Regular Expression to DFA 29/11/19.

Step 1: Argumented Regular Expression.

Step 2: construct syntax tree.

Step 3: Fixed nullable, lastpse, firstpost

Step 4: Fixed folowpos.

Step 5 : Convert DFA.

Direct Method.

Fixed nullable

Node	constant	Nullable
ϵ		T
a		F
*		T
/		N(C) or N(R)
.		N(C) and N(R)

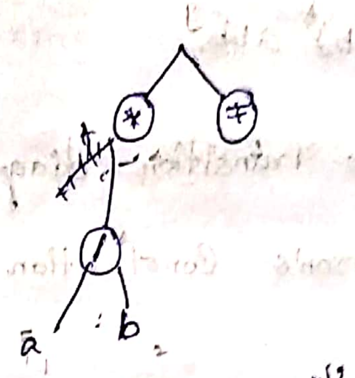
Node	function Firstpos	Lastpos
*	Firstpos (c)	Lpos (c)
/	Firstpos (L) ∪ Firstpos (R)	Lpos(L) ∪ Lpos(R)
.	if N(L) is true firstpos (L) ∪ firstpos(R) else firstpos(L)	if N(R) is true Lpos(L) ∪ Lpos(R) else Lpos(R)
Leaf label with ϵ	ϕ	ϕ
leaf node	pos(i)	pos(i)

1) $(a/b)^*$.

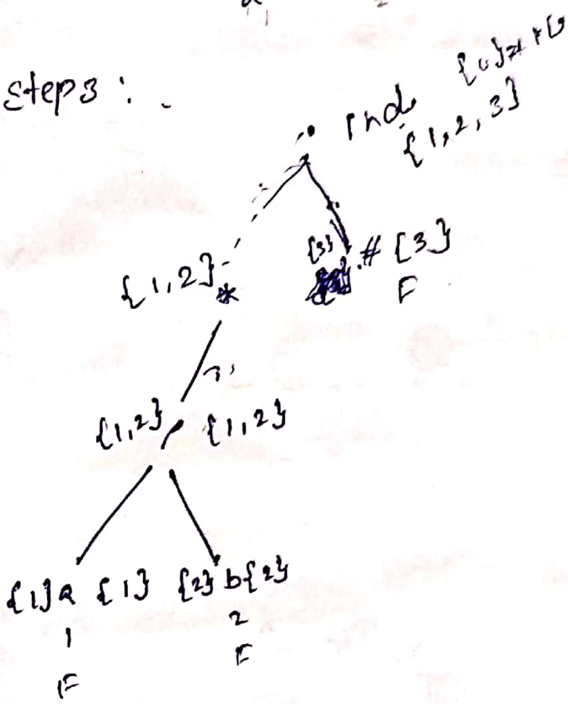
Step1: Augumented Regular Expression:

$$(a/b)^* = (a/b)^* \cdot \#/$$

Step 2: construct syntax Tree.



Step3:



$*$, . only have in followposition.

Find followposition:-

Node	followposition.
1	{1, 2, 3}
2	{1, 2, 3}
3	

construct the DFA:-

DFA = {Q, A, Σ, q_0, F}; Q → q_0, Σ = {a, b},

q_0 = {1,2,3} F → {3}

$\delta\{(q_0), b\} = \{(1,2,3), b\}$

$\delta\{(q_0), a\} = \{(1,2,3), b\}$

followpos(1) = {1, 2, 3}.

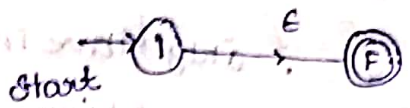
construct DFA ~~for~~ (a/b)* abb..

Regular Expression to Transition diagram.

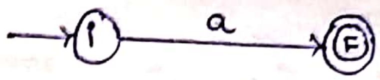
Using Thompson's Construction

Rules :-

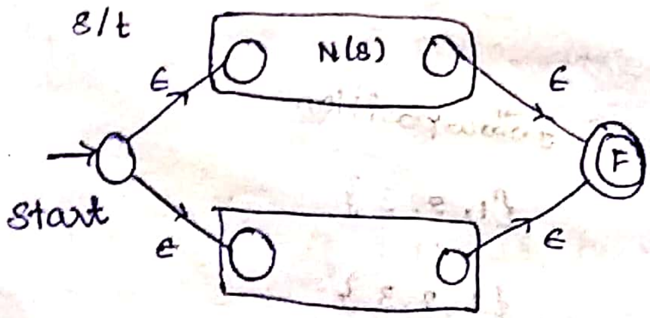
1) For ϵ, construct NFA



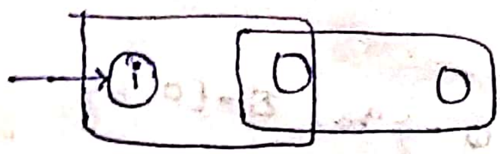
2) For a



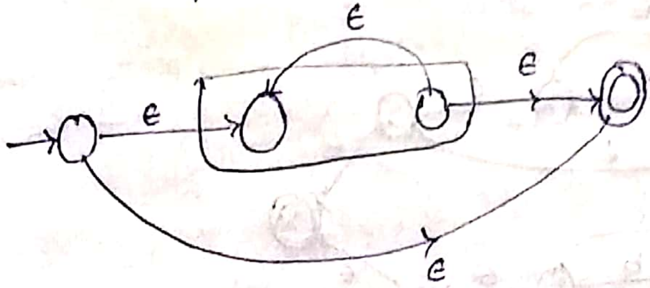
3) For regular expression



4). For regular expression st

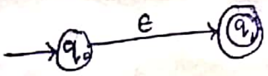


5) For regular express s^*

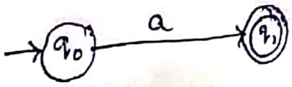


Regular Expression to NFA :-

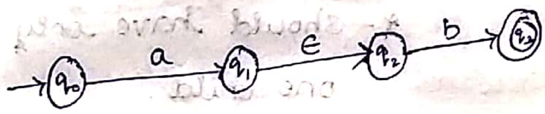
1) ε .

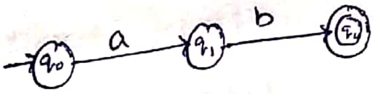


2) a

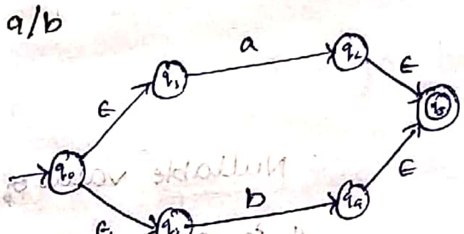


3) ab

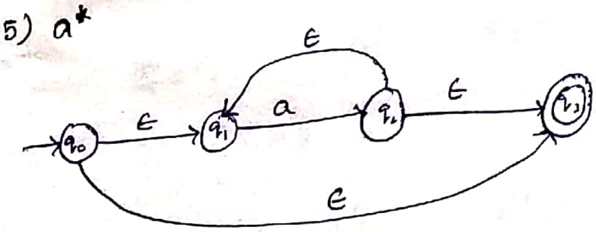




4) a/b



5) a^*



Transition function (δ):

$\delta\{[q_0], a\} = \{[1,2,3], a\} = \{(1,a) \cup (2,a) \cup (3,a)\}$

$= (1,a)$

$\delta\{[q_0], b\} = \{[1,2,3], b\} = \{(1,b), (2,b), (3,b)\}$

$= (2,b)$

Follow posi (1) = {1,2,3}

Follow position (2) = {1,2,3}

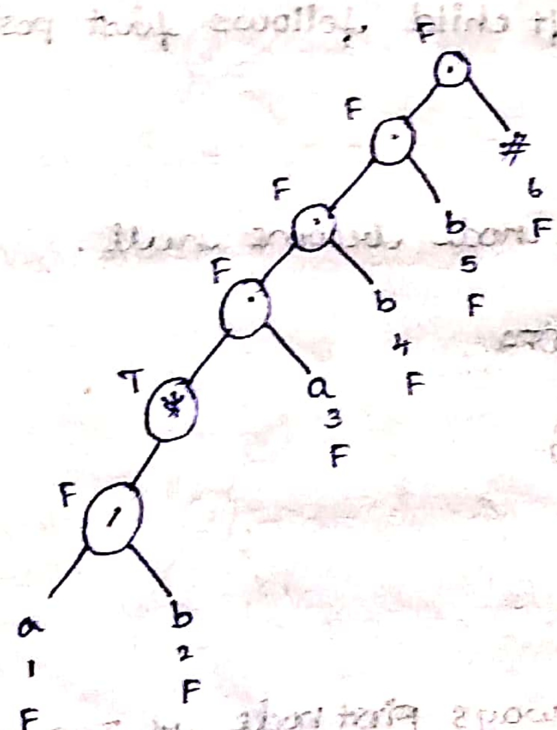
Transition table:

		a	b
*→	q_0	{1,2,3}	{1,2,3}

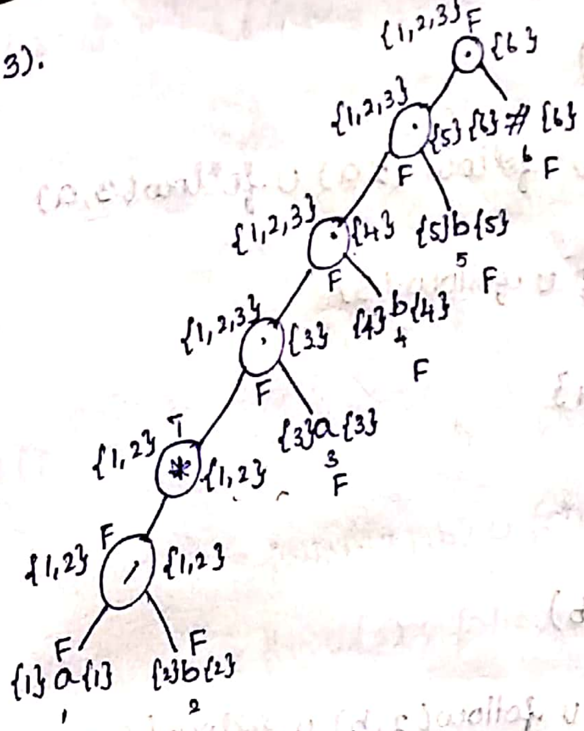
2). $(a/b)^*.abb$.

Step 1: Augumented Regular Expression: $(a/b)^*.abb.\#$

Step 2: construct syntax tree and nullable:



3).



4) Find follow position :-

Node	follow position
1	{1,2,3}
2	{1,2,3}
3	{4}
4	{5}
5	{6}
6	

5) construct DFA :

DFA, M = (Q, Σ, δ, q_0 , F }

Q -> q_0

Σ -> {a,b}

q_0 -> {1,2,3}

F = { 6 }

Transition table :

	a	b
-> q_0	{1,2,3,4} - q_1	{1,2,3} - q_2
q_1	{1,2,3,4}	{1,2,3,5} - q_3
* q_2	{1,2,3,4}	{1,2,3,6}
* q_3	{1,2,3,4}	{1,2,3}

$\delta([q_0], a) = ([1,2,3], a)$

$= \text{follow}(1,a) \cup \text{follow}(2,a) \cup \text{follow}(3,a)$

$= \text{follow}(1) \cup \phi \cup \text{follow}(3)$

$= \{1,2,3\} \cup \{4\}$

$= \{1,2,3,4\}$

$\delta([q_0], b) = ([1,2,3], b)$

$= \text{follow}(1,b) \cup \text{follow}(2,b) \cup \text{follow}(3,b)$

$= \phi \cup \text{follow}(2,b) \cup \phi$

$= \{1,2,3\}$

$\delta([q_1], a) = ([1,2,3,4], a)$

$= \text{follow}(1,a) \cup \text{follow}(2,a) \cup \text{follow}(3,a) \cup \text{follow}(4,a)$

$= \text{follow}(1) \cup \text{follow}(3)$

$= \{1,2,3\} \cup \{4\} = \{1,2,3,4\}$

$\delta([q_1], b) = ([1,2,3,4], b)$

$= \text{follow}(1,b) \cup \text{follow}(2,b) \cup \text{follow}(3,b) \cup \text{follow}(4,b)$

$= \text{follow}(2) \cup \text{follow}(4)$

$= \{1,2,3\} \cup \{5\}$

$= \{1,2,3,5\}$.

$\delta([q_2], a) = ([1,2,3,5], a)$

$= \text{follow}(1,a) \cup \text{follow}(2,a) \cup \text{follow}(3,a) \cup \text{follow}(5,a)$

$= \text{follow}(1) \cup \text{follow}(3)$

$= \{1,2,3,4\}$

$\delta([q_2], b) = ([1,2,3,5], b)$

$= \text{follow}(1,b) \cup \text{follow}(2,b) \cup \text{follow}(3,b) \cup \text{follow}(5,b)$

$= \text{follow}(2) \cup \text{follow}(5)$

$= \{1,2,3,6\}$

$\delta([q_3], a) = ([1,2,3,6], a)$

$= \text{follow}(1,a) \cup \text{follow}(2,a) \cup \text{follow}(3,a) \cup \text{follow}(6,a)$

$= \text{follow}(1) \cup \text{follow}(3)$

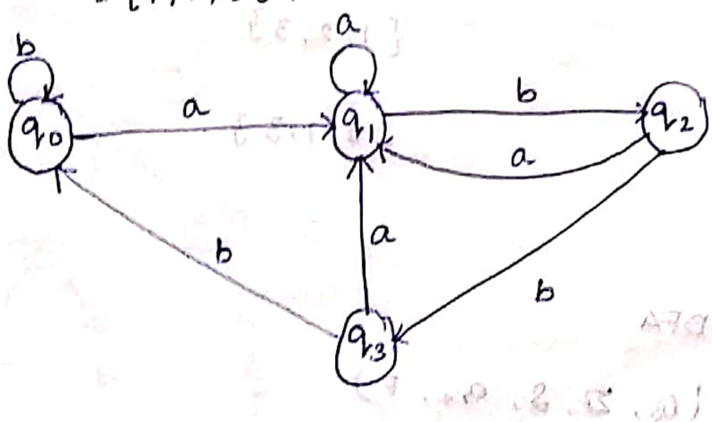
$= \{1,2,3,4\}$

$\delta([q_3], b) = ([1,2,3,6], b)$

$= \text{follow}(1,b) \cup \text{follow}(2,b) \cup \text{follow}(3,b) \cup \text{follow}(6,b)$

$= \text{follow}(2) \cup \text{follow}(6)$

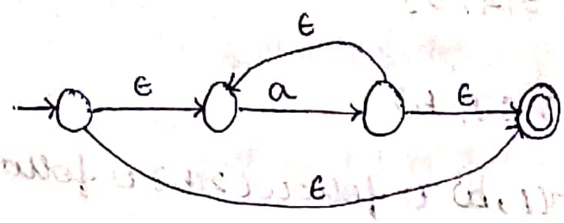
$= \{1,2,3\}$.



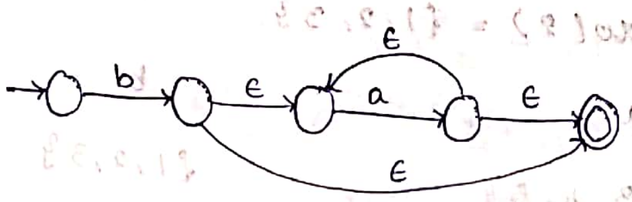
3/12/19

Construct NFA for Regular Expression : $a+b.a^*$ by using Thomson's construction method:-

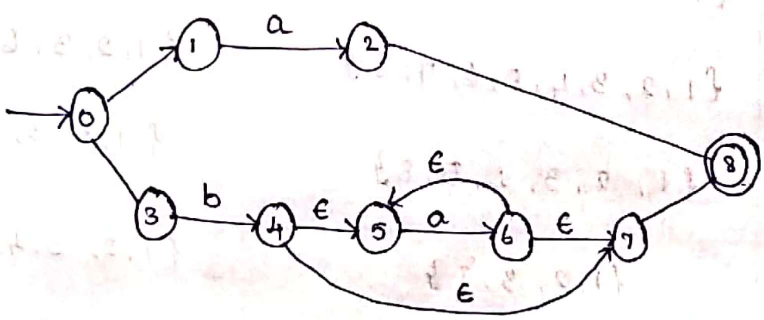
Step 1:



Step 2:

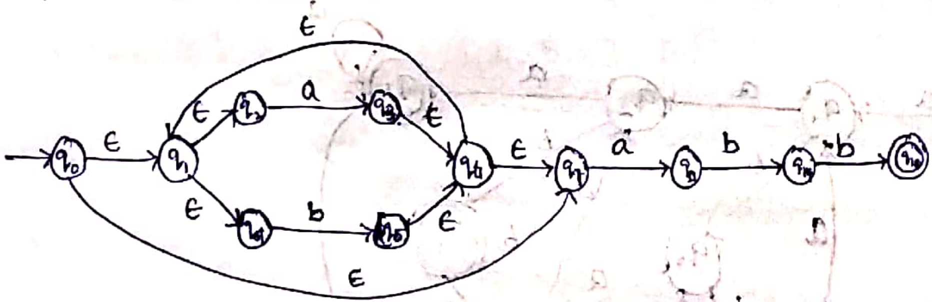


Step 3:



For the regular Expression $(a/b)^*abb$ find the DFA using subset construction method.

Step 1: construct ε-NFA:



ε-closure (q_0) = [q_0, q_1, q_2, q_4, q_7] - A

ϵ- closure (q_1) = [q_1, q_2, q_4]

ϵ- closure (q_2) = [q_2]

ϵ -closure (q_3) = [$q_3, q_6, q_7, q_1, q_2, q_4$]

ϵ- closure (q_4) = [q_4]

ϵ- Closure (q_5) = [$q_5, q_6, q_7, q_1, q_2, q_4$].

ϵ- closure (q_6) = [q_6, q_7, q_1, q_2, q_4]

ϵ- closure (q_7) = [q_7]

ϵ - closure (q_8) = [q_8]

ϵ- closure (q_9) = [q_9]

ϵ- closure (q_{10}) = [q_{10}]

ϵ- closure (A, a) = ϵ - closure (Move (q_0, q_1, q_2, q_4, q_7), a)

= ϵ-closure ($(q_0, a) \cup (q_1, a) \cup (q_2, a) \cup (q_4, a)$
$\cup (q_7, a)$)

= ϵ- closure ($\phi \cup \phi \cup q_3 \cup \phi \cup q_8$)

= ϵ- closure (q_3, q_8)

= ϵ - closure $(q_3) \cup \epsilon$. closure (q_8)

= [$q_3, q_6, q_7, q_1, q_2, q_4$] $\cup$ [q_8]

= [$q_1, q_2, q_3, q_4, q_6, q_7, q_8$] —(B)

ϵ- closure (A, b) = ϵ- closure (Move (q_0, q_1, q_2, q_4, q_7), b)

= ϵ- closure ($(q_0, b) \cup (q_1, b) \cup (q_2, b) \cup$
$(q_4, b) \cup (q_7, b)$)

= ϵ- closure ($\phi \cup \phi \cup \phi \cup q_5 \cup \phi$)

= ϵ - closure (q_5)

= [$q_1, q_2, q_4, q_5, q_6, q_7$] —— (C)

ϵ-closure (B, a) = ϵ-closure (Move $(q_3, q_8, q_6, q_7, q_1, q_2, q_4), a)$

= ϵ-closure $((q_3, a) \cup (q_6, a) \cup (q_7, a) \cup (q_8, a)$

$\cup (q_1, a) \cup (q_2, a) \cup (q_4, a))$

= ϵ-closure $(\phi \cup \phi \cup q_8 \cup \phi \cup q_3 \cup \phi)$

= ϵ-closure $(q_8) \cup \epsilon$-closure(q_3)

= $[q_1, q_2, q_3, q_4, q_6, q_7, q_8] \rightarrow B$

ϵ-closure (B, b) = ϵ-closure (Move $(q_8, q_1, q_2, q_3, q_4, q_6, q_7), b)$

= ϵ-closure $((q_1, b) \cup (q_2, b) \cup (q_3, b) \cup (q_4, b) \cup$

$(q_8, b) \cup (q_6, b) \cup (q_7, b))$

= ϵ-closure $(\phi \cup \phi \cup \phi \cup q_5 \cup \phi \cup \phi \cup q_9)$

= $(q_1, q_2, q_4, q_5, q_6, q_7, q_9) \rightarrow D$

ϵ-closure (C, a) = ϵ-closure (Move$(q_1, q_2, q_4, q_5, q_6, q_7), a)$

= ϵ-closure$((q_1, a) \cup (q_2, a) \cup (q_4, a) \cup (q_5, a)$

$\cup (q_6, a) \cup (q_7, a))$

= ϵ-closure $(q_3 \cup q_8)$

= $(q_1, q_2, q_3, q_4, q_6, q_7, q_8)$ — (B)

ϵ-closure(C, b) = ϵ-closure (move$(q_1, q_2, q_4, q_5, q_6, q_7), b)$

= ϵ-closure $((q_1, b) \cup (q_2, b) \cup (q_4, b) \cup (q_5, b)$

$\cup (q_6, b) \cup (q_7, b))$

= ϵ-closure (q_5)

= $(q_1, q_2, q_4, q_5, q_6, q_7)$ — (C)

ϵ-closure $(D, a) = \epsilon$-closure $(move(q_1, q_2, q_4, q_5, q_6, q_7, q_9), a)$

$= \epsilon\text{-closure}((q_1, a) \cup (q_2, a) \cup (q_4, a) \cup (q_5, a) \cup$

$(q_6, a) \cup (q_7, a) \cup (q_9, a))$

$= \epsilon\text{-closure}(q_3 \cup q_8)$

$= (q_1, q_2, q_3, q_4, q_6, q_7, q_8)$ — (B)

ϵ-closure $(D, b) = \epsilon$-closure $(move(q_1, q_2, q_4, q_5, q_6, q_7, q_9), b)$

$= \epsilon\text{-closure}((q_1, b) \cup (q_2, b) \cup (q_4, b) \cup (q_5, b)$

$\cup (q_6, b) \cup (q_7, b) \cup (q_9, b))$

$= \epsilon\text{-closure}(q_5 \cup q_{10})$

$= (q_1, q_2, q_4, q_5, q_6, q_7, q_{10})$ — (E)

ϵ-closure $(E, a) = \epsilon$-closure $(move(q_1, q_2, q_4, q_5, q_6, q_7, q_{10}), a)$

$= \epsilon\text{-closure}((q_1, a) \cup (q_2, a) \cup (q_4, a) \cup (q_5, a)$

$\cup (q_6, a) \cup (q_7, a) \cup (q_{10}, a))$

$= \epsilon\text{-closure}(q_3 \cup q_8)$

$= (q_1, q_2, q_3, q_4, q_6, q_7, q_8)$ — (B).

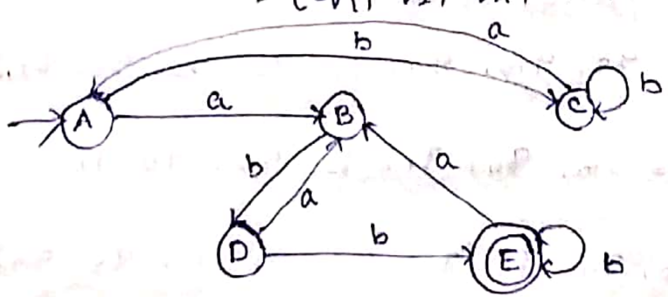
ϵ-closure $(E, b) = \epsilon$-closure $(move(q_1, q_2, q_4, q_5, q_6, q_7, q_{10}), b)$

$= \epsilon\text{-closure}((q_1, b) \cup (q_2, b) \cup (q_4, b) \cup (q_5, b) \cup$

$\cancel{(q_6, b)} \cup (q_6, b) \cup (q_7, b) \cup (q_{10}, b))$

$= \epsilon\text{-closure}(q_5)$

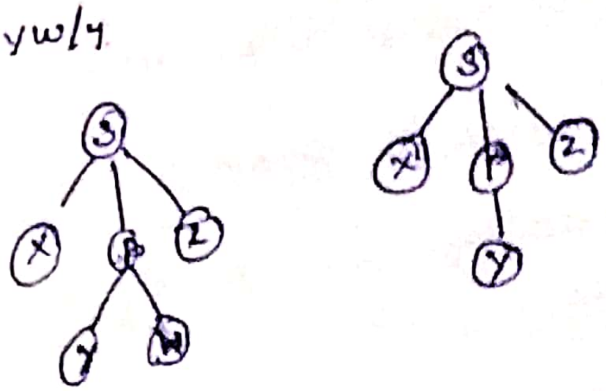
$= (q_1, q_2, q_4, q_5, q_6, q_7)$ — (C)



eg: for Backtracking

S → xPz

P → yw/y



Ex for predictive follows

- First()
- Follow();
- Parsing table

Left Recursion:

if a production $A \xrightarrow{*} A\alpha / B$ and remove

then
$A \to \beta A'$
$A' \to \alpha A' / \epsilon$.

Left side & right side of the production is LR

1). Remove the left Recursive

$E \to E + T / T$;

$T \to T * F / F$;

$F \to (E) / id$;

$A \to A\alpha / \beta$

$A \to \beta A'$

$E \to E + T / T$

$T \to T * F / F$

$E \to E + T / T$;
$E \to TE'$
$E' \to +TE' / \epsilon$

(A) for $E \to E + T / T$;

construct a predictive parser table for the following production

$E \rightarrow E+T \mid T$ — (1)

$T \rightarrow T*F \mid F$ — (2)

$F \rightarrow (E) \mid id$ — (3)

Step 1: production (1) and (2) have left recursion.

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \epsilon$

$T \rightarrow FT$

$T' \rightarrow *F' \mid \epsilon$

$F \rightarrow (E) \mid id$

First(E) = {First(T)}

First(T) = {First(F)}

First(F) = {(, id}.

Step 2:

First (E) = { (, id }

First (E') = { +, ε }

First (T) = { (, id }

First (T') = {*, ε}

First (F) = { (, id }

step 3 :

Follow(E) = { \$,) } => [Follow(E) =) = First( ))
=)]

Follow (E') = { Follow(E) } [E' have no follow so
Follow(E') = follow(E)]
= { \$,) }

Follow(T) = { ~~Follow (E')~~ }
= { ~~Follow(E)~~

Follow(T) = { + , Follow(E) } [E -> TE' => First(E')
= { + , \$,) } => { + , ε }
so Follow(E)]

Follow(T') = { Follow(T) } [T' has no follow so
= { + , \$,) } Follow(T') = Follow(T)]

Follow(F) = { * , Follow(T) } [T -> FT' => First(T')
= { * , + , \$,) } => { * , ε }
Follow(T)]

2) $S \rightarrow Bb \,/\, Cd$

$B \rightarrow aB \,/\, \epsilon$

$C \rightarrow cC \,/\, \epsilon$

Foll

16|12|19.

$S \rightarrow Bb \,/\, aBb \,/\, b \,/\, Cd \,/\, cCd \,/\, d$

Step 1 : No left recursion.

Step 2 :

First $(S) = \{ a, b, c, d \}$

First (B) = { a, ε }

First (c) = { c, ε }

Step 3 :

Follow(S) = { \$ }

Follow (B) = { b }

Follow(C) = { d }

4) S → ACB / CbB / Ba

A → da / BC   B → g / ε   C → h / ε

Step1 :
S → ACB / ACg / AC / cb / cbB / hb / ga / Ba / a / AhB / AB

No left recursion.   A → da / BC / gc / c / Bh / B   B → g / ε   C → h / ε.

Step 2 :

First (S) = { d, g, b, h, a }

First (A) = { d, g }

First (B) = { g, ε }

First (C) = { h, ε }

Step 3:

Follow (S) = { \$ }

Follow(A) = { h, g, ε }

Follow (B) = { a, \$, h, ε }

Follow (C) = { \$, b, h, g, ε }.

1) Parse tree

	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \epsilon$	$T' \to *F'/\epsilon$		$T' \to \epsilon$	
F	$F \to id$			$F \to (E)$		

Stack	input	Action
$E	id + id * id $	
$E'T	id + id * id $	
$E'T'F	id + id * id $	
$E'T'id	id + id * id $	POP
$E'T'	+ id * id $	
$E'	+ id * id $	
$E'T+	+ id * id $	POP
$E'T	id * id $	
$E'T'F	id * id $	
$E'T'id	id * id $	
$E'T'	* id $.	
$E'T'F*	* id $	POP
$E'T'F	id $	
$E'T'id	id $	POP
$E'T'	$	
$E'	$	
$	$	