \geq 0\}$

Sol^n

let $G = (V_n, V_t, P, S)$ be required CFG where

$V_n = \{S\}$
$V_t = \{a, b\}$
$P = \{S \to asb, S \to \epsilon\}$

lets test for aaaabbbb

$S \Rightarrow asb$
$\Rightarrow aasbb$
$\Rightarrow aaasbbb$
$\Rightarrow aaaasbbbb$ [$\because S \to \epsilon$]
$\Rightarrow a^4 b^4$ produced #

2. Write CFG for the language $L = \{a^m b^n : m \geq n\}$

Sol^n

let $G = (V_n, V_t, P, S)$ be the required CFG when,

$V_n = \{S\}$, $V_t = \{a, b\}$
$P = \{ S \to aS,$
$S \to asb,$
$S \to \epsilon \}$

3. For equal number of 'a' and 'b'.

Date
Page

$P = \{ S \to aSb,$
$S \to bSa,$
$S \to SS,$
$S \to \epsilon \}$ #

4. Write a content free grammer (CFG) for
$w = w^R$

$S \to aSa$
$S \to bSb,$
$S \to \epsilon,$
$S \to a,$
$S \to b \}$

5. $w = ww^R$

$P = \{ S \to aSa,$
$S \to bSb,$
$S \to \epsilon \}$

6. w has even length

$P = \{ S \to aSa,$
$S \to bSb,$
$S \to aSb,$
$S \to bSa,$
$S \to \epsilon \}$

7. w has odd length

$P = \{ S \to aSa,$
$S \to bSb,$
aSb
bSa

Date
Page

Derivation Table

A derivative tree or purse tree is an ordered tree in which nodes are labelled with the left side of ~~phenomena~~ productio and in which the children of a node represent its corresponding right sides. It is tree representation of derivation.

Derivation:

Let $G = (V_n, V_t, P, S)$ be a cfG, Then a tree is a derivation tree ~~is a derivation tree~~ for G if,

(i) Every verten has a label, which is a symbol of $V_n \cup V_t \cup \{\epsilon\}$

(ii) The label of root is i.e starting non-terminal,

(iii) If a verten is interior and has label A, then A must be in V_n.

(iv) If n has label A and vertices $n_1, n_2, n_3 \dots n_k$ are the children of verten n, involve from the left, with labels $x_1, x_2 \dots x_k$ must be a production in P.

(v) If verten n has ϵ, then n is a leaf and is only son of its father (i.e only child of the parent)

Date
Page

Yield of parse tree

If we look at the leaves of any parse tree and concatenate them from the left we get a string, which is called "the yeild of the tree".

Eg:- Given $G = (V_n, V_t, P, S) = (\{S, A\}, \{a, b\}, P, S)$ where P is defined as

$$S \rightarrow aAS \mid a \mid SS,$$
$$A \rightarrow SbA \mid ba$$

The derive the string as aabaa.

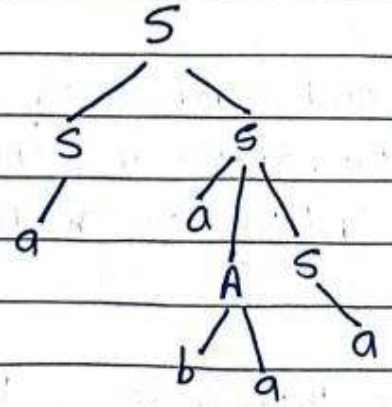
here, $S \rightarrow SS$

$\Rightarrow as$ [$\because S \rightarrow a$]

$\Rightarrow aAS$ [$\because S \rightarrow aAS$]

$\Rightarrow aabas$ [$\because A \rightarrow ba$]

$\Rightarrow aabaa$ [$\because S \rightarrow a$]



$\therefore$ yeild is aabaa #

Date
Page

A2. Given $G = (V_n, V_t, P, S)$ where P is defined as

$S \rightarrow aAS \mid a$

$A \rightarrow SbA \mid SS \mid ba$

Show that $S \overset{*}{\Rightarrow} aabbaa$ & construct a derivation tree where yield is aabbaa

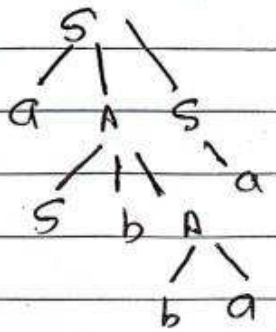
here, $S \rightarrow aAS$

~~$\Rightarrow aAa$~~ ~~[S → a]~~

$\Rightarrow aSbA$ [A → SbA]

$\Rightarrow aabSS$ [A → SS]

$aabaa$ [S → a]



$S \rightarrow aAS$

$\Rightarrow aSbAS$ [A → SbA]

$\Rightarrow aabAS$ [S → a]

$\Rightarrow aabbas$ [A → ba]

$\Rightarrow aabbaa$ [S → a]

yield is aabaa //

∴

Ambiguty of Grammar and language.

A content free grammar is called ambiguous if for at least one word in the language that it generates, there are two possible derivations of the word that can correspond to different syntax trees. If a CFG is not ambigious, it is unambigious.

Consider a CFG with production as:

Date
Page

$S \to asb \mid ss \mid \epsilon$. prove that the given CF ambigious for aabb .

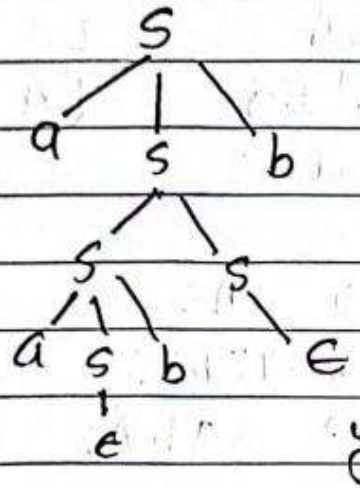
Soln

$S \Rightarrow asb$

$\Rightarrow$ ~~aasb~~ $assb$ [$\because S \to ss$]

$\Rightarrow$ ~~aas~~ $aasbsb$ [$\because S \to asb$]

$\Rightarrow aabb$ [$\because S \to \epsilon$]



yield = aabb //

Date
Page

Simplification of CFG

We get simplified grammar by applying -

(1) We must eliminate usless /non- generating/ unreachable symbols from given grammar.

(2) We must eliminate empty - productions.

(3) We must eliminate unit productions.

(1) Elimination of useless symbols :-

→ We ~~must eliminate usele~~ say A is generating if $A \overset{*}{\Rightarrow} w$ for some terminal string w.

→ We say A is reachable if there is a derivation $S \overset{*}{\Rightarrow} \alpha A \beta$ for some α and β

Eg :- 1: Identify and eliminate useless symbols from following grammar.

$S \rightarrow aB \mid bX$

$A \rightarrow BAd \mid bsX \mid a$

$B \rightarrow asB \mid bBX$

$X \rightarrow SBD \mid aBX \mid ad$

Here, A and X are generating due to $A \rightarrow a$ S $X \rightarrow ad$, S is also useful due to production

$S \rightarrow bX$

$A \rightarrow bsX \mid a$

$X \rightarrow ad$

Again, A is unreachable symbol, so, we must eliminate this symbol to get,

$S \rightarrow bX$

$X \rightarrow ad$

This is simplified grammar

Eg:-2

$S \rightarrow AB \mid CA$

$B \rightarrow Bc \mid AB$

$A \rightarrow a$

$C \rightarrow AB \mid b$

2. Elimination of empty production (i.e ϵ- production)

Eg-1 Eliminate ϵ- production from following grammar

$S \rightarrow AB$

$A \rightarrow AaA \mid \epsilon$

$B \rightarrow b$

Sol^n, here $A \rightarrow \epsilon$ is an empty production. We can eliminate this production as follows:

Date
Page

$S \rightarrow AB \mid B$

$A \rightarrow AaA \mid aA \mid Aa \mid a$

$B \rightarrow b$

3. Elimination of unit production

Eg.1:- Eliminate unit productions from following grammar.

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow C \mid d$

$C \rightarrow D$

$D \rightarrow E$

$E \rightarrow a$

here, $B \rightarrow C$, $C \rightarrow D$ and $D \rightarrow E$ are unit productio we can eliminate these as follows:-

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow a \mid d$

$C \rightarrow a$

$D \rightarrow a$

$E \rightarrow a$

Again, C, D and E are unreachable symbols so, we have to eliminate these symbols. finally, we get

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow a \mid d$

Eg.-2 Eliminate unit production from following grammar.

Date
Page

$S \rightarrow aA \mid B$

$A \rightarrow bA \mid c$

$B \rightarrow abA \mid bc$

here, $S \rightarrow B$ is unit production,

$S \rightarrow aA \mid abA \mid bc$

$A \rightarrow bA \mid c$

$B \rightarrow abA \mid bc.$

Date
Page

Chomsky Normal Form (CNF)

If a CFG has only production of the form

(i) $A \to BC$ where A, B, and C are each non-terminals
i.e non-terminal → exactly two non-terminals

(ii) $A \to a$ where 'A' is a non-terminals
i.e non-terminal → exactly one terminal

is said to be chomsky normal form or CNF.

conversion to CNF

consider the grammar $G = (\{S, A, B\}, \{a, b\}, P, S)$, where, S is the start symbol and the production P is given by

$S \to bA \mid aB$
$A \to bAA \mid as \mid a$
$B \to aBB \mid bS \mid b$

find the equivalent CNF of it.

Soln

Since a CNF has either two non-terminals or one terminal at the right side, we have to replace the terminal 'b' by a non-terminal 'Cb' and 'a' by 'Ca' in the first production. Then we get,

Date
Page

3. Design a pushdown automata (PDA) M that accepts a language $L = \{w \in (a,b)^*$; number of a is equal to number of b$\}$.

Date
Page

$S \rightarrow C_b A \mid C_a B$
$A \rightarrow C_b AA \mid C_a S \mid a$
$B \rightarrow C_a BB \mid C_b S \mid b$
$C_a \rightarrow a, \quad C_b \rightarrow b$

Now in the second & third rule $C_b AA$ and $C_a BB$ are not in the CNF, as they contain 3 non-terminals. Replace AA by D and BB by E,

we get,

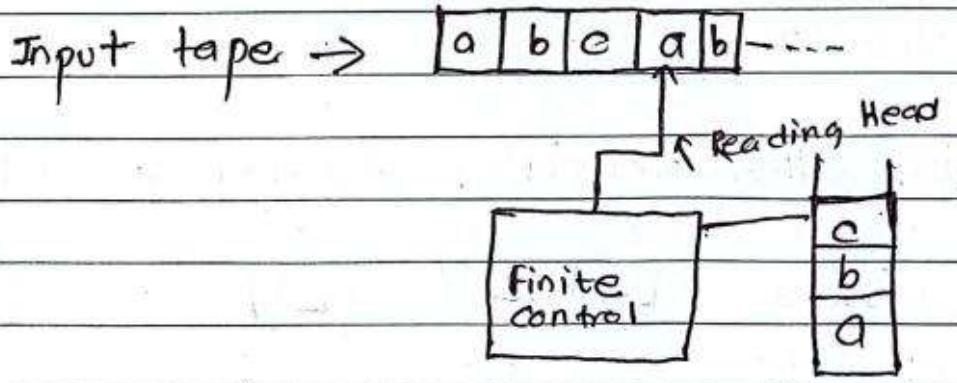
$S \rightarrow C_b A \mid C_b B$
$A \rightarrow C_b D \mid C_b S \mid a$
$B \rightarrow C_a E \mid C_b S \mid b$
$D \rightarrow AA,$
$E \Rightarrow BB$
$C_a \rightarrow a,$
$C_b \rightarrow b$ which is the equivalent CNF

Greibach Normal form (GNF)
GNF is of the form $A \rightarrow a\alpha$, where A is a non-terminal and 'a' is exactly one terminal and 'α' is the string of zero or more non-terminals.

Date
Page

Pushdown Automata (PDA)

Push down Automata (PDA) are essentially the finite Automata (FA) with auxiliary storage element called stack. Hence PDA has control of both an input tape and a stack to store what it has read.



Formal Definition

A pushdown automata is a system which is mathematically defined as follows.

Date
Page

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

Where,

Q : is a finite set of states

Σ : is an alphabet (finite set of input symbols)

Γ : is an alphabet (finite set of stack symbols or pushdown symbols.

δ : is a mapping or transition relation which maps.

$(Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^*) \rightarrow$ final subset of $Q \times \Gamma^*$

s : is the initial state $s \in Q$

F : is the set of final ~~set~~ states $F \subseteq Q$.

Explanation of ~~mass~~ moves of Pushdown Automata:

Given the transition function of PDA for

$(Q \times \Sigma^* \times \Gamma^*) \rightarrow (Q \times \Gamma^*)$

(a) The interpolation of $[(q, a, z), (p, y)] \in \delta$ (where $q, p \in Q$, a is an alphabet, z and y in Γ^*) is that PDA whenever in state of with z an the top of the State may read 'a' from the input tape, replace z by y an the top of the stack and enter stack p.

Date
Page

(b) To push a symbol on the ~~state~~ stack, just add it on the top of the stack. This can be achieved by the transistion

$[q, a, \epsilon], (P, a)$ or, $(q, a, \epsilon) \rightarrow (P, a)$

(c) Similarly to pop a ~~support~~ symbol is to remove it from the top of the stack. The transition $[(q, a, z), (P, \epsilon)]$ pop z from the stack.

d) PDA can also behave as do nothing machine. Just read the input from the input tape and does not make any change in the state and symbol at the stack.

Eg :- $(P, q, z), (P, z)$,

here, PDA just read input 'a' without making any change on the state and top of the state. waits for another input from input tape.

Pushdown Automata (PDA)

Examples #1 Design a Pushdown automata (PDA) M that accepts a language $L = \{wcw^R : w \in \{a, b\}^* \}$

Soln Let $M = (Q, \Sigma, \Gamma, \delta, s, F)$ be the required PDA

Where,

$Q = \{S, F\}$

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

$\Gamma = \{a, b\}$

$F = \{F\}$

Find the transition relation δ is defined by

i) $((S, a, \epsilon), (S, a))$; push a

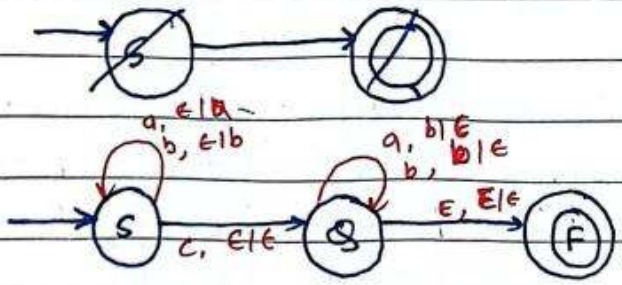
ii) $((S, b, \epsilon), (S, b))$; push b

iii) $((S, c, \epsilon), (F, \epsilon))$: Just switch the state

iv) $((F, a, a), (F, \epsilon))$; pop a

(v) $((F, b, b), (F, \epsilon))$; pop b

vi) $((Q, \epsilon, \epsilon), (F, \epsilon))$; state change.



let us check for string abcba

State	Unread symbol	Stack	Transition used
S	abcba	—	—
S	bcba	a	(i)
S	cba	ba	(ii)
Q	ba	ba	(iii)
Q	a	a	(v)
Q	ε	ε	(iv)
Q		ε	vi)

Date
Page

#2 Design a pushdown automata (PDA) M that accepts a language $L = \{a^n b^n : n > 0\}$

soln

Let $M = (Q, \Sigma, \tau, \delta, S, F)$ be the required PDA

where, $Q = \{S, Q, F\}$

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

$\tau = \{a, \epsilon\}$

$F = \{F\}$, $S = S$

And the transition relation δ is defined by

i) $((S, a, \epsilon), (S, a))$, push a ||

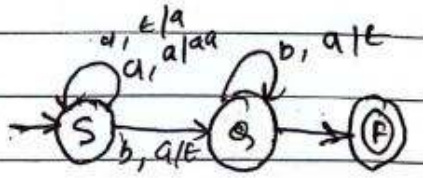
ii) $((S, a, a), (S, aa))$;

iii) $((S, b, a), (Q, \epsilon))$;

iv) $((Q, b, a), (Q, \epsilon))$;

v) $((Q, \epsilon, \epsilon), (F, \epsilon))$

vi) $((Q, \epsilon, \epsilon), (F, \epsilon))$



State	unread symbol	stack	Transition used
S	aaa bbb	-	-
S	aabbb	a	(i)
S	abbb	aa	(ii)
S	bbb	aaa	(ii)
Q	bb	aa	(iii)
Q	b	a	(iv)
Q	ϵ	ϵ	(iv)
F	ϵ	ϵ	(v)

Date
Page

Pushdown Automata and CFG

PDA can recognize only language for which there exists a CFG. That is the class of language accepted by PDA is exactly the class of content free language.

Construction of PDA equivalent to a CFG.

Let $G = (V_n, V_t, P, S)$ be a CFG, we must construct a pushdown automata P such that $L(P) = L(G)$.
The machine we construct has only two states, p and q and remains permanently in state q after its first move. Also, P uses V_n, the set of non-terminals and V_t the set of terminals, as its state alphabet. Now we assume

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

where $Q = \{p, q\}$, $\Sigma = V_t$

, $\Gamma = V_n \cup V_t$ (i.e set of non terminals

$S = p$

And the transition relation δ is defined as follows

i) $((p, \epsilon, \epsilon), (q, S))$ as S is ~~strong~~ ~~starting~~ starting non terminal ~~of~~ of CFG

ii) $((q, \epsilon, A), (q, z))$ for each rule $A \to z$ in CFG

iii) $((q, a, a), (q, \epsilon))$ for each $a \in V_t$.

Date
Page

The PDA P begins by pushing S, the starts symbol of grammar G, on its initially empty push down storage and entering state q, using transition 1. On each subsequent step, it either replaces the top most symbol A on the stack provided that it is a non terminal by the right hand side. i.e x of some rule A→x the grammar (transition 2. OR pops the top most element from the stack. provided that it is a terminal the next input symbol (transition 3)

Example 1.

Design a PDA for the grammar
$G = (V_n, V_t, P, S)$ where $V_n = \{S\}$
$V_t = \{a, b, c\}$ and P is defined as

$S \rightarrow asa$
$S \rightarrow bsb$
$S \rightarrow c$

=) Soln
let the PDA be $P = (Q, \Sigma, \Gamma, \delta, S, F)$
where,

$Q = \{P, q\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{s, a, b, c\}$
$S = \{P\}$
$F = \{q\}$

Date
Page

And the transition δ is defined as

i $((p, \epsilon E), (q, S))$
ii) $((q, \epsilon, S), (q, aSa))$
iii) $((q, \epsilon, S), (q, bSb))$
iv) $((q, \epsilon, S), (q, c))$
v) $((q, a, a), (q, \epsilon))$
vi) $(q, b, b), (q, \epsilon))$
vii) $((q, c, c), (q, \epsilon))$

Now, let us apply the transition for the string

$w = abbcbba$

S.N.	State	Unread i/p	Stack	Transition us
1	p	abbcbba	ϵ	–
2	q	abbcbba	S	(i)
3	q	abbcbba	aSa	(ii)
4	q	bbcbba	Sa	(v)
5.	q	bbcbba	bSba	(iii)
6.	q	bbcbba	Sba	(v)
7.	q	bcbba	bSbba	(iii)
8	q	cbba	Sbba	(vi)
9.	q	cbba	cbba	(iv)
10.	q	bba	bba	(vii)
11.	q	ba	ba	(vi)
12.	q	a	a	(v)
13.	q	ϵ	ϵ	–

Hence accepted

Date
Page

#2) Design a PDA for the CFG

$G = (V_n, V_t, P, S)$

$V_n = \{S\}$

$V_t = \{(,)\}$

and P is defined as

$S \to SS$

$S \to (S)$

$S \to \epsilon$

consider the string $w = ()()$

Soln : Let the PDA be $P = (Q, \Sigma, \Gamma, \delta, S, F)$

where, $Q = \{P, q\}$

$\Sigma = \{(,)\}$

$\Gamma = \{S, \epsilon, (,)\}$

$S = \{P\}$

$F = \{q\}$

And the transition δ is defined as:

i) $((P, \epsilon, \epsilon), (q, S))$

ii) $((q, \epsilon, S), (q, SS))$

iii) $((q, \epsilon, S), (q, (S))$

iv) $((q, (, (), (q, \epsilon))$

v) $((q,),)), (q, \epsilon))$

vi)

i) $((P, \epsilon, \epsilon), (q, S))$

ii) $((q, \epsilon, S), (q, SS))$

iii) $((q, \epsilon, S), (q, (S)))$

iv) $((q, \epsilon, (), (q, \epsilon))$

v) $((q, (, (), (q, \epsilon))$

vi) $((q,),)), (q, \epsilon))$

Date
Page

S.N	state	Unread	Stack	Transition u
1.	p	( ) ( )	$	–
2.	q	( ) ( )	S$	(i)
3.	q	( ) ( )	SS	(ii)
4.	q	( ) ( )	(S) S	(iii)
5.	q	) ( )	S) S	(iv)
6.	q	) ( )	) S	(v)
7.	q	( )	S	(iii)
8.	q	)	(S)	(iii)
9.	q	)	S)	(iv)
10.	q	)	)	iv
11	q	∈	∈	vi

hence Accepte

Properties of content free languages

1) Closure Properties of content free language.

– CFL are closed under union, concatination and kleene star.

Theorem: The family of content free languages is closed under union, concatenation and kleene star.

Proof: Let L_1 and L_2 be the conten-free languages generated by the following content free grammars respectfvely.

$G_1 = (V_n, V_t, P, S)$ and

$G_2 = (V_{n2}, V_{t2}, P_2, S_2)$

Date
Page

- Union :-

let S be a new symbol and also assume $G = (V_n, V_t, P, S)$ be the new grammer obtained from the union of G_1 and G_2 where,

$V_n = V_{n_1} \cup V_{n_2} \cup \{S\}$

$V_t = V_{t_1} \cup V_{t_2}$

S is the start symbol and productions is defined as follows

$P = \{P_1 \cup P_2 \cup \{S \to S_1 / S_2\}\}$

here, only rules involving S are $S \to S_1$ and $S \to S_2$. So, G_1 and G_2 have disjoint non terminals.

So, $L(G) = L(G_1) \cup L(G_2)$ is equivalent to saying $\omega \in L(G_1) \cup L(G_2)$

hence, content free languages are closed under union. proved. #

- Concatination :

Let $G = (V_n, V_t, P, S)$ be a new grammar obtained from concatenation (i.e $L(G) = L(G_1) \cdot L(G_2)$) of G_1 and G_2 where,

$V_n = V_{n_1} \cup V_{n_2} \cup \{S\}$

$V_t = V_{t_1} \cup V_{t_2}$

$P = P_1 \cup P_2 \cup \{S \to S_1 S_2\}$

Date
Page

here, $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} w_1$ & $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w_2$ then we can claim that $S \overset{*}{\underset{G}{\Rightarrow}} w_1 w_2$ due to the production $S \to S_1 S_2$.

- **Kleene star:-**

Let $G = (V_n, V_t, P, S)$ be a new grammar, obtained from kleene star. (i.e $L(G) = L(G_1)^*$) or, G_1 where,

$$V_n = V_{n_1} \cup \{S\}$$
$$V_t = V_{t_1}$$
$$P = P_1 \cup \{S \to S_1 S_2, S \to \epsilon\}$$

here,

If $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} w$ then S gives w_1^* (i.e strings like ϵ, w_1, w_2 etc) due to productions $S \to S_1 S$ and $S \to \epsilon$. So, we can claim in that $L(G_1) = L(G_1)^*$

hence, content free languages are closed under kleene star proved.

2. **Negative Properties of content-free languages**

- Content-free languages are not closed under intersection and complementation.

Theorem: The family of content free languages are not closed under intersection.

Let $L = \{a^n b^n c^n : n \geq 1\}$ which is not content free languages. However, following two languages are content free.

$L_1 = \{a^n b^n c^m : n \geq 1, m \geq 1\}$

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

A grammar for L_1 is

$S \rightarrow Ac$

$A \rightarrow aAb \mid \epsilon$

$C \rightarrow cC \mid \epsilon$

Similarly, grammar for L_2 is

$S \rightarrow AB$

$A \rightarrow aA \mid \epsilon$

$B \rightarrow bBc \mid \epsilon$

however, the languages $L = L_1 \cap L_2 = \{a^n b^n c^n : n \geq 1\}$ is not CFL. Hence, content free languages are not closed under intersection.

Proved ##

Theorem :- content free languages are not closed under complementation.

Proof :-

Assume that content free languages are closed under complementation. Let L_1 and L_2 be two content free languages.

According to our assumption, $\overline{L_1}$ and $\overline{L_2}$ are also content free languages.

We know that content free languages are

also content free languages.

we know that content free languages are closed under union. so, $\overline{L_1} \cup \overline{L_2}$ is also content free.

Again, from assumption, we can say that $\overline{L_1 \cup L_2}$ is also content free language.

From de-margan's law:-

$\overline{L_1} \cup \overline{L_2} = L_1 \cap L_2$ is content free.
which contradicts with theorem or assumption that CFL are not closed under intersection.

hence,
statement (content f.l. are not closed under complementation). proved #

Decision algorithm for CFLs :-

Decide whether a given CFL is empty, finite or infinite or whether a word is CFL

Emptyness :- Prove that for a given cfg, there is an algorithm to determine whether or not it can be generated any word or if the production of the form $S \to a$ or if it is empty or not.

Date
Page

Proof:- we have already known about nullable non-terminals and to decide the emptiness problem, we decide whether the start symbol 'S' is nullable.

i.e $S \overset{*}{\Rightarrow} \epsilon$

Now, if the production is of the form $S \to a$, where 'a' is a terminal then 'a' is a word generated by CFG. Then, if there is no such production then we follow the following algorithm.

Step 1: For each non terminal N, that has some productions of the form $V_n \to a$ where 'a' is a terminal a string of terminals, we choose one of them and put 'a' in the other production having V_n at right side, thus eliminating non-terminals V_n all thog together.

Finiteness:-

There is an algorithm to decide whether a given CFG generates a finte or infinite language.

Proof:-

let us assume that a grammar contains no ϵ-productions, no unit production or no useless symbols.

Assume a grammer with production of the form

$X \overset{*}{\Rightarrow} xXy$

since the grammar is assumed to have no ϵ production or no unit production or no useless symbols x and y can't be empty simultaneously.

since X is neither nullable or no use-less symbol

so,

$S \overset{*}{\Rightarrow} uXu \overset{*}{\Rightarrow} wsX \overset{*}{\Rightarrow} z$

where,

u, v, z are in V_t^*. But then,

$S \overset{*}{\Rightarrow} uXu \overset{*}{\Rightarrow} ux^nXy^nv \Rightarrow ux^nzy^nu$ is possible for all n, so that $L(G)$ is infinite. But if no variable can ever repeat, then the lengths of any derivation is bounded by $|V_n|$. In that case, $L(G)$ is finite.

Step 2 :- Repeat Step 1 until it eliminates S or if eliminates no new non-terminals. If S has been eliminated, then CFG produces some words, if not then not.

Example :-

Consider a CFG

S

Date
Page

$S \to XY$
$X \to AX$
$X \to AA$
$A \to a$
$Y \to BY$
$Y \to BB$
$B \to Bb$

soln

Replace all A's by a and all B's by b.

Then, $S \to XY$
$X \to aX$
$X \to aa$
$Y \to bY$
$Y \to bb$

Now, Replace X's by aa and Y's by bb.

Then, $S \to aabb$

Replaces S by aabb

Step2: Terminate step1 and discover that S has been eliminated. So, CFG, produces at least one word.

Date
Page

The pumping lemma for CFL

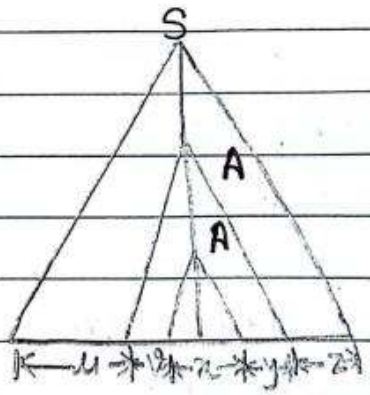
Statement:

Let G be any CFG, there is a constant 'n', depending only on G, such that if w is in L(G) and $|w| \geq n$, then we may write $w = uvxyz$ such that:

i) $|vy| \geq 1$ that is either v or y is non empty
ii) $|vxy| \leq n$ then for all $i \geq 0$
iii) $uv^i xy^i z$ in its L(G)

Proof:

let us take a following diagram



from above diagram, we can write

$S \rightarrow uAz$

$A \rightarrow vAy$

$A \rightarrow x$

By using these productions,

$S \Rightarrow uAz$

$\Rightarrow uvAyz$

$\Rightarrow uvvAyyz$

$\Rightarrow uv^2Ay^2z$

Date
Page

Show that $L = \{a^n b^n c^n : n \geq 1\}$ is not content free language.

Proof:

Let L be a content free language and w be any string in L i.e $w \in L$ and m be pumping constant with $|w| \geq m$. Then w can be written as:

$w = uvxyz = a^m b^m c^m$

such that $|vxy| \leq m$ & $|vy| \geq 1$

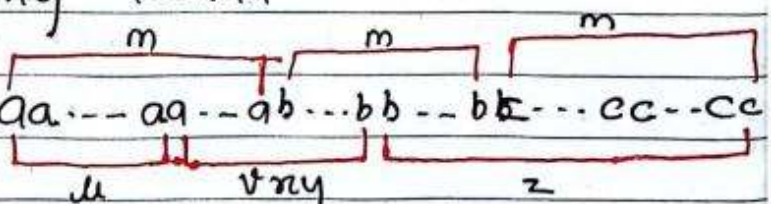
If the given language L is CFL, then it must satisfy $uv^i xy^i z \in L$ for $i \geq 0$

We can see that vxy cannot contain both as and cs as they are separated by at least m+1 positions.

Following cases may occur:

i) vxy contains both as and b's

Then from pumping lemma



$w = uvxyz =$

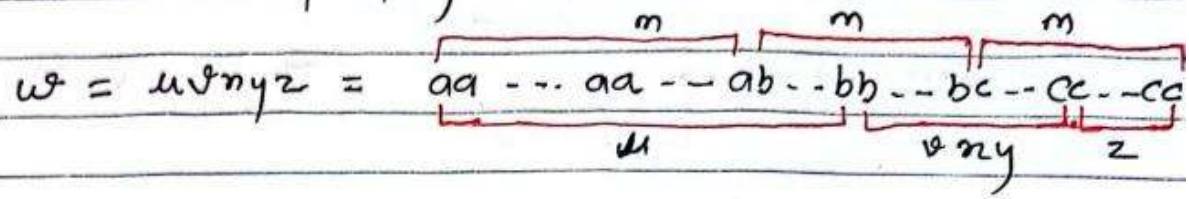
let $u = a^p$, $x = a^q$, $y = b^r$, $p+r \geq 1$

Then for $i = 2$,

$uv^2xy^2z = a^{m+p} b^{m+r} c^m \notin L$

ii) vxy contains both b's and c's.

Then from pumping lemma



let $v = b^p$, $n = b^q$, $y = c^r$; $p+r \geq 1$

Then, for $i = 2$

$$uv^2ny^2z = a^m b^{m+p} c^{m+r} \notin L$$

which contradicts out assumption.

hence, the given language $L = \{a^n b^n c^n : n \geq 1\}$ is not CFL.

Theorem:- The intersection of regular language with content free language is content free.

Proof: let L be a content free language and R be a regular language. Then $L = L(M_1)$ for some PDA

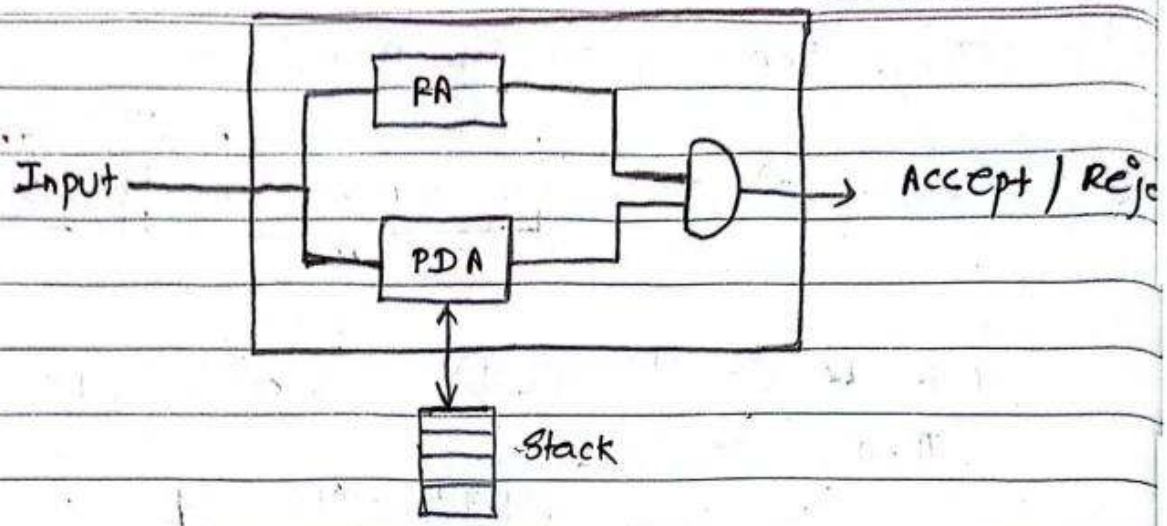
$M_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, s_1, F_1)$ and $R = L(M_2)$ for

finite automater $M_2 = (Q_2, \Sigma_2, \delta_2, s_2, f_2)$

Then, we can combine both to make new PDA

$M = (Q, \Sigma, \Gamma, \delta, s, F)$ as follows:

Date
Page



here new PDA M has following components

$$Q = Q_1 \times Q_2$$
$$\Gamma = \Gamma_1$$
$$S = (S_1, S_2)$$
$$F = F_1 \times F_2$$

And transition relation δ is defined as:

$((p_1, p_2), w, \alpha), ((q_1, q_2), \beta)) \in \delta$ if and only if $((p_1, w, \alpha), (q, \beta)) \in \Delta_1$ and $(p_2, w) \xrightarrow[M_1]{*} (q_1, \epsilon)$

Here PDA M goes from state (p_1, p_2) to (q_1, q_2) in the same way that M_1 passes from state p_1 & p_2 and also keeps track that M_2 changes its state on same input.
Therefore, $w \in L(M)$, if $w \in L(M_1) \cap L(M_2)$

hence, the intersection of regular language with content free language is content free

proved #