Theory of Computation

PAGE NO
DATE: / /

Lecture-1

Types of problems: (4 models)

1. No formal Definition:
   e.g Why are u late? (No well defined ans.)

2. formal definition exist but no sol^n till now
   eg $a^n + b^n = c^n$ $a, b, c > 0$ & $n > 2$
   (also known as Last fermat Theorem).

3. Sol^n exists for small inputs but not sure for large inputs.
   eg All exponential time complexity algos).

4. Sol^n exists for all inputs:
   eg Polynomial time sol^n from any algo.

Models of computation

1. Regular language → DFA / NFA
2. Context free language → DPDA / NPDA
3. Context sensitive language → L.B.A
4. Recursive Enumerable lang. → TM (turing machine)
5. Undecidability
6. Classes of problm: P, NP, NP-complete, NP-hard

Scanned by CamScanner

Language (L): It is sequence of words which are arranged on the basis of rules / grammar.

word (w): sequence of characters / alphabets
↳ set of symbols

Σ Alphabet: set of symbols

$L = \{ w \mid w$ contains at least one a', $\Sigma = \{a, b\} \}$

$= \{a, abab, bba, aab, \ldots\}$ inf

Language = problem

let $x = ab$, $y = ba$

$P = \{a, bb\}$, $q = \{b, ab\}$ $\Sigma = \{a, b\}$

1. Union: $X \cup Y = Z$

$X, Y \in$ String / set

$Z \in$ Set

$x \cup y = \{ab, ba\}$

$x \cup q = \{b, ab\}$

Anything$^0 = 1$

just real no.

2. Concatenation: $X.Y = Z$

$X, Y \in$ Set / String

$Z \in$ Set / String

$x.P = \{aba, abbb\}$

3. Power: X^i $X.X.X \ldots$ i times

$X \in$ Set / String

$i \in$ non-negative integer

$x^2 = xx = abab$

$P^2 = PP = \{a, bb\}\{a, bb\} = \{aa, abb, bba, bbbb\}$

$X^0 =$ Empty String

Empty str: having length 0 $\Rightarrow \wedge \mid \lambda \mid e \mid \in$

length of str: no. of characters in str.

$|w|$

Empty set: $\{\} = \phi$

Null set $= \{\} = \phi$

Empty str.: $\wedge$

Null str.: $\wedge$

Note: set has cardinality & string has length.

Scanned by CamScanner

Kleen Star $X^* = \bigcup_{i=0}^{\infty} X^i$

$X \in$ set/string

if $X = a$

$X^* = \{a^0, a^1, a^2, \ldots\}$

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

$= \{\lambda + a + aa + aaa \ldots\}$

$= \{\lambda \cup a \cup aa \cup aaa \cup \ldots\}$

A OR B = A ∪ B

↓ ↓

A, B, AB, BA either A or B not both

Precedence : $* > . > \cup$

Positive closure: $X^+ = \bigcup_{i=1}^{\infty} X^i$

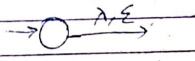
$\Rightarrow a^+ = aa^*$

$a^* = a^+ \cup \lambda$

Lecture-2

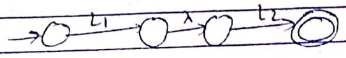
Transition Diagram / finite state Diagram

Directed graph have one initial state & many final states (even 0).

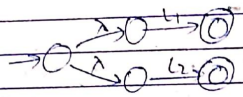


target to final state(s) which start from initial state & ends at final state.

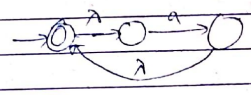
$L = L_1 L_2$



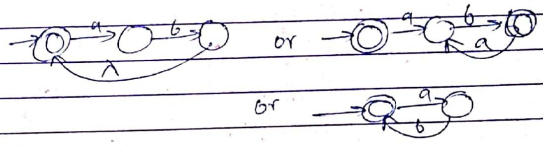
$L = L_1 \cup L_2$



$a^* \Rightarrow$



$(ab)^*$



Scanned by CamScanner

(1) .X.Y

put λ transition from all finals of X to initial state of Y & change all final states of X as non-final states.

(2) X ∪ Y

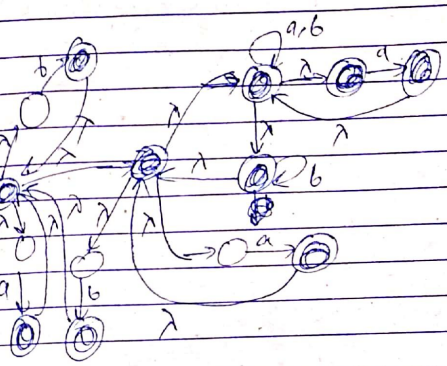
Create a new state & put λ transition from new state to initial state of X & Y. & make new state, to initial state.

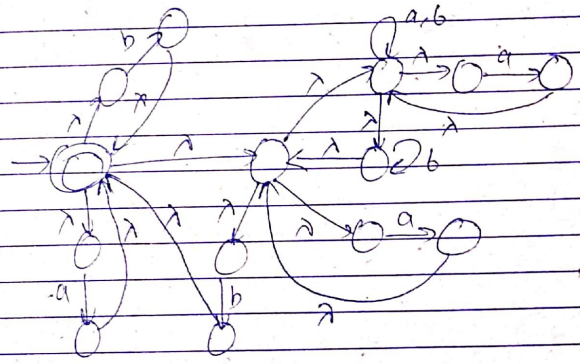
(3) X^*

create a new state & put λ from new state to initial of X & from all final states to new state.

Change all finals to non-finals & set new state as initial & final state.

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



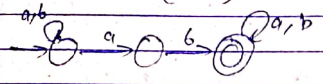


Scanned by CamScanner

Book: Peter linz Chp.: 2.1 Ex

$L = \{ w \mid w \text{ contains ab substring}, \Sigma = \{a,b\} \}$

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

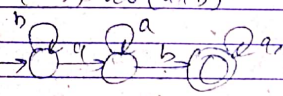


It is non-deterministic diagram.

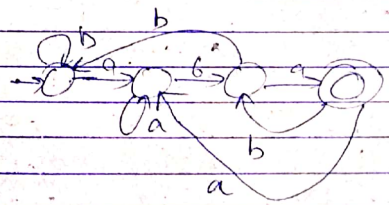
Deterministic Transition Diagram

- from every state there should be exactly one transition for every Σ.

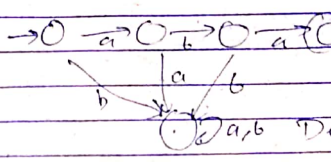
$(a+b)^* ab (a+b)^*$



$L = \{ w \mid w \text{ ends with aba}, \Sigma = \{a,b\} \}$



$L = \{ w \mid w \text{ starts from aba}, \Sigma = \{a,b\} \}$



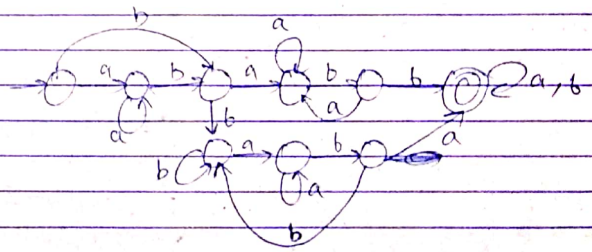
$L = \{ w \mid w \text{ does not contain bbb substr}, \Sigma = \{a,b\} \}$

draw from does & then turn all final → non-final & non-final → final.

$L = \{ w \mid w \text{ contains two a's}, \Sigma = \{a,b\} \}$

at least 2 (by default)
exactly
at most

$L = \{ w \mid w \text{ contains aba \& bb}, \Sigma = \{a,b\} \}$

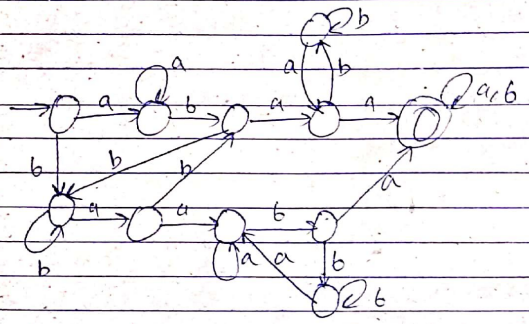


Scanned by CamScanner

Ques $L = \{ w \mid w \text{ contains aba \& baa substr} ; \Sigma = \{a,b\} \}$

Ans. Nao aba, baa — overlapping

small correct str. → abaa



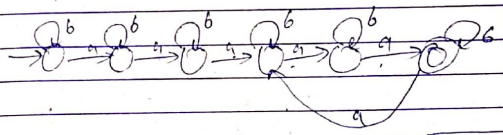
Lecture 32&4

Diff. b/w ∧ & ϕ

$L = \{\wedge\}$ → (q) —a,b→ ((q)) $\Sigma = \{a,b\}$

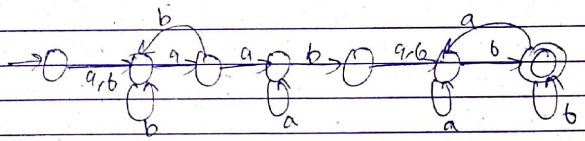
$L = \phi$ → (q) —a,b→

$L = \{ w \mid w \text{ contains } 3K+5 \text{ a's}, K \geq 0, \Sigma = \{a,b\} \}$



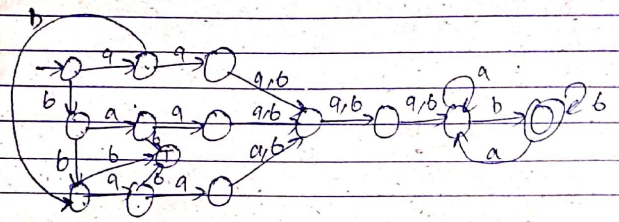
☆ If w contains pK+q a's then min. no of states reqd is $\max(p, q+1)$

$L = \{ w_1 a^2 b w_2 b, \quad \Sigma = \{a,b\}$
$w_1, w_2 \in \{a,b\}^* \}$

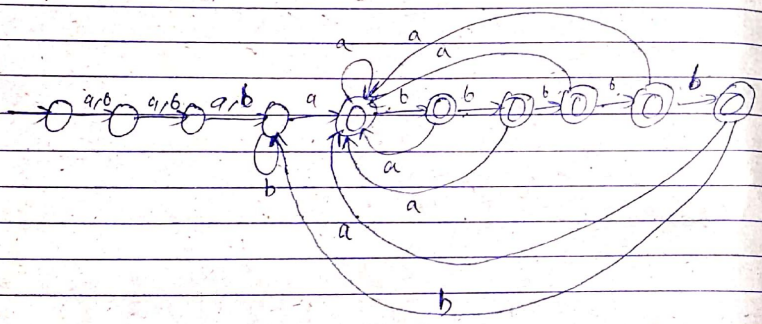


Scanned by CamScanner

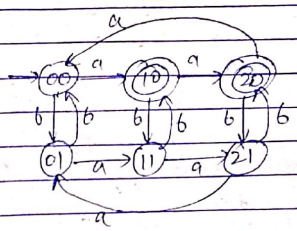
$L = \{ w_1 a^2 w_2 b \;,\; w_1, w_2 \in (a+b)^* \;,\; |w_1| <= 2 \;\&\; |w_2| >= 3 \}$



$L = \{ w_1 a w_2 \mid |w_1| >= 3 \;\&\; |w_2| <= 5 \quad \Sigma = (a+b) \}$



$L = \{ w \mid w \text{ contains } 3k+1 \text{ a's } \& \text{ } 2k \text{ b's} \;,\; \Sigma = (a+b) \}$



Scanned by CamScanner

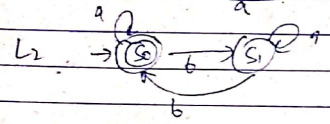
lecture-5

$L = \{w \mid$ no. of a's should be $3K+1$ & $n_b(w) \% 2 = 0$, $\Sigma = \{a,b\}\}$

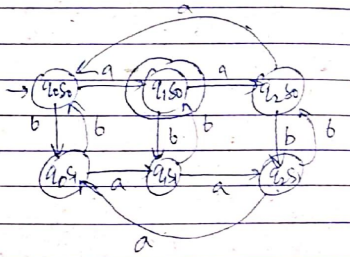
$L_1 = \{w \mid n_a(w) \% 3 = 1, \Sigma = \{a,b\}\}$

$L_2 = \{w \mid n_b(w) \% 2 = 0, \Sigma = \{a,b\}\}$

$L = L_1 \cap L_2$ (combine)



	a	b
$\to q_0 S_0$	$q_1 S_0$	$q_0 S_1$
$q_1 S_0$	$q_2 S_0$	$q_1 S_1$
$q_0 S_1$	$q_1 S_0$	$q_0 S_0$
$q_2 S_0$	$q_0 S_0$	$q_2 S_1$
$q_1 S_1$	$q_2 S_1$	$q_1 S_0$
$q_2 S_1$	$q_0 S_1$	$q_1 S_0$



How to decide final state

$L_1 \cap L_2$: When both final states is available in any state of new diagram.

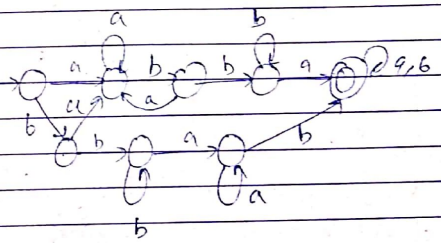
$L_1 \cup L_2$: if any final states is present in new state diagram.

either L_1 or L_2 : $L_1 \cup L_2 - L_1 \cap L_2$

neither L_1 nor L_2 : All $- L_1 \cup L_2$ $\Rightarrow$ $\overline{L_1 \cup L_2}$

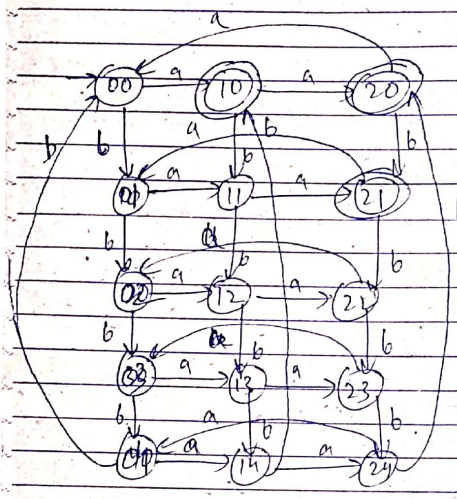
L_1 but not L_2 : $L_1 - L_2$
if state contains final of L_1 but not final of L_2

$L = \{w \mid w$ contains substr. ab & bba, $\Sigma = \{a,b\}\}$



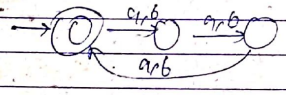
Scanned by CamScanner

$L = \{ w \mid n_a(w) \% 3 \geq n_b(w) \% 5 , \Sigma = \{a, b\} \}$



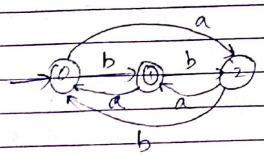
This is not the minimum DFA

$L = \{ w \mid (n_a(w) + n_b(w)) \% 3 = 0 \}$



Easy-peesy ^_^

$L = \{ w \mid (2 \times n_a(w) \% 3 + n_b(w)) \% 3 = 1 \}$



Run: It is a substring of length at least 2 or as many as possible which contains similiar character/alphabet.

eg abababaaaba bbba

Run of a's → 3 length

Run of b's → 2 length

abbabbbbbaaaaab

Size of run → 5

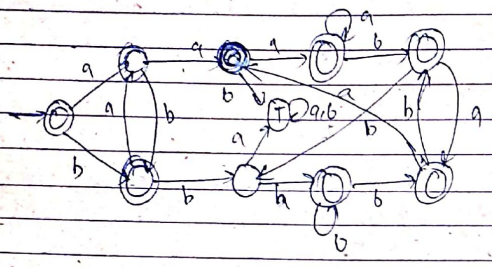
no. of runs of b's of length 2 → 1

abababa

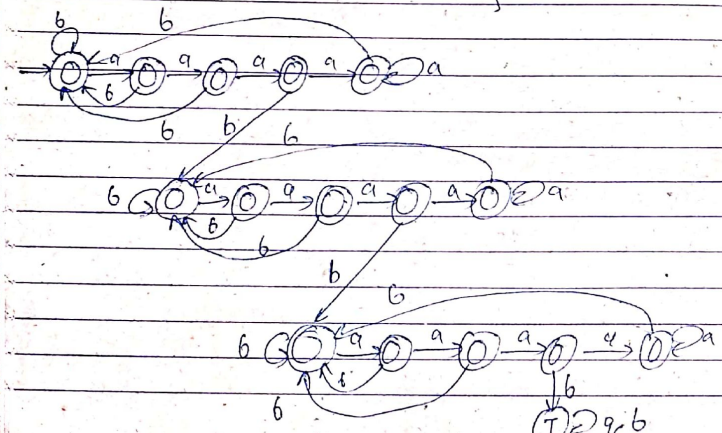
no run

Scanned by CamScanner

$L = \{ w \mid w$ contains runs of length not less than 3, $\Sigma = \{a,b\} \}$



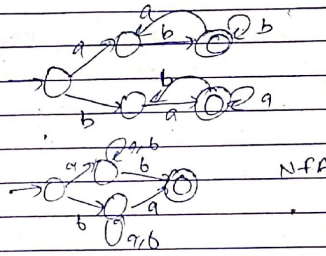
$L = \{ w \mid w$ contains at most 2 runs of length 3 of a's $\Sigma = \{a,b\} \}$



Lecture-6

PAGE NO
DATE: / /

$L = \{ w \mid |w| \geq 2 \quad xwy, \ x \neq y, \ w \in \{a,b\}^* \}$

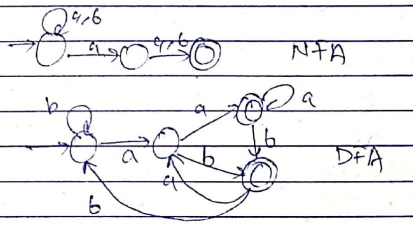


In the ques. is no. of min. states in finite automata. of the above ques :

→ 4 not 5

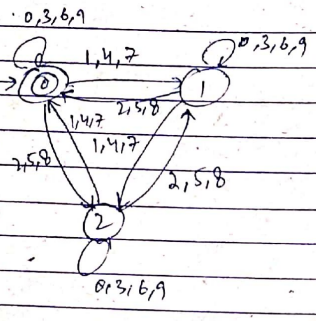
$L = \{ w \mid 2^{nd}$ last char. $= a$, $\Sigma = \{a,b\} \}$

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

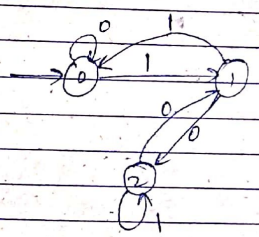


Scanned by CamScanner

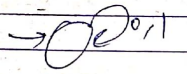
$L = \{ w \mid f(w) \% 3 = 0$, $f(w)$ returns decimal val. of decimal str. w, $\Sigma = \{0,9\} \}$



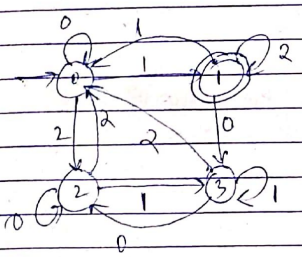
$L = \{ w \mid f(w) \% 3 = 0$, $f(w)$ returns decimal val. of binary str. w, $\Sigma = \{0,1\} \}$



$L = \{ w \mid f(w) \% 5 = 6 \}$



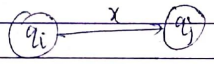
$L = \{ w \mid f(w) \% 4 = 1$, $f(w)$ returns decimal val. of w, where w in base 3 $\Sigma = \{0,1,2\} \}$



$L = \{ w \mid f(w) \% p = 0$, $f(w)$ returns decimal val. of w, w in base b $\Sigma = \{0,1 \ldots b-1\} \}$

Step 1: create p states $q_0, q_1, \ldots q_{p-1}$

Step 2:

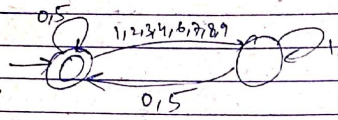


$j = (i \times b + x) \% p$

Step 3: q_0 is initial & q_f is final.

Scanned by CamScanner

$L = \{ w \mid f(w) \% 5 = 0 \}$ $f(w)$ returns decimal val. of decimal str. w, $\Sigma = \{0 \ldots 9\}$ }

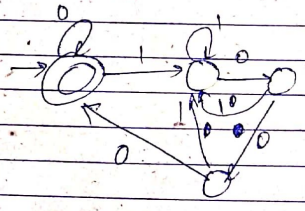


So, min. no. of states are 5

$L = \{ w \mid f(w) \% 8 = 0$, $f(w)$ returns decimal val. of binary str. w, $\Sigma = \{0,1\} \}$

$$\frac{(d_n d_{n-1} \ldots d_2 d_1 d_0)_2}{8} = \frac{d_n \times 2^n + d_{n-1} \times 2^{n-1} + \ldots + d_2 \times 2^2 + d_1 \times 2^1 + d_0 \times 2^0}{8}$$

$$= \frac{d_2 \times 2^2 + d_1 \times 2^1 + d_0 \times 2^0}{8} = \frac{(d_2 d_1 d_0)_2}{8}$$



So, min. no. of states are 4.

$$\frac{(d_n d_{n-1} \ldots d_2 d_1 d_0)_{10}}{3} = \frac{d_n \times 10^n + d_{n-1} \times 10^{n-1} + \ldots + d_1 \times 10^1 + d_0 \times 10^0}{3}$$

$10^x \% 3 \Rightarrow (10 \% 3)^x = 1^x = 1$

So, $\frac{d_n \times 10^n}{3} + \frac{d_{n-1} \times 10^{n-1}}{3} + \ldots + \frac{d_2 \times 10^2}{3} + \frac{d_1 \times 10^1}{3} + \frac{d_0 \times 10^0}{3}$

$$= \frac{d_n + d_{n-1} + d_{n-2} + \ldots + d_1 + d_0}{3}$$

Regular Expression

operators: $*, +, \cup, .$ operands: $\phi, \varepsilon, \lambda$

Any regex may be $*, +, \cup$ & . (concatination).
(RE)

further even something using ( ) that that is RE.

Eg, ab^* ✓ $L = \{w \mid w$ ends with $a, \Sigma = \{a,b\}\}$ ✓

a^2 ✗ $(a+b)^*(a+b)$ ✓

aa ✓

$a^n, n > 0$ ✗

Scanned by CamScanner

Regular language : If there exist ~~other~~ DFA/NFA or RE of any lang. then that lang. is ~~known~~ as Regular language.

~~L = (a+b)*~~ ✓

L = {a²} ✓

~~L = {aⁿbᵐ , n≥2 & m≤3}~~ ✓

aaa*(λ+b+bb+bbb)

$L_1 = \{ a^n b^m \mid |n-m| \% 2 = 0 \}$ ✓

$L_2 = \{ a^n b^m \mid m,n \geq 0 \}$ ✓

$L_3 = \{ a^n b^m \mid n+m \in \text{even no.} \}$ ✓

$L_4 = \{ a^n b^m \mid n \times m \in \text{even no.} \}$ ✓

$L_5 = \{ w \mid$ in every prefix no. of a's should be larger than no. of b's $\Sigma = \{a,b\} \}$

$L_6 = \{ w \mid |n_a(s) - n_b(s)| \leq 2, s \text{ is prefix of } w, \Sigma = \{a,b\} \}$

$L_7 = \{ w w^R \mid \Sigma = \{a,b\} \}$

$L_8 = \{ w(w^R)^* \quad \Sigma = \{a,b\} \}$ ✓

$L_9 = \{ w(ww)^* \quad w \in \{a,b\}^* \}$ ✓

$L_{10} = \{ w w^* \quad w \in \{a,b\}^* \}$ ✓

Scanned by CamScanner

lecture-7

Transition Diagram / Finite Automata

finite Automata → automatic / less interaction with human

machine should contain finite states / finite inputs.

$M = (Q, \Sigma, \delta, S, f)$

Q: set of states $\quad \delta(\ldots) \to 1$ | 1

Σ: set of input symbols

δ: transition f^n $\quad \delta(Q, \Sigma^*)$

$\delta(Q, \{\lambda \cup \Sigma\}) \to 2^Q$ (any subset of total states)

$\to X, \quad X \subseteq Q$

S: initial state $\quad S \in Q$

F: set of final states $\quad f \subseteq Q$

FA

DFA $\quad$ NFA / FA

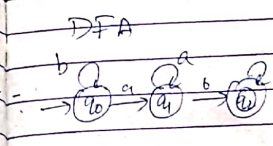
for DFA

δ: $\delta(Q, \Sigma) \to Q$

$L = \{ w \mid w$ contain ab substr, $\Sigma = \{a, b\} \}$

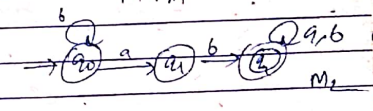
$M = (Q, \Sigma, \delta, S, f)$

DFA



M_1

NFA



M_2

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

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

$S = q_0$

$f = \{q_2\}$

δ: $\delta(q_0, a) \to q_1$

$\delta(q_0, b) \to q_0$

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

$\delta(q_1, b) \to q_2$

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

$\delta(q_2, b) \to q_2$

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

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

$S = q_0$

$f = \{q_2\}$

δ: $\delta(q_0, a) \to \{q_0, q_1\}$

$\delta(q_0, b) \to \{q_0\}$

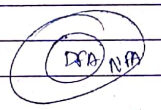
$\delta(q_1, a) \to \phi$

$\delta(q_1, b) \to \{q_2\}$

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

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

all DFA are NFA



in context of transition f^n

Scanned by CamScanner

Acceptance of str. in DFA & NFA.

Let w is input str.

DFA	NFA
if $\delta^*(s,w) \in f$ then w is acceptable otherwise not	if any one member of $\delta^*(s,w) \in f$ then w is acceptable otherwise

w = aaba in M_1

$\delta^*(q_0, aaba) \to \delta^*(q_1, aba) \to \delta^*(q_1, ba) \to \delta^*(q_2, a) \to (q_2)$

w = bba

$\delta^*(q_0, bba) \to \delta^*(q_0, ba) \to \delta^*(q_0, a) \to (q_1)$ ✗

w = aaba in M_2

$\delta^*(q_0, aaba) \Rightarrow \delta^*(\{q_0,q_1\}, aba) \to \delta^*(\{q_0,q_1\}, ba)$

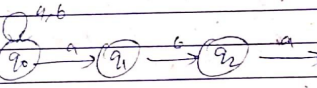
$\to \delta^*(\{q_0,q_1\}, a) \to \{q_0, q_1, (q_2)\}$ ✓

$\delta^*(q_0, bba) \to \delta^*(\{q_0\}, ba) \to \delta^*(\{q_0\}, a) \to \{q_0,q_1\}$ ✗

DFA = NFA in context of language

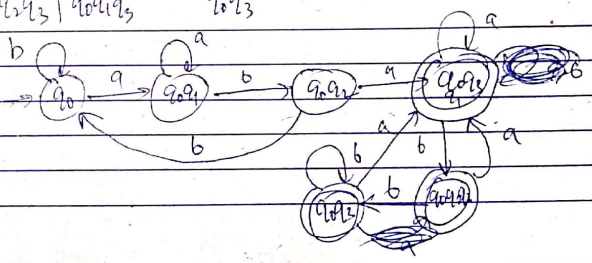
Conversion from NFA → DFA

$L = \{ w \mid w \text{ contains aba substr.}, \Sigma = \{a,b\} \}$

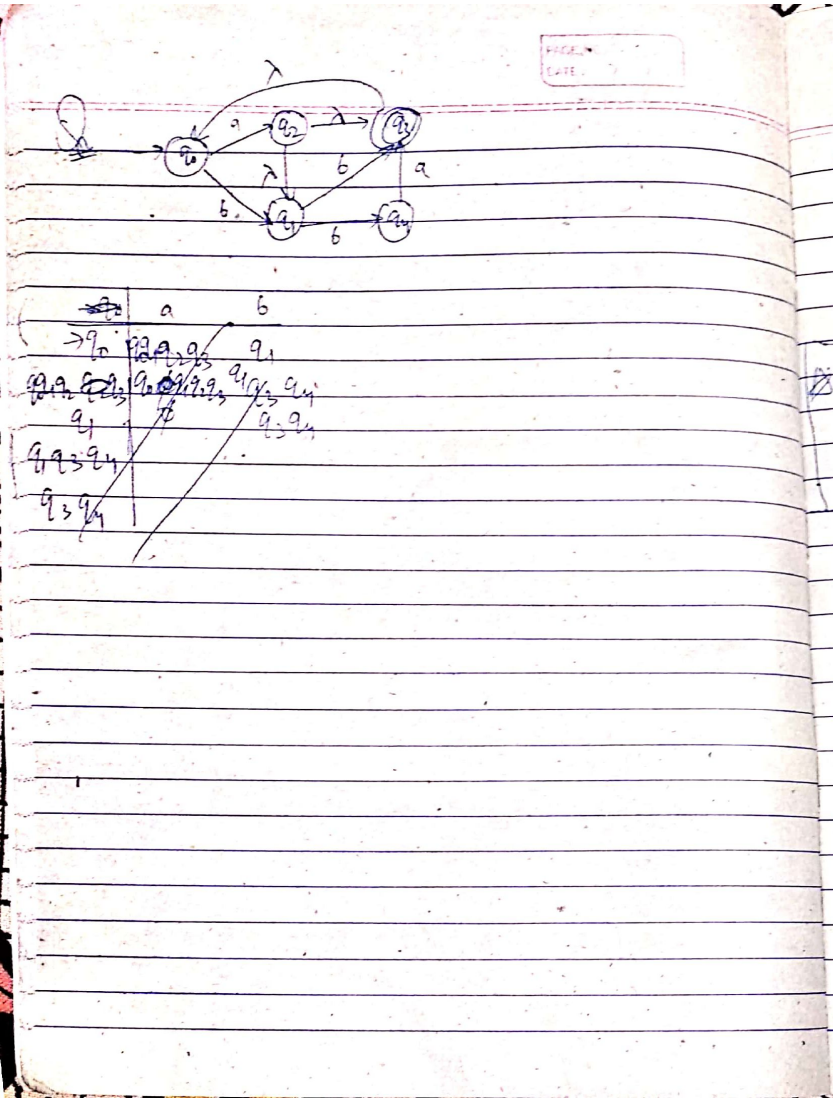


	a	b
$\to q_0$	q_0q_1	q_0
q_0q_1	q_0q_1	q_0q_2
q_0q_2	q_0q_3	q_0
q_0q_3	q_0q_3	q_0q_3
$q_0q_2q_3$	$q_0q_1q_3$	q_0q_3

q_0q_3	$q_0q_1q_3$	q_0q_3

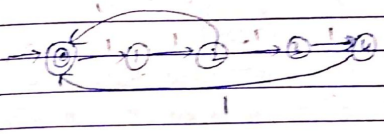


Scanned by CamScanner



	a	b
$\to q_0$	q_2	q_1
q_1	ϕ	$q_3 q_4$
q_2	q_0	q_4
q_3	q_4	$q_3 q_4$
q_4	ϕ	$q_3 q_4$

$L = (11 + 111)^*$



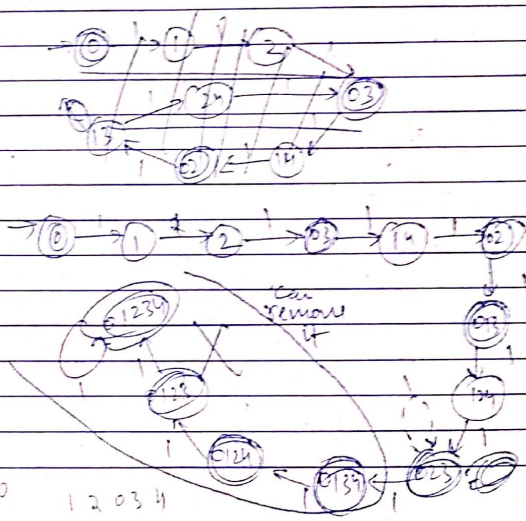
NFA

Q. If there are n states in NFA then what is the maximum no. of states in equivalent DFA?

At most $\to 2^n$

At least $\to 1$

	1
$\to 0$	1
1	2
2	0 3
03	14
14	02
02	013
013	124
124	023
023	0134
0134	0124
0124	0123
0123	01234
01234	01234



Scanned by CamScanner

$(111 + 11111)^*$

0, 3, 5, 6, 8, 9, 10, 11, 12 . . . .

all are acceptable

So, min. no. of states reqd are 9.

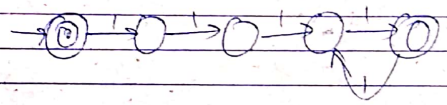
(ii) $(11 + 111)^*$

0, 2, 3, 4, 5 . . .

min. no. of states 3.

(iii) $(111 + 11111)^*$

0, 4, 6, 8, 10, 12, 14, . . .



min no. of states 5.

No algo to minimise NFA

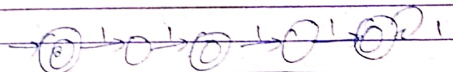
min. no. of states in

~~NFA~~ → 4

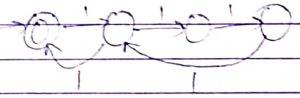
DFA → 5

for $L = (11 + 11111)^*$

0, 2, 4, 5, 6, 7 . . . .



~~DFA~~ min 5 in DFA



min. 4 in NFA

Scanned by CamScanner

lecture-6

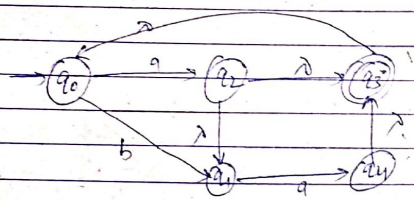
λ-NFA → NFA conversion

Step 1: find closure of all states using all ε + λ

closure (q_i, x) → it contains all that states which is found after accepting x, $x \in \Sigma$ or λ

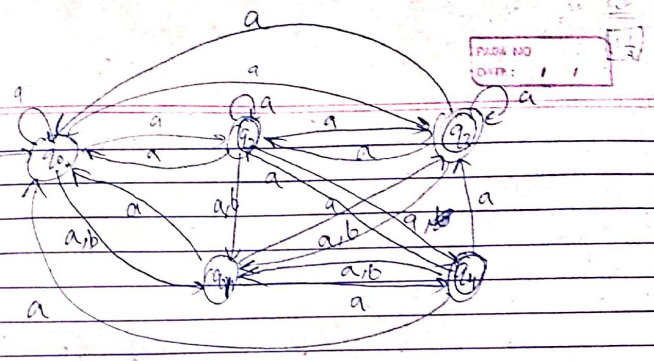
Step 2: create a new NFA according to closure table

Step 3: q_i is final if q_i can reach to final in original dia. using λ acceptance.



Closure Table

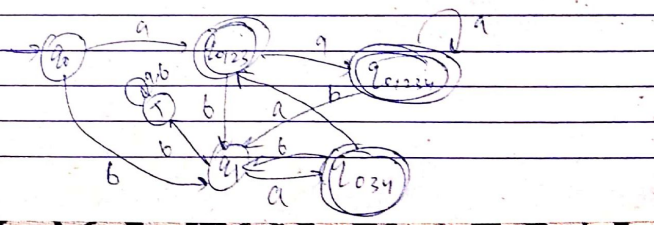
	a	b	λ
q_0	$\{q_2 q_1 q_3 q_0\}$	$\{q_1\}$	$\{q_0\}$
q_1	$\{q_4 q_3 q_0\}$	ϕ	q_1
q_2	$\{q_2 q_3 q_0 q_3 q_4\}$	$\{q_1\}$	$\{q_2, q_3, q_0 q_4\}$
q_3	$q_2 q_3 q_0 q_1$	q_1	$\{q_3, q_0\}$
q_4	$q_2 q_3 q_0 q_1$	q_1	$\{q_4, q_3, q_0\}$



Convert it to DFA

- using closure table

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



Scanned by CamScanner

RE → NFA → DFA

DFA minimisation

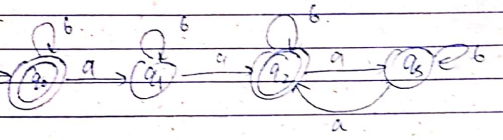
Methods :

1. merging of equivalence states
2. grouping of states

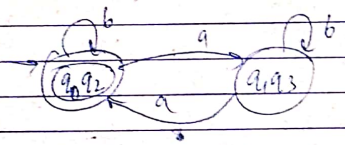
→ If $q_i \approx q_j$ then merge into single state

$q_i \approx q_j$ if all strings $w \in \Sigma^*$,

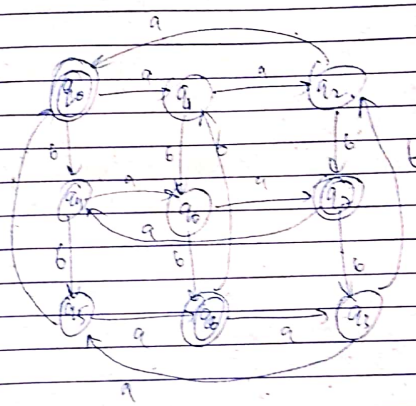
$\delta^*(q_i, w)$ & $\delta^*(q_j, w)$ are either both final or both non-final



q_1	X		
q_2	✓	X	
q_3	X	✓	X
	q_0	q_1	q_2



$q_i \approx q_j$ & $q_j \approx q_k \Rightarrow q_i \approx q_k$



$q_0q_2q_5$ $q_1 \rightarrow q_3q_6$ $q_4q_7q_5$

q_1							
q_2							
q_3	X	✓	X	✓			
q_4	X	X	✓	X	✓		
q_5	X	X	✓	X	X	✓	
q_6	✓	X	X	X	X	X	X
q_7	✓	X	X	X	X	X	X
	q_0	q_1	q_2	q_3	q_4	q_5	q_6

Scanned by CamScanner

RE → NFA → DFA

DFA minimisation

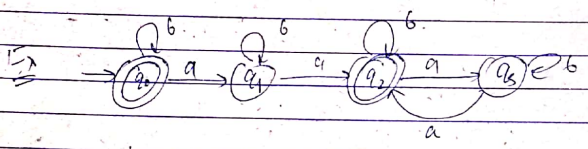
Methods:

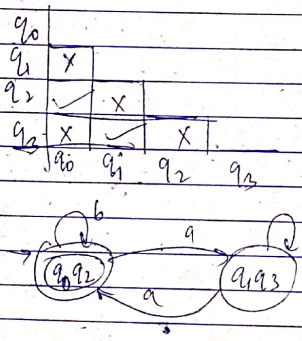
1. merging of equivalence states
2. grouping of states

→ if $q_i \simeq q_j$ then merge into single state

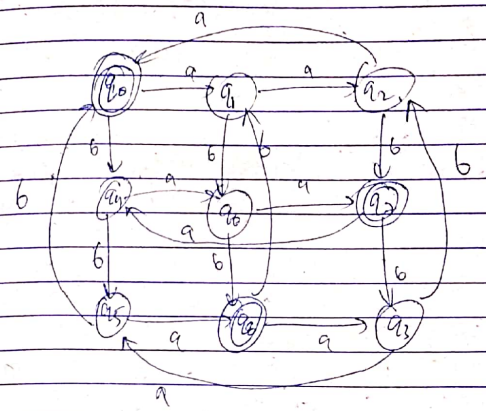
$q_i \simeq q_j$ if all strings $w \in \Sigma^*$,

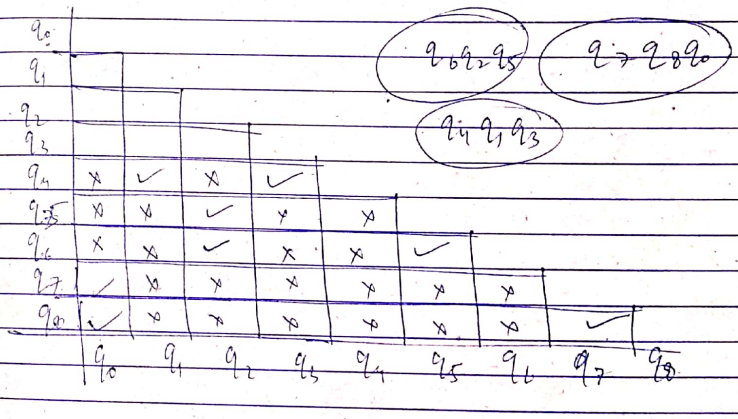
$\delta^*(q_i, w)$ & $\delta^*(q_j, w)$ ∈ either f or ~~δ~~ f (final)





$q_i \simeq q_j$ & $q_j \simeq q_k \Rightarrow q_i \simeq q_k$

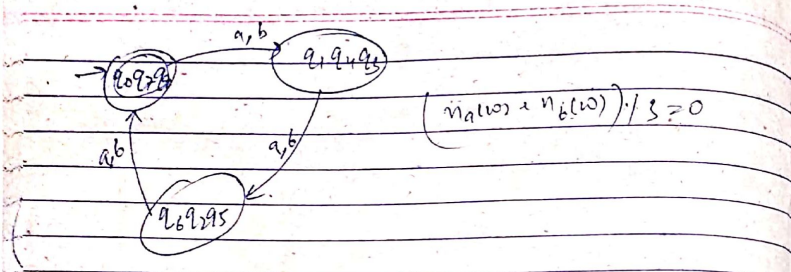




$q_0 q_6 q_5$ $q_1 \to q_0 q_0$

$q_4 q_1 q_3$

Scanned by CamScanner

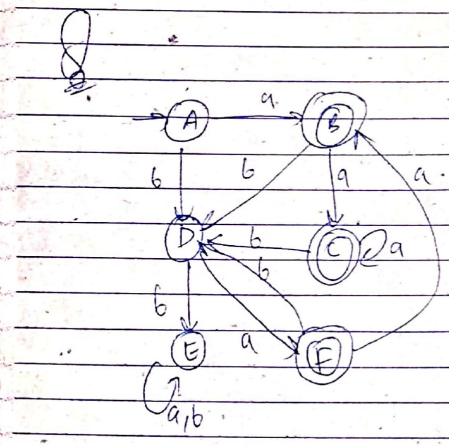


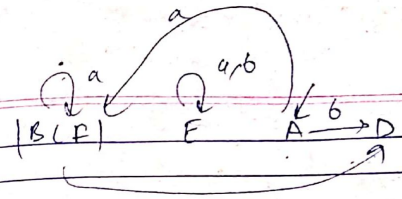
$(n_a(w) + n_b(w)) \% 3 = 0$

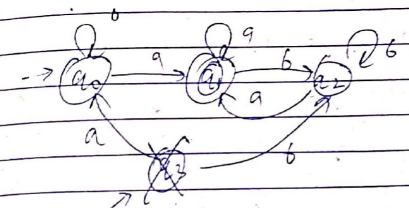
2. grouping of states

same written in class

make 2 groups of final & non-final states





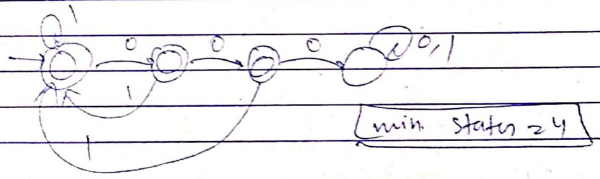


if any state isn't reachable from q_0 then remove it

Input - Output

type of given L × DFA — min. no. of states

Q $L = \{ w \mid$ after every 00 there should be at least one 1, $\Sigma = \{0,1\} \}$

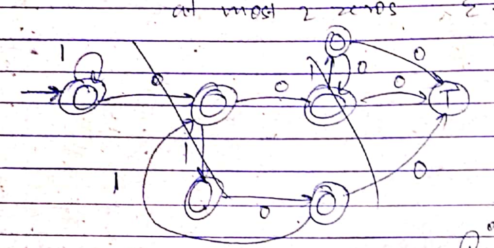


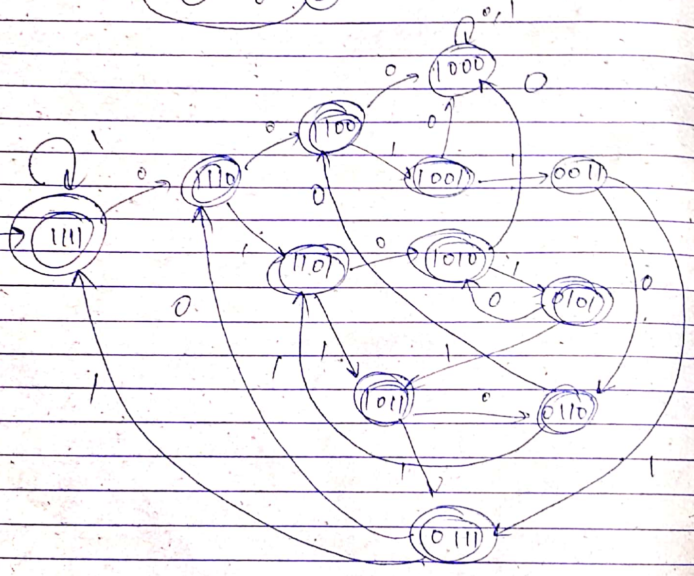
min states = 4

Scanned by CamScanner

0110 1000 0000 n.K 0101 0010

L = { w | every substr. of length 4 contains at most 2 zeros, Σ = {0,1} }





lecture-9

$L_1 - L_2$

Closure Operation over Regular Language

let S is a set

$x, y \in S$

if $x * y \in S$ then

$*$ is closed operation over set S.

Qn I is integer Set

$x, y \in I$

$x \cdot y \in I$ ✓ (closed op.)

$x / y \in I$ ✗ (not-closed)

let L_1 & L_2 are Regular Languages

1. Union : $L_1 \cup L_2$ (it is closed)
2. Concatination : $L_1 . L_2$ (it is closed)
3. Kleen Star : L_1^* (it is closed)
4. Compliment : $\overline{L_1}$ (it is closed)

   Note: design only DFA then change final to non-final & non-final to final.
5. $L_1 - L_2$:
6. $L_1 \oplus L_2$: ✓

⑦ Reverse $(L_1) = \{ w^R \mid w \in L_1 \}$ ✓

~~(reverse the direction of edges)~~.

$L_1 = \{ a^n ; n > 0 \}$ regular

$L_2 = \{ b^n ; n > 0 \}$ regular

$L_3 = L_1 . L_2 \Rightarrow a^n b^m$ regular

✗ $L_4 = \{ a^n b^n ; n > 0 \}$ not-regular

$L_5 = \{ w w^R ; w \in (a,b)^* \}$ ✗

$L_6 = L_1 L_1^R \Rightarrow \{ w w^R ; w, w, \in (a,b)^* \}$ ✓

⑧ Truncate (L_1) : $\{ w \mid wx \in L_1 , x \in \Sigma \}$ ✓

Remove last (L_1) :

Let $M = (Q, \Sigma, \delta, S, f)$

Truncate $(M) = (Q, \Sigma, \delta, S, f')$

$q_j \in f'$ if $\delta(q_j, \varepsilon) \in f$

⑨ Prefix (L_1) : $\{ w \mid wx \in L_1 , x \in \Sigma\Sigma^* \}$ ✓

Prefix $(M) = (Q, \Sigma, \delta, S, f')$

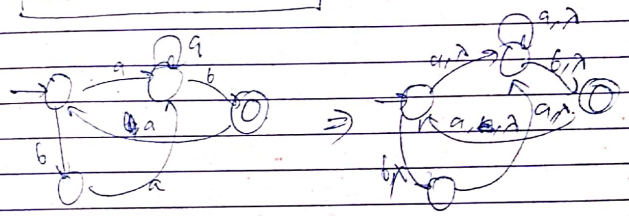
$q_j \in f'$ if $\delta(q_j, \Sigma^*) \in f$

⑩ Suffix (L_1) : $\{ w \mid xw \in L_1 , x \in \Sigma^* \}$ ✓

Create new state, put λ transition from ~~start~~ new state to all states (except dead state) & made new state to ~~start~~ start state.

⑪ Subsequence $(L_1) = \{ x \mid x$ is subsequence of w where $w \in L_1 \}$ ✓

Take it or Leave it



put λ transition on all edges

Scanned by CamScanner

Homework

(11) Exchange $(L_1) = \{ xwy \mid ywx \in L_1, x,y \in \Sigma \}$

(12) Even $(L_1) = \{ w_1 w_2 \ldots w_{2n} \mid w_1 w_2 \ldots w_{2n} \in L_1 \}$

(13) Odd (L_1)

(14) Half $(L_1) = \{ w_1 w_2 \ldots w_{n/2} \mid w_1 w_2 \ldots w_n \in L \}$

(15) Double $(L_1) = \{ ww \mid w \in L_1 \}$ ✗

(16) RDouble $(L_1) = \{ ww^R \mid w \in L_1 \}$ ✓

lecture-18

Regular language

Context free language

Pumping Lemma — used to prove lang. is not regular. like $a^n b^n$

disadvantage

Pumping lemma may also prove non-regular lang. to be regular.

Pumping Lemma

Step-1 : Assume L is regular & there exists DFA M having K states.

Step-2 : let $w \in L$, $|w| \geq K-1$, w visits all states of DFA M.

[If a language has finite no. of strings (of finite language), then the language is regular.]

let $w = xyz$

let y is part of loop $|y| \geq 1$

$|xy| \leq r$

or

$|yz| \leq r$

Scanned by CamScanner

Step3: $xy^iz \in L$ then anyhow show that $xy^iz \notin L$ due to this contradiction L isn't regular

Sum Prove $L = \{a^n b^n ; n>0\}$ isn't regular

Step1:

Step2:

Step3: $xy^iz \in L$

Case1: $y \in a^+$ then z is b^+ when i increases then no. of a's increases & no. of b's fixed. So $xy^ib \notin L$

Case2: $y \in b^+$ then

like case 1:

$xy^iz \notin L$

Case3: $y \in (ab)^+$ then

abab... substr. is not pair of language $xy^iz \notin L$

So, language isn't regular.

$\therefore L = \{a^n b^n ; n>0\}$

$L_2 = \{a^p ; p \in$ prim no. $\}$

$L_3 = \{a^{n^2} ; n>0\}$

$L_4 = \{ww \mid w \in \{a,b\}^+\}$

$L_5 = \{a^{n^2} ; n>0\}$

$xy^iz \in L$

let $|y| = m$

$|xyz| = r$, $|xy^iz| = |xyz| + |y^{i-1}|$

$= r + (i-1)m$

$r+(i-1)m \in$ Prin

$\therefore \forall i$ let $i = r+1$

$r+(i-1)m = r(1+m) \notin$ Prin

So it isn't regular

$xy^iz \in L$

$xyz \in L \Rightarrow |xyz| = 2^u$

let $|y| = m$

$|x y^i z| = 2^u + (i-1)m = 2^{u+v}$

$\forall i \; |xy^iz \notin L|$

Scanned by CamScanner

$L_6 = \{ w \mid n_a(w) = n_b(w), \Sigma = \{a, b\} \}$

Case 3: $y \in a^+$
$y \in b^+$
$y \in (ab)^+$
$y \in (ba)^+$

$L = \{ a^n b^n ; n \leq 100 \}$

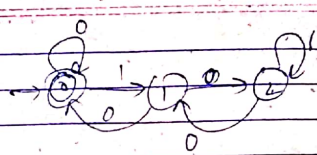
it is regular because it is finite

Ques Write a program in C to identify whether a given string S is divisible by 3 or not where S in binary form & length of string is unknown.

```
r=0, i=0;
while (s[i] != '\0')
{
  r = (r*2 + s[i] - '0') % 3;
  i++
}
return cout << r << endl;
```

OR

```
i=0, r=0;
while (s[i] != '\0')
{
  if (s[i] == '0')
    if (r==1) r=2;
    else if (r==2) r=1;
  else
    if (r==0) r=1;
    else if (r==1) r=0;
  i++
}
cout << r << endl;
```



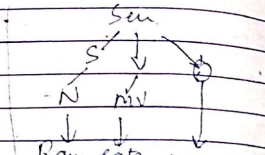
Scanned by CamScanner

lecture-11

Context Free Language / Grammer

Grammer: set of rules

Sentence → S + V + O
Sen → Noun
Noun → Ram | Mohan | - -
V → AV | mV
mV → eat | jump | - -
O → mango. - -



α → β
α = variable
β ⇒ (variable ∪ terminal)*

Formal Definition of Grammer

G = (V, T, P, S)

V: set of variables
T: set of terminals / alphabets
P: set of production rules α → β
α, β ∈ |V, T|* & α should contain at least 1 var.
S: starting symbol S ∈ V

lang. is generated by grammer

Chomsky classification

4 types of grammer based on production rules

Type 0: Unrestricted grammer / Recursive Enumerable Grammer / Language

α → β α, β ∈ |V, T|*, α should contain at least one variable

Type 1: Context sensitive grammer

α → β , α, β ∈ |V, T|* , α should contain at least one var. & |α| ≤ |β|
⇒ β can't empty

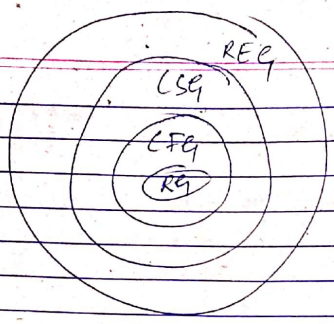
Type 2: Context free grammer

α → β α ∈ V , β ∈ |V, T|*

Type 3: Regular Grammer

α → β : α ∈ V , β either TV | T | λ
or
VT | T | λ

Scanned by CamScanner



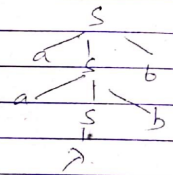
Context free Grammer

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

$S \to aSb \mid \lambda$

∵ we can choose any prod. rule it is called context free

aabb



Parse Tree

or

$S \to XY \mid \lambda$

$X \to AS$

$Y \to b$

$A \to a$

Q. $L = \{ a^{2n} b^{2n} ; n \geq 0 \}$

$S \to aaSbb \mid \lambda$

or

$S \to aXb \mid \lambda$

$X \to aSb$

Q. $L = \{ a^n b^m ; n > m \}$

$S \to aSX \mid a$

$X \to b \mid \lambda$

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

or $S \to aSb \mid aX$

$X \to aX \mid \lambda$

$S \to ASb \mid a$

$A \to aA \mid \lambda$

X ∵ we have to accept at least 1 b.

Q. $L = \{ a^n b^m ; n \neq m \}$

$\to n < m$

$n > m$

$S_1 = aS_1b \mid S_1b \mid b$

$S_2 = aS_2b \mid S_2a \mid a$

$S = S_1 \mid S_2$

Scanned by CamScanner

or

$S = aSb \mid aX \mid bY$

$X \to aX \mid \lambda$

$Y \to bY \mid b\lambda$

or

$S \to AX \mid XB$

$X \to aXb \mid \lambda$

$A \to aA \mid a$

$B \to bB \mid b$

$L = \{ a^n b^m ; n \neq m+2 \}$

$L = \{ a^n b^m ; n \leq m+3 \}$

n	m
0,1,2,3	0
0,1,2,3	1

$S \to \lambda \mid a \mid aa \mid aaa \mid aSb \mid Sb$

$L_1 = \{ a^n b^m ; n \leq m+1 \}$

$S_1 \to \lambda \mid a \mid aS_1b \mid S_1b$

$L_2 = \{ a^n b^m ; n > m+2 \}$

$S_2 \to aaaX \mid aaa$ or $S_2 \to aaa \mid aS_2b \mid aS_2$

$X \to aXb \mid aX \mid \lambda$

$L = \{ a^n b^m ; n \neq m+2 \}$

$S \to S_1 \mid S_2$

$S_1 \to \lambda \mid a \mid aS_1b \mid S_1b$

$S_2 \to aaa \mid aS_2b \mid aS_2$

$L = \{ a^n b^m ; n = m \text{ or } 2n = m \}$

$S \to S_1 \mid S_2$

$S_1 \to aS_1b \mid \lambda$

$S_2 \to aS_2bb \mid \lambda$

$L = \{ a^n b^m ; n \leq m \leq 2n \}$

$S \to aSb \mid aSbb \mid \lambda$

or

$S \to aSbB \mid \lambda$

$B \to b \mid \lambda$

Scanned by CamScanner

$L_1 = \{ a^n b^m c^p ;\ n=m ;\ n,m,p \geq 0 \} \Rightarrow$

$S = S_1 S_2$
$S_1 \to a S_1 b \mid \lambda$
$S_2 \to c S_2 \mid \lambda$

$L_2 = \{ a^n b^m c^p ;\ n = p \ldots \} \Rightarrow$

$S = aSc \mid B$
$B \to bB \mid \lambda$

$L_3 = \{ a^n b^m c^p ;\ n = m+p \}$

$L_4 = \{ a^n b^m c^p ;\ m = n+p \}$

$L_5 = \{ a^n b^m c^p ;\ 2m = n+p \}$

$L_6 = \{ a^n b^m c^p ;\ m = 3n+2p \} \Rightarrow$

$a^n b^{3n+2p} c^p$
$a^n b^{3n} \cdot b^{2p} c^p$
$S \to S_1 S_2$
$S_1 \to a S_1 bbb \mid \lambda$
$S_2 \to bbS_2 c \mid \lambda$

$L_7 = \{ a^n b^m c^p ;\ 2n = 3m+2p \}$

$L_8 = \{ a^n b^m c^p d^q ;\ n+m = p+q \}$

(3) $\because n = m+p$

$a^{m+p} b^m c^p \Rightarrow a^p [a^m b^m] c^p$

$S \to aSc \mid X$
$X \to aXb \mid \lambda$

(4) $m = n+p$

$a^n b^{n+p} c^p \Rightarrow a^n b^n \; b^p c^p$

$S \to S_1 S_2$
$S_1 \to a S_1 b \mid \lambda$
$S_2 \to b S_2 c \mid \lambda$

$L = \{ a^n b^n ;\ 3n = 2m \}$

$S \to aaSbbb \mid \lambda$

(5) $2m = n+p$

$a^{2n} b^n . b^p c^{2p}$

$S \to S_1 S_2$
$S_1 \to aaS_1 b \mid \lambda$
$S_2 \to bS_2 cc \mid \lambda$

for $n+p$ = odd

$\Rightarrow S \to S_1 S_2 \mid aS_1 b S_2 c$
$S_1 \to aaS_1 b \mid \lambda$
$S_2 \to bS_2 cc \mid \lambda$

(6) $L_9 = \{ a^n b^m c^p ;\ 5n = 2m+p \}$

$L = \{ a^n b^m c^p ;\ n = 3m+2p \}$

$a^{2p+3m} b^m c^p \Rightarrow a^{2p} \{ a^{3m} b^m \} c^p$

$S \to aaSc \mid X$
$X \to aaaXb \mid \lambda$

(7) $2n = 3m+2p$

$a^{p+\frac{3m}{2}} b^m c^p \Rightarrow a^p \mid a^{\frac{3m}{2}} b^m \mid c^p$

m = even

$S \to aSc \mid X$
$X \to aaaXbb \mid \lambda$

Scanned by CamScanner

⑧ $n+m = p+q$

cases :

$m = p+q$ S_1

$m = p+q$ S_2

$n+m = p$ S_3

$n+m = q$ S_4

~~$n=p$, $m=q$~~ S_5

$n=q$, $m=p$ S_6

$S_1 \rightarrow aS_1d \mid X_1$ | $S_2 \rightarrow bS_2d \mid X_2$

$X_1 \rightarrow aX_1c \mid \lambda$ | $X_2 \rightarrow bX_2c \mid \lambda$

$S_3 \rightarrow aS_3c \mid X_3$ | $S_4 \rightarrow aS_4d \mid X_4$

$X_3 \rightarrow bX_3c \mid \lambda$ | $X_4 \rightarrow bX_4d \mid \lambda$

$S_6 \rightarrow aS_6d \mid X_6$ | $S_5 \rightarrow aS_5d \mid S_3 \mid S_2$

$X_6 \rightarrow bX_6c \mid \lambda$

$S \rightarrow S_1 \mid S_2 \mid S_3 \mid S_4 \mid S_5 \mid S_6$

Homework

$L_9 = \{a^n b^m ; 2n \neq m\}$

$L_{10} = \{a^n b^m c^p ; m \neq n+p\}$

$L_{11} = \{a^n b^m c^p ; n \neq m+p\}$

$L_{12} = \{a^n b^{kn} ; n \geq 0\}$ find L_{12}

$L_{13} = \{a^n b^m c^p ; n \neq m \text{ or } m \neq p \text{ or } n \neq p\}$

$L_{14} = \{a^n b^m c^p d^q ; n+m \neq p+q\}$

⑨ $2n < m \Rightarrow S_1 \rightarrow aS_1bb \mid X$ | or $S_1 \rightarrow aS_1bb \mid bS_1 \mid b$

$2n > m$

$X \rightarrow bX \mid b$

$S_2 \rightarrow X_1Ab \mid X_2$ | or $S_2 \rightarrow aS_2bb \mid a \mid ab \mid aS_2$

$A \rightarrow S_2b \mid X_2$

$X \rightarrow aX_1 \mid a$

$X_1 \rightarrow aX_2 \mid \lambda$

⑩ $m \neq n+p$

$m < n+p$

$m > n+p$

to solve this first generate grammer for $m = n+p$

$\Rightarrow S \rightarrow AC$

$A \rightarrow aAb \mid \lambda$

$C \rightarrow bCc \mid \lambda$

Scanned by CamScanner

To solve ≠ grammar, first write grammar for '=' & then divide it into two cases. (< & >)

increase at least 1 extra b

now $m > n + p$ ⇒

$m < n + p$

⇓

$S_1 \to ABC$

$A \to aAb \mid \lambda$

$C \to bCc \mid \lambda$

$B \to bB \mid b$

$S_2 \to XAC \mid ACY \mid XACY$

$X \to aX \mid a$ } extra a

or

$Y \to cY \mid c$ } extra c

⇒ $S \to S_1 \mid S_2$

(11.) $n \neq m + p$

$n > m + p$

$S_1 \to aS_1c \mid X$

$X \to aXb \mid X_1$

$X_1 \to aX_1 \mid a$

$n = m + p$

$S \to aSc \mid X$

$X \to aXb \mid \lambda$

$n < m + p$

$S_2 \to aXc$

$X \to aXB \mid \lambda$

$C \to cC \mid c$

$B \to bB \mid b$

$n < m$: $S_2 \to aS_2c \mid X$

$X \to aXb \mid Xb \mid b$

$n < p$: $S_3 \to aS_3c \mid S_3c \mid Yc$

$Y \to aYb \mid \lambda$

$n < m$ && $n < p$: $Y \to aYb \mid Yb \mid b$

So, $S \to S_1 \mid S_2 \mid S_3$

$\bar{L}$ is super difficult

(14) $n + m \neq p + q$

$n \neq p + q$

$m \neq p + q$

$n + m \neq p$

$n + m \neq q$

$n \neq q$ && $m \neq p$

$n \neq p$ && $m \neq q$

$n = q$ & $m \neq p$

$n \neq q$ & $m = p$

$n \neq q$ & $m \neq p$

↓ solve this part

$n > q$: $m > p$, $m < p$

$n < q$: $m > p$, $m < p$

$n > q$ ⇒ $S_1 \to aS_1d \mid aS_1 \mid aX$

$X \to bXc \mid bZ \mid cR$

$Z \to bZ \mid \lambda$

$R \to cR \mid \lambda$

(12) $L = \{a^n b^n, n \geq 0, \Sigma = \{a, b\}\}$

$S \to aSb \mid \lambda$

$\bar{L} \Rightarrow \Sigma^* - L$

$S \to aSb \mid Xa \mid bX$

$X \to aX \mid bX \mid \lambda$

Scanned by CamScanner

$L = \{ w \mid n_a(w) = n_b(w), \Sigma = \{a,b\} \}$

$S \to aSb \mid bSa \mid SS \mid \lambda$

$L = \{ w \mid n_a(w) \neq n_b(w), \Sigma = \{a,b\} \}$

$n_a(w) > n_b(w)$ $\qquad$ $n_b(w) > n_a(w)$

$S_1 \to XaX$ $\qquad$ $S_2 \to YbY$

$X \to aXb \mid bXa \mid XX \mid Xa \mid \lambda$ $\qquad$ $Y \to aYb \mid bYb \mid YY \mid Y \mid \lambda$

$L_1 = \{ w \mid 2n_a(w) = n_b(w) \}$

$L_2 = \{ w \mid 2n_a(w) = n_b(w) + n_c(w) \}$

$L_3 = \{ w \mid 3n_a(w) = |w|, \Sigma = \{a,b,l\} \}$

(1) $S \to abb \mid bba \mid bab \mid \lambda$

$S \to SaSbSba \mid SbSbaS \mid SbSaSbS \mid \lambda$

(2)

abb	$\Rightarrow$	abb bab bba
acc		acc cac cca
abc		abc acb bac bca cab cba

I

aabbba

$S \to aSbSbS \mid \ldots\ldots$ Use I.

(3) $3n_a(w) = |w|$

$\Rightarrow 2n_a(w) = n_b(w) + n_c(w)$

So same as (2)

Terms of Context free Grammer

1. Production
2. Derivation
3. Partial Derivation
4. Parse Tree
5. Left most derivation
6. Rightmost derivation
7. Ambigous Grammer
8. Inherently Ambigous language
9. Removal of λ - prod^n
10. Removal of unit prod^n
11. find useful prod^n

Scanned by CamScanner

1. Production

$\alpha \to \beta$ where $\alpha \in V$, $\beta \in (V,T)^*$

2. Derivation:

$S \overset{*}{\to} T^*$ here $\overset{*}{\to}$ means using multiple pred^n

eg $S \to aSbS \mid bSaS \mid \lambda$ w = abab

multiple derivation

$S \to aSbS \to abS \to abaSbS \to ababS \to abab$

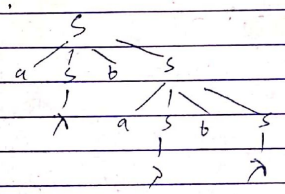
$S \overset{*}{\to} abab$

$S \to aSbS \to aSb \to abSaSb \to abaSb \to abab$

3. Partial Derivation

$S \overset{*}{\to} \beta_1\beta_2 \ldots \beta_n$ where at least one $\beta_i \in V$

4. Parse Tree: Derivation tree is known as parse tree where leaf node contains empty or terminal.



leftmost & rightmost

We can find actual string using traversal of parse tree (Inorder) & perform concatination of leaf node characters.

5. Left Most Derivation

$S \overset{*}{\to} \beta_1\beta_2 \ldots \beta_n \to T^*$

In every intermediate derivation we use any one pred^n of β_i where β_i is leftmost var. in all β,

or

In every intermediate derivation use pred^n of leftmost variable.

6. Right Most Derivation

In every partial derivation use pred^n of rightmost variable.

$S \to aSb \mid \lambda$

leftmost & rightmost both

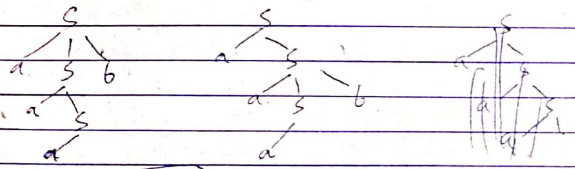
Scanned by CamScanner

7. Ambiguous Grammar

If there exist more than 1 leftmost derivation or more than 1 rightmost derivation or more than 1 parse tree

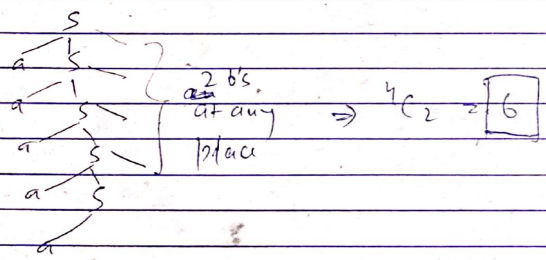
Ques How many parse tree are possible for str. "aaab" in grammar

$S \to aS \mid abSb$ $S \to aS \mid a \mid BaSb$



ans: 2

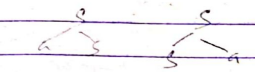
Q. Str. = aaaaabb



2 b's at any place $\Rightarrow {}^4C_2 = 6$

Q. Str. = "aaa"

$S \to aS \mid Sa \mid \lambda$



→ 2 choice at any level.

$\Rightarrow 2^3 = 8$

Q. Removal of λ-prod^n

$S \to aSB \mid aAaB \mid a$	$aS \mid aAa \mid aaB \mid aBA \mid aa \mid baA \mid baBA \mid aBaA$
$A \to aBAB \mid bBaBA$	aS
$B \to aSB \mid bb \mid \cancel{\lambda}$	

use these after removing λ

Steps

Step 1: find λ prod^n list
i.e. $\alpha \in L$ if $\alpha \xrightarrow{*} \lambda$
$L = \{B\}$

Step 2: Add extra prod^n by removal of all variables of list L using all combinations.

Step 3: remove λ prod^n

Scanned by CamScanner

PAGE NO
DATE: / /

$S \to aAB \mid as$
$A \to BB \mid as \mid a$
$B \to \lambda$

$\Rightarrow$ $S \to aAB \mid as \mid aB \mid aA \mid a$
$A \to BB \mid as \mid a \mid B$
~~$B \to \lambda$~~

after removing $\{B \to \lambda\}$. B becom useless, so remove it

$\Rightarrow$ $S \to aA \mid as \mid a$
$A \to as \mid a$
} useful prod^n

Useful prod^n:
$\alpha \to \beta$ is useful prod^n
if α is reachable from S & α should generate T^*.

$S \to (aA) \mid (aSBC) \mid b$
$A \to (aA) \mid (aS) \mid (BC) \mid (BB)$
$B \to aBX$
$D \to as \mid a$

$\Rightarrow$ $S \to aA \mid b$
$A \to aA \mid aS$

~~Removal of~~ Removal of useless prod^n

Let G_1 is given grammer & G_1' is grammer after removal of empty prod^n. if $L(G_1)$ & $L(G_1')$ is language generated by G_1 & G_1'.

Then
whether $L(G_1) = L(G_1')$?
No.
iff. language does not contain 'λ'.

eg $S \to aSb \mid \lambda \Rightarrow \mid S \to aSb \mid ab$

Hence.
$L(G_1') = L(G_1) - \lambda$

Removal of unit prod^n

Unit prod: $\alpha \to \beta$ is unit prod^n
if ~~$\alpha \to \beta$~~ $\alpha, \beta \in V$

Scanned by CamScanner

$S \to aA \mid bAb \mid aS \mid b$

$A \to aSB \mid aa \mid \lambda \quad | aA | bbAb | aS | b$

$B \to aA \mid bBa \mid \beta$

$S \to a \mid B \quad S \to a \mid b$

$B \to C \quad B \to b$

$C \to D \quad C \to b \quad \Rightarrow S \to a \mid b$

$D \to b \quad D \to b$ Useless

Step 1: find reachable list using unit prod^n for each var.

let α be var. then α' is reachable list of α

$\beta \in \alpha'$ if $\alpha \xrightarrow{*} \beta$

Step 2: Add also prod^{n}s of β in α $\forall \beta \in \alpha'$

Step 3: remove unit prod^{n}s

after removing unit prod^n

$L(G) = L(G')$

Lecture-14

Ambigous Grammar

- There exists a string which has more than one parse tree.

Q. find ambigen

L_1: $S \to aS \mid Sa \mid \lambda$

L_2: $S \to aS \mid a \mid \lambda$

L_3: $S \to aS \mid aSb \mid a$

L_4: $S \to aS \mid a\lambda$
$\lambda \to a\lambda b \mid \lambda$

L_5: $S \to aS \mid aSb \mid \lambda$

L_6: $S \to aS \mid Sb \mid \lambda$

L_7: $S \to aS \mid \lambda$
$\lambda \to b\lambda \mid \lambda$

L_8: $S \to aSb \mid aS \mid bS \mid \lambda$

L_9: $S \to AB$
$A \to aA \mid \lambda \quad B \to aB \mid \lambda$

L_{10}: $S \to S_1 \mid S_2$
$S_1 \to aS_1b \mid \lambda \quad S_2 \to aS_2bb \mid \lambda$

L_{11}: $S \to aSbb \mid aSb \mid \lambda$

L_{12}: $S \to aSbB \mid \lambda$
$B \to b \mid \lambda$

L_{13}: $S \to 0S \mid 10S \mid 0S1 \mid 1S0 \mid \lambda$

Inherently Ambigous language

If all grammers of lang. are ambigous, then that lang. is known as inherently ambigous.

Scanned by CamScanner

$S \to aSbb \mid aSb \mid \lambda$ — ambigous

$L = \{ a^n b^m ;\ n \le m \le 2n \}$

$S \to aSbb \mid X$
$X \to aXb \mid \lambda$ } unambigous

$L = \{ a^n b^m c^p ,\ n=m \text{ or } m=p \}$

$S \to S_1 \mid S_2$

$S_1 \to XY$ $\quad S_2 \to PQ$

$X \to aXb \mid \lambda$ $\quad P \to aPb \mid \lambda$

$Y \to cY \mid \lambda$ $\quad Q \to bQc \mid \lambda$

$a^5 b^5 c^5$ ambigous

inherently ambigous lang.

Ex-b

$L = \{ ww^R ;\ w \in (a,b)^* \}$

$S = (a,b)^*$

$L = ?$

. taller all odd length & even which is not ww^R.

$S \to asa \mid bsb \mid aXb \mid bXa \mid a \mid b$

$X \to aXa \mid bXb \mid aXb \mid bXa \mid \lambda \mid a \mid b$

or

$X \to aX \mid bX \mid \lambda$

Q1 $L = \{ w \mid w = w^R ,\ w \in (a,b)^* \}$

$L = ?$

Q2 $L = \{ ww \mid w \in (a,b)^* \}$ $\quad L = ?$

Q3 $L = \{ a^n b^n c^n ;\ n \ge 0 \}$ $\quad L = ?$

Ans

Q $L = \{ a^n b^m c^p ;\ n = m \}$

$L = ?$

$S \to DC \mid EC' \mid CX'$

$D \to aDb \mid Xa \mid bX$

$X \to aX \mid bX \mid cX \mid \lambda$

$C \to ac \mid bc \mid cC \mid \lambda$

$E \to aEb \mid \lambda$

$C' \to cC' \mid ac \mid bc$

$X' \to cX' \mid ac \mid bc$

Q3 $L = \{ a^n b^m c^p ;\ n = m = p \}$

$L \Rightarrow n \ne m ,\ n \ne p ,\ m \ne p$

Scanned by CamScanner

lecture-15

Normal Forms: It's a method to convert a grammer for computer processing i.e, how comp can understand whether given str. is member of given grammer or not.

Types —

- Chomnskey normal form
- Greibach normal form ⇒ $\alpha \to \beta$, $\alpha \in V$, $\beta \in TV^*$, $\beta \neq \epsilon$

It's as cfg in which prod^n is diff.
$(\alpha \to \beta ;\ \alpha \in V,\ \beta \in (V \cup T)^*)$
$\beta \neq \{\epsilon, V\}$

Conversion of cfg → Chomnskey

Step 1: remove ϵ-prod^n ⇒, then unit prod^n

Step 2: if $A \to \beta_1 \beta_2 \ldots \beta_{n-1} \beta_n$, $\beta_i = \{V, T\}$

- if size = 1 no problm
- if size = 2 cases: $VV \mid TV \mid VT \mid TT$

for every terminal create independent variables i.e, $T_i \to \xi_i$

convert T to its corresponding variable (terminal)

if size > 2 convert it as follows

$A \to \beta_1 \beta_2 \ldots \beta_{n-1} \beta_n$

$\Rightarrow A \to \beta_1 X_1$

$X_1 \to \beta_2 X_2$

$X_{n-2} \to \beta_{n-1} \beta_n$

So, $A \to \beta_1 \beta_2 \ldots \beta_{n-1} \beta_n$ requires $n-2$ new var. to convert above into chomskey

Q: What is min. & max. size of str. generated by chomisky normal form where height of parse tree is h.

[height of node = max. no. of edges from given node to leaf.]

A: max → 2^h

min → h

Scanned by CamScanner

Greibach normal form (GNF)

$\alpha \to \beta$ $\quad \alpha \to TV^*$

$S \to aSBa \mid aaaB \mid ab$ | $T_1 \to a$, $T_2 \to b$ | $S \to aSBT_1 \mid aT_1T_1B \mid aT_2$

like this

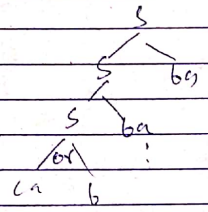
$S \to aA \mid baB \mid a$
$A \to aA \mid Sb \mid Bb$ (problm)
$\Rightarrow A \to aA \mid aAb \mid BaBb \mid ab$

$S \to aA \mid baB \mid a$

// reduce like first

$S \to Ba \mid b$
$B \to Sb \mid c$ | $S \to Sba \mid Bca \mid b$
$B \to Bab \mid bb \mid c$ — since it is unreachable

So, $S \to Sba \mid cab$ ← left recursion



removal of left recursion

$\Rightarrow S \to caA \mid bA \mid ca \mid b$
$A \to baA \mid ba$

$\alpha \to T_1$, $T_1 \to a$, $T_2 \to b$, $T_3 \to c$

$S \to cT_1A \mid bA \mid cT_1 \mid b$
$A \to bT_1A \mid bT_1$

GNF

Removal of left recursion

$A \to A\beta_1 \mid A\beta_2 \mid \ldots \mid A\beta_n \mid \gamma_1 \mid \gamma_2 \mid \ldots \mid \gamma_n$

$\Rightarrow$

$A \to \gamma_1 A' \mid \gamma_2 A' \mid \ldots \mid \gamma_n A'$
$A' \to \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A' \mid \lambda$

Steps to convert CFG $\to$ GNF

1. remove left recursion
2. perform $S \to SBS \mid aS \mid b$
   $B \to BSB \mid bB \mid a$

$S \to aSS' \mid bS'$
$S' \to BSS' \mid \lambda$
$B \to bBB' \mid aB'$
$B' \to SBB' \mid \lambda$

2. perform substitution

$S \to aSS' \mid bS'$
$S' \to bBS'SS' \mid aB'SS' \mid \lambda$
$B \to bBB' \mid aB'$
$B' \to aSS'BB' \mid bS'BB' \mid \lambda$

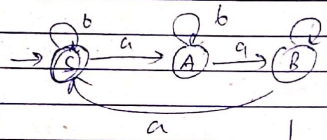
Scanned by CamScanner

3. removal of ϵ & unit prod^n.

★ $CNF(L) = GNF(L) = (reg(L) - \epsilon$

Regular grm

Design cfg for $L = \{w \mid w$ contains $3k+1$ a's $\}$

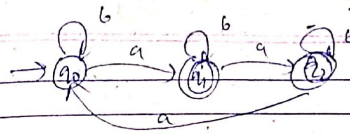


$S \to bS \mid aA \mid \lambda$	$S \to Ba \mid Sb \mid \lambda$
$A \to bA \mid aB$	$A \to Sa \mid Ab$
$B \to bB \mid aS$	$B \to Aa \mid Bb$
(~~left~~-linear) right	(left-linear)
$\alpha \to TV \mid T \mid \lambda$	$\alpha \to VT \mid T \mid \lambda$

★ ~~How to find left-linear grammar of regular lang. grammar:~~ ⇒

if transition of $q_i \xrightarrow{\Sigma} q_j \Rightarrow q_j \to q_i \Sigma$

& add empty in initial state & starting var. of grammar is final state.



$q_0 = q_0 b \mid q_2 a \mid \lambda$

$q_1 = q_0 a \mid q_1 b$

$q_2 = q_1 a \mid q_2 b$

Stuk

What if there is more than 1 final state?

let say q_1 & q_2 are final, so

$S \to q_1 \mid q_2$ ∵ it isn't left/right linear.

replace q_1 & q_2 with their respective prod^n(s)

$S \to q_0 a \mid q_1 b \mid q_1 a \mid q_2 b$

$q_0 \to q_0 b \mid q_2 a \mid \lambda$

$q_1 \to q_0 a \mid q_1 b$

$q_2 \to q_1 a \mid q_2 b$

Scanned by CamScanner

PAGE NO
DATE: / /

* How to find right-linear

if transition $q_i \xrightarrow{\Sigma} q_j \Rightarrow q_i = \Sigma q_j$

add λ in final states & initial state is starting symbol of grammar.

PAGE NO
DATE: / /

lecture-16

Homomorphism :

$H(L) = \{ h(w) \mid w \in L$ & h is homomorphism fn where w is not empty $\}$

$h(\Sigma) \rightarrow T^*$ ← other alphabets.

↑ input alphabet

Eg $L = \{ w \mid w$ contains even no. of a's $\Sigma = \{a, b\} \}$

$h(a) \rightarrow cc$

$h(b) \rightarrow 01$

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

$H(L) = (01 + cc(01)^*cc)^*$

* So, if L is RL $\Rightarrow$ $H(L)$ is also RL.

So, Homomorphism is closed under RL.

Inverse Homomorphism

if $h(L) \rightarrow L_1 \Rightarrow h^{-1}(L_1) \rightarrow L$ ✗

$L^{-1}(T^*) \rightarrow \Sigma$

Scanned by CamScanner

$H^{-1}(L) = \{ h^{-1}(w) \mid w \in L,\ L^{-1} \text{ is a } \uparrow^n \text{ like } h^{-1}(T^*) \to \varepsilon \}$

eg $L = ab^*(a^*b)^*$

$h^{-1}(a) \to 21$

$h^{-1}(ba) \to 11$

$h^{-1}(bb) \to 22$

$21(22)^*((21)^*(22)^*)^*$

eg $(a^*b)^*$

$(21(21)^*22(22)^* + (22)^*)^*$

or

$(21(21)^*(11)^*22(22)^* + (22)^*)^*$

* Inverse Homomorphism is also closed.

DFA/NFA/RE $\longrightarrow$ left/right linear grammer

RL $\longrightarrow$ left/right linear grammer

RG: either $A \to VT \mid T \mid \lambda$ or $A \to TV \mid T \mid \lambda$ — not both together

Linear grammer: $\alpha \to \beta$ is linear if $\alpha \in V$ & β contains at most one variable.

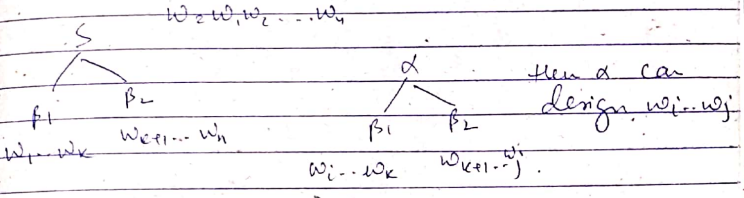
Parsing Algorithm

We have given string w & grammer G determine whether $w \in L(G)$ or not?

CYK algo:

Convert G into G' in CNF

$w = w_1 w_2 \ldots w_n$



if $S = \alpha$, $i = 1$, $j = n$ then $w \in L(G)$

CYK is based on Bidynamic programming

Scanned by CamScanner

(a) $S \to AB \mid SA \mid b$ $\quad w = "abba"$ $\quad w_1 w_2 w_3 w_4$

$A \to BB \mid a$

$B \to AB \mid SS$

	1	2	3	4
1	A	ϕ	S, B (1)	S, S, B (3)
2		S	B	B (2)
3			S	S
4				A

if containing S then it is accepted

$w_1 \mid w_2$

A S → no prod" AS

$w_2 \mid w_3$

B S → B

$w_3 \mid w_4$

S A → S

$w_1 w_2 w_3$

$w_1 \mid w_2 w_3$ $\quad w_1 w_2 \mid w_3$

AB → B $\quad \phi$

$w_2 w_3 w_4$

$w_2 \mid w_3 w_4$ $\quad w_2 w_3 \mid w_4$

SS → B $\quad$ BA → ϕ

$w_1 w_2 w_3 w_4$

$w_1 \mid w_2 w_3 w_4$ $\quad w_1 w_2 \mid w_3 w_4$ $\quad w_1 w_2 w_3 \mid w_4$

AB → B, S $\quad \phi \quad$ SA, BA → S, ϕ

$|w| = n$

Complexity $\frac{n^2}{2} \times \frac{n}{2} = O(n^3)$

CNF

$w \in L(G)$ where G is in CNF.

$S \to aAB \mid aS \mid b$ $\quad w = abaa$

$A \to bS \mid a$

$B \to aSA \mid bA$

parse using ~~back tracking~~

Simple Grammar:

$\alpha \to TV^*$

s.t. at most one pair of (α, T) should be in prod" rule.

$S \to aB \mid bAS$

$A \to bS$

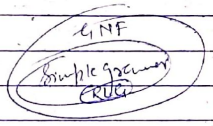
$B \to b \mid aAS$

Simple

$S \to AB \mid aS \mid b$ ✗ (conflict)

$CFG_1 \to G$ whether we can convert into simple grammer or not?

Ans No



Scanned by CamScanner

$L = \{a^n b^n : n > 0\}$

- write simple grammer

Closure Prop. of CFL

let L_1 & L_2 are CFL

① Union $L_1 \cup L_2$ $\Rightarrow S \to S_1 | S_2$ ✓

② Contenation $L_1 \cdot L_2$ $S \to S_1 S_2$ ✓

③ Kleen Star L_1^* $S \to L_1 S | \lambda$ ✓

4. Compliment L_1

5. Intersection $L_1 \cap L_2$

$L_1 = \{a^n b^n c^m ; n,m \geq 0\}$

$L_2 = \{a^m b^n c^n ; n,m \geq 0\}$

$L_1 \cap L_2 = \{a^n b^n c^n , n \geq 0\}$

not CFL

Gr Grammer (not CFG)

$S \to aSBC | \lambda$

$CB \to BC$

$B \to b$

$C \to c$

$S \to aSBC \to aaSBCBC$

$\to aaaBCBCBC$

$\to aaaBBCBCCC$

$\to aaabBCBCCC$

$\to aaabbCBCCC$

$\to aaabbBCCCC$

$\to aaabbbCCCC$

4. Compliment L_1

assum compliment is context free

$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}} \to CF$

Complement CF $\overline{L_1} \cup \overline{L_2} \to CF$

$\Rightarrow L_1 \cap L_2$ is context-free

which is wrong

So, compliment is not context-free

Scanned by CamScanner

$L_1 = \{a^n b^m,\ n,m>0\}$ $L_2 = \{b^m a^n,\ n,m>0\}$

$L_1 \cap L_2 = a^+ b^+$

There are infinite languages which aren't context-free but complement is.

$L = \{a^n b^n c^n;\ n>0\}$ - not CFL

$\to \{a^n b^m c^p;\ n=m=p\}$

$\overline{L} = L_1 \cup L_2 \cup L_3$

$L = \{ww \mid w \in \{a,b\}^*\}$ not CFL

$\overline{L} = L_1 \cup L_2$

L_1 (odd) $S_1 \to aX \mid bX$
$X \to aS_1 \mid bS_1 \mid \lambda$

L_2 $S_2 \to PQ \mid QP$
$P \to aPa \mid bPb \mid aPb \mid bPa \mid a$
$Q \to aQa \mid bQb \mid aQb \mid bQa \mid b$

$\underbrace{w_1}_{d}\ \underbrace{w_2}_{d}$ mid should be diff.

(a) Prefix(L_1)
Suffix(L_1)
Half(L_1)
Truncate(L_1)

In subset operation CFL & RL is not closed

e.g.

$a^* b^* c^* \to a^n b^n c^m \to a^n b^{n^2} c^n$

CFL — not CFL

Scanned by CamScanner

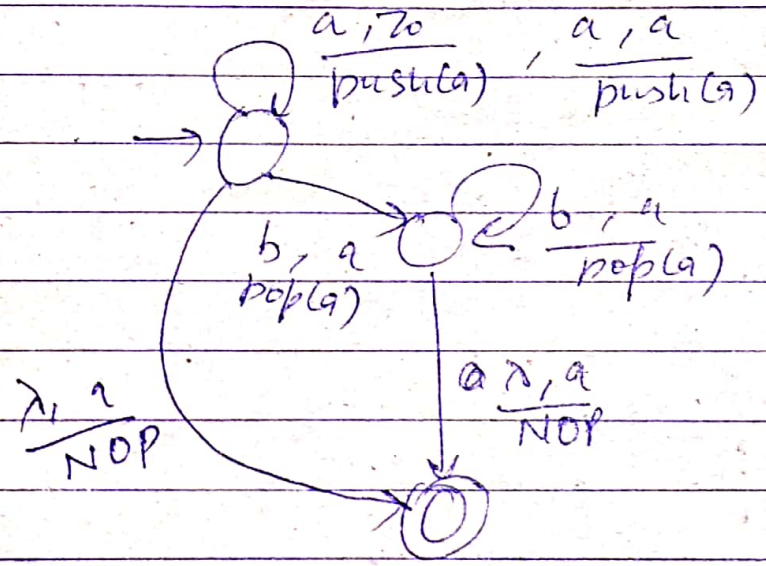
TOC

lecture-17

PAGE. NO
DATE: / /

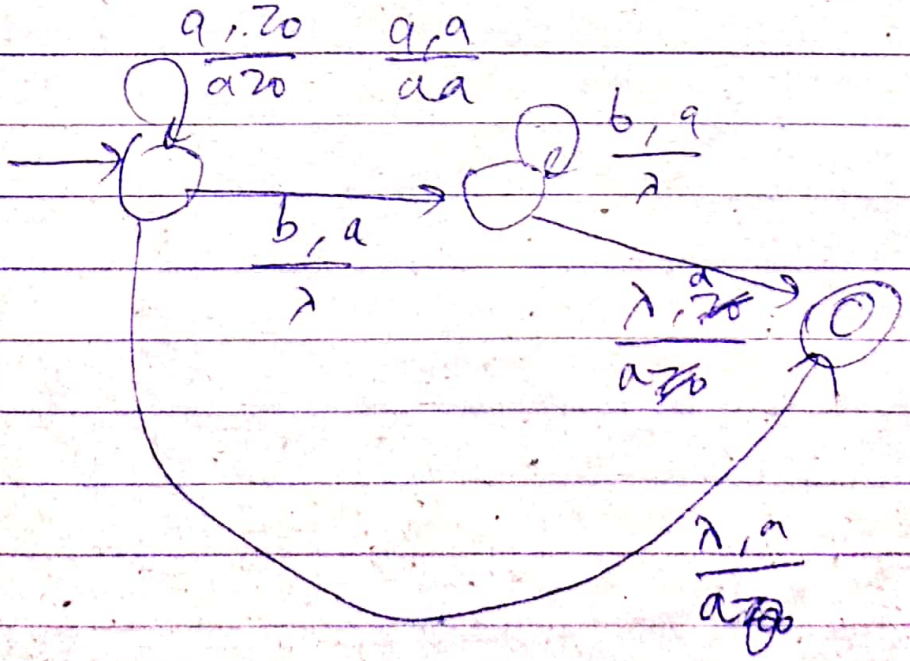
PDA : push down automata which contain one stack and use to verify string of cfg.

$L = \{ a^n b^m , n > m \}$ aaaa b



push : push
pop : delete
NOP : no operation

⇓ good / right way



Scanned by CamScanner

for every CFL there exists PDA & vice-versa.

Examples

$L_1 = \{ a^n b^m \; ; \; n \neq m \}$

$L_2 = \{ a^n b^m \; ; \; n+2 < m \}$

$L_3 = \{ a^n b^m c^p \; ; \; n = m \}$

$L_4 = \{ a^n b^m c^p \; ; \; n = p \}$ $\neq$

$L_5 = \{ a^n b^m c^p \; ; \; n = m+p \}$ $\neq$

$L_6 = \{ a^n b^m c^p \; ; \; m = n+p \}$ $\neq$

$L_7 = \{ a^n b^m c^p ; m = 3n+2p \}$ $\neq$

$L_8 = \{ a^{2n} b^{3m} c^p ; n = m \}$ $\neq$

$L_9 = \{ a^n b^m c^p, d^q ; n+m = p+q \}$

$L_{10} = \{ a^n b^{2n} ; n > 0 \}$

$L_{11} = \{ w c w^R ; w \in (a,b)^* \}$

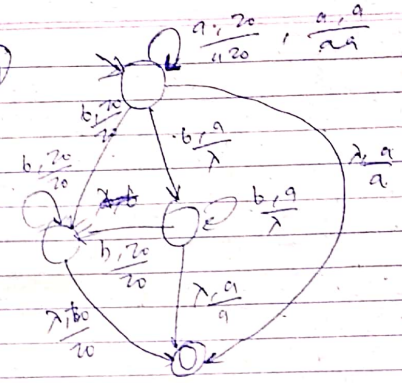
$L_{12} = \{ w w^R \}$

$L_{13} = \{ w \mid n_a(w) = n_b(w) \; , \; \Sigma = (a,b) \}$

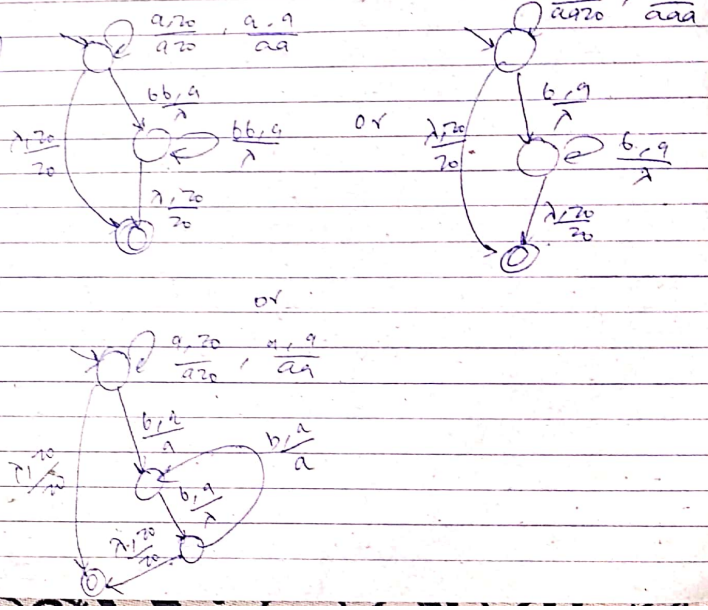
$L_{14} = \{ w \mid n_a(w) = n_b(w) + n_c(w) \; , \; \Sigma = (a,b,c) \}$

$L_{15} = \{ w \mid w = w^R \; , \; \Sigma = (a,b) \}$

(1)



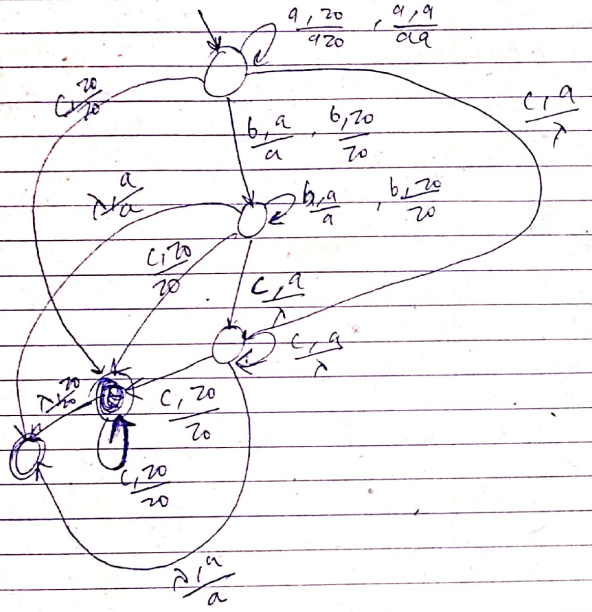
(10)



or

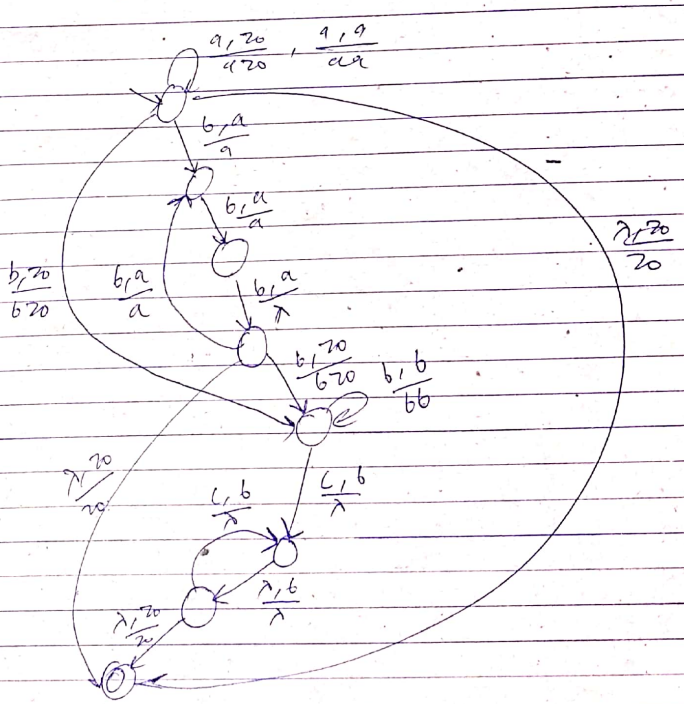
Scanned by CamScanner

(L4) $\{a^n b^m c^p ;\ n \neq p\}$



(L7) $a^n b^m c^p \quad m = 3n + 2p \quad n, m, p \geq 0$

$a^n \; b^{3n} \; b^{2p} c^p$



Scanned by CamScanner

PAGE NO
DATE : / /

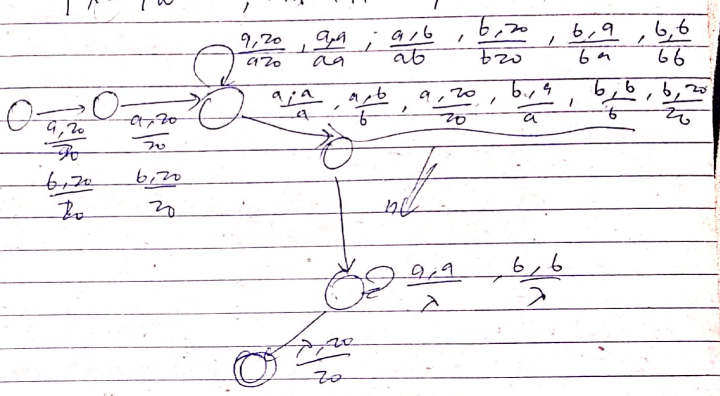
★ V.V.Imp

complement in PDA isn't final → non-final ← non-final → final. It is different

PAGE NO
DATE : / /

Lecture-18

1. → { $xwy w^R$, $|x| = |y| = 2$ }

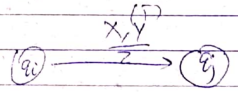


PDA

$M = (Q, \Sigma, \delta, s, f, \Gamma, z_0)$

↓ set of push stack variable → initial marker

PDA/NDPDA

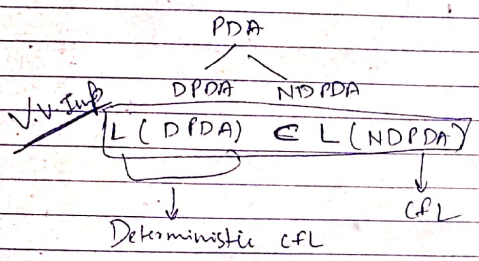


$\delta(q_i, x, \Gamma_1) \rightarrow \{(q_j, \Gamma_2^*), (q_r, \Gamma_3^*), \ldots\}$

Scanned by CamScanner

PAGE. NO
DATE : / /

DPDA : $\delta(q_i, X, r)$ contains at most one element in set δ



Deterministic cfL : If a lang. is generated by DPDA or there exists a DPDA of that lang.

Q. Identify DCFL

$L_1 = \{a^n b^m ; n \leq m\}$ ✓

$L_2 = \{a^n b^m ; n > m$ or $n = m+1\}$ ✓

$L_3 = \{a^n b^m ; n, m \geq 0\}$ ✓

$L_4 = \{wcw^R ; w \in (a,b)^*\}$ ✓

$L_5 = \{ww^R ; w \in (a,b)^*\}$ ✓

$L_6 = \{w \mid w = w^R, w \in (a,b)^*\}$ ✗

$L_7 = \{a^n b^m ; n = m$ or $2n = m\}$ ✗

✗ Unambiguous grammar ≠ Deterministic CFL

PAGE. NO
DATE : / /

$L_8 = \{a^n b^m ; n \leq m \leq 2n\}$ ✗

Closure operation of DCFL / DPDA

	DCFL	CFL
Union	No	Yes
Concatination	No	Yes
Kleen star	No	Yes
✗ Compliment	Yes	No
Intersection	No	No

If a lang. is DCFL → its compliment is also DCFL

$L_1 = \{a^n b^n ; n > 0\}$

✗ L_1^* is DCFL

Scanned by CamScanner

Lecture-19

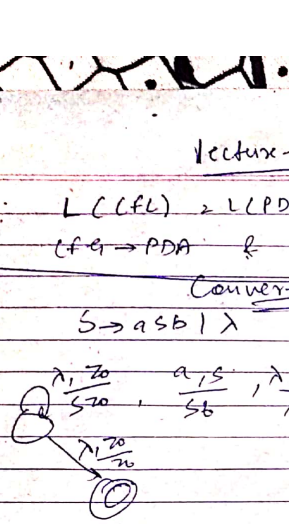
PAGE NO
DATE : / /

L(CFL) = L(PDA)

CFG → PDA & PDA → CFG

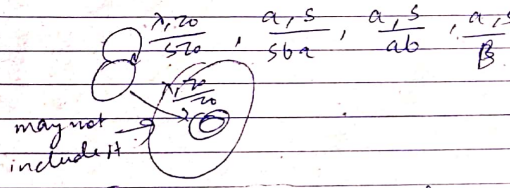
Convert CFG to PDA

$S \to asb \mid \lambda$



$S \to asba \mid aaab \mid aB$

$B \to aSaa \mid b$



Every PDA can be designed in one state.

PAGE NO
DATE : / /

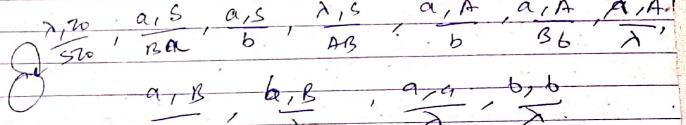
PDA

- final state version
- Empty stack version — stack must be empty

$S \to AB \mid aBA \mid ab$

$A \to ab \mid aBb \mid \lambda$

$B \to aaB \mid b$



$\alpha \to T\beta$	$\delta(q_0, T, \alpha)$	$\to (q_0, \beta)$
$\alpha \to V\beta$	$\delta(q_0, \lambda, \alpha)$	$\to (q_0, V\beta)$
$\alpha \to \lambda$	$\delta(q_0, \lambda, \alpha)$	$\to (q_0, \lambda)$

Add

$\delta(q_0, \lambda, Z_0) \to (q_0, SZ_0)$

$\delta(q_0, \Sigma, Z_0) \to (q_0, \lambda)$

Scanned by CamScanner

PDA → Grammer

DPDA → Grammer

↳ It will be unambigous grammer

⇒ If Langugi has DPDA, then it is will be unambigous. grammer

but converse isn't true.

eg $L = \{a^n b^m ; n = m \text{ or } 2n = m\}$

It has no DPDA

but it is unambigous.

$S \to S_1 | S_2 | \lambda$

$S_1 \to aS_1b | ab$

$S_2 \to aS_2bb | abb$

So, If DCFL / DPDA ⟶ Unambigous

If Unambigous doesn't gaurantee DCFL / DPDA

$L = \{ww | w \in (a,b)^*\}$

www

$a^n w b^n w c^n$

RL → FA (DFA / NFA)

CFL → PDA

CSL

RE → TM (turing machine)

Turing Machine

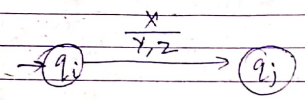
Informal definition : It is similiar to DFA / NFA but we can read any char. many times because of we can move left also when we read a character (in PDA/FA we move only at right direction).

Non-deterministic → Parallel processing (explore all possibilities parallely)

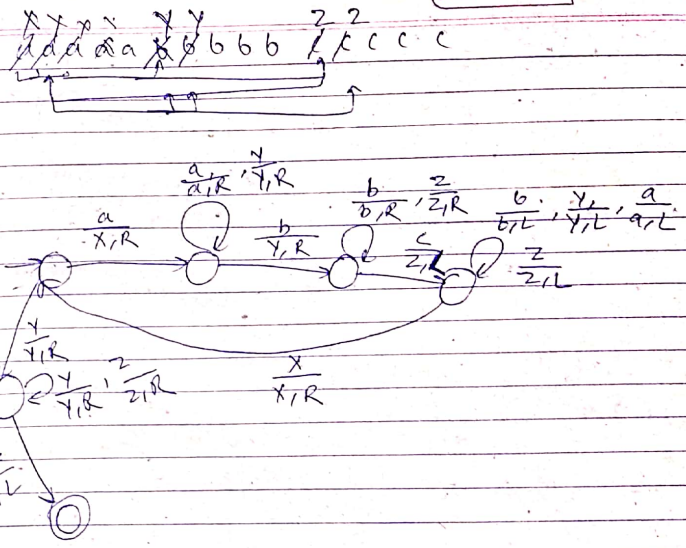
Scanned by CamScanner

We have also extra power to change content of tape. i.e, input string is available as tape, whose size of tape is infinite.
i.e, if input str. is w then input tape contains
... # # w # # ... ∞ times
finite

Eg: $a^n b^n c^n$



X → curr. symbol of tape
Y → new symbol at X
Z → dir. Left(L), right(R), nowhere(N)



$L = \{ a^n b^m c^p d^q ;\ n=p \ \& \ m=p+q \ \& \ p=q$
or
$n+m = 2p+3q \}$

$a^p b^{2p} c^p d^{3p}$

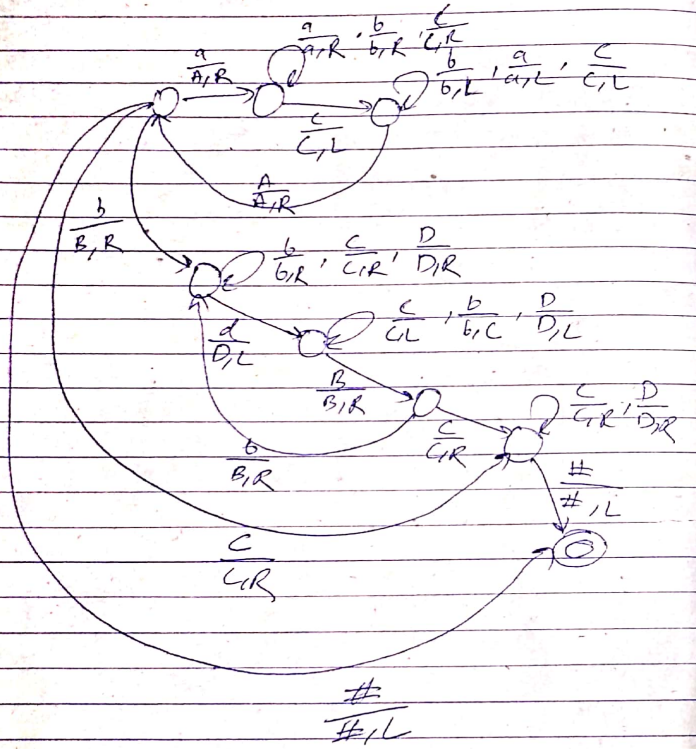
a a b b b b c c d d d

So, cut 1 a, 2 b, 1 c & 1 d at each iteration

Scanned by CamScanner

$L = \{ a^n b^m c^p d^q \; ; \; n = p \text{ & } m = q \}$

first change all a & c, then all b & d.



$L_4 = \{ a^n b^m c^p \; ; \; m = 2n \text{ & } p = 2m \}$

$L_2 = \{ w \mid n_a(w) = n_b(w) = n_c(w) \; , \Sigma = \{a, b, c\} \}$

$L_3 = \{ w \mid n_a(w) = 2 n_b(w) \neq n_c(w) \; , \Sigma = \{a, b, c\} \}$

(CFL) $L_4 = \{ a^n b^m c^p \; ; \; 2m = 5n + 3p \}$

$L_5 = \{ a^p \; ; \; p \in \text{prime} \}$

$L_6 = \{ ww \mid w \in \{a, b\}^* \}$

$L_7 = \{ a^n$

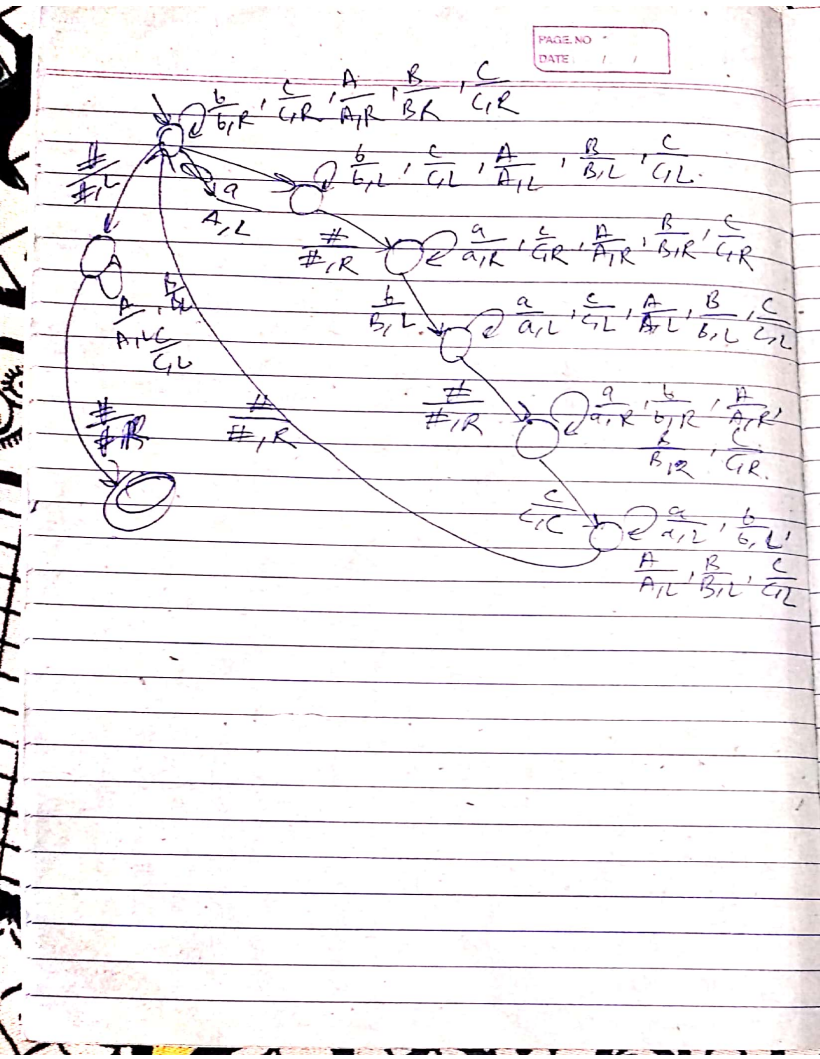
$L_8 = \{ wvw^R \mid \& w = \{a, b\}^* \}$

$L_1 : a^n b^{2n} c^{4n}$

change $1a, 2b, 4c$

$L_2 :$ #baaacccbbab

Scanned by CamScanner



TM

acceptance computations

Scanned by CamScanner

PAGE NO
DATE : / /

lecture-20

Regular language

RE → λ-NFA → DFA

NFA

FA → RE
(convert)

3 methods to convert FA to RE

1. ~~R(i,j,k+1) = R(i,j,k) + R(k)~~
2. $R(i,j,k+1) = R(i,j,k) \cup R(i,k,k)R(k,k,k)^* R(k,j,k)$

i → initial state
j → final state
k+1 is no. of states
↳ no. of intermediate states

2. Arden's method
3. ~~State elimination method~~

PAGE NO
DATE : / /

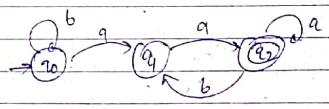
(2) Arden's method

$P \rightarrow xP + y \Rightarrow P \rightarrow x^*y$

FA to RE

Step 1 : find eqⁿ of all states
Step 2 : Add λ in final state
Step 3 : Use substitution method & find initial state.

eg



$q_0 = aq_1 + bq_0 \Rightarrow q_0 = b^*aq_1$

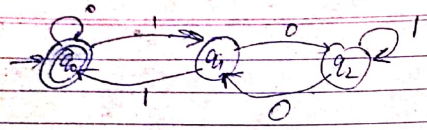
$q_1 = aq_2$

$q_2 = aq_2 + bq_1 + \lambda \Rightarrow q_2 = a^*(bq_1 + \lambda)$

$q_1 = aq_2$
$= a^+(bq_1 + \lambda)$
$= a^+bq_1 + a^+$
$q_1 = (a+b)^*a^+$

$q_0 = \boxed{b^*a(a+b)^*a^+}$

Scanned by CamScanner



$f(w) \bmod 3 = 0$

$q_0 = 0q_0 + 1q_1 + \lambda$

$q_1 = 0q_2 + 1q_0$

$q_2 = 1q_2 + 0q_1 \Rightarrow q_2 = 1^*0q_1$

$q_1 = 01^*0q_1 + 1q_0 \Rightarrow q_1 = (01^*0)^*1q_0$

$q_0 = (0 + 1(01^*0)^*1)q_0 + \lambda$

$q_0 = (0 + 1(01^*0)^*1)^*$

* How to prove two regex are equal.

Method

$RE_1 \cong RE_2$

RE_1 → NFA → DFA → minimize

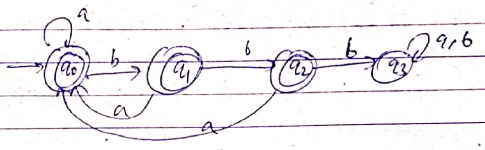
RE_2 → NFA → DFA → minimize

if they are isomorphic ⇒ they are equal

V.V.I: $R + \phi = \phi + R = R$, $R \cdot \phi = \phi \cdot R = \phi$

$\phi^* = \lambda$

Sol:



$q_0 = aq_0 + bq_1 + \lambda$

$q_1 = aq_0 + bq_2 + \lambda$

$q_2 = aq_0 + bq_3 + \lambda$

$q_3 = aq_3 + bq_3 + \phi \Rightarrow q_3 = q_3(a+b) + \phi$

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

$q_2 = aq_0 + b\phi + \lambda$

$= aq_0 + \phi + \lambda = aq_0 + \lambda$

$q_1 = aq_0 + b(aq_0 + \lambda) + \lambda$

$= (a + ba)q_0 + b + \lambda$

$q_0 = aq_0 + b[(a+ba)q_0 + b + \lambda] + \lambda$

$= (a + ba + bba)q_0 + bb + b + \lambda$

$= (a + ba + bba)^*(bb + b + \lambda)$

Scanned by CamScanner

PAGE NO
DATE : / /

Thumb rule : If language is given & asked to check with RE is correct then always draw DFA first then derive RE from it.

In Ardan's method :
add λ in all final states & find initial state

Original Ardan's

$\delta(q_i, \Sigma) \to q_j$

$q_j = q_i \Sigma$

all λ in initial state

find all final states & perform union

Modified (use this)

$q_i = \Sigma q_j$

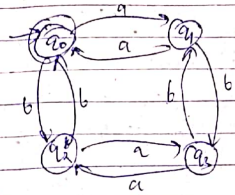
all λ in all final states

&

find initial state

PAGE NO
DATE : / /

Lecture-21



$q_0 = aq_1 + bq_2 + \lambda$

$q_1 = aq_0 + bq_3$

$q_2 = aq_3 + bq_0$

$q_3 = aq_2 + bq_1$

$q_2 = aaq_2 + abq_1 + bq_0$

$q_2 = (aa)^* (abq_1 + bq_0)$ — (1)

$q_1 = aq_0 + baq_2 + bbq_1$

$q_1 = (bb)^* (baq_2 + aq_0)$ — (2)

$q_1 = (bb)^* baq_2 + (bb)^* aq_0$

$= (bb)^* ba \left[(aa)^* abq_1 + (aa)^* bq_0 \right] + (bb)^* aq_0$

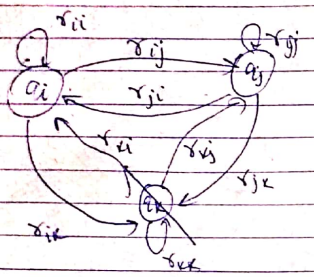
$q_1 = \left((bb)^* ba (aa)^* ab \right)^* \left((bb)^* ba (aa)^* bq_0 + (bb)^* aq_0 \right)$

long method

Scanned by CamScanner

PAGE. NO
DATE : / /

State elimination method

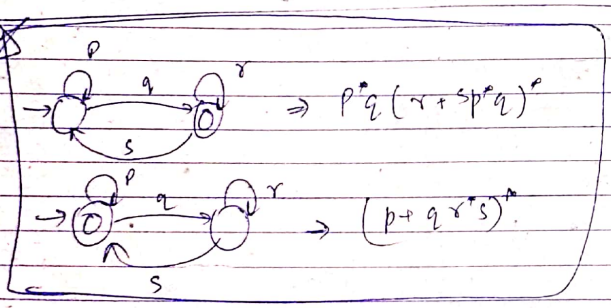


$r_{ii} + r_{ik} r_{kk}^* r_{ki}$

$r_{ij} + r_{ik} r_{kk}^* r_{kj}$

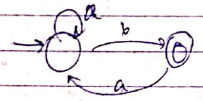
$r_{jj} + r_{jk} r_{kk}^* r_{kj}$

$r_{ji} + r_{jk} r_{kk}^* r_{ki}$



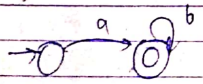
$\Rightarrow p^* q (r + s p^* q)^*$

$\Rightarrow (p + q r^* s)^*$

eg 

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

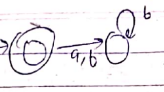
$\Rightarrow a^* b (a a^* b)^*$



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

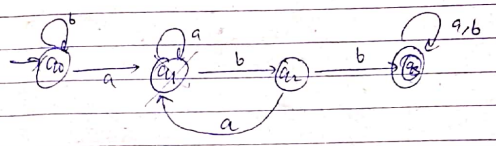
$\Rightarrow a b^*$

PAGE. NO
DATE : / /

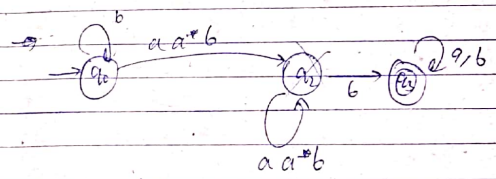


$\Rightarrow (\phi + (a+b) b^* \phi)^*$

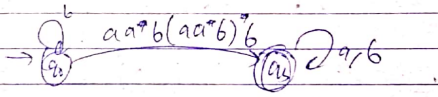
$\Rightarrow \phi^* = \lambda$



$\Rightarrow$ let's delete q_1



delete q_2



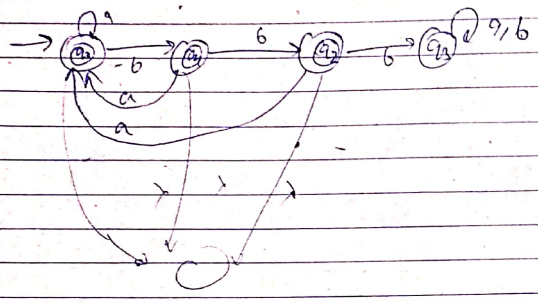
$\Rightarrow b^* (aa^*b(aa^*b)^*b)((a+b) + \phi)^*$

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

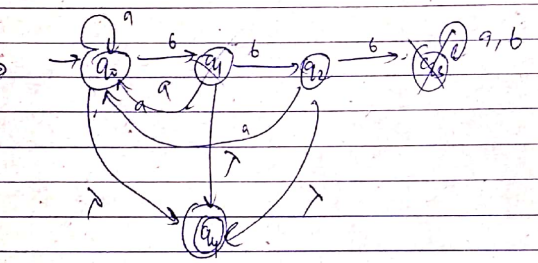
Scanned by CamScanner

Thumb rule: Except initial & final states, eliminate all states one by one. Solve initial & final by using parsvian method.

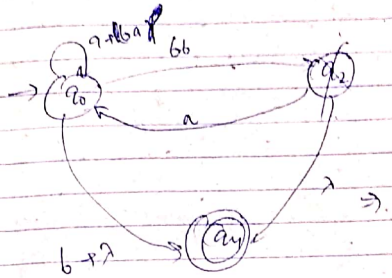
If more than one final → create new final, put λ-trans from all finals & make them non-final.



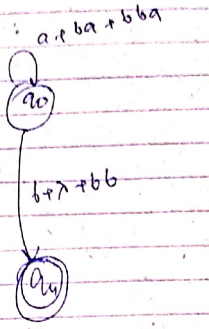
⇒



delete q_1



⇒



⇒ $(a+ba+bba)^*(b+bb+\lambda)$

Scanned by CamScanner

New method

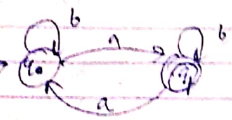
$R(i,j,k+1) = R(i,j,k) + R(i,k,k)\,R(k,k,k)^*\,R(k,j,k)$

$\gamma_{ij}^{k+1} = \gamma_{ij}^{k} + \gamma_{ik}^{k}\,\gamma_{kk}^{k*}\,\gamma_{kj}^{k}$

i → initial state j → final states

k+1 → no. of states

eg



$R(0,0,0) = \lambda + b$
$R(0,1,0) = a$
$R(1,0,0) = a$
$R(1,1,0) = \lambda + b$

$R(0,1,2) = ?$

∴ i→0, j→1, no. of states → 2

$R(0,1,2) = R(0,1,1) + R(0,1,1)\,R(1,1,1)^*\,R(1,1,1)$

$R(0,1,1) = R(0,1,0) + R(0,0,0)\,R(0,0,0)^*\,R(0,1,0)$

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

$R(1,1,1) = R(1,1,0) + R(1,0,0)\,R(0,0,0)^*\,R(0,1,0)$

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

$R(0,1,2) = a + (\lambda+b)^* a + \left(a + (\lambda+b)^* a\right)\left(\lambda + b + a(\lambda+b)^* a\right)^*$

$= a + b^* a + (a + b^* a)(\lambda + b + ab^* a)^*$

$= a + b^* a + (a + b^* a)(\lambda + b + ab^* a)^*$

$= b^* a + b^* a\,(b + ab^* a)^* \quad [\because a + b^* a = b^* a]$

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

8 Find correct regex for lang

L = { w | w contains even no. of 'a's }

(a) $(aa+b)^*$ ✗

(b) $b^*(ab^*a)b^*$ ✗

(c) $(ab^*ab^*)b^*$ ✗

(d) $b^*(ab^*a)^*$ aab ✗

(e) $(b^*(ab^*a^*)^*)^*$ ✓

(f) $(b^* ab^* + b^*)^*$ ✓ but ✗

(g) $b^*(ab^*ab^*)^*$ ✓

(h) $ab^*(b^*ab^*a)b^*$ ✗

Scanned by CamScanner

lecture-22

$P = (ab^*)^*$ $\quad P \subseteq Q$? ✗ $\quad R \subseteq Q$? ✗

$Q = (a^*b)^*$ $\quad Q \subseteq P$? ✗ $\quad P \subseteq R$? ✗

$R = a^*b^*$ $\quad Q \subseteq R$? ✗ $\quad R \subseteq P$? ✗

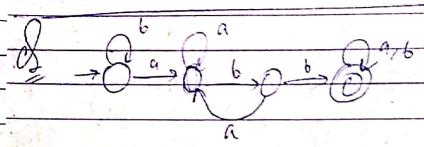
$P \subseteq Q \Rightarrow Q$ contains P.. (subset)

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

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

$(a+b)^* \not\subseteq (ab^* + a^*)^*$ [∵ b isn't in $(ab^* + a)^*$]

So, $(ab^* + a)^* \neq (a+b)^*$



M final correct RE

(a) $b^*a(a+ba)^*b(a+b)^*$ ✗

(b) $b^*aa^*b(aa^*b)bb^*(a+b)^*$ ✗

(c) $b^*aa^*b(ab)^*b(a+b)^*$ ✗

(d) $b^*a^*a(b^*a^*b)^*bb(a+b)^*$ ✗ aabbbb

(e) $(b^*a)^*a(a+ba)^*bb(a+b)^*$ ✗ babbb

(f) $(a+b)^*abb(a+b)^*$ ✓

abaabb

(g) $b^*a^*b^*(ab+b)^*abb(a+b)^*$ ✗ abaabbb

(h) $b^*aa^*(a+bb)^*bb(a+b)^*$ ✗ abaabb

(i) $b^*aa^*(a+\overset{ba}{ab}+bb)^*bb(a+b)^*$ ✓

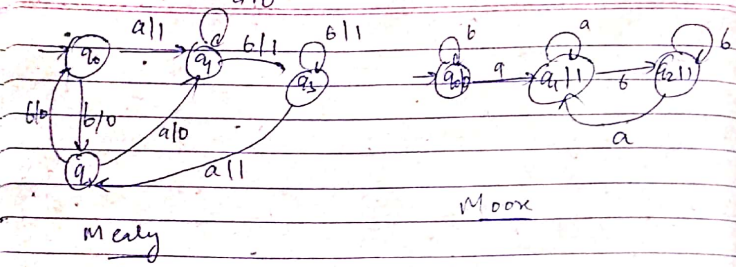
Mealy machine & Moore machine

It is also known as output machine in which there is one initial state but no final state. When we supply input, it produce output.

In Mealy, output is available on transition while in moore, it is available at state.

Mealy	Moore
$M = (Q, \Sigma, \delta, S, \Delta, T)$	$M_2 = (Q, \Sigma, \delta, \Delta, T)$
Q:	
Σ:	
δ: $\delta(Q, \Sigma) \to Q$	
$S \subseteq Q$	
Δ: output fⁿ $\Delta(Q, \Sigma) \to T$	$\Delta(Q) \to T$
T: output alphabets	

Scanned by CamScanner

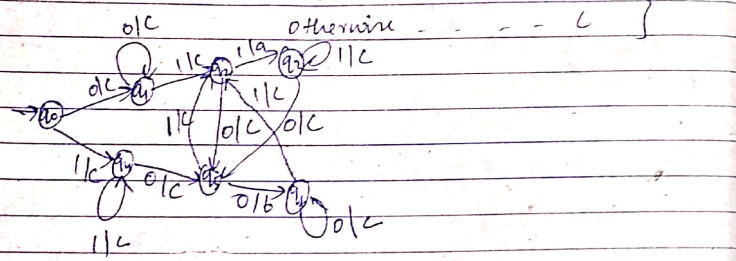


Q. Design Mealy & Moore for

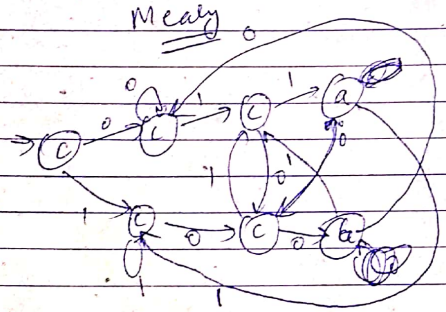
$L = \{w \mid$ when 011 occurs output 'a'

4 100 ... b

otherwise ... c $\}$



Mealy



Every moore → converted to mealy

Moore $\Delta(q_i) \to x \quad x \in \Gamma, \; q_i \in Q$

$\delta(q_j, \varepsilon) \to q_i$

Mealy $\Delta(q_j, \varepsilon) \to x$

Minimisation of mealy machine

Two states are equivalent if both states goto similar state & output is also similar at same input. So we can merge them.

	0	1	0	1
q_0	q_1	q_4	c	c
q_1	q_1	q_2	c	c
q_2	q_5	q_3	c	a
q_3	q_5	q_3	c	c
~~q_4~~	q_5	q_4	c	c (same as q_3)
q_5	q_4	q_2	b	c
~~q_6~~	q_1	q_2	c	b (same as q_1)

Scanned by CamScanner

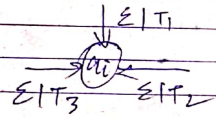
PAGE. NO
DATE: / /

How to identify two DFA are equivalent?

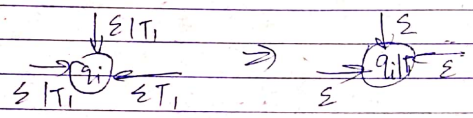
If initial state is equivalent then DFA(s) are equivalent.

⇒ for every inp., both should reach to final or non-final.

Mealy to moore



⇒ make 3 copies of q_i



size of output in moore machine =
1 + size of input

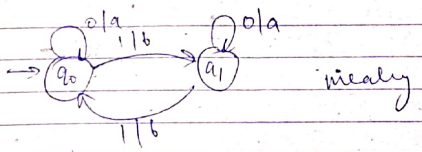
PAGE. NO
DATE: / /

In mealy machine

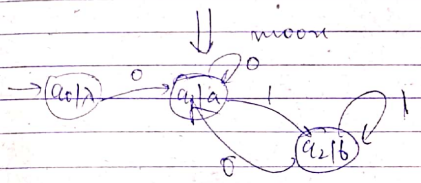
$|Q| = n$, $|\Sigma| = m$, $|T| = r$

then how many max. no. of states in equivalent moore machine

⇒ $n \times r + 1$



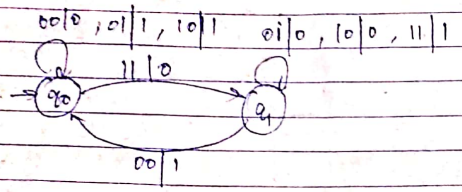
mealy



Scanned by CamScanner

V.V. Imp

Mealy machine to add two binary no.



q_0 : 0 carry

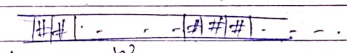
q_1 : 1 carry

Lecture-23

Turing Machine

Machine having two powers
- Acceptance of string or Enumeration of str
- Computation power

It has 1 tape of inf. size which contain a str. After str. it has inf. blank space represented by '#'.
There is 1 head that can read single char. at a time & it can decide new char. in curr. man & move left & right accordingly.



Computational machine

It is a turing machine which can compute/ perform any computation b/w numbers.

like a×b ; a+b, a^b, a/b etc.

Scanned by CamScanner

$L = \{ a^p : p = \text{prime} \}$

to perform addition

5 ⇒ IIIII 4 ⇒ IIII

~~input tape~~ : IIIII # IIII

for addⁿ : remove # in b/w & remove I from last

IIIII I IIII #

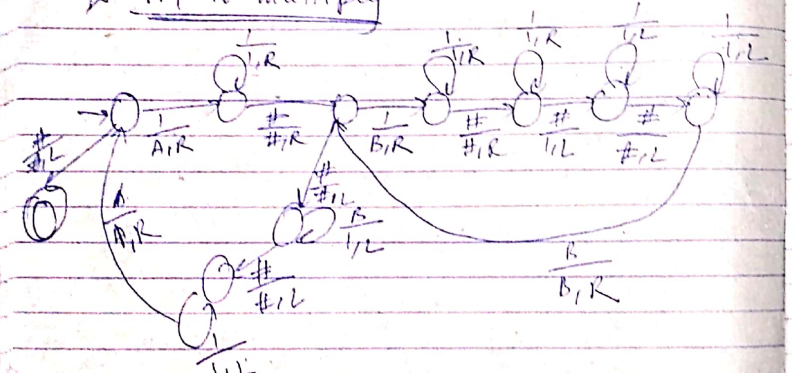
to perform multiply

III # II ⇒

III # II # II IIII

AAA 6

☆ TM to multiply



a^n , $n!$

TM of $n!$ present in video.

Idea is,

e.g. 5!

IIIII

copy all 5 ⇒ # IIIII# IIIII

delete 1 (I) & copy 5 4 time

⇒ # IIII# IIIII IIIIIIIIIIIIIIIII I#

20

repeat same

like delete 1 (I) & copy 20 3 time

when no (I) is present

then give string will be accepted.

Scanned by CamScanner

PAGE NO
DATE : / /

lecture-29

TM

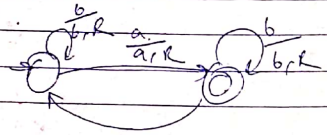
$M = (Q, \Sigma, \delta, S, f, \Gamma)$

Γ → tape symbol

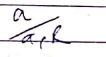
f: final state

h: halting state: when we reach there further there is no move.

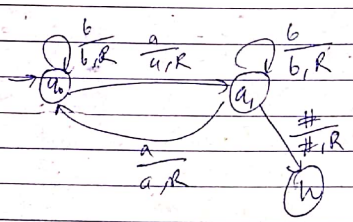
Design TM for $L = \{w \mid n_a(w)\%2=1, \Sigma=\{a,b\}\}$



just like DFA



final state TM



Halting state TM

$Q = \{q_0, q_1, h\}$, $\Sigma = \{a,b\}$, $S = q_0$, $\Gamma = \{a,b,\#\}$

$\delta(Q,\Gamma) \rightarrow \{(Q,\Gamma,D), (Q,\Gamma,D), \dots\}$

direction

PAGE NO
DATE : / /

FA ⟷ RL

PDA ⟷ CFL

(linear bounded automata) LBA ⟷ CSL

TM ⟷ Recursive Enumerable Lang.

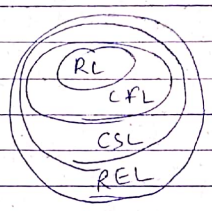
LBA (linear Bounded Automata)

$M = (Q, \Sigma, \delta, S, f, \Gamma, <, >)$

almost like TM except '<' & '>'

Ex. w is available in b/w < & >. & head can't move before < & after > symbols. There maybe some extra space b/w < & > but it is linear.

☆ It means we have not unlimited extra space in LBA like TM



Input format: < w >

Scanned by CamScanner

CSL

$L_1 = \{a^p ; p \in prime\}$ — REL not CSL

$L_2 = \{a^{2^n} ; n>=0\}$ — CSL

$L_3 = \{a^{n^2} ; n>=0\}$ — REL

$L_4 = \{wcw \mid w \in (a,b)^*\}$ — CSL

in L_4 ⇒ first ~~first~~ find mid

* Context-Sensitive Grammar

eg $L = a^n b^n c^n , n>0$

$S \rightarrow aSBC \mid \lambda$

$CB \rightarrow BC$

$aB \rightarrow ab$

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

Lang. = Problem

Decidable problm/lang.

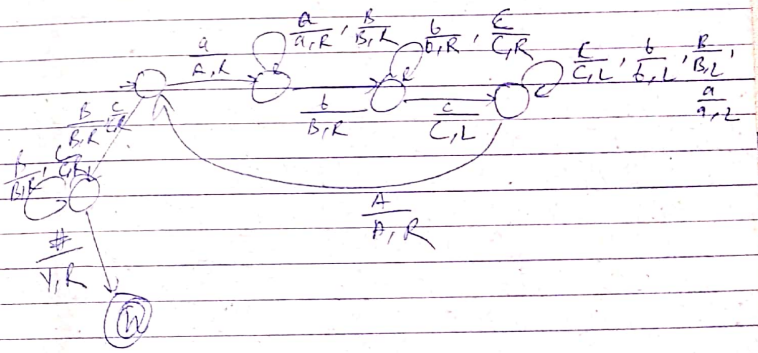
Def 1 A lang. L is known as decidable if there exists for L & $\bar{L}$. OR

Def 2 If L is acceptable & $\bar{L}$ is also acceptable then L is decidable language.

Acceptable lang.: If ∃ TM of L then it is acceptable

Def 3 TM reaches to halting state for any input w by writing YES when w∈L otherwise NO if w∉L.

$L = \{a^n b^n c^n ; n>0\}$



Scanned by CamScanner

Decidable lang. also known Recursive lang. / Tractable problm / solvable problm

def

If there exists deterministic TM having polynomial time complexity then it is decidable

↳ definition of Algorithm

Time Complexity : no. of moves in TM is known as its complexity.

$L = \{a^n b^n c^n\}$

$n(2n+2n) + 3n = 4n^2 + 3n$

def 5 If there exist Algo (in any lang) having polynomial time complexity.

problm / Intractable problem

Undecidable problm :

If a problm is not decidable → undecidable

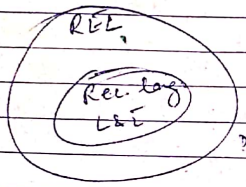
Lecture-25

PAGE NO
DATE: / /

Undecidibility : depends on time

Recursive lang: If L & $\bar{L}$ both are acceptable

Recursive Enumerable lang: Acceptable lang.



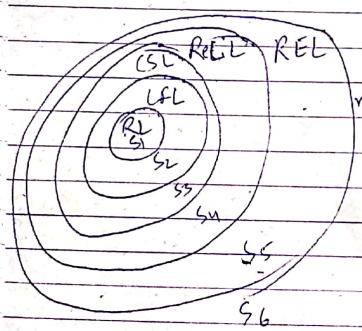
REL ⇒ L is acceptable

RL ⇒ L & $\bar{L}$ is acceptable

All recursive lang. are recursive enumerable.

If L is recursive then $\bar{L}$ is also recursive

Complement of same not REL is REL.



$L_i \in S_i$ & $L_j \in S_j$ & $i \neq j$

then :

$L_i \cap L_j \in S_j$

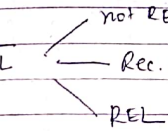
$L_i \cup L_j \in S_j$

PAGE NO
DATE: / /

Prop.

REL ∩ REL = REL ∪ REL = REL

Rec.L ∩ Rec.L = Rec.L ∪ Rec.L = $\overline{\text{Rec.L}}$ ∪ Rec.L = Rec.L

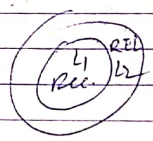


let $L_1 \in$ Rec.L

$L_2 \in$ REL but not Rec.L

$L_3 \in$ REL.

$\bar{L}_1 \cap L_2$ = REL or (REL − Rec.L)



$\bar{L}_2 \cup L_1$ = not RE

$L_1 \cap L_3$ = Rec.L or REL or (REL − Rec.L)

$L_1 \cap \bar{L}_3$ = not REL, Rec.L, REL

Scanned by CamScanner

PAGE NO
DATE: / /

List of Undecidible Problems

About TM

1. Whether TM M accepts string w or not. It is also known as halting problm.

TM → halting state : no further transition
→ final State : further transition from this state is possible.

2. Whether two TM(s) M_1 & M_2 accepts same string or not!

3. Let M_1 & M_2 are two TM(s)
   (a) $L(M_1) \subseteq L(M_2)$
   (b) $L(M_1) \cap L(M_2) = \phi$
   (c) $L(M_1) = L(M_2)$

4. Whether TM M accepts ϕ or empty language

5. Whether TM M accepts
   (a) RL
   (b) CFL
   (c) CSL

Decidable

6. Whether TM M reads 6 tape symbols or not.

7. Whether TM M reaches to given state q_x or not ?

~~If q_x It is undecidable~~

∴ If it is decidable & q_x = halting state then halting problm is also decidable.

About Grammers

All problms of TM(s) is similiar to Grammer also both of every grammer can be converted into TM & vice versa.

About CFL :

1. Let G_1 & G_2 are two CFG(s)
   G_1 is ambigous
   $L(G_1) = L(G_2)$
   $L(G_1) \subseteq L(G_2)$
   $L(G_1) \cap L(G_2) = \phi$

2. Whether $L(G_1)$ accepts RL

Scanned by CamScanner

$L = \{ w \mid w \text{ contains } 3k\ q's \}$

Eg $S \to SaSaSaS \mid bS \mid \lambda$

CFG but not RL

∴ It is RL but not RG so undecidable

1 Decidable (using parsing)

3. Whether $w \in L(G)$?

Decidable

4. Whether $L(G_{11})$ contains infinite strings or not.

5. Any closure prop. which isn't satisfied closed or not.

2 $L_1 \cup L_2 \to$ may be DCFL or not

DCFL DCFL

2 So, it is undecidable

6. Inherently Ambiguous or not ?

3

Algorithm

decidable

1. M_1 & M_2 are DFA

$L(M_1) = L(M_2)$?

It is decidable by checking that initial state of both DFA is equivalent.

i.e, $\delta(q_{0_{M_1}}, w) = \delta(q_{0_{M_2}}, w) =$ final or non final

decidable

2. Whether subset of $L(M_1)$ is regular

Union (U) $\to$ RL

$RL_1 \cup RL_2 \to RL$

Infinite Union of RL $\to RL_X$

Eg $\lambda, ab, anbn, anbnbb, \ldots$

regular lang

Union of all

$a^n b^n$ (not regular)

Scanned by CamScanner

PAGE NO
DATE: / /

Countable Set :

q^* → countable set

a^*b^* → ~~countable set~~

Set of all programs ∧ are countable (or TM)

Yes ⇒ bcoz finite person on earth, who has finite life span.

So, we can count them.

subset of languages over Σ^* $\Sigma = \{a,b\}^*$ is not countable

$\because 2^{\Sigma^*}$ is not countable

$2^{\text{finite set}}$ is

$2^{\text{infinite set}}$ is

no. of real no. b/w [0-1] is infinite & uncountable

Scanned by CamScanner