Theory of Computation
(FAFL - Finite Automata and Formal Language)

Introduction to Finite Automata

Central concepts of Automata Theory

i) **Alphabet**: It is finite, non-empty set of symbols

Ex: $\{0, 1\}$, $\{a, b, c\}$

ii) **String**: Finite sequence of symbols chosen from particular alphabet

"W" denotes a string.

Ex $W = 0101$

iii) **Empty string**: It is a string with zero occurance or no occurance of symbols

ϵ denotes empty string.

iv) **Length of a string**:- The no. of positions for the symbols in a string

$|w|$ denotes the length of the string

$w = 0101$. Length of the string is 4

v) **Power of an Alphabet**: If Σ is an alphabet, the set of strings of certain length is represented by exponential notation. Σ^i is the set of words of length i.

Eg: Σ^*

$\Sigma \rightarrow$ set of strings/symbols

$\Sigma = \{0, 1, \ldots a, \ldots z, A, \ldots Z\}$

If $\Sigma = \{0, 1\}$ then $\Sigma^0, \Sigma^1, \Sigma^2 = ?$

$\Sigma^0 = \{\phi\}$, $\Sigma^1 = \{0, 1\}$ $\Sigma^2 = \{00, 01, 10, 11\}$

Scanned by CamScanner

vi) Language: Set of strings which taken from Σ^* where Σ is a particular alphabet

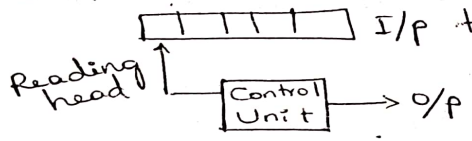
Eg: An empty language is denoted by $\alpha = \{\phi\}$

A language of strings consisting of equal no. of zero's & one's

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

$\alpha = \{0^n 1^n \mid n \geq 1\}$.

Finite Automata:



aabb

5 states

* Reading head is responsible for reading the I/p tape from left to right.
* Once all the symbols are read. At that time if the control unit is at the final state then we say the string is accepted otherwise rejected.

Finite Automata is a mathematical model which is used to study the abstract machines, abstract computing devices with the ~~tape~~ I/p's chosen from Σ

It mainly has 3 components

I/p tape:-

It is divided into cells, each of which can hold only one symbol. The string to be processed is stored in these cells.

Scanned by CamScanner

Control Unit

The machines has some states, of which is start state denoted as q_0 and atleast one final state.

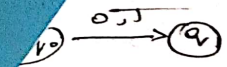
O/P

It may be accept / reject.

Symbols used in designing the Finite Automata.

Symbols	Meaning
① q_0	a circle is used to represent a state. Here q_0 is the state of the machine
② $\rightarrow q_0$	the circle with an arrow represents start state of the machine
③ q_0	Two circles are used to represent a final state
④ $q_0 \xrightarrow{1} q_1$	An arrow with label 1 indicates that there is a transition (change of states) from state q_0 on the I/p symbol 1 to the state q_1 it is represented $\delta(q_0, 1) = q_1$
⑤ q_0 (loop labelled 0)	An arrow with label 0, starts & ends with q_0 indicates that the machine is in the state q_0 on reading 0 I/p it remains in the same state q_0. It is represented as $\delta(q_0, 0) = q_0$

Scanned by CamScanner

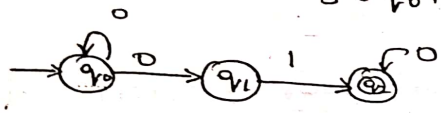


An arrow with label 0,1 goes from state q_0 to q_1 indicates that the machine on the state q_0 on reading 0 or 1 enters into state q_1

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_1$

1.



$\delta(q_0, 0) = q_0$

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

$\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_2$

δ	0	1
$\rightarrow q_0$	q_0, q_1	
q_1		q_2
$* q_2$	q_2	

2.

$\delta(q_0, 0) = q_0$

$\delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_1$

$\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_2$

$\delta(q_2, 1) = q_2$

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_1	q_2
$* q_2$	q_2	q_2

FA
- Deterministic Finite Automata (DFA)
- Non deterministic FA (NFA)
- Epsilon NFA (ϵ-NFA)

Scanned by CamScanner

Deterministic Finite Automata (DFA)

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

- Σ: i/p symbol
- F: final state
- q_0: start state
- Q: $(q_0, \ldots, q_n)$
- δ: Transition → $\delta : Q \times \Sigma = Q$.

DFA is defined with 5 tuples i.e.

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

where Q is set of finite states

Σ is finite set of input symbols

δ is transition function where

$\boxed{\delta : Q \times \Sigma = Q}$

q_0 is initial state where $q_0 \in Q$

F is set of final states where $\boxed{F \subseteq Q}$

Construct the transition table for the following.

δ	0	1
→ q_0	q_1	q_0
q_1	q_1	q_2
* q_2	q_2	q_2

Extended Transition function ($\hat{\delta}$)/(δ^*)

It describes what happens when we start any state and follow any input symbol

$\hat{\delta}$ is constructed by δ

Def^n: Basis: $\hat{\delta}(q, \epsilon) = q$

This indicates that, if the machine is in state q and reads no I/p symbols then the machine is still in the state q.

Induction: Let $w = xa$, where a is the last symbol & x is the remaining string of w.

$$\hat{\delta}(q, w) = \hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$$

1.

Write the extend transition fun for DFA for I/p symbol $w = 101$

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

$$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1)$$
$$= \delta(q_0, 1)$$
$$= q_0$$

$$\hat{\delta}(q_0, 10) = \delta(\hat{\delta}(q_0, 1), 0)$$
$$= \delta(q_0, 0)$$
$$= q_1$$

$$\hat{\delta}(q_0, 101) = \delta(\hat{\delta}(q_0, 10), 1)$$
$$= \delta(q_1, 1)$$
$$= q_2$$

2. $w = 100$

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

$$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1)$$
$$= \delta(q_0, 1)$$

Scanned by CamScanner

$w = abaab$

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

$\hat{\delta}(q_0, a) = \delta(\hat{\delta}(q_0, \epsilon), a)$

$= \delta(q_0, a)$

$= q_1$

$\hat{\delta}(q_0, ab) = \delta(\hat{\delta}(q_0, a), b)$

$= \delta(q_1, b)$

$= q_2$

$\hat{\delta}(q_0, aba) = \delta(\hat{\delta}(q_0, ab), a)$

$= \delta(q_2, a)$

$= q_1$

$\hat{\delta}(q_0, abaa) = \delta(\hat{\delta}(q_0, aba), a)$

$= \delta(q_1, a)$

$= q_1$

$\hat{\delta}(q_0, abaab) = \delta(\hat{\delta}(q_0, abaa), b$

$= \delta(q_1, b)$

$= a.$

Scanned by CamScanner

This indicates that, if the machine is in state q and reads no I/p symbols then the machine is still in the state q.

Induction: Let $w = xa$, where a is the last symbol & x is the remaining string of w.

$$\hat{\delta}(q, w) = \hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$$

1.

Write the extend transition fun for DFA for I/p symbol $w = 101$

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

$$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1)$$
$$= \delta(q_0, 1)$$
$$= q_0$$

$$\hat{\delta}(q_0, 10) = \delta(\hat{\delta}(q_0, 1), 0)$$
$$= \delta(q_0, 0)$$
$$= q_1$$

$$\hat{\delta}(q_0, 101) = \delta(\hat{\delta}(q_0, 10), 1)$$
$$= \delta(q_1, 1)$$
$$= q_2$$

2. $w = 100$

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

$$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1)$$
$$= \delta(q_0, 1)$$

Scanned by CamScanner

$\hat{\delta}(q_0, 10) = \delta(\hat{\delta}(q_0, 1), 0)$

$= \delta(q_0, 0)$

$= q_1$

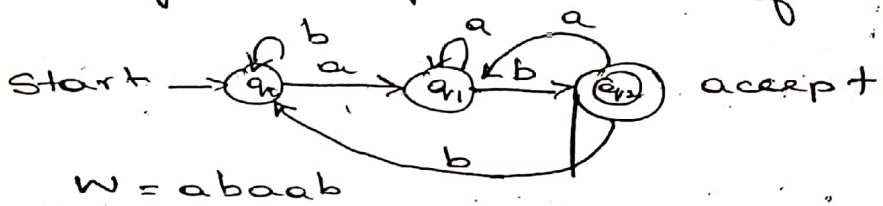
$\hat{\delta}(q_0, 100) = \delta(\hat{\delta}(q_0, 10), 0)$

$= \delta(q_1, 0)$

$= q_1$

String is rejected $\because$ it didn't reach final state.

3.



$w = abaab$

$\hat{\delta}(q_0, \epsilon) = q_0$.

$\hat{\delta}(q_0, a) = \delta(\hat{\delta}(q_0, \epsilon), a)$

$= \delta(q_0, a)$

$= q_1$

$\hat{\delta}(q_0, ab) = \delta(\hat{\delta}(q_0, a), b)$

$= \delta(q_1, b)$

$= q_2$

$\hat{\delta}(q_0, aba) = \delta(\hat{\delta}(q_0, ab), a)$

$= \delta(q_2, a)$

$= q_1$

$\hat{\delta}(q_0, abaa) = \delta(\hat{\delta}(q_0, aba), a)$

$= \delta(q_1, a)$

$= q_1$

$\hat{\delta}(q_0, abaab) = \delta(\hat{\delta}(q_0, abaa), b)$

$= \delta(q_1, b)$

$= a$.

Scanned by CamScanner

Language Accepted by DFA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. a string 'w' is accepted by the machine 'M', if it takes the machine from initial state q_0 to the final state i.e. $\hat{\delta}(q_0, w)$ is in F. Thus the lang. accepted by the DFA represented as

L(M) can be formally written as

$L(M) = \{ w \mid w \in \Sigma^* \text{ and } \hat{\delta}(q_0, w) \text{ is in } F \}$

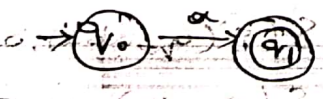
1. Construct a DFA to accept empty lang.

$L = \{\phi\}$

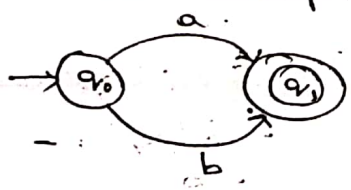
2. DFA to accept an empty string.

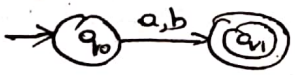
$L = \{\epsilon\}$

3. DFA to accept exactly one 'a'



4. DFA to accept one a or one b





Scanned by CamScanner

5. DFA to accept 0 or more a's.

6. Draw a DFA to accept strings of a's having atleast one a.

Transition diagram

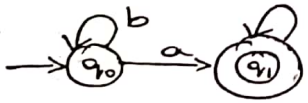
δ	a
$\rightarrow q_0$	q_1
$* q_1$	q_1

Transition table

7. Design a DFA to accept strings of a's & b's having atleast one a.

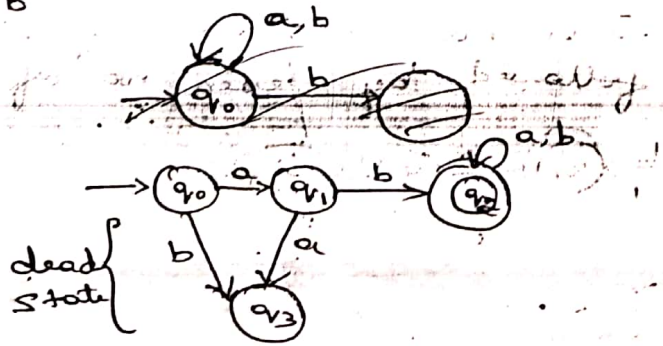
δ	a	b
$\rightarrow q_0$	q_1	q_0
$* q_1$	q_1	q_1

8. Design a DFA to accept strings of a's & b's having exactly one a.



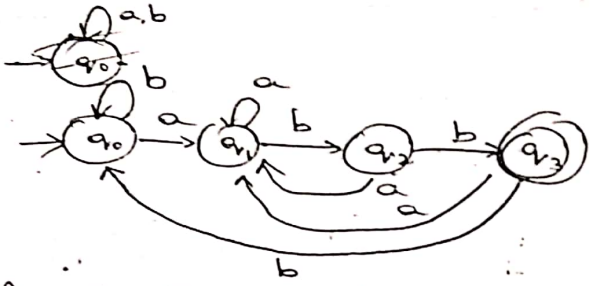
δ	a	b
$\rightarrow q_0$	q_1	q_0
$* q_1$	—	q_1

9. Design a DFA to accept strings of a's & b's starting with the string ab

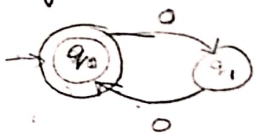


Scanned by CamScanner

10. Draw a DFA to accept strings of a's & b's ending with the string abb.

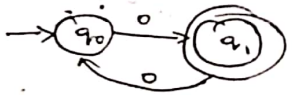


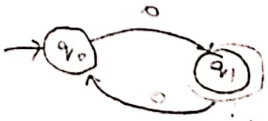
11. Design a DFA for the set of strings having even length of 0's



δ	0
→ q_0	q_1
q_1	q_2
* q_2	q_1

12. Design a DFA for odd length of 0's

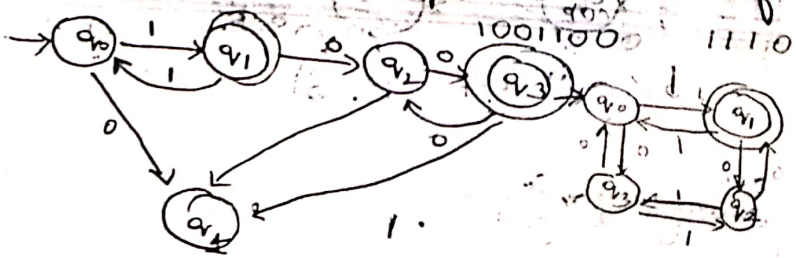




13. Design DFA for foll. lang.

L = {w | w has odd no. of 1's followed by even no. of zeros}

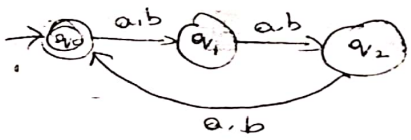
1001100 1110000



Scanned by CamScanner

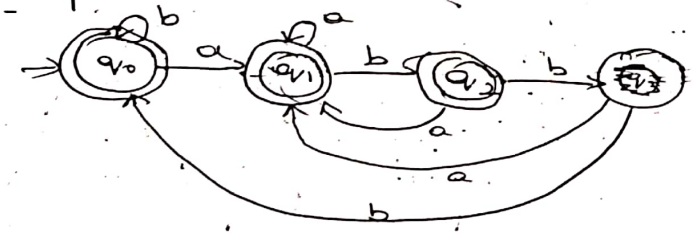
14. Design a DFA

$L = \{ |w| \bmod 3 = 0 , w \in \{a, b\} \}$



L = {Set of strings a's or b's doesn't end with abb}

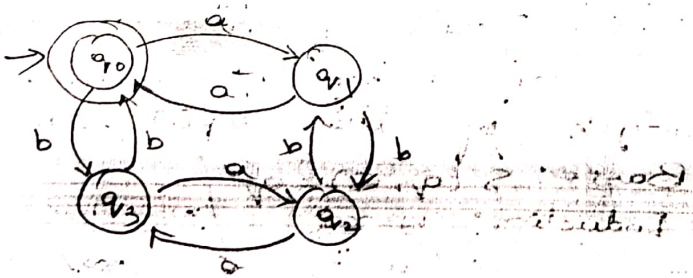
ababab



L = {w : w has even no. of a's or even no. of b's}

aabb

abab

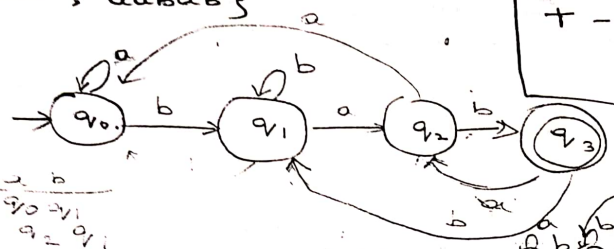


$L = \{ wbab \mid w \in \{a,b\}^* \}$

$\{abab, aabab\}$

$* - 0$ or more occurance

$+ - 1$ or more occurence



	a	b
$\rightarrow * q_0$	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_3
$\rightarrow q_3$		

Non-Deterministic finite automata (NFA)

NFA is defined as set of five tuples

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

where Q is a finite set of states

Σ is finite set of I/P symbols

$\delta \rightarrow$ transⁿ funⁿ $\delta : Q \times \Sigma = 2^Q$

(power set of Q)

q_0 is initial state where $q_0 \in Q$

F set of final states where $F \subseteq Q$

Note: i) When compared to DFA, NFA has more options to design.

ii) DFA is more efficient than NFA

Extended Transiⁿ function ($\hat{\delta}$)

$\hat{\delta}$ is defined by $\delta \rightarrow$ induction on the length of the input string

Basis: $\hat{\delta}(q, \epsilon) = q$

Induction: $w = xa$

$\hat{\delta}(q, w) = \hat{\delta}(q, xa) = \delta(\hat{\delta}(q, w), a)$

1.

$w = 001$

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

$\delta(q_0, 0) = \delta(\hat{\delta}(q_0, \epsilon), 0)$

$= \delta(q_0, 0)$

$= \{q_1\}$

$\delta(q_0, 00) = \delta(\hat{\delta}(q_0, 0), 0)$

$= \delta(q_1, 0)$

$= \{q_1\}$

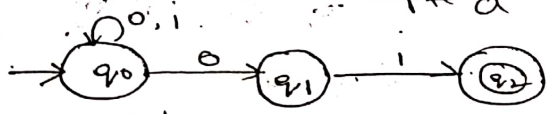
$\delta(q_0, 001) = \delta(\hat{\delta}(q_1, 0), 1)$

$= \delta(q_1, 1)$

$= \{q_2\}$

w is accepted

2.



$w = 001$

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

$\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \epsilon), 0)$

$= \delta(q_0, 0)$

$= \cancel{q_1 / q_0}$ $\delta(q_0, q_1, 0) = \delta((q_0, 0) \cup (q_1, 0))$

$\cancel{\hat{\delta}(q_0, 00) = \delta(\hat{\delta}(q_0, 0), 0)}$

$\cancel{= \delta(q_0, 0)}$ $= \{q_0, q_1\} \cup \{\phi\}$

$\cancel{= q_1}$ $= \{q_0, q_1\}$

$\hat{\delta}(q_0, 00) = \delta(\hat{\delta}(q_0, 0), 0)$

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

$= \delta((q_0, 0) \cup (q_1, 0))$

$= \delta(\{q_1, q_0\} \cup \{\phi\})$

$= \delta((q_0, 0) \cup (q_1, 0))$

$= \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\}$

$\hat{\delta}(q_0, 001) =$

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

$= \delta((q_0, 1) \cup (q_1, 1))$

$= \{q_0, q_2\}$

$w = 00101$

$\hat{\delta}(q_0, 001) = \{q_0, q_2\}$

$\hat{\delta}(q_0, 0010) = \delta(\hat{\delta}(q_0, 001), 0)$

$= \delta(\{q_0, q_2\}, 0)$

$= \delta(q_0, 0) \cup \delta(q_2, 0)$

$= \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\}$

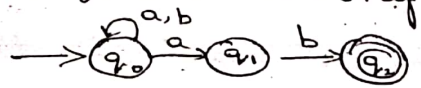
$\delta(q_0, 00101) = \delta(\hat{\delta}(q_0, 0010), 1)$

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

$= \delta(q_0, 1) \cup \delta(q_1, 1)$

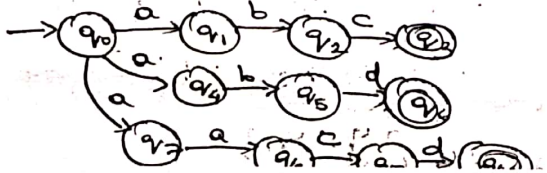
$= \{q_0, q_2\}$

1. Design an NFA to accept strings of a's & b's ending with ab



	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
$*q_2$	ϕ	ϕ

2. Obtain an NFA to recognize the set of strings abc, abd & aacd



Scanned by CamScanner

Conversion from NFA to DFA

a) Subset construction method :

Language accepted by NFA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA, a string w is accepted by the machine M. If it takes initial state as q_0 to the final state i.e $\hat{\delta}(q_0, w) = F$

Thus the lang. accepted by NFA is represented as $L(M)$ & can be formally written as $L(M) = \{ w \mid w \in \Sigma^* \text{ \& } \hat{\delta}(q_0, w) \text{ is in } F \}$

#

Procedure:

i) Given an NFA, $M_N = (\Sigma_N, \delta_N, q_0, F_N, Q_N)$ which accepts the lang. $L(M_N)$, we can find an equivalent DFA, $M_D = (\Sigma_D, \delta_D, q_0, F_D, Q_D)$ such that $L(M_N) = L(M_D)$

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, q_0 is the start state of DFA

Step 2: Identify the alphabets of DFA - The I/p alphabets of NFA are I/p alphabets of DFA i.e $\Sigma = \{a, b\}$, $\Sigma_N = \Sigma_D$

Step 3: Identify Q_D which are the states of DFA: Set of subsets of Q_N will be the set of states of Q_D.

Ex: $Q_N = \{q_0, q_1, q_2\}$ then $|Q_N| = 2^Q = 2^3 = 8$

$Q_D = \{\}, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}$

Step 4: Identify the final states of DFA

If $\{q_1, q_2, \ldots, q_k\}$ is a state in Q_D

Scanned by CamScanner

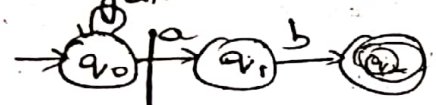
of DFA, provided one of the $q_i, q_j, \ldots q_k$ is the final state of Q_N

Step 5 : Identify the transition of DFA:
For each state $\{q_i, q_j, \ldots q_k\}$ in Q_D & for each input symbol a in Σ, trans^n can be obtained as follows.

$$\delta_D((q_i, q_j, \ldots q_k), a) = \delta_N(q_i, a), (q_j, a), \ldots (q_k, a))$$

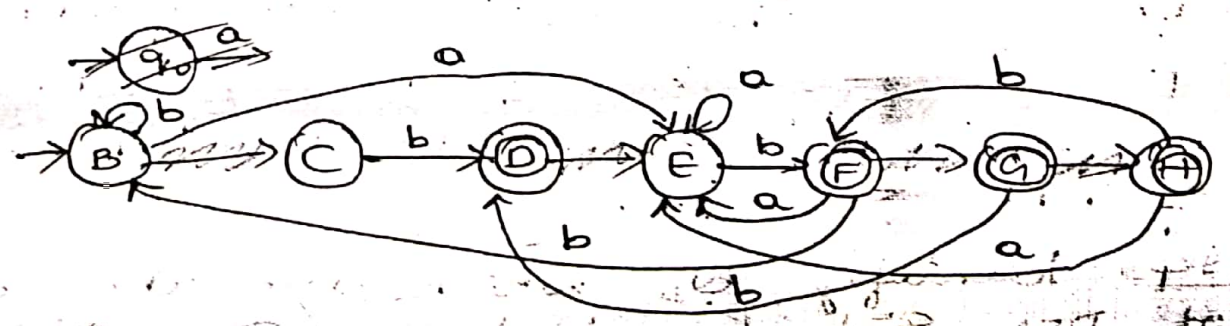
* Obtain DFA for the foll. NFA using subset construc^n method

1.



δ	a	b
→ q_0	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
* q_2	ϕ	ϕ

δ		a	b
$\{\ \}$	A	ϕ (A)	ϕ (A)
→ q_0	B	$\{q_0, q_1\}$ (E)	q_0 (B)
q_1	C	ϕ (A)	q_2 (D)
* q_2	D	ϕ (A)	ϕ (A)
$\{q_0, q_1\}$	E	$\{q_0 q_1\}$ (E)	$\{q_0 q_2\}$ (F)
* $q_0 q_2$	F	$\{q_0 q_1\}$ (E)	$\{q_0\}$ (B)
* $q_1 q_2$	G	ϕ (A)	$\{q_2\}$ (D)
* $q_1 q_2 q_2$	H	$\{q_0 q_1\}$ (E)	$\{q_0 q_2\}$ (F)



Conversion of NFA to DFA using Lazy method

Procedure

i) Given an NFA, $M_N = (Q_N, \Sigma_N, \delta_N, q_0, F_N)$ which accepts the lang $L(M_N)$, we can

Such that $L(M_D) = L(M_N)$.

Step 1: Identify the start state of DFA.
Since q_0 is the start state of NFA q_0 is the start state of DFA.

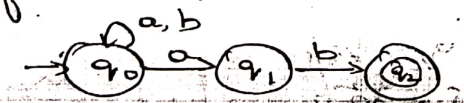
S-2: Identify the alphabets of DFA: the I/p alphabets of NFA are I/p Σ of DFA i.e. $\Sigma = \{a, b\}$ for both NFA & DFA.

S-3 Identify the transitions of DFA(δ)
For each state $\{q_i, q_j, \ldots q_k\}$ in Q_D & for each I/p symbol 'a' in Σ, the transition can be obtained as below

$$\delta_D(\{q_i, q_j, \ldots q_k\}, a) = \delta_N(q_i, a) \cup (q_j, a) \cup (q_k, a))$$

Add the state $\{q_l, q_m, \ldots q_n\}$ to Q_D if it is not already in Q_D
Add the transition from $\{q_i, q_j \ldots q_k\}$ to $\{q_l, q_m, \ldots q_n\}$ on the I/p symbol a.

S-4 Identify the final states of DFA;
If $\{q_i, q_j, q_k\}$ is a state in Q_D & if one of the q_i, q_j, q_k is the final state of NFA, then $\{q_i, q_j, q_k\}$ will be the final states of DFA.



δ	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
$*q_2$	ϕ	ϕ

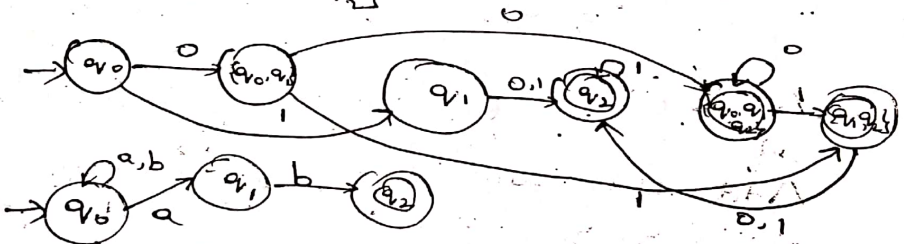
δ	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

Scanned by CamScanner

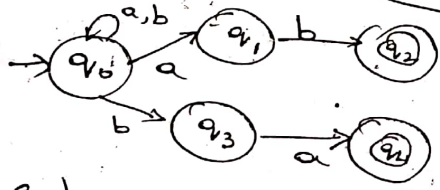
2.

δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1	q_2	q_2
$*q_2$	ϕ	q_2

δ	0	1
q_0	$\{q_0, q_1\}$	$\{q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
$\{q_1\}$	q_2	q_2
$\{q_2\}$	ϕ	q_2
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_2\}$	$\{q_2\}$

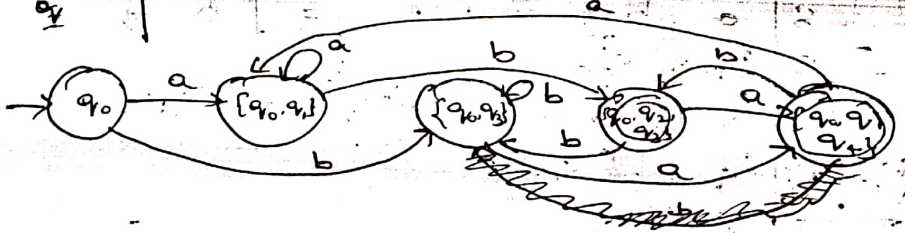


3.



δ	a	b
q_0	$\{q_0, q_1\}$	$\{q_0, q_3\}$
q_1	ϕ	q_2
q_2	ϕ	ϕ
q_3	q_4	ϕ
q_4	ϕ	ϕ

δ	a	b
q_0	$\{q_0, q_1\}$	$\{q_0, q_3\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2, q_3\}$
$\{q_0, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
$\{q_0, q_1, q_4\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
q_4		



Scanned by CamScanner

By subset consⁿ

	a	b
{ }	ϕ	ϕ
q_0		
q_1		
q_2		
q_3		
q_4		
$q_0\ q_1$		
$q_0\ q_2$		
$q_0\ q_3$		
$q_0\ q_4$		
$q_1\ q_2$		
$q_1\ q_3$		
$q_1\ q_4$		
$q_2\ q_3$		
$q_2\ q_4$		
$q_3\ q_4$		
$q_0\ q_1\ q_2$		
$q_0\ q_1\ q_3$		
$q_0\ q_1\ q_4$		
$q_1\ q_2\ q_3$		
$q_1\ q_2\ q_4$		
$q_2\ q_3\ q_4$		
$q_0\ q_1\ q_2\ q_3$		
$q_0\ q_1\ q_2\ q_4$		
$q_1\ q_2\ q_3\ q_4$		
$q_0\ q_2\ q_3\ q_4$		
$q_0\ q_1\ q_2\ q_3\ q_4$		
$q_0\ q_1\ q_3\ q_4$		

Pplec
Decision
Used
Digital

Scanned by CamScanner

Applications of final Automata

1. Designing of digital circuit: It is used to check the behaviour of Digital Ckt. Also used for circuit verification

2. Compiler construction: It is used in the design of Lexical analyzer. (one of the phase of a compiler. Which breaks the I/p text into identifier, keywords, punctuations etc.

3. String matching: It is used for designing the software for identifying the words, phrases or other patterns

4. String processing. To write the software for processing the natural lang.

5. Software Design: It is used in building the software, to verify the system having finite no. of states

NFA with Epsilon moves (ϵ-NFA)

ϵ-NFA is designed as set of 5 tuples

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

where Q: Finite set of states

Σ: Finite —"— input symbols

δ: Transition function

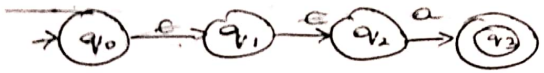
$\delta : Q \times \Sigma \cup \{\epsilon\} = 2^Q$

q_0: Initial state $q_0 \in Q$

F: Set of final states $F \subseteq Q$

Epsilon closure / (ϵ-closure)

Scanned by CamScanner

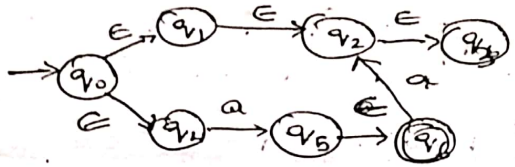


E-close / E-closure $(q_0) = \{q_0, q_1, q_2\}$

——"—— $(q_1) = \{q_1, q_2\}$

——"—— $(q_2) = \{q_2\}$

——"—— $(q_3) = \{q_3\}$



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

—"— $(q_1) = \{q_1, q_2, q_3\}$

—"— $(q_2) = \{q_2, q_3\}$

—"— $(q_3) = \{q_3\}$

—"— $(q_4) = \{q_4\}$

—"— $(q_5) = \{q_5\}$

—"— $(q_6) = \{q_6\}$

Extended transⁿ funⁿ of ε-NFA

Basis: $\hat{\delta}(q, \epsilon) = \text{E-close}(q)$

The extended transition funⁿ $(\hat{\delta})$ describes what happens to the state of machine when I/P is a string (sequence of symbols) The extended tranⁿ funⁿ $\hat{\delta} \neq \delta : \Sigma \times Q \cup \{\epsilon\} \to 2^Q$ is also defined recursively wrt basis & Induction.

Basis: $\hat{\delta}(q, \epsilon) = \text{E-close}(q)$

It indicates that if the machine is in state Q, & reads no I/P, then the machine is still in the state Q

Induction: Let $w = xa$ where a is the last symbol of w & x is the [illegible] Let q is

the current state of x is the string to be processed and after consuming the string x, Let the states of machine be

$\{p_1, p_2, p_3, \ldots p_m\}$

ie $\hat{\delta}(p, x) = \{p_1, p_2, \ldots p_m\}$

Let the trans^n from $\{p_1, p_2, \ldots p_m\}$ on the trans^n a is $\delta(\{p_1, p_2, \ldots p_m\}, a)$

$$= \delta(p_1, a) \cup \delta(p_2, a) \cup \ldots \delta(p_m, a)$$

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

Now $\hat{\delta}(q, w) = E\text{-close}(r_1, r_2, r_3, \ldots r_m)$

$$= E\text{-close}(r_1) \cup E\text{-close}(r_2) \cup E\text{-close}(r_3) \ldots E\text{-close}(r_m)$$

Thus, various properties of extended trans^n function for an ϵ-NFA can be

$\rightarrow \hat{\delta}(q, \epsilon) = E\text{-close}(q)$

if $w = xa$ $\rightarrow \hat{\delta}(q, w) = \hat{\delta}(q, xa)$

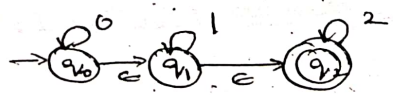
$= E\text{-close}(\delta(\hat{\delta}(q, x), a))$

where $w = xa$

$\hat{\delta}(q, w) = \hat{\delta}(q, ax)$

$= E\text{-close}(\delta(\hat{\delta}(q, a), x))$

where $w = ax$

1. 

$w = 012$

$\hat{\delta}(q_0, \epsilon) = \epsilon\text{-close}(q_0) = \{q_0, q_1, q_2\}$

$\hat{\delta}(q_0, w) = \hat{\delta}(q_0, 012) = ?$

$\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \epsilon), 0)$

$= \delta(q_0, q_1, q_2, 0)$

$= \delta(E\text{-close}(q_0, 0) \cup E\text{-close}(q_1, 0) \ldots$

$\hat{\delta}(q_0, 0)$

$\hat{\delta}(q_0, 0$

$\hat{\delta}(q_0$

$\hat{\delta}($

$\hat{\delta}(q$

Scanned by CamScanner

$= \delta\{$ ~~E-close~~$(q_0) \cup$ E-close$(\phi) \cup$ E-clos

$\hat{\delta}(q_0, 0) = \{q_0, q_1, q_2\}$

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1)$

$= \delta(\{q_0, q_1, q_2\}, 1)$

$= \delta[$ E-close$(q_0, 1) \cup$ E-close$(q_1, 1) \cup$ E-close$(q_2, 1)]$

$= \delta[$ E-close$(\phi) \cup$ E-close$(q_1) \cup$ E-close(

$= \{q_1, q_2\}$

$\hat{\delta}(q_0, 012) = \delta(\hat{\delta}(q_0, 01), 2)$

$= \delta(\{q_1, q_2\}, 2)$

$= \delta[$ E-close$(q_1, 2) \cup$ E-close$(q_2, 2)]$

$= \delta[$ E-close$(\phi) \cup$ E-close$(q_2)]$

$= \{q_2\}$

$\therefore$ w is accepted

2.

$w = 0112$

$\hat{\delta}(q_0, \in) =$ E-close(q_0)

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

$\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \in), 0)$

$= \delta(\{q_0, q_1, q_2, q_3\}, 0)$

$= \delta[$ E-close$(q_0, 0)$, E-close$(q_1, 0)$, E-close$(q_2, 0)$, E-close$(q_3, 0)]$

$= \delta($ E-close(q_0), E-close(ϕ), E-close(ϕ), E-close$(\phi)]$

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

Scanned by CamScanner

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1)$

$= \delta(\{q_0, q_1, q_2, q_3\}, 1)$

$= \delta(\text{E-close}(q_0, 1) \cup \text{E-close}(q_1, 1) \cup \text{E-close}(q_2, 1) \cup \text{E-close}(q_3, 1))$

$= \delta[\text{E-close}(\phi) \cup \text{E-close}(q_1) \cup \text{E-close}(q_2) \cup \text{E-close}(\phi))$

$= \delta\{q_2, q_1, q_3\} \cup \{q_2, q_3\}$

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

$\hat{\delta}(q_0, 011) = \delta(\hat{\delta}(q_0, 01), 1)$

$= \delta(\{q_1, q_2, q_3\}, 1)$

$= \delta[\text{E-close}(q_1, 1) \cup \text{E-close}(q_2, 1) \cup \text{E-close}(q_3, 1)$

$= \delta[\text{E-close}(q_1) \cup \text{E-close}(q_2) \cup \text{E-close}\phi)$

$= \{q_1, q_2, q_3\} \cup \{q_2, q_3\}$

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

$\hat{\delta}(q_0, 011 2) = \delta(\hat{\delta}(q_0, 011), 2)$

$= \delta(\{q_1, q_2, q_3\}, 2)$

$= \delta[\text{E-close}(q_1, 2) \cup \text{E-close}(q_2, 2) \cup \text{E-close}(q_3, 2)]$

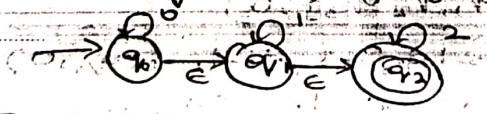
$= \delta[\text{E-close}(\phi) \cup \text{E-close}(\phi) \cup \text{E-close}(q_3)]$

$= \{q_3\}$

$\therefore$ w is accepted

Eliminating Epsilon transition

1.



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

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

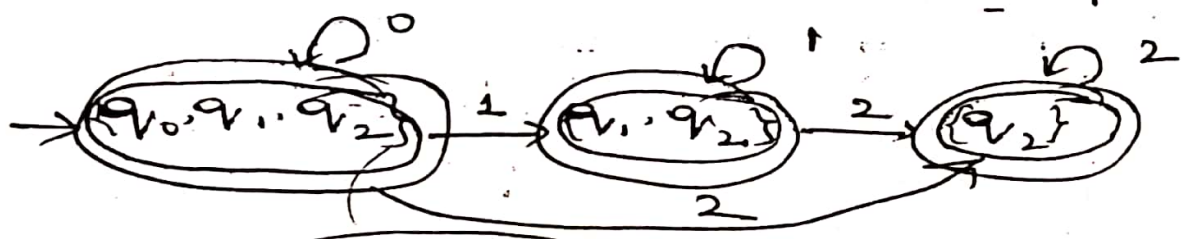
E-close$(q_2) = \{q_2\}$

Scanned by CamScanner

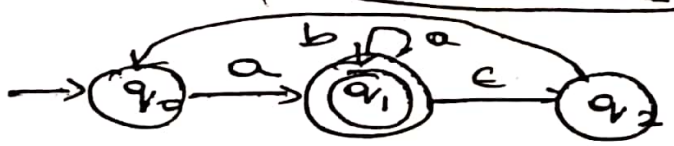
δ	ϵ	0	1	2
$\rightarrow q_0$	$\{q_1\}$	$\{q_2\}$	ϕ	ϕ
q_1	$\{q_2\}$	ϕ	$\{q_1\}$	ϕ
$*q_2$	ϕ	ϕ	ϕ	$\{q_2\}$

$\Rightarrow$

δ	0	1	2
E-close(q_0)	E-close(q_0)		
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$*\{q_1, q_2\}$	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
$*\{q_2\}$	ϕ	ϕ	$\{q_2\}$



2.



E-close(q_0) = $\{q_0\}$

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

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

δ	ϵ	a	b

a) **Indistinguishable**: Let p & q are the states of a DFA and are said to be indistinguishable if and only if

$\hat{\delta}(p,w) \in F$ & $\hat{\delta}(q,w) \in F$ (or)

$\hat{\delta}(p,w) \notin F$ or $\hat{\delta}(q,w) \notin F$ $\forall\, w \in \Sigma^*$

b) **Distinguishable**: Let p & q are the states of DFA and are said to be distinguishable if and only if

$\hat{\delta}(p,w) \in F$ & $\hat{\delta}(q,w) \notin F$

$\hat{\delta}(p,w) \notin F$ or $\hat{\delta}(q,w) \in F$

Note: Distinguishable state is represented by ×
Indistinguishable state by ✓

Table filling algorithm: II (Myhill - Nerode Theorem)

Minimizn of DFA

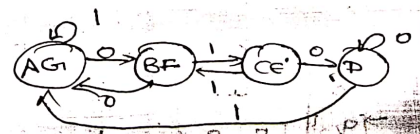
1.

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

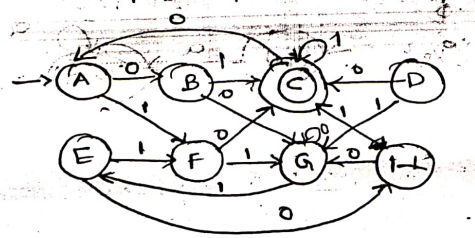
	A	B	C	D	E	F
B	×					
C	×	×				
*D	×	×	×			
E	×	×	✓	×		
F	×	✓	×	×	×	
G	✓	×	×	×	×	×

CE, BF, AG, D

δ	0	1
→AG	BF	AG
BF	AG	CE
CE	D	BF
*D	D	AG



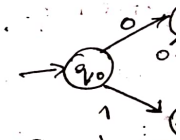
2.



δ	0
→A	B
B	G
*C	A
D	C
E	H
F	C
G	G
H	G

δ	0
→AE	BH
BH	G
DF	C
*C	AE
G	G

3.



δ	0
→q0	q1
q1	q2
q2	q3
q3	q
*q4	q

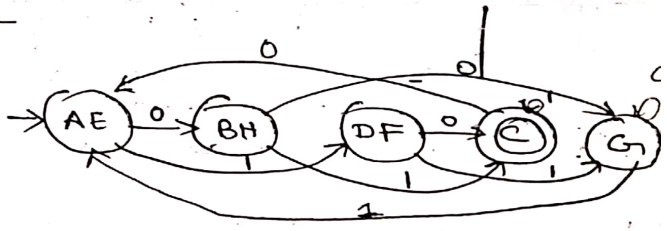
δ	0	1
→q0q3	{q1,q2,q1,q	
q1q2	{q1,q2}	
q1q3	{q1,q2,q	
q2q3	{q1,q2	

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

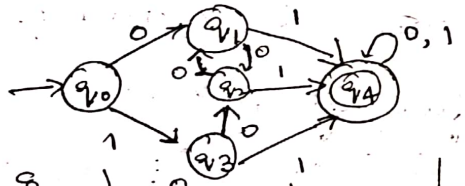
	A	B	C	D	E	F	G
B	×						
C	×	×					
D	×	×	×				
E	✓	×	×	×			
F	×	×	×	✓	×		
G	×	×	×	×	×	×	
H	×	✓	×	×	×	×	×

AE, BH, DF, C, G

δ	0	1
→AE	BH	DF
BH	G	C
DF	C	G
*C	AE	C
G	G	AE



3.



δ	0	1
→q_0	q_1	q_3
q_1	q_2	q_4
q_2	q_1	q_4
q_3	q_2	q_4
*q_4	q_4	q_4

	q_0	q_1	q_2	q_3
q_1	×			
q_2	×	✓		
q_3	×	✓	✓	
*q_4	×	×	×	×

$q_0, q_1 q_2, q_1 q_3, q_2 q_3, q_4$

δ	0	1
→q_0	$\{q_1 q_2, q_1 q_3\}$	$\{q_1 q_3, q_2 q_3\}$
$q_1 q_2$	$\{q_1 q_2\}$	$\{q_4\}$
$q_1 q_3$	$\{q_1 q_2, q_2 q_3\}$	$\{q_4\}$
$q_2 q_3$	$\{q_1 q_2\}$	$\{q_4\}$

$q_0 \{q_1, q_2, q_3\}, q_4$

Scanned by CamScanner

δ	0	1
$\rightarrow q_0$	$\{q_1 q_2 q_3\}$	$\{q_1 q_2 q_3\}$
$\{q_1 q_2 q_3\}$	$\{q_1 q_2 q_3\}$	q_4
$* q_4$	q_4	q_4

Basis: If p is a final state & q is a non-final state the pair p-q is distinguishable.

Induc^n : Let p & q belongs Q, & $a \in \Sigma$, $r = \delta(p,a)$, $s = \delta(q,a)$, then (r,s) is known to be distinguishable, then (p,q) is distinguishable.

1 Define the foll. terms with an example
a) Alphabet b) String c) Σ^n d) L e) DFA f) NFA
g) ϵ-NFA
h) Extendended trans fun of DFA, NFA, ϵ
i) E-close (Basis & Induc^n)
j) Distinguishable & Indistinguishable state

2. Define NFA, DFA with given problem
3. Write subset const^n algorithm & conversion from NFA to DFA
4. Write Algorithm to eliminate ϵ-trans^n & convert ϵ-NFA to DFA
5. Write table filling algorithm (basis & ind^n) & minimize to DFA

Unit 2

Regular expression.

Def^n: Regular express^n is defined recursively as followed

1) Basis: ϕ is a regular expression denoting the lang. $L\{\phi\} = \phi$. ε is a regular expression $L\{\varepsilon\} = \varepsilon$.

a is a regular expr^n where $a \in \Sigma$ denoting the lang. $L\{a\} = \{a\}$

2) Induction: If E & F are the regular expressions denoting the lang. L(E) & L(F) respectively.

i) $E + F$ is a regular expression denoting union of L(E) & L(F) i.e.

$$L(E+F) = L(E) \cup L(F).$$

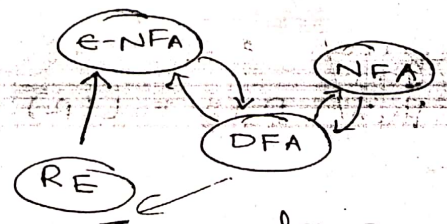
ii) $E.F$ is a regular expression denoting concatenation of L(E) follow by L(F) i.e.

$$L(E.F) = L(E).L(F)$$

iii) E^* is a regular expression denoting the lang. L(E) i.e

$$L(E^*) = (L(E))^*$$

Equivalence b/w regular expression & finite automata



Theorm: Every lang. defined by regular expression is also defined by finite automata.

Scanned by CamScanner

Proof: Consider $L = L(R)$ for some regular expression R, then ST $L = L(E)$ with some ϵ-NFA with the foll. conditions.

i) Only one accepting state (final state)

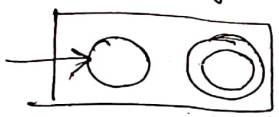
ii) No arc into the initial state (self loop)

iii) No arc out of accepting state

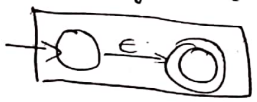
Proof is by using the def^n of regular expression.

Basis:

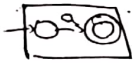
i) ϕ is a regular expression denoting the lang. ϕ, i.e. $L(\phi) = \phi$ or $\{\}$



ii) ϵ is a regular expression denoting the lang ϵ, i.e. $L(\epsilon) = \{\epsilon\}$

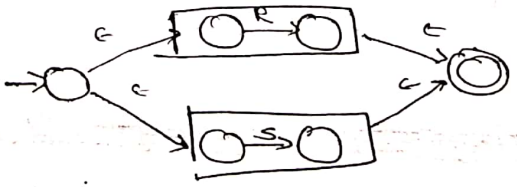


iii) a is a regular expression where $a \in \Sigma$. $L(\epsilon) = \{a\}$



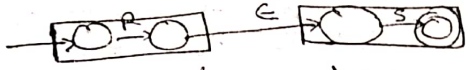
Induction: If R & S are regular exp denoting the lang $L(R)$ & $L(S)$

i) $R+S$ is a regular expression denoting the lang $L(R) \cup L(S)$, then $R+S = L(R) \cup L(S)$

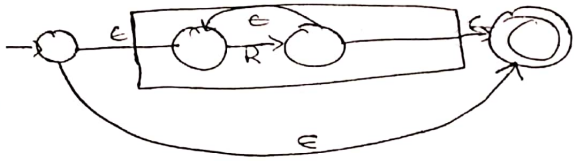


Scanned by CamScanner

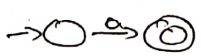
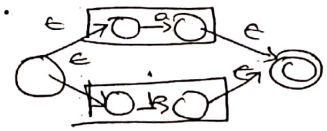
ii) R.S is a regular expr. a
of L(R) & L(S) ts



iii) R^* is a regular expⁿ. then
$L(R^*) = (L(R))^*$

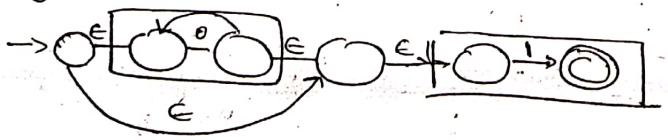


Hence the following assumptions are proved

R.E	ε-NFA
ϕ	
ε	
a	
a+b	
a.b	
a*	

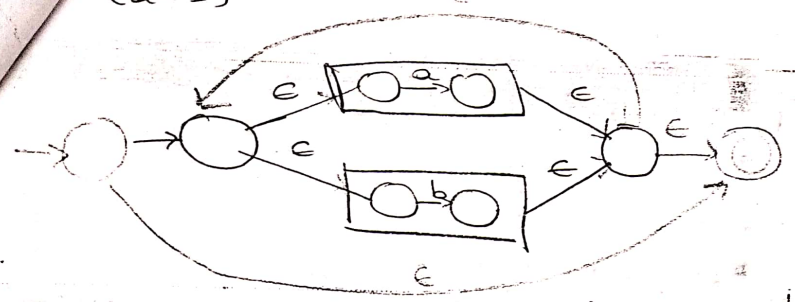
Convert the foll. regular expression to ε-NFA

① 0^*1

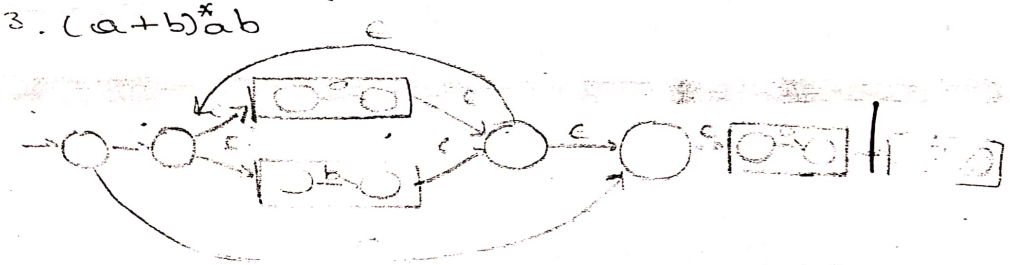


Scanned by CamScanner

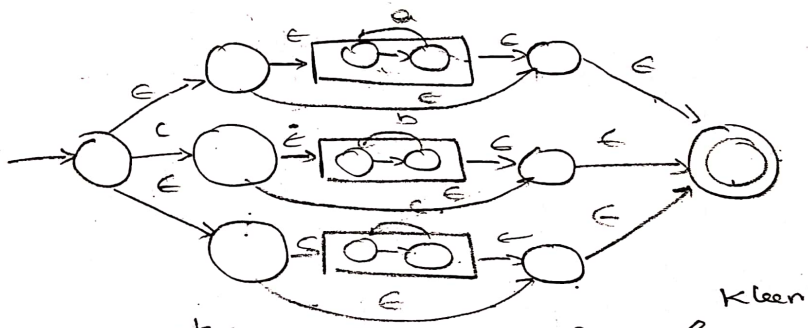
$(a+b)^*$



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



4) $a^*+b^*+c^*$



2. Converting DFA to RE < Kleen's theorm / State elimination method.

Laws of algebra

Two lang. L & M

1) $L+M = M+L$ => Commutative law of union

2) $(L+M)+M = L+(M+M)$ => Associative law for union.

3) $(L\{M)M = L(MM)$ => Associative law for concatenation

4) $\phi+L = L+\phi = L$ => Identity law for union

5) $\epsilon L = L\epsilon = L$ => Identity law for concatenation

Laws of closure

1) $(L^*)^* = L^*$

2) $\phi^* = \epsilon$

3) $\epsilon^* = \epsilon$

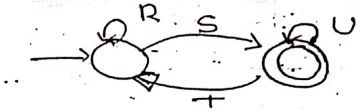
4) $L^+ = LL^* = L^*L$

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

(3) State elimination method

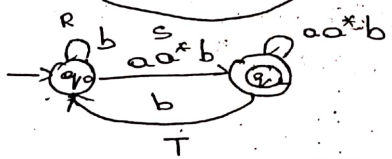
It includes only the start state and the end state, all the intermediate states are removed and the final DFA is represented as

$(R + SU^*T)^* SU^*$



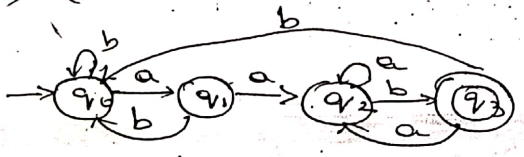
1.

$\Rightarrow$

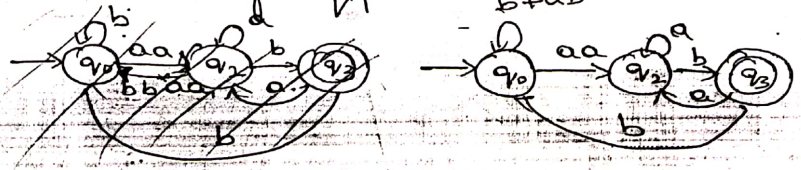


$\Rightarrow (b + aa^*b(aa^*b)^*b)^* aa^*b(aa^*b)^*$

2.

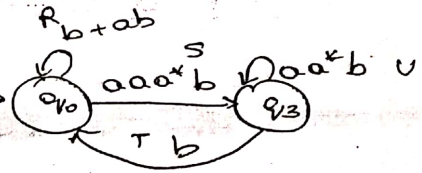


Eliminating q_1



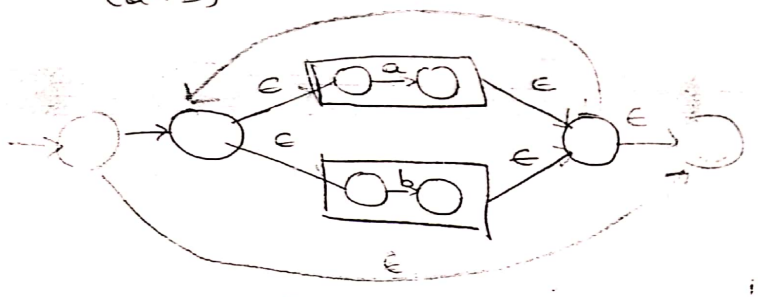
Eliminating q_2

$\Rightarrow (b+ab+aaa^*b(aa^*b)^*b)^* aaa^*b(aa^*b)^*$

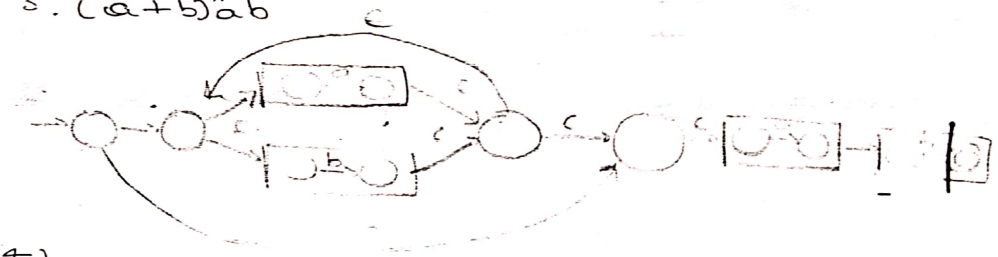


Scanned by CamScanner

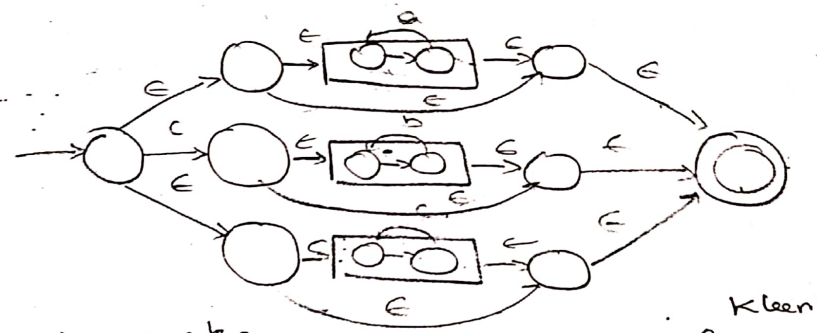
2. $(a+b)^*$



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



4) $a^*+b^*+c^*$



Converting DFA to RE < Kleen's theorm / State elimination method.

Laws of algebra

Two lang. L & M

1) $L+M = M+L$ => Commutative law of union

2) $(L+M)+M = L+(M+M)$ => Associative law for union.

3) $(L\,\dot{\varepsilon}\,M)M = L(MM)$ => Associative law for concatenation

4) $\phi+L = L+\phi = L$ => Identity law for union

5) $\epsilon L = L\epsilon = L$ => Identity law for concatenation

Laws of closure

1) $(L^*)^* = L^*$

2) $\phi^* = \epsilon$

3) $\epsilon^* = \epsilon$

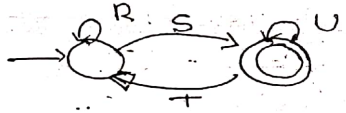
4) $L^+ = LL^* = L^*L$

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

(3) State elimination method

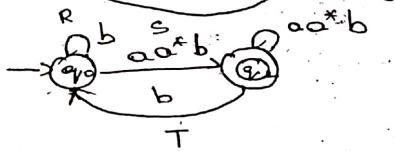
It includes only the start state and the end state, all the intermediate states are removed and the final DFA is represented as

$(R + SU^*T)^* SU^*$



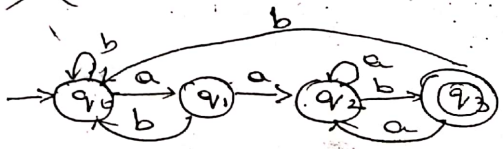
1.

$\Rightarrow$

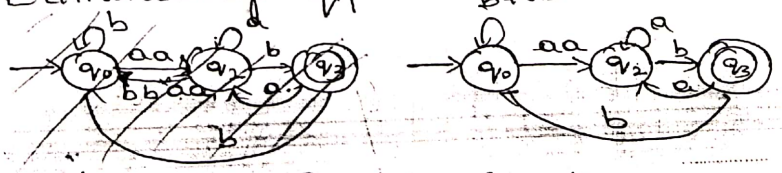


$\Rightarrow (b + aa^*b(aa^*b)^*b)^* aa^*b(aa^*b)^*$

2.

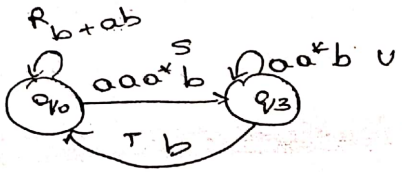


Eliminating q_1

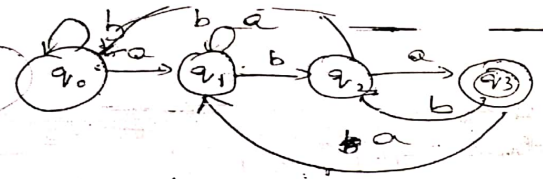


Eliminating q_2

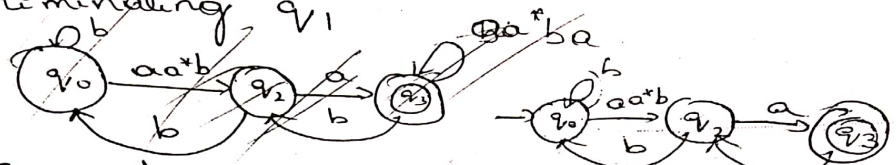
$\Rightarrow (b + ab + aaa^*b(aa^*b)^*b)^* aaa^*b(aa^*b)^*$



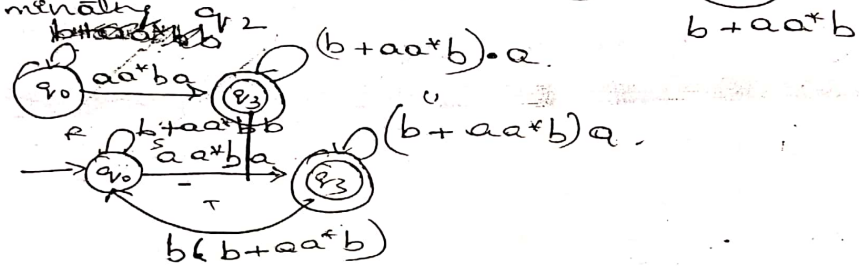
3)



Eliminating q_1



Eliminating q_2

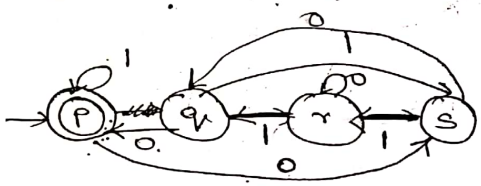


$(R + SU^*T)^* SU^*$

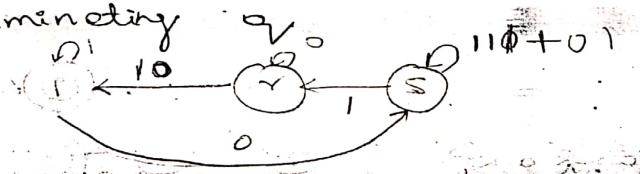
$= [b + aa^*bb + aa^*ba[(b+aa^*b)a]^* b(b+aa^*b)]^*$
$\cdot\ aa^*ba[(b+aa^*b)a]^*$

4)

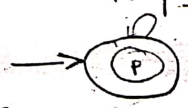
	0	1
$\rightarrow$ p	s	p
q	p	s
r	r	q
s	q	r



Eliminating q_0



R $1 + 0(111+01)^* 10^* 01$



R

$[1 + 0(111+01)^* 10^* 01 + \phi \in \phi] \in \phi$

$[1 + 0(111+01)^* 10^* 01 + \phi]^* \phi$

$[1 + 0(111+01)^* 10^* 01]^*$

CONTEXT FREE GRAMMAR (CFG)

- It is defined with 4 tuples.

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

where V is a finite set of variable or non terminals, T is a finite set of terminals, P = set of rules or productions i.e. $A \to \alpha$, where $A \in V$, $\alpha \in (V \cup T)^*$, S is a start variable where $S \in V$

1. Construct a context free grammar for the lang.

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

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

$S \to \epsilon$ $\quad S \to ab$

$S \to aSb$

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

$S \to \epsilon$

$S \to aSb$

2. $L = \{a^n b^{n+1} \mid n \geq 0\}$

$= \{b, abb, aabbb, aaabbbb, \ldots\}$

$S \to b$

$S \to aSb$

$S \to b$

$S \to abb$

$S \to aSb$

aabbb

3. $L = \{a^{2n} \mid n \geq 0\}$

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

$S \to \epsilon$

$S \to aaSa$

4. $L = \{a^{2n+1} \mid n \geq 0\}$

$= \{a, aaa, aaaaa, \ldots\}$

$S \to a$

$S \to aaSa$

5. $L = \{a^n b^n \mid n > 0\} = \{\epsilon, ab, aabb, \ldots\}$

Scanned by CamScanner

6. $L = \{(ab)^n \mid n \geq 1\}$ → $a^{n+2} b^n$, $n \geq 1$ $L = \{aaabb, aaaabb$

$= \{ab, abab, \ldots\}$

$S \to ab$

$S \to abS$

$S \to aaA$

$A \to ab \mid aAb$

7. $L = \{w w^R \mid w \in \{a,b\}^*\}$ (Reverse of string)

$= \{\epsilon, aa, bb, abba, baab, \ldots abba, \ldots\}$

$S \to \epsilon$

$S \to aSa$

$S \to bSb$

8. $L = \{a^m b^m c^n \mid m \geq 1, n \geq 0\}$

$= \{ab, abc, aabbc, aabbcc, \ldots$

$S \to abABC$ (or)

$A \to aA \mid \epsilon$

$B \to bB \mid \epsilon$

$C \to cC \mid \epsilon$

$S \to AB$

$A \to ab \mid aAb$

$B \to \epsilon \mid cB$

Derivatives

The process of obtaining a string from ~~starting~~ start variable of Grammar (G) by applying one or more prod^n is called derivatives.

Two types:

i) Left Most Derivative

ii) Right Most Derivative

i) LMD: Deriv^n is said to be left most if in each step the left most variable in the sentential form is replaced by its production

ii) RMD: Deriv^n is said to right most if in each step the right most [illegible] for

1.

$S \to AB$

$A \to aaA \mid \epsilon$

$B \to Bb \mid \epsilon$

$S \xrightarrow{lm} AB$

$S \xrightarrow{lm} aaAB$

$S \xrightarrow{lm} aaB$

$S \xrightarrow{lm} aaBb$

$S \xrightarrow{lm} aab.$

$S \xrightarrow{rm} A$

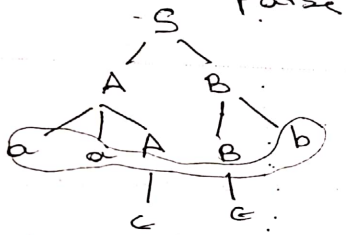
$S \xrightarrow{rm} A\epsilon$

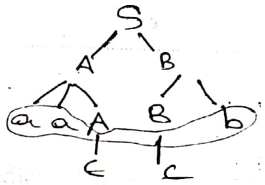
$S \xrightarrow{rm} Ab$

$S \xrightarrow{rm} aaAb$

$S \xrightarrow{rm} aab$

Parse Tree / Derivation Tree (D





Write the LMD ~~or~~ RMD ~~or~~ DT for the I/P $+*-xyxy$ to the given

$E \to +EE \mid *EE \mid -EE \mid x \mid y$

$E \xrightarrow{lm} +EE$

$E \xrightarrow{lm} +*EEE$

$E \xrightarrow{lm} +*-EEEE$

$E \xrightarrow{lm} +*-xEEE$

$E \xrightarrow{lm} +*-xyEE$

$E \xrightarrow{lm} +*-xyxE$

$E \xrightarrow{lm} +*-xyxy$

$E \xrightarrow{rm} +EE$

$E \xrightarrow{rm} +E\underline{E}y$

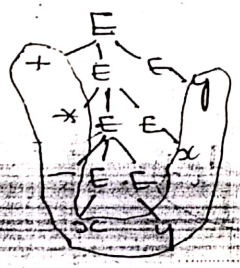
$E \xrightarrow{rm} +*EEy$

$E \xrightarrow{rm} +*-EEEy$

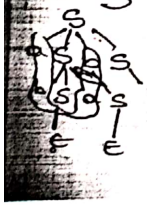
$E \xrightarrow{rm} +*-xEEy$

$E \xrightarrow{rm} +*-xyEy$

$E \xrightarrow{rm} +*-xyxy$



$S \to aSbS \mid bSaS \mid \epsilon$ abab.



$S \xrightarrow{lm} aSbS$

$S \xrightarrow{lm} abSaSbS$

$S \xrightarrow{lm} abaSbS$

$S \xrightarrow{lm} ababS$

$S \xrightarrow{rm} aSbS$

$S \xrightarrow{rm} aSb\epsilon$

$S \xrightarrow{rm} abSab$

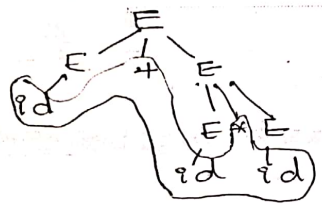
$S \xrightarrow{rm} abab$

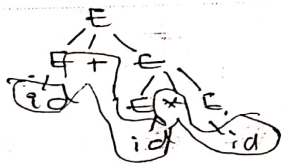
Scanned by CamScanner

$E \rightarrow E+E \mid E*E \mid id$, $id+id*id$

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

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





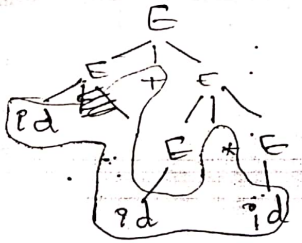
Ambiguous grammar

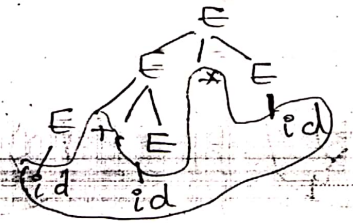
Let CFG, $G = (V, T, P, S)$ is ambiguous if there is atleast one string w in T^* for which we can find 2 diff parse tree, each with root labelled as an yield w.

$E \rightarrow E+E \mid E*E \mid id$: $id+id*id$

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

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





Since grammar has 2 DT's. It is ambiguous.

$E \rightarrow T \mid E+T$
$T \rightarrow F \mid F*T$

2) $S \to aS \mid X$ aa

$X \to aX \mid a$

$S \xrightarrow{lm} aS$
$S \xrightarrow{lm} aX$
$S \xrightarrow{lm} aa$

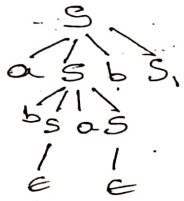
$S \xrightarrow{lm} X$
$S \xrightarrow{lm} aX$
$S \xrightarrow{lm} aa$

Ambiguous.

3) $S \to aSbS \mid bSaS \mid \epsilon$ w = abab

$S \xrightarrow{lm} aSbS$
$S \xrightarrow{lm} abSaSbS$
$S \xrightarrow{lm} abasbS$
$S \xrightarrow{lm} ababS$
$S \xrightarrow{lm} abab$

~~$S \xrightarrow{lm} bSaS$~~
~~$S \xrightarrow{lm} b$~~
$S \xrightarrow{lm} aSbS$
$S \xrightarrow{lm} abS$
$S \xrightarrow{lm} abasbS$
$S \xrightarrow{lm} ababS$
$S \xrightarrow{lm} abab$



Ambiguous

Simplification of CFGs & normal forms.

Chomsky Normal Form (CNF)

The context free Grammar is in Chomsky Normal Form (CNF) if all the productions are of the form. $A \to BC$ or $A \to a$ (single terminal) where production should have one terminal or two variable.

Ex: $S \to AS \mid a$

$A \to SA \mid b$

Simplifying context free grammar

Methods

i) Eliminating ϵ - production

ii) Elimination of unit production

iii) Elimination of useless production

Elimination of ϵ-production

1) $S \to BAAB$

$A \to 0A2 \mid 2A0 \mid \epsilon$

$B \to AB \mid 1B \mid \epsilon$

$A \to \epsilon$

$B \to \epsilon$

$S \to \epsilon$

Nullable variables

$S \to BAAB \mid AAB \mid BAB \mid BAA \mid AB \mid BB \mid AA \mid BA \mid AB$

$A \to 0A2 \mid 02 \mid 2A0 \mid 20$

$B \to AB \mid B \mid A \mid 1B \mid 1$

2) $S \to AaB \mid aaB$

$A \to \epsilon$

$B \to bbA \mid \epsilon$

$S \to AaB \mid aB \mid Aa \mid aaB \mid aa$

$B \to bbA \mid bb$

3) $S \to ABaC$

$A \to BC$

$B \to b \mid \epsilon$

$C \to D \mid \epsilon$

$D \to d$

$S \to ABaC \mid BaC \mid aC \mid a \mid AaC \mid ABa \mid Aa \mid Ba$

$A \to BC \mid B \mid C$

$B \to b$

$C \to D$

$D \to d$

Scanned by CamScanner

~~Elim~~

4) $S \to aA \mid aBB$

$A \to aaA \mid \epsilon$

$B \to bB \mid bbC$

$C \to B$

$S \to aA \mid a \mid aBB$

$A \to aaA \mid aa$

$B \to bB \mid bbC$

$C \to B$

Elemention of unit production,

1. $S \to AB$

$A \to a$

$B \to C \mid b$

$C \to D$

$D \to E \mid bc$

$E \to d \mid Ab$

Unit	Non-Unit
$C \to D$	$S \to AB$
$B \to C$	$A \to a$
$D \to E$	$B \to b$
	$D \to bc$
	$E \to d \mid Ab$

$B \to C \to D \to E$

$B \to C$ $\quad C \to D$ $\quad D \to E$

$B \to D$ $\quad C \to E$

$B \to E$

$B \to C$ in non-unit procn - α

$B \to D$ $\quad C \to D$ $\quad D \to E$

$\boxed{B \to bc}$ $\quad \boxed{C \to bc}$ $\quad \boxed{D \to d \mid Ab}$

$B \to E$ $\quad C \to E$

$\boxed{B \to d \mid Ab}$ $\quad \boxed{C \to d \mid Ab}$

$S \to AB$

$A \to a$

$B \to bc \mid b \mid d \mid Ab$

$C \to bc \mid d \mid Ab$

$D \to d \mid Ab \mid bc$

$E \to d \mid Ab$

Scanned by CamScanner

2. $S \to Aa / B$
$B \to A / bb$
$A \to a / bc / B$

Unit	Non-unit
$S \to B$	$S \to Aa$
$B \to A$	$B \to bb$
$A \to B$	$A \to a / bc$

$S \to B \to A \to B$

$S \to B$ $\quad B \to A$ $\quad A \to B$.
$S \to A$ $\quad B \to B$
~~$S \to B$~~
$S \to B$
$\boxed{S \to bb}$

$A \to B$
$\boxed{A \to bb}$

$S \to A$
$\boxed{S \to a / bc}$

$B \to A$
$\boxed{B \to a / bc}$

$B \to B$
$\boxed{B \to bb}$

Ans: ~~$S \to Aa / a / bc / bb$~~
~~$B \to bb$~~
~~$A \to a / bc / bb$~~

Ans: $S \to a / bc / bb / Aa$
$B \to a / bc / bb$
$A \to bb / a / bc$

Any production of context free grammar of the form $A \to B$ where $A, B \in V$ is called <u>unit production</u>.

Any prod^n of CFG of the form $A \to \epsilon$ is called <u>ϵ-prod^n</u>. Any variable A for which the derivation $A \xrightarrow{*} \epsilon$ is possible is called Nullable.

3. $S \to aA / aBB$
$A \to aaA / \epsilon$
$B \to bB / bbC$
$C \to B$

Nullable
$A \to \epsilon$

Unit	Non-unit
$C \to B$	$S \to aA / a / aBB$
	$A \to aaA / aa$
	$B \to bB / bbc$

$\boxed{\begin{array}{l} S \to aA / a / aBB \\ A \to aaA / aa \\ B \to bB / bbc \\ C \to B \end{array}}$

$C \to B$
$C \to bB / bbC$

Ans.
$\boxed{\begin{array}{l} S \to aA / aBB / a \\ A \to aaA / aa \\ B \to bB / bbc \end{array}}$

Scanned by CamScanner

4. $S \to abAB$
$A \to bAB / \epsilon$
$B \to BAa / A / \epsilon$

Nullable
$A \to \epsilon$
$B \to \epsilon$

$$\boxed{\begin{aligned} S &\to abAB / abA / abB / ab \\ A &\to bAB / bA / bB / b \\ B &\to BAa / Ba / Aa / a / A \end{aligned}}$$

Unit
~~$B \to B$~~
$B \to A$

Non-unit
$S \to abAB / abA / abB / ab$
$A \to bAB / bA / bB / b$
$B \to BAa / Ba / Aa / a$.

$B \to A$
$B \to bAB / bA / bB / b$.

$$\boxed{\begin{aligned} S &\to abAB / abA / abB / ab \\ A &\to bAB / bA / bB / b \\ B &\to BAa / Ba / Aa / a / bAB / bA / bB / b \end{aligned}}$$

Computing nullable variables.

Let $G = (V, T, P, S)$ be a context free grammar. Find all the nullable variables of G by the following iterative algorithm:

Basis: If $A \to \epsilon$ is a production of G, then A is nullable.

Induction: If there is a production $B \to C_1, C_2, \ldots C_k$, where each C_i is nullable then B is nullable.

Theorem: Let G be any CFG with ϵ not in $L(G)$ then there exist an equivalent grammar G_1 having no ϵ production.

Proof ?? # PTO

Scanned by CamScanner

$S \to a \mid aA \mid BAC$

$A \to aB \mid a$

$B \to Aa$

$C \to cCD$

$D \to d$

Ans

$S \to \cancel{a \mid a}\, aS_1 \mid \cancel{BA}\, BAS_2$

$S_1 \to A$

$S_2 \to \cancel{A}\, C$

$A \to aS_3 \mid a$

$S_3 \to B$

$B \to S_1 a$

$C \to CS_4$

$S_4 \to CD$

$D \to dS_5$

$S_5 \to d$

$A \to BC$

$A \to a$

2) $S \to BAAB$

$A \to 0A2 \mid 2A0 \mid \epsilon$

$B \to AB \mid 1B \mid \epsilon$

$S \to BAAB \mid BAA \mid BA \mid B \mid AAB \mid AB \mid A \mid AA \mid BB \mid BAB$

$A \to 0A2 \mid 02 \mid 2A0 \mid 20$

$B \to AB \mid A \mid B \mid 1B \mid 1$

$S \to BAS_1 \mid AS_1 \mid BAS_2 \mid BS_1 \mid BA \mid AB \mid AS_2 \mid BS_3 \mid S_2 \mid$

$S_1 \to AB$

$S_2 \to A$

$S_3 \to B$

$A \to \cancel{0AS_4} \; 0AS_4 \mid 0S_4 \mid S_2 0 \mid S_4 0$

$\cancel{S_4 \to 02}$ $S_4 \to 2$

$A \to 0S_5 \mid 0S_4 \mid$

$S_5 \to AS_4$

2) $S \to 0A \mid 1B$

$A \to 0AA \mid 1S \mid 1$

$B \to 1BB \mid 0S \mid 0$

$S \to S_1 A \mid S_2 B$

$S_1 \to 0$

$S_2 \to 1$

Scanned by CamScanner

$S_3 \to AA$

$B \to S_2 S_4 \mid S_1 S \mid 0$

$S_4 \to BB$

3) $S \to aA \mid aBB$

$A \to aaA \mid e$

$B \to bB \mid bbC$

$C \to B$.

$\cancel{S \to S_1 A \mid S_1 S_2}$

$\cancel{S_1 \to a}$

$\cancel{S_2 \to BB}$.

$\cancel{A \to S_1 S_3 \mid \lambda}$

$S \to aA \mid a \mid aBB$.

$A \to aaA \mid aa$

$B \to bB \mid bbC$

$C \to B$

$S \to aA \mid a \mid aBB$.

$A \to aaA \mid aa$

$B \to bB \mid bbC$

$C \to bB \mid bbC$

S_1

$S \to S_1 A \mid a \mid S_1 S_2$

$S_1 \to a$

$S_2 \to BB$.

$A \to S_1 S_3 \mid S_1 a$. $\quad A \to S_1 S_3 \mid S_1 S_1$

$S_3 \to aA$. $\quad S_3 \to S_1 A$

$B \to S_4 B \mid S_4 bC$. $\quad B \to S_4 B \mid S_4 S_5$

$S_4 \to b$

$\cancel{S_5 \to bC}$

$S_5 \to bC$

$S_5 \to S_4 C$

$C \to S_4 B \mid S_4 S_5$.

Scanned by CamScanner

Thm:

Proof: We first find the set V_n for all nullable variables of G using following steps.

i) Find all the production $A \to \epsilon$, put A into V_n

ii) Repeat the following steps until no other variables are added to V_n

iii) For all productions $B \to A_1 A_2 \ldots A_k$ where $A_1, A_2, \ldots A_k$ are in V_N, put B into V_n

Elemination of useless symbols.

a) Non-generating symbols

b) Non-reachable symbol

a) Eg: $S \to ABa \mid \boxed{BC}$

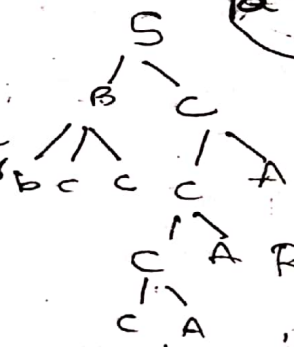
$B \to bcc$

$\boxed{C \to CA}$ — Recursive grammar

$D \to E$

$\boxed{A \to aC} \mid Bcc \mid a$

$E \to e$



Recursive grammar

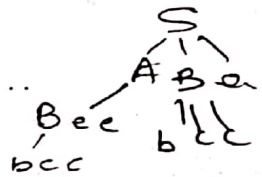
$S \to ABa$

$B \to bcc$

Non reachable $\boxed{D \to E}$

$A \to Bcc \mid a$

$\boxed{E \to e}$



$\Rightarrow$ $S \to ABa$

$B \to bcc$

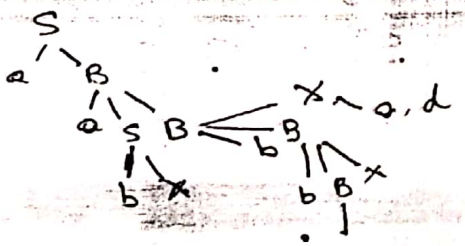
$A \to Bcc \mid a$

e) $S \to \boxed{aB} \mid bX$

$A \to \boxed{BAd} \mid bSX \mid a$

$\boxed{B \to aSB \mid bBX}$

$X \to \boxed{SBD} \mid \boxed{aBx} \mid ad$



$S \to bX$

$A \to bSX \mid a$

$S \to bX$

$X \to ad$

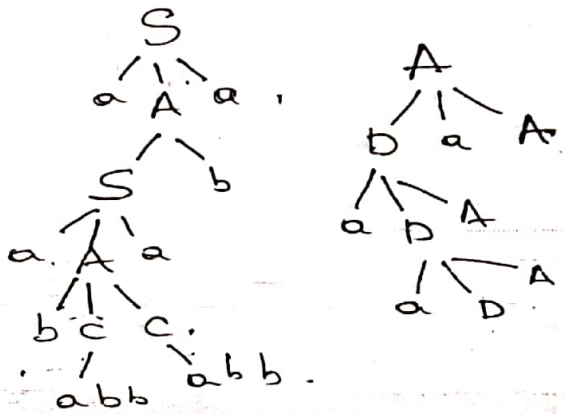
3) $S \to aAa$

$A \to Sb \mid bcc \mid \boxed{DaA}$

$C \to abb \mid \boxed{DD}$

$E \to aC$

$\boxed{D \to aDA}$

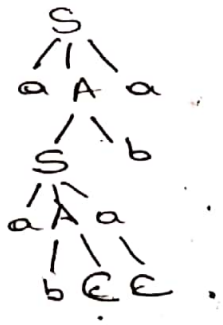


$S \to aAa$

$A \to Sb \mid bCC$

$C \to abb$

$\boxed{E \to aC}$



$S \to aAa$

$A \to Sb \mid bCC$

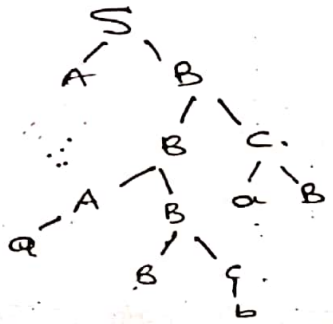
$C \to abb$

4. $S \to \boxed{AB} \mid CA$

$\boxed{B \to Bc \mid AB}$

$A \to a$

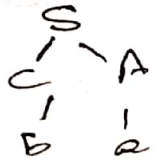
$C \to \boxed{aB} \mid b$



$S \to CA$

$A \to a$

$C \to b$

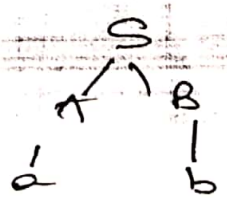


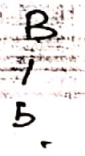
5. $S \to AB$

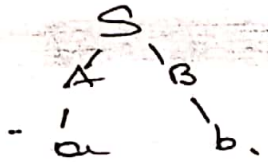
$A \to a$

$B \to b \mid C$

$\boxed{E \to c}$







$S \to AB$

$A \to a$

$B \to b \mid C$

Scanned by CamScanner

6) $S \to a \mid aA \mid B$

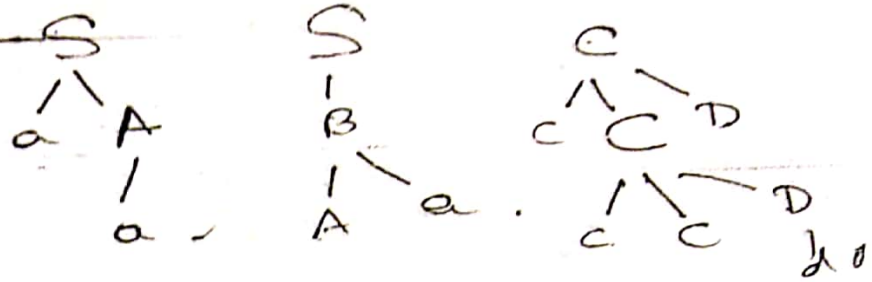
$A \to aB \mid a$

$B \to Aa$

$\boxed{E \to cCD}$

Non Reach. $\boxed{D \to ddd}$

~~S →~~



$S \to a \mid aA \mid B$

$A \to aB \mid a$

$B \to Aa$.

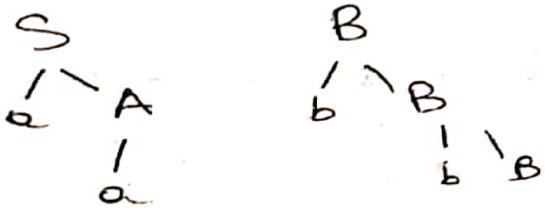
7) $S \to aA \mid bA$

$A \to aA \mid a$

Non Term. $\boxed{B \to bB}$

N Reachable $\boxed{D \to ab \mid Ea}$.

$\boxed{E \to ac \mid d}$.

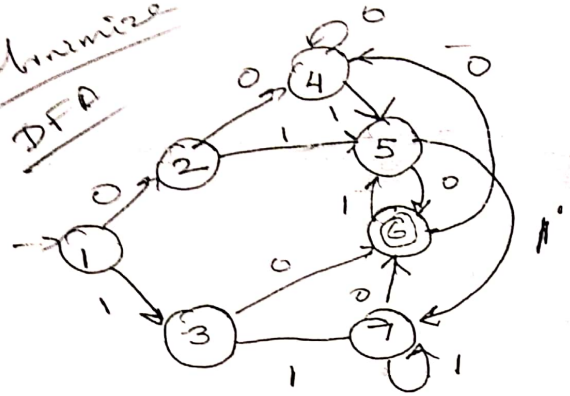


$S \to aA \mid bA$

$A \to aA \mid a$.

1) Write the applications of context free grammar.

Minimize DFA



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

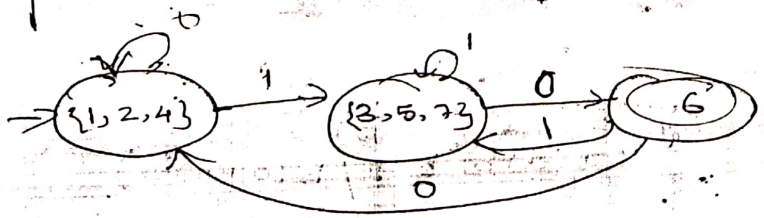
	1	2	3	4	5	6
2	✓					
3	×	×				
4	✓	✓	×			
5	×	×	✓	×		
6	×	×	×	×	×	
7	×	×	✓	×	✓	×

12, 14, 24, 35, 37, 57, 6

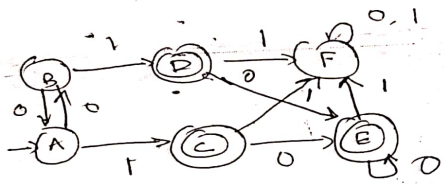
{124}, {3, 5, 7}, 6

δ	0	1
12	24	35
14	24	35
24		
35		
37		
57		
* 6		

δ	0	1
→ {1, 2, 4}	{1, 2, 4}	{3, 5, 7}
{3, 5, 7}	6	{3, 5, 7}
* 6	{1, 2, 4}	{3, 5, 7}



Scanned by CamScanner



Design R.E over $\Sigma = \{a, b\}$

i) String a's & b's where starting & ending symbol are diff

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

ii) having a's exactly 2.

$\therefore b^*ab^*ab^*$

iii) a's atleast 2

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

Scanned by CamScanner

iv) a's ~~& b's~~ are even.

$b^*(aa)^*b^*$ $\quad (aa+b)^*$

v) Strings of a's & b's starts with a.

$a(a+b)^*$

vi) end's with a.

$(a+b)^*a$.

vii) ~~ending with~~ contains a.

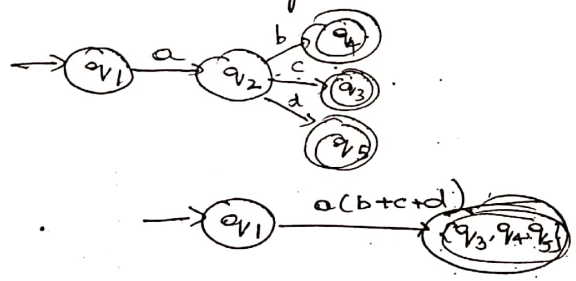
$b^*a(a+b)^*$ // $(a+b)^*a(a+b)^*$ Harsh

viii) starting & ending with ~~same~~ symbol

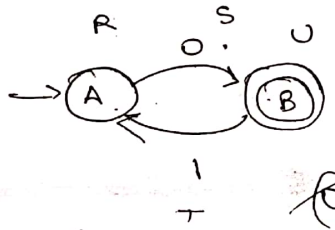
aba bab

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

DFA using state elimination method

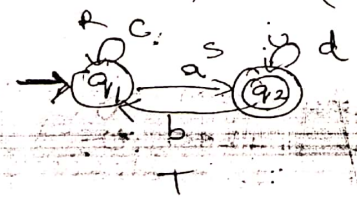


$a(b+c+d)$



$(R+SU^*T)^*SU^*$

$\cancel{(0+1)}$ $\quad 0(10)^*$



$\Rightarrow \cancel{(a+a.d^*b)^*ad^*}$

$c^*a(d^*+bc^*a)^*$

Strings of a's & b's having a's atmost 2.

$\cancel{ba^*b^*a^*}$

Set of a's & b's ending with ab,
$(a+b)^* ab$

Set of strings of a's & b's with no consecutive pair of a
$(b+ab)^* a^*$

Set of strings of a's & b's with atleast one consecutive pair of a's

Set of strings of a's & b's with atmost one consecutive pair of a's

Set of strings of a's & b's with even length.

with odd length

Write the RE for the foll Lang.
$L = \{ |w| \bmod 3 = 0 \mid w \in \{a,b\}^* \}$

Let $G = \{V, T, P, S\}$ be CFG, Let
$V = \{E, I\}$ $T = \{a, b, c, d, +, *, (,)\}$
$P = \{ E \to I \mid (E+E) \mid (E*E) \mid (E)$
$I \to a \mid b \mid c \mid d$
$\}$ Obtain the derivation for the string $(a+b)^* c^* d$ & $a+b*c$ & parse tree for each derivation.

~~$E \to I \mid E+E \mid E*E \mid (E)$~~

~~$E \to I+E*E$~~

~~$E \to E+E$~~

$E \to E*E$

$E \to E*E*E$

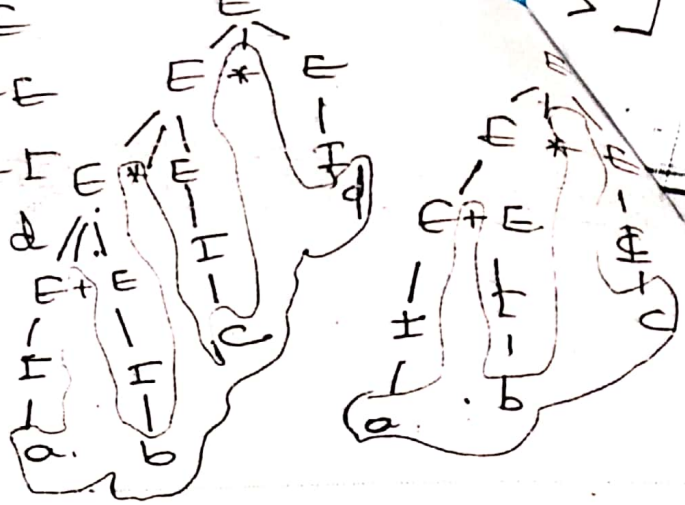
$E \to (E+E)*E*E$

$E \to (a+I)*E*E$

$(a+b)*I*E$

$(a+b)*c*I$

$(a+b)*c*d$



$E \to E*E$

$E \to E+E*E$

$E \to I+E*E$

$E \to a+I*E$

$E \to a+b*I$

$E \to a+b*C$

1. $S \to A0 / B$

$B \to A / 11$

$A \to 0 / 12 / B$

2. $S \to iCts / iCtSeS / a$

$C \to b$

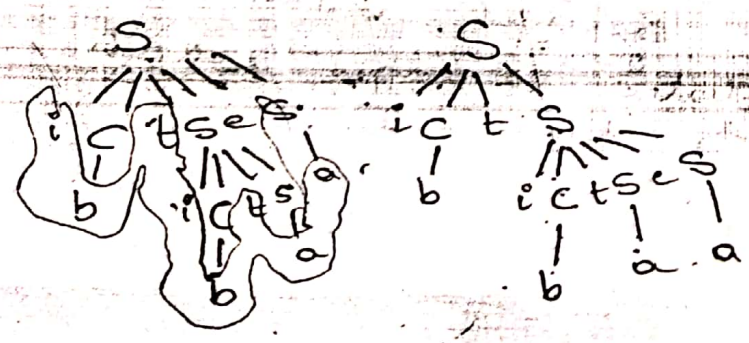
$S \xrightarrow{lm} iCtSeS$

$S \xrightarrow{lm} ibtSeS$

$S \xrightarrow{lm} ibtiCtseS$

$S \xrightarrow{lm} ibtibtSeS$

$S \xrightarrow{lm} ibtibtaea$



ibtibtaea

b

$S \xrightarrow{rm} ictSeS$

$S \xrightarrow{rm} ictSea$

$S \xrightarrow{rm} ictiCtSea$

$S \xrightarrow{rm} ictictaea$

$S \xrightarrow{rm} ictibtaea$

$S \xrightarrow{lm} iCtS$

$S \xrightarrow{lm} ibtS$

$S \xrightarrow{lm} ibtiCtSeS$

$S \xrightarrow{lm} ibtibtSeS$

$S \xrightarrow{lm} ibtibtaea$

Ambiguous.

Scanned by CamScanner

Write CFG for the foll lang

$L = \{ ww^R / w \in \{a, b\}^* \}$ $L = \{aabb, abba, \ldots\}$

$S \to \epsilon$

$S \to aSa$

$S \to bSb$

$L = \{ 0^n w w^R 1^n \mid n \geq 1, w \in \{a, b\}^* \}$

$L = \{ 0aa1, 0bb1, 0abba1, 00abba11, \ldots \}$

$S \to \epsilon$

$S \to 0S1$

$S \to aSa$

$S \to bSb$

$S \to 0S1$

$S \to aSa \mid bSb$

$0S1$

$00S11$

Applications of CFG & RE's

CFG

→ There are numerous applications of grammar which includes the following

Applications
- → Parser, YACC parser, markup lang, XML
- → Finite Automaton
- → Digital design

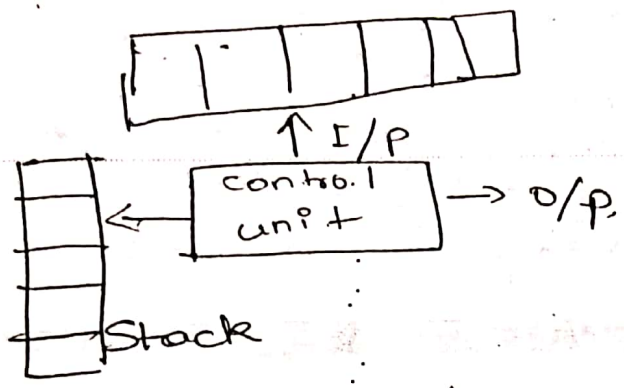
Grammar is formal lang. are very suitable to express any programming lang like C, C++, Pascal etc. also grammars are used to identify an identifier

An automation can be represented using directed graph with vertices representing states & edges which certain represents transition.

Automation is useful in digital design because many of the concepts are automation theory. Even though logical implementation is through transistors, flip flops etc.

its implementation ~~of~~ functional description of the circuit,

Pushdown automata



DPA — NDPA

$\delta : Q \times \Sigma \times \Gamma = Q \times \Gamma^*$

$\delta : Q \times \Sigma \times \Gamma \to 2^{Q \times \Gamma^*}$

Pushdown automata is defined as a set of 7 tuples

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

Tow

top of the stack

Stack symbol

q_0 is initial state $q_0 \in Q$.

Q is finite set of states

XML F is set of final states

δ is transition function.

$\delta : Q \times \Sigma \times \Gamma = Q \times \Gamma^*$

Σ is set of finite I/P symbols

Γ is stack alphabet

Z_0 is initial top of stack

where $Z_0 \in \Gamma$

$a \to$ push

$b \to$ pop

$L = \{a^n b^n \mid n \geq 1\}$

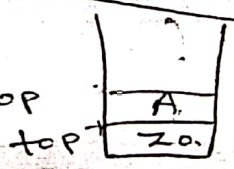
$L = \{ab, aabb, aaabbb, \ldots\}$

Consider, aabb

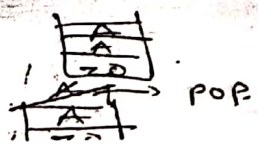
state I/P sym top of stack

$\delta(q_0, a, Z_0) = (q_0 A Z_0)$

state top top-1



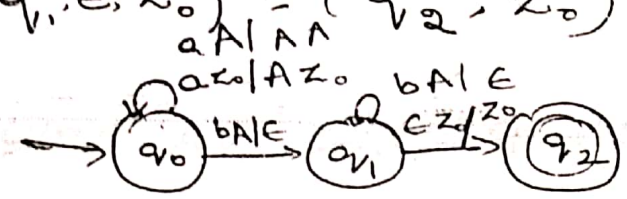
$\delta(q_0, a, A) = (q_0, AA)$



$\delta(q_0, b, A) = (q_1, \epsilon)$

Scanned by CamScanner

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



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

$L = \{a, aab, aaabb, \ldots\}$
aaabb

$\delta(q_0, a, z_0) = (q_0, Az_0)$

$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_0, a, A) = (q_0, AA)$

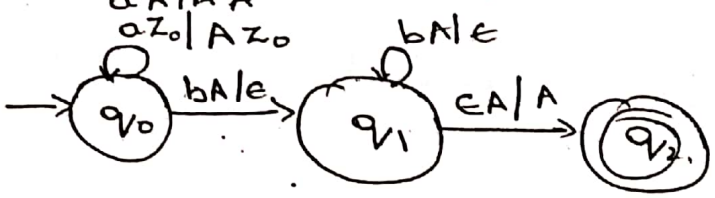
$\delta(q_0, b, A) = (q_1, \epsilon)$

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

$\cancel{\delta(q_1, \epsilon, A) = (q_1, A\epsilon)}$

$\cancel{\delta(q_1, \epsilon, z_0) =}$

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



3) $L = \{a^n b^m \mid n \geq m,\ n, m \geq 0\}$

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

$\delta(q_0, \epsilon, z_0) = (q_2, z_0)$

$\delta(q_0, a, z_0) = (q_0, Az_0)$

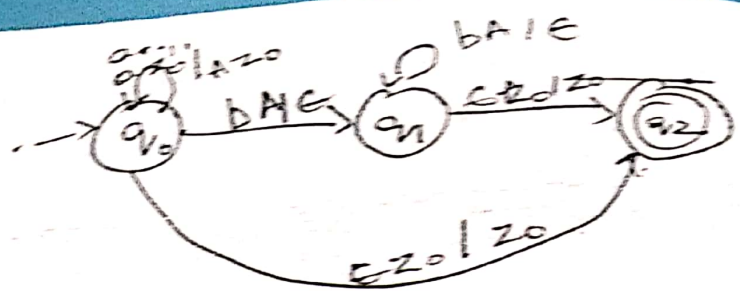
$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_0, b, A) = (q_1, \epsilon)$

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

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

Scanned by CamScanner



PDA

Design PDA for the foll. lang.

4) $L = \{ w \mid n_a(w) = n_b(w) \}$

$= \{ bbaa, aabb, abab, baba, \ldots \}$

b	a
b	a
	Z_0

b	a
b	Z_0

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$

$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_0, b, Z_0) = (q_0, BZ_0)$

$\delta(q_0, b, b) = (q_0, BB)$

$\delta(q_0, a, B) = (q_0, AB)$

$\delta(q_0, b, A) = (q_0, ABA)$

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$

$\delta(q_0, b, Z_0) = (q_0, BZ_0)$

$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_0, b, B) = (q_0, BB)$

$\delta(q_0, a, B) = (q_0, \epsilon)$

$\delta(q_0, b, A) = (q_0, \epsilon)$

$\delta(q_0, \epsilon, Z_0) = (q_f, Z_0)$

5) $L = \{ a^n b^m \mid n \geq m, n, m \geq 0 \}$

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

$\delta(q_0, \epsilon, Z_0) = (q_f, Z_0)$

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$

$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_1, b, A)$ ... ϵ

Scanned by CamScanner

$\delta(q_0, b, A) = (q_1, \epsilon)$

$\cancel{\delta(q_0, \epsilon, Z_0) = (q_1, Z_0)}$

$\delta(q_0, \epsilon, A) = \delta(q_1, A)$

6) $L = \{ wcw^R \mid w \in \{a, b\}^* \}$

$= \{ c, aacaa, bbcbb, abcba, bacab, \ldots \}$

$\delta(q_0, c, Z_0) = (q_2, Z_0)$

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$

$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_0, a, B) = (q_0, AB)$

$\cancel{\delta(q_0, c, A) = (q_0}$

$\delta(q_0, b, Z_0) = (q_0, BZ_0)$

$\delta(q_0, b, B) = (q_0, BB)$

$\delta(q_0, b, A) = (q_0, BA)$

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

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

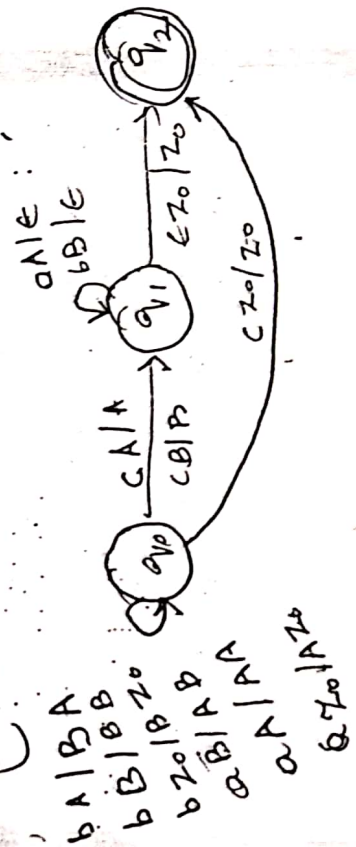
$\delta(q_1, a, A) = (q_1, \epsilon)$

$\cancel{\delta(q_1, a, B) = (q_1, \epsilon)}$

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

$\cancel{\delta(q_1, b, A) = (q_1, \epsilon)}$

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



7) $L = \{ ww^R \mid w \in \{a, b\}^+ \}$

$\{ \epsilon, aa, bb, abba, baab \}$

$\delta(q_0, \epsilon, Z_0) = (q_2, Z_0)$

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$

$\delta(q_0, b, Z_0) = (q_0, BZ_0)$

$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_0, a, B) = (q_0, AB)$

$\delta(q_0, b, A) = (q_0, BA)$

$\cancel{\delta(q_0, \epsilon, A) = (q_1, A)}$

$\cancel{\delta(q_0}$

$\delta(q_0, b, B) = (q_0, \epsilon)$

Scanned by CamScanner

$\delta(q_0, a, A) = (q_0$

$\delta(q_0, \epsilon, Z_0) = (q_1, Z_0)$.

$\delta(q_0, \epsilon, A) = ($

7. $\delta(q_0, \epsilon, Z_0) = (q_2, Z_0)$

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$

$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_0, a, B) = (q_0, AB)$

$\delta(q_0, b, Z_0) = (q_0, BZ_0)$

$\delta(q_0, b, B) = (q_0, BB)$

$\delta(q_0, b, A) = (q_0, BA)$

$\delta(q_0, \epsilon, A) = (q_1, A)$

$\delta(q_0, \epsilon, B) = (q_1, B)$

$\delta(q_1, a, A) = (q_1, \epsilon)$.

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

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

8. $L = \{a^n b^n c^m \mid n, m \geq 1\}$

$L = \{abc, aabbc, aabbcc, aaabbbccc, \ldots\}$

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$

$\delta(q_0, a, A) = (q_0, AA)$

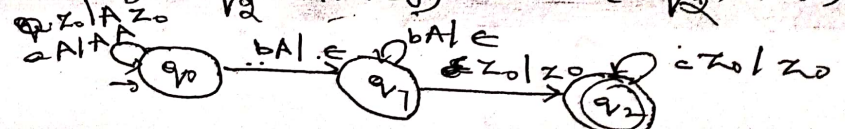
$\delta(q_0, b, A) = (q_1, \epsilon)$

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

$\delta(q_1, c, B) = (q_2, B)$

$\delta(q_1, \epsilon, Z_0) = (q_2, Z_0)$.

$\delta(q_2, c, Z_0) = (q_2, Z_0)$



Scanned by CamScanner

$L = \{ a^m b^n \mid n \geq m,\ n, m \geq 0 \}$

$\delta(q_0, \epsilon, z_0) = (q_2, z_0)$

$\delta(q_0, a, z_0) = (q_0, A z_0)$

$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_0, b, z_0) = (q_2, z_0)$

$\delta(q_0, b, A) = (q_1, \epsilon)$

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

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

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

$\delta(q_2, b, B) = (q_2, B)$

1. $L = \{ a^n b^m b^n \mid n \geq 1,\ m \geq 1 \}$

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

Scanned by CamScanner

Deterministic PDA :

PDA. $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is said to be deterministic if and only if foll. conditions are met.

① $\delta(q, a, X)$ has at most one member for any q in Q & a in Σ or $a = \epsilon$ & X in Γ

② If $\delta(q, a, X)$ is non-empty for some a in Σ then. $\delta(q, \epsilon, X)$ must be empty.

1. Determine whether the given PDA is DPDA or not

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

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

$\delta(q_0, \epsilon, Z_0) = (q_2, Z_2)$

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$

$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_0, b, A) = (q_1, \epsilon)$

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

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

i) All δ fun^n are unique.

ii) $\delta(q_0, a, Z_0)$ is already defined so $\delta(q_0, \epsilon, Z_0)$ ~~need~~ is not to be defined for PDA. be a deterministic PDA. Hence, this grammar is not Deterministic PDA

2. $\{a^n b^n \mid n \geq 1\}$

Scanned by CamScanner

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

$\delta(q_0, a, Z_0) = \delta(q_0, AZ_0)$

$\delta(q_0, a, A) = \delta(q_0, AA)$

$\delta(q_0, b, A) = \delta(q_1, \epsilon)$

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

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

Hence DPDA

3. $L = \{ a^n c^m b^n \mid n \geq 1, m \geq 1 \}$

$\delta(q_0, a, Z_0) = \delta(q_0, AZ_0)$

$\delta(q_0, a, A) = \delta(q_0, AA)$

Scanned by CamScanner

Convert the grammar to PDA

1. $S \to 0AA$ $\qquad$ $0 \& 1 \to T$

$A \to 0S \mid 1S \mid 0$ $\qquad$ $S \& A \to V$.

$\delta(q, \epsilon, S) = (\{q, 0AA\})$

$\delta(q, \epsilon, A) = \{\{q, 0S\}, \{q, 1S\}, \{q, 0\}\}$

$\delta(q, 0, 0) = \{(q, \epsilon)\}$

$\delta(q, 1, 1) = \{(q, \epsilon)\}$

2. $S \to S0S \mid S0S0S1S \mid S \mid S0S0S \mid \epsilon$

$0, 1 \to T$

$S \to V$

$\delta(q, \epsilon, S) = \{\{q, S0S\}, \{q, S0S0S1S\}, \{q, S\}, \{q, S0S0S\}, \{q, \epsilon\}\}$

$\delta(q, 0, 0) = \{\{q, \epsilon\}\}$

$\delta(q, 1, 1) = \{\{q, \epsilon\}\}$

Formal def^n:

Let $G = (V, T, P, S)$, Let G be a context free grammar. construct PDA 'P', that accepts $L(G)$ by empty stack as follows.

$P = (\{q\}, T, V \cup T, \delta, q, S)$ where δ is defined by

i) For each variable A.

$\delta(q, \epsilon, A) = (q, \beta) \mid A \to \beta$ is a production(β) of A

ii) For each terminal 'a'

$\delta(q, a, a) = \{(a, \epsilon)\}$

Scanned by CamScanner

To generate
Whi

a/x.R a/n

2) $E \to E*E \mid E+E \mid (E) \mid I$

$I \to b \mid a \mid Ia \mid Ib$

$E, I \to V$

$(,), +, *, a, b \to T$

$\delta(q, \epsilon, E) = (\{q, E*E\}, (q, E+E), (q, E), \{q, I\})$

$\delta(q, \epsilon, I) = (\{q, b\}, \{q, a\}, \{q, Ia\}, \{q, Ib\})$

$\delta(q, a, a) = \{(q, \epsilon)\}$

$\delta(q, b, b) = \{(q, \epsilon)\}$

$\delta(q, +, +) = \{(q, \epsilon)\}$

$\delta(q, *, *) = \{(q, \epsilon)\}$

$\delta(q, (, (\} = \{(q, \epsilon)\}$

$\delta(q,),)\} = \{(q, \epsilon)\}$

Language of PDA

A string can be accepted by PDA by 2 cases.

i) acceptance by final state
(PDA accepts its I/P by consuming it and entering to accepting states)

ii) acceptance by stack which is empty.
(set of strings that cause PDA to empty its stack).

Accepta

Let P
a
accept
i.e.
L(P
for
stack

Acceptan

Let P
a PDA
accepted
is
L(P) =
for
i.e. L
P can
stack

Derive

E → I
E → E+
E → E*
E → (E
I → a
I → b
I → Ia
I → Ib
I → Ia
I → Ib

Scanned by CamScanner

Acceptance by Final state

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA, then $L(P)$ is a lang. accepted by P by final state

i.e.

$$L(P) = \{ w \mid (q_0, w, Z_0) \overset{*}{\underset{P}{\vdash}} (q_f, \epsilon, \alpha) \}$$

for some state q_f in F & any stack string α.

Acceptance by empty stack.

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA, then $L(P)$ is a lang. accepted by P by empty stack

is

$$L(P) = \{ w \mid (q_0, w, Z_0) \overset{*}{\underset{P_0}{\vdash}} (q, \epsilon, \epsilon) \}$$

for any state q.

i.e $L(P)$ is a set of I/ps w, that P can consume & at the same time stack is empty.

Derive $(a101+b1)*(a1+b)$ using LMD & RMD

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

$E \to E*E$
$E \to (E)*E$
$E \to (E+E)*E$
$E \to (I+E)*E$
$E \to (I1+E)*E$
$E \to (I01+E)*E$
$E \to (I101+E)*E$
$E \to (a101+E)*E$
$E \to (a101+I1)*E$
$E \to (a101+b1)*E$
$E \to (a101+b1)*(E)$

$E \to (a101+b1)*(I1+E)$
$E \to (a101+b1)*(a1+E)$
$E \to (a101+b1)*(a+I)$
$E \to (a101+b1)*(a1+b)$

Scanned by CamScanner

$E \to E * E$

$E \to E * (E)$

$E \to E * (E+E)$

$E \to E * (E + I)$

$E \to E * (E + b)$

$E \to E * (I + b)$

$E \to E * (I1 + b)$

$E \to E * (a1 + b)$

$E \to (E) * (a1 + b)$

$E \to (E+E) * (a1+b)$

$E \to (E + I1) * (a1 + b)$

$E \to (E + b1) * (a1 + b)$

$E \to (I1 + b1) * (a1 + b)$

$E \to (I01 + b1) * (a1 + b)$

$E \to (I101 + b1) * (a1 + b)$

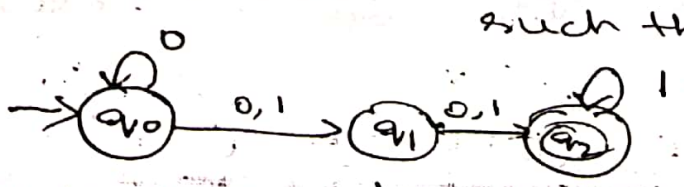
$E \to (a101 + b1) * (a1 + b)$

Define DFA & design DFA to accept set of strings of a's & b's. ends with the string aba & also compute $\hat{\delta}(q_0, ababa)$

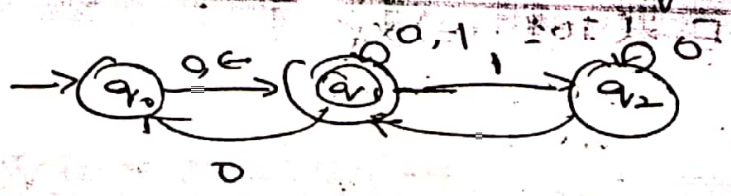
Write subset constru^n algo & convert the foll. finite automata into its equivalent DFA

I/P NFA, $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$

O/P Equivalent DFA, $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$ such that $L(N) = L(D)$



Define ε-NFA & convert foll. finite automata to its equivalent DFA

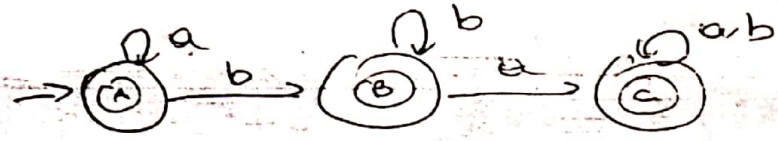


Diff b/w DFA, NFA & ε-NFA

Convert R.E into ε-NFA

Scanned by CamScanner

Convert DFA to RE by state elimin^n



Instantaneous Description (ID)

It is a set of 3 elements (q, w, α) where q is current state, w is remaining I/P, α is stack content

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

Ans: $w = aabb$

$\delta(q_0, a, z_0) = (q_0, A z_0)$

$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_0, b, A) = (q_1, \epsilon)$

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

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

ID

$(q_0, aabb, z_0) \vdash (q_0, abb, A z_0) \vdash$

$(q_0, bb, AA z_0) \vdash (q_1, b, A z_0) \vdash (q_1, \epsilon, z_0)$

$\vdash (q_2, z_0)$ Accepted

$w = aab$

$(q_0, aab, z_0) \vdash (q_0, ab, A z_0) \vdash (q_0, b, AA z_0)$

$\vdash (q_1, \epsilon, A z_0)$. rejected

2) $L = \{a^n b^m, n < m\}$ ~~w =~~

$w = aabbb$

$\delta(q_0, a, z_0) = (q_0, A z_0)$

$\delta(q_0, a, A) = (q_0, AA)$

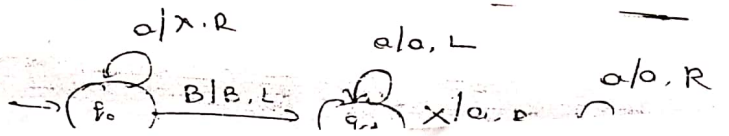
$\delta(q_0, b, A) = (q_1, \epsilon)$

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

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

Scanned by CamScanner

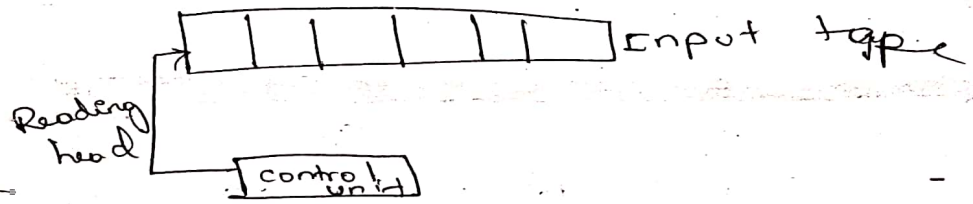
To generate
Wh



$(q_0, aabbb, Z_0) \vdash (q_0, abbb, AZ_0) \vdash$
$(q_0, bbb, AAZ_0) \vdash (q_1, bb, AZ_0) \vdash$
$(q_1, b, AZ_0) \vdash (q_2, Z_0)$

Unit 5

Turing Machine



The reading head moves in both the direction. There is no o/p for this machine

Formal defn: Turing machine is defined by seven tuples

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

where Q is set of finite states

Σ is set of finite I/P symbol

Γ is tape symbols

where Σ is part of Γ.

δ is transition where $\delta: Q \times F \to Q \times F \times \{L, R\}$

q_0 is start state $q_0 \in Q$

B is blank symbol

$B \in \Gamma$

F is set of final states

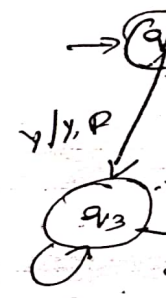
$F \subseteq Q$

Write

Design lang

$L =$

Consider a



ID

Create of a's

$L = \{a$

aabb

aabb B

$\vdash XXq_1$

$\vdash q_2 XXq$

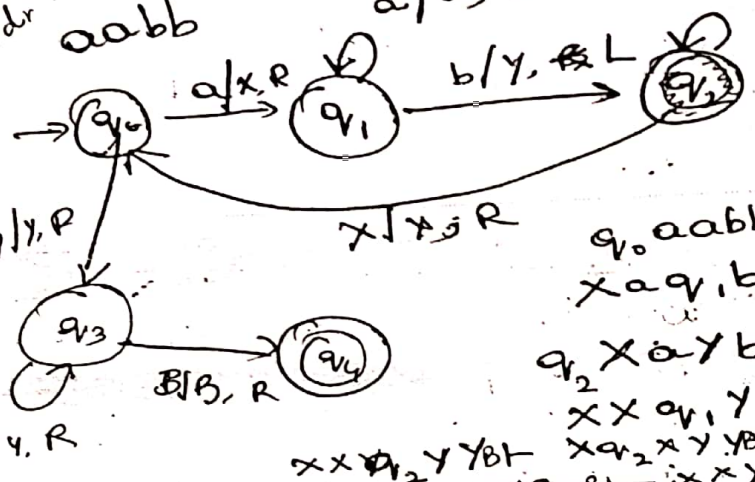
$XXYY$

Write the

Design a turing machine for lang. $L = \{a^n b^n \mid n \geq 1\}$

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

Consider aabb



$a \to x$

$b \to y$

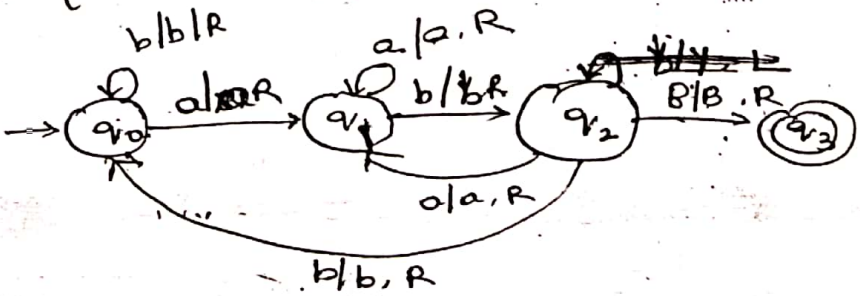
$q_0 aabbB \vdash x q_1 abbB \vdash$
$x a q_1 bbB \vdash x q_2 a y bB \vdash$
$q_2 x a y bB \vdash x q_0 a y bB \vdash$
$x x q_1 y bB \vdash x x y q_1 bB \vdash$
$x x q_1 y y B \vdash x x q_0 y y B \vdash x x y q_3 y B$
$x x q_2 y y B \vdash x x q_2 y y B$ (accepted)
$\vdash x x y y q_3 B \vdash x x y y B$

~~ID — $q_0 aabbb \vdash q_3 xxyy$~~

~~$\vdash x x y y q_3 B \vdash x x y y B \vdash q_4$~~

Create TM to accept set of strings of a's & b's ending with ab.

$L = \{ab, abab, \ldots\}$



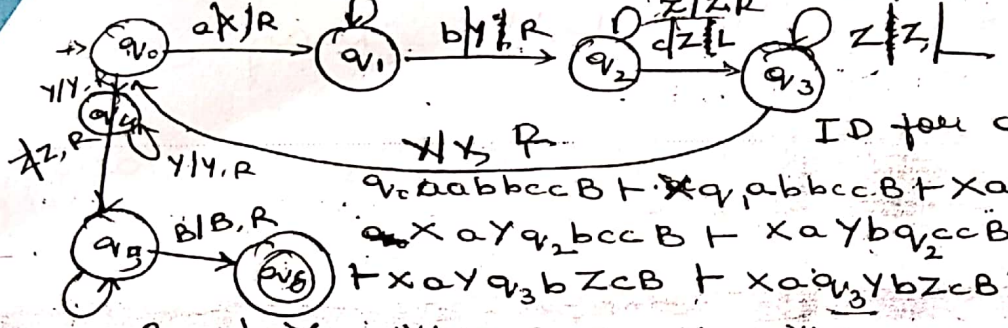
aabb

$aabbB \vdash x q_1 bbB \vdash x q_2 a y bB \vdash q_2 x a y bB$
$\vdash x x q_1 bB \vdash x x q_2 y y B \vdash x x q_2 x y y B \vdash$
$\vdash q_0 x x q_0 y y B \vdash x x y q_3 y B \vdash x x y q_3 B \vdash$
$x x y y B q_4$

Scanned by CamScanner

$L = \{ a^n b^n c^n \mid n \geq 1 \}$

$L = \{ abc, aabbcc, \ldots \}$



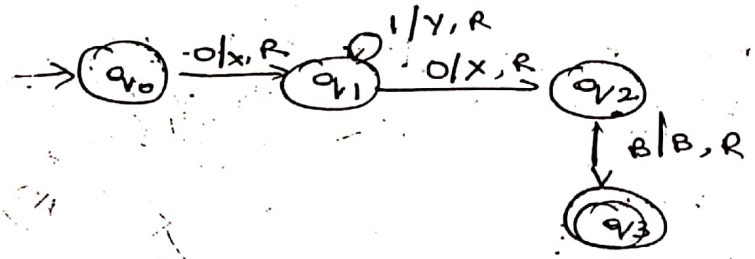
ID for aabbc

$q_0 aabbccB \vdash X q_1 abbccB \vdash Xa q_1 bbccB \vdash$
$XaY q_2 bccB \vdash XaYb q_2 ccB \vdash XaYb q_3 ZcB$
$\vdash XaY q_3 bZcB \vdash Xa q_3 YbZcB \vdash X q_3 aYbZcB$
$\vdash q_3 XaYbZcB \vdash X q_0 aYbZcB \vdash XX q_1 YbZcB$
$\vdash XXY q_1 bZcB \vdash XXY Y q_2 ZcB \vdash XXYYZ q_2 cB$
$\vdash XXYYZ q_3 ZB \vdash XXYY q_3 ZZB \vdash XXY q_3 YZZB \vdash$
$XX q_3 YYZZB \vdash X q_3 XYYZZB \vdash XX q_0 YYZZB$
$\vdash XX q_4 YZZB \vdash XXY Y q_4 ZZB \vdash XXYYZ q_5 ZB \vdash$
$XXYYZZ q_5 B \vdash XXYYZZB$
(accepted)

4) $L = 0 1^* 0$

$L = \{ 00, 010, 0110 \ldots \}$

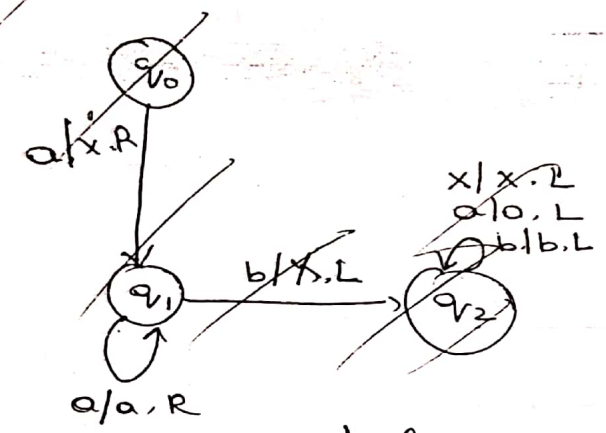
0110!



5]. $L = \{ ww \mid w \in \{a,b\}^* \}$

6]. Even & odd length palindrome

$n_a = n_b$



$q_3 Z_c$

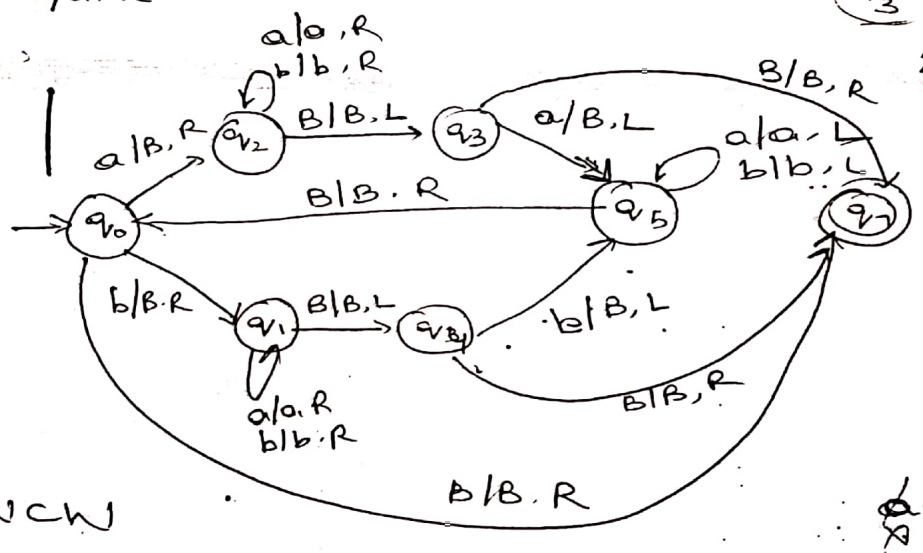
$q_3 bZ$

$q_3 YbZ$

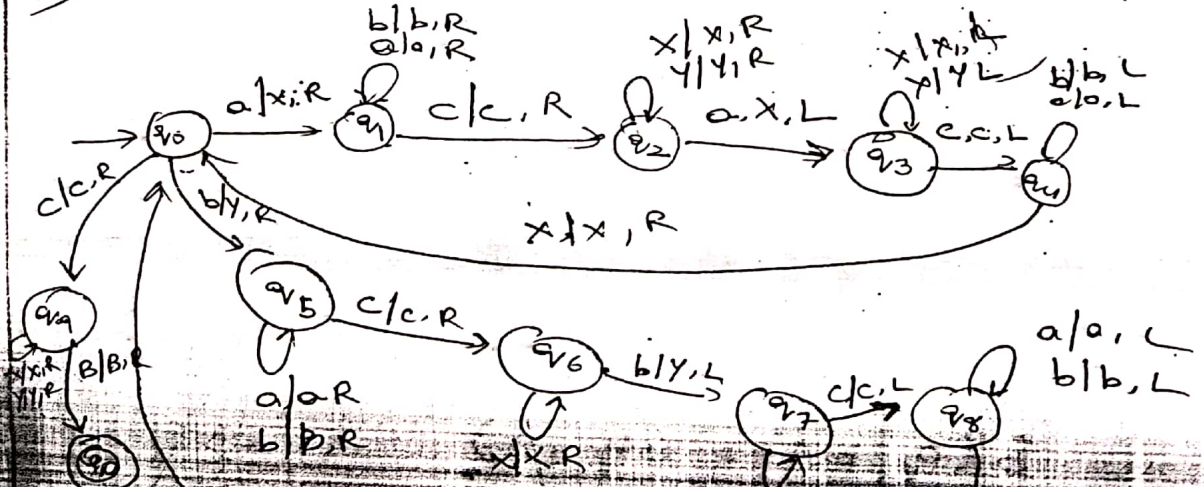
$q_3 aYbZ$

$q_3 XaYbZ$

Xq_0a

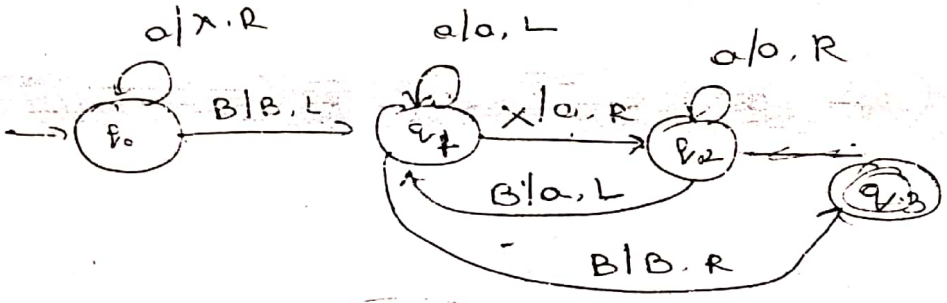


5) WCW



Scanned by CamScanner

To generate
WH



aabbb aabbbbb

$\delta(q_0, a, Z_0) = (q_0, AZ_0)$

$\delta(q_0, a, A) = (q_0, AA)$

$\delta(q_0, b, A) = (q_1, \epsilon)$

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

$\delta(q_1, b, Z_0) = (q_1, BZ_0)$

$\delta(q_1, b, B) = (q_1, BB)$

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

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

$\delta(q_1, \epsilon, B) = (q_2, Z_0)$

aabbbbbb

$\delta(q_0, a, Z_0) = (q_0, AAZ_0)$

$\delta(q_0, a, A) = (q_0, AAA)$

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

$\delta(q_1, b, A) = (q_2, \epsilon)$

$\delta(q_2, b, A) = (q_2, \epsilon)$

$\delta(q_2, \epsilon, Z_0) = (q_3, \epsilon)$

Scanned by CamScanner

Write the applic^n of CFG.

Applications of R.E

State & prove pumping lemma for regular lang.

Consider the foll. lang. defined over $\Sigma = \{0, 1\}$.

$L_1 = \{$Set of all strings containing atleast one zero$\}$

$L_2 = \{$Set of all strings containing atleast one 1$\}$

Draw a DFA to recognize the language $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 - L_2$

Construct a PDA for the lang.

$L = \{a^n b^m c^{n+m} \mid n \geq 1, m \geq 1\}$

Write the context free gramma. for the lang. $L = \{ab(bbaa)^n bba(ba)^n \mid n \geq 0\}$

Construct a turing machine having equal no. of 0's & 1's. ID for 110100

Scanned by CamScanner