PureLogics
No magic. Just logic.

Date:

Pumping Lemma for Regluar Lang.

Palendrome	$0^p 1 0^p$
Non-Palindrome	$0^p 1 0^{p-1}$
ww	$0^p 1 0^p 1$ $\because w = 0^p 1$
0^{n^2}	0^{p^2} $u: 0^r$ $v: 0^s$ $y: 0^{p^2-r-s}$
0^{2n}	0^{2p} $u: 0^r$ $v: 0^s$ $y: 0^{2p-r-s}$
prime numbers	0^p $u: 0^{p-r-s}$ $v: 0^r$ $y: 0^s$

valid string $\longrightarrow$ not a constant string
contain variable

$|x|$ $|uv| \leq p$; $|v| \geq 1$

$\downarrow$

Decompose into $u\, v^i\, y$

$i = 0$ pump Down $i > 0$ pump up

if resultant $x \notin L$ then

L is not regular Language.

Context Free Grammer (CFG)

(Have base and recursion cases)

four tuples

1) Σ Terminals   2) V Variable

3) S Start variable   4) $P \to \alpha \underbrace{(V \cup \Sigma)^*}_{\text{could be } \Lambda}$

production

(1) one or more rule

$a^n b^m$ $m > n$

$a^n b^n . b^{1,2,3,\ldots}$ $S \longrightarrow S_1 S_2$

$S_1 \to aSb$

$S_2 \to bS_2 | b$

Date:

$a^n b^m$ $\quad n \geq m \quad$ $n>0, m>0$

(•) These are base cases

Two Different Languages

$(n > m) \cdot M$ $\qquad$ $(n == m) \cdot L$

as discussed earlier

$S \rightarrow M \mid L$

When $(a^n b^m)^k$ $\qquad n=0$

$S \rightarrow AS$

$A \rightarrow aAb \mid \wedge$

Trace Tree or Derivation Method for a string

(•) string ababaa

$S \rightarrow AA^{(1)}$

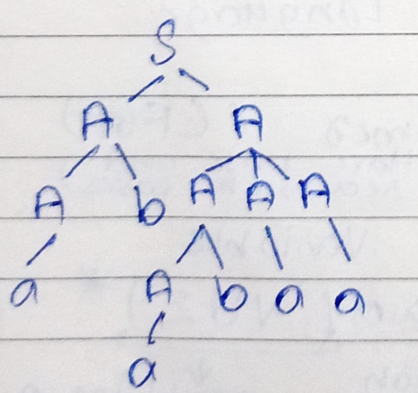
$A \rightarrow AAA^{(2)} \mid bA^{(3)} \mid Ab^{(4)} \mid a^{(5)}$

$S \xrightarrow{1} AA \xrightarrow{4} AbA \xrightarrow{5} abA$

$\xrightarrow{2} abAAA \xrightarrow{4} abAbAA$

$\xrightarrow{5} ababAA \xrightarrow{5} ababaA$

$\xrightarrow{5} ababaa$



CFG to CNF

When not used:

(•) Null production is only acceptable in start variable.

(•) $V \rightarrow VV$: Only two variable production

(•) $V \rightarrow \Sigma$: single terminal production

Rules for conversion :

1) Make new start variable

2) Remove null production

, if $A \rightarrow SaS \mid C$
$S \rightarrow ASa \mid \wedge$ then

$A \rightarrow SaS \mid C \mid aS \mid Sa$
$S \rightarrow ASa$

3) Remove unit production, if

$A \rightarrow S \mid AB$
$S \rightarrow aA \mid bA \mid C$

then,

$A \rightarrow aA \mid bA \mid C \mid AB$
$S \rightarrow aA \mid bA \mid C$

4) Ensures $V \rightarrow \Sigma$

for defaulty terminals make new productions like $X \rightarrow x$ or $Y \rightarrow y$

5) Ensures $V \rightarrow VY$ for $P \rightarrow ABC$

make $T \rightarrow AB$, $P \rightarrow TC$

Date: ____________________

CYK Parser Algo:

(1) Grammer should be in CNF:
then |x| length of string
make 2D matrix of size |x| × |x|

y-axis iterations

j=4	S_0 {{{}}			
j=3	ϕ {{}}	ϕ {{}}		
j=2	ϕ {{	S_0, S {}	ϕ }}	
j=1	X	X	Y	Y
	{	{	}	}

x-axis input string

for
j = 1 : { { } }
↓
found through X in grammer

for j = 2: {{ , {}, }}
XX XY ϕY
↓
found by
S_0/S

for j = 3
{{ } , { }}
→ { , { } cross product of both cells → { , }}
→ {{ } then search in grammer → {} , }

Pure Logics
No magic. Just logic.

Date:

FA + stack

PDA (Push Down Automata)

7 tuples

- Σ set of alphabets
- Q set of states
- S start state
- F_n set of final states
- δ_0 transition function
- Γ stack elements $\{Z_0, A, B\}$
- Z_0 top of stack

$a, \underline{pop} / \underline{push}$

- (•) No operation $a, Z_0 / Z_0$
- (•) pop $a, Z_0 / \epsilon$
- (•) push $a, Z_0 / AZ_0$ (push A)
- (•) replace $a, Z_0 / A$ (replace)

Date:

Regular Expression

Define / Express Regular Language

(•) Operation   union $(+)$ concate $(\bullet)$ star $*$ positive star $+$

parenthesis $()$

$(a+b) \rightarrow$ anyone from a and b

$ab \rightarrow$ first a then b

$(a+b)^* \rightarrow (a+b)^0 | (a+b)^1 | (a+b)^2 | (a+b)^3$

concate result $(a+b)(a+b)$

$(a+b)^+ \rightarrow$ same as above except $(a+b)^0$ can't null.

Operands (will be alphabets)

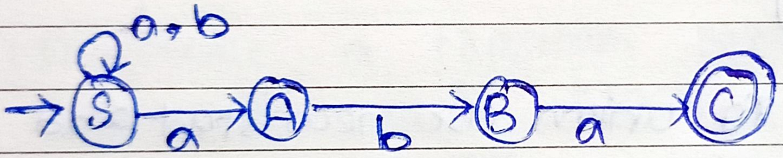
NFA to DFA
(subset construction Method)

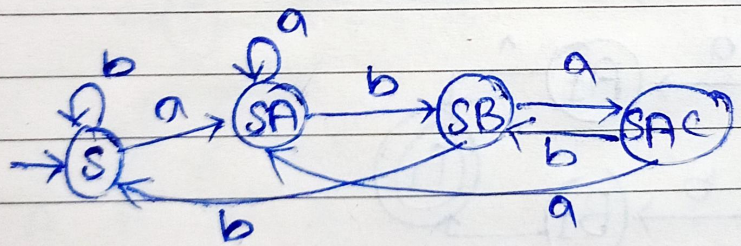
(•) from NFA make transition table to clear every transition for simplicity

(•) Now start with start state and make a combine state for its 'a' transition (combine all nodes (states) that can be reachable by a by start state), same for b.

(•) Now for next state take into consideration all combine states value of 'a' and 'b' and make new states.

e.g :





Transition Table

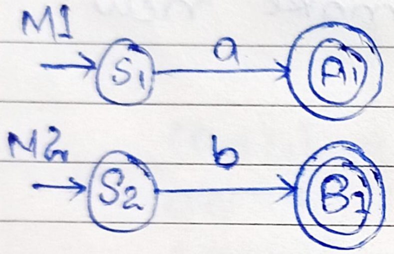
	a	b
S	SA	S
A	X	B
B	C	X
C	X	X

(•) As C is final state so whereever C exists it becomes final state.

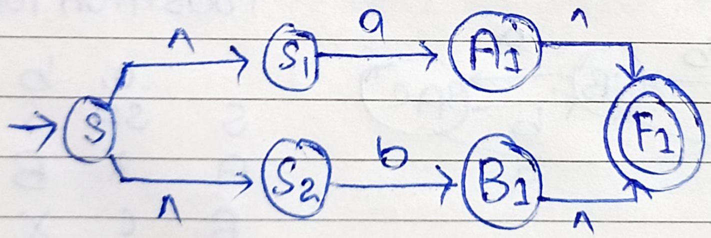
RE to NFA-Λ

$$\underbrace{\underbrace{(\underbrace{a}_{M1}+\underbrace{b}_{M2})}_{M3}{}^{*}}_{M4}\ \underbrace{a}_{M5}$$

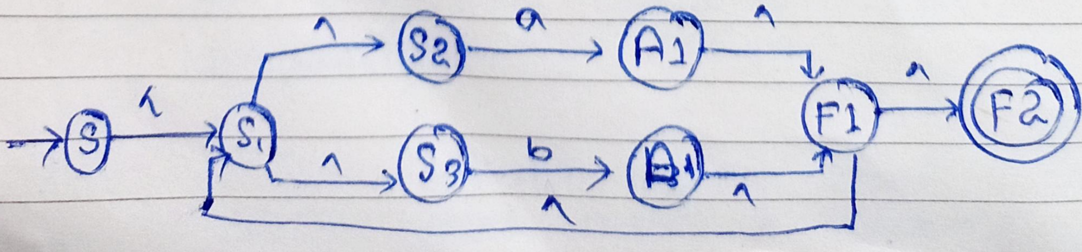
$M6$ = the whole expression $(a+b)^* a$



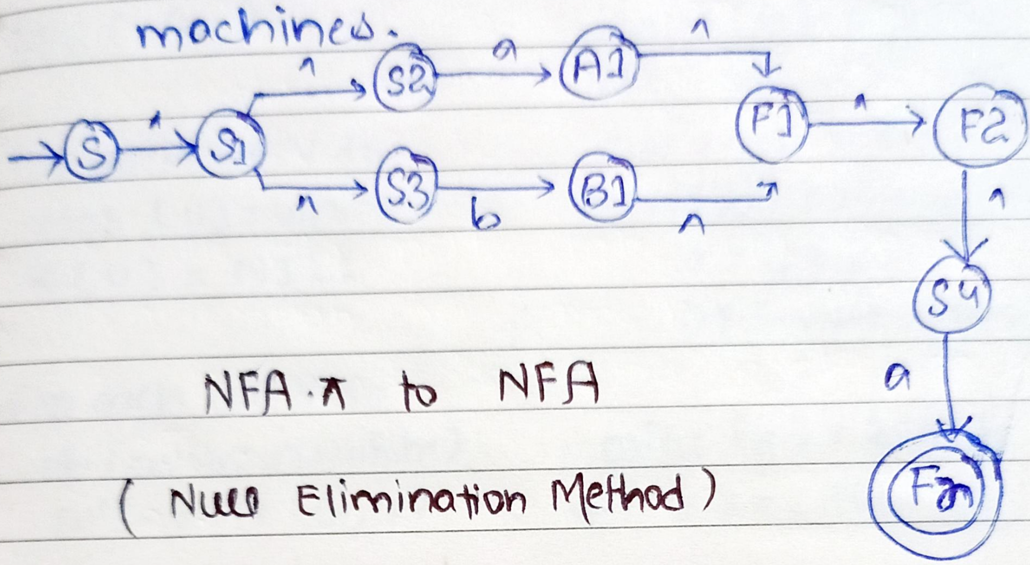
(•) To union use new start and end state by Λ transitions



(•) for * use new start and end state and make transition then make null transition of previous start and final state



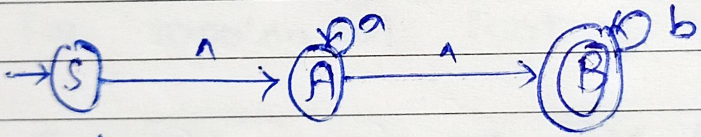
(•) For contatination just combine 2 machines.



NFA-Λ to NFA

(Null Elimination Method)

(1) Make a transition table also include all the null transitions.



	a	b	Λ
S	x	x	A, B
A	A	x	B
B	x	B	x

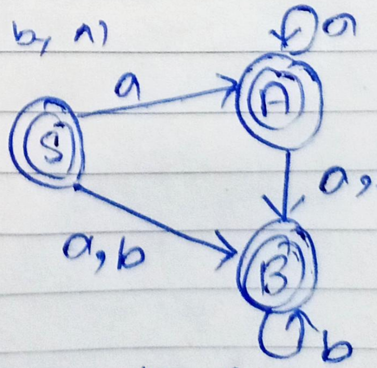
new Table

	a	b
S	A, B	B
A	A, B	B
B	x	B

Method $\Lambda^* \Sigma \Lambda^*$ (input alphabets) (state reachable by Λ)

	Λ^*	Σ a Λ^*	b Λ^*
S	S	x	x
	A	A < B, A	x
	B	x	B B

similarly all



(•) states reachable by Λ to final are all final states.