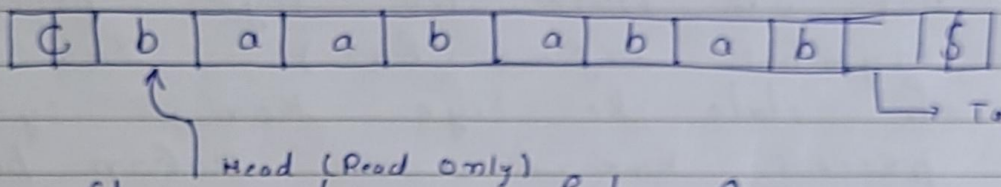
Automata

Ab automata mein machine's ke abstract our problem solving ability sekhenge



Tape — Jo String process hoti hai ussey kehte hai Tape

Head — Vo part Jo read karega our then eh hee direction mein aage Jaula hai → ← (depends on manufacture)

Diagram Samjo

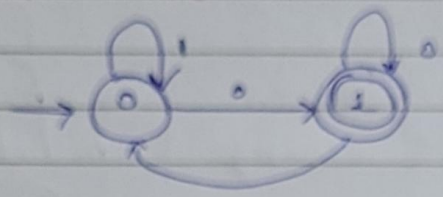
→ (q0) ⟶ Initial State

(qF) ⟶ final State

Baaki Saari State Intermediate hoti hai

(q1)

Ab isme aap state ka relation bhi likhte ho



* har eh state he liye processing (symbol define hai) Yani har eh state ke liye Konsa symbol process karega aur kahan pahuchega. vo define hai

Finite Automata => N : no° of state

Important Terms

Symbol Symbol is the smallest building block, which can

Alphabets Set of Symbol always finite

Σ

CLASSMATE
Date :
Page :

String — String is a finite sequence of symbols from some alphabet.

w

|w| — length of String

Language — Language is a collection of appropriate strings. A language that is formed over Σ can be finite or infinite

Language of Finite Language

→ L_1 = { set of stringth of length 2 }
→ L_1 = { xy, yx, xx, yy }

Grammar — It is a finite set of formal rules for generating symetically correct sentences

Terminal => Component of Sentence's generated using grammar
a , b , c - - - -

Non Terminal => Symbol which take part in composition of sentence but Not part of it
A , B , C - - - -

Grammar can be 4 tuples (V, T, S, P)

V — Set of Non Terminal Symbols.
T — Set of Terminal Symbol
S — It is a special Symbol called Start Symbol
P — Ye hai Production Rules for Terminals & Non Terminals.

$$\alpha \longrightarrow \beta$$

α & β are strings on $V_N \cup \Sigma$
atleast one symbol of α belongs to V_N.

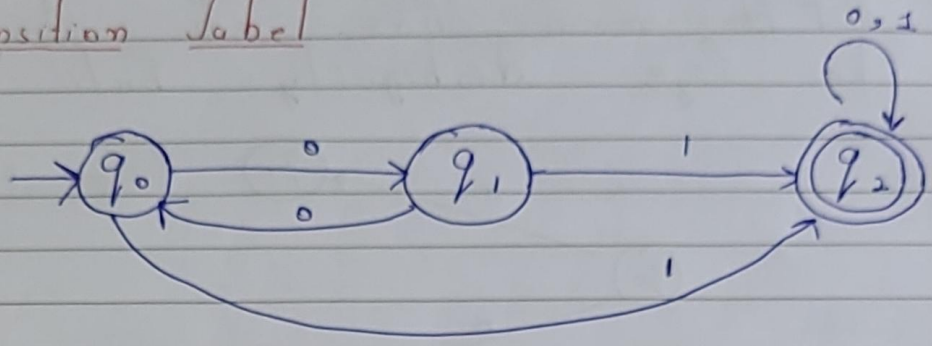
$(\{S, A, B\}, \{a, b\}, S, \{S \to AB, A \to a, B \to b\})$

Finite Automata Five Tuples.

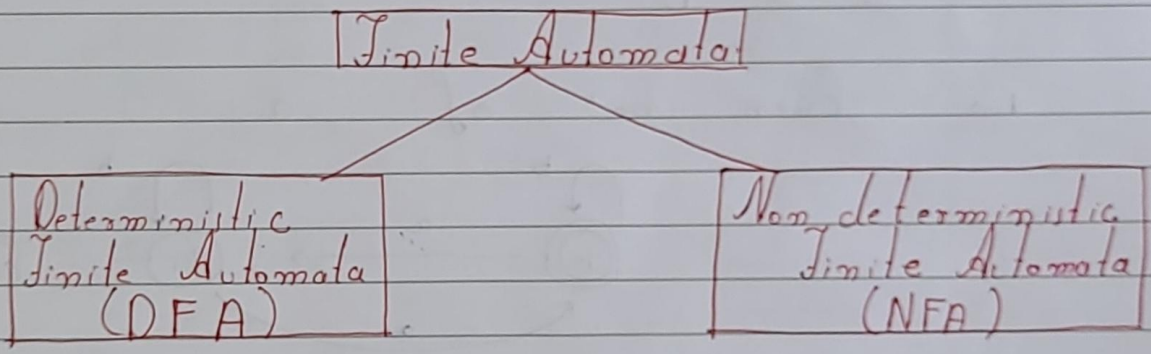
$(Q, \Sigma, q, F, \delta)$

Q Finite set of states
Σ set of Input Symbols
q initial State
F set of final State
δ Transition Function

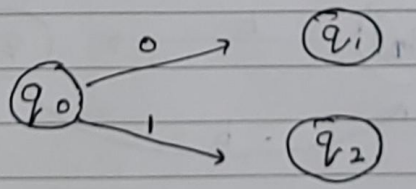
Transition Table



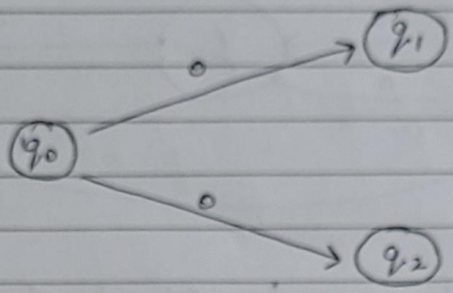
Present State	Next for 0	Next for 1
→ q_0	q_1	q_2
q_1	q_0	q_2
* q_2	q_2	q_2



DFA mein har ek Input Symbol ke liye sirf ek hee path chuna.



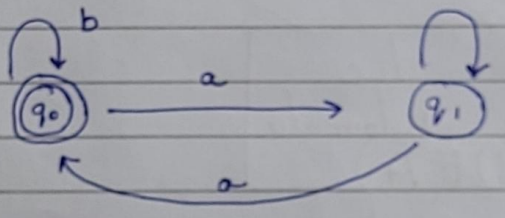
NFA mein har ek Input Symbol ke liye Ambiguity hoti hai



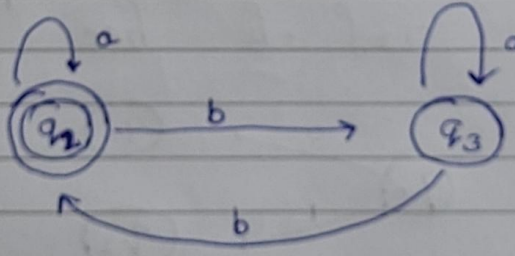
Cross product Seekho

Q Design a DFA which accept even no° of a & b

1→ Even no° of a



Even no° of a.



Even no of b.

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

$\{(q_0,q_2), (q_0,q_3), (q_1,q_2), (q_1,q_3)\}$

$q_0 \xrightarrow{a} [q_1]$
$q_2 \xrightarrow{a} [q_2]$

$q_0 \xrightarrow{b} [q_0]$
$q_2 \xrightarrow{b} [q_3]$

$q_0 \xrightarrow{a} [q_1]$
$q_3 \xrightarrow{a} [q_3]$

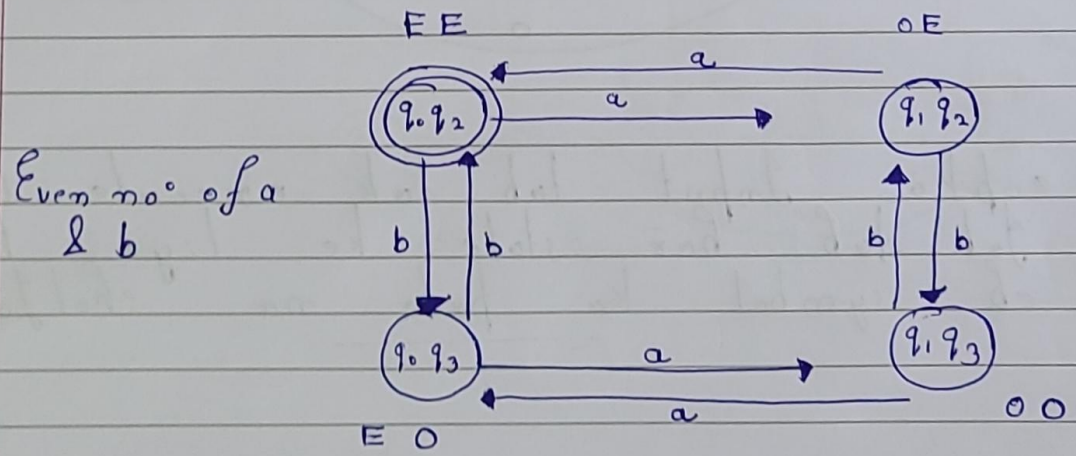
$q_0 \xrightarrow{b} [q_0]$
$q_3 \xrightarrow{b} [q_2]$

$q_1 \xrightarrow{a} [q_0]$
$q_2 \xrightarrow{a} [q_2]$

$q_1 \xrightarrow{b} [q_1]$
$q_2 \xrightarrow{b} [q_3]$

$q_1 \xrightarrow{a} [q_0]$
$q_3 \xrightarrow{a} [q_3]$

$q_1 \xrightarrow{b} [q_1]$
$q_3 \xrightarrow{b} [q_2]$

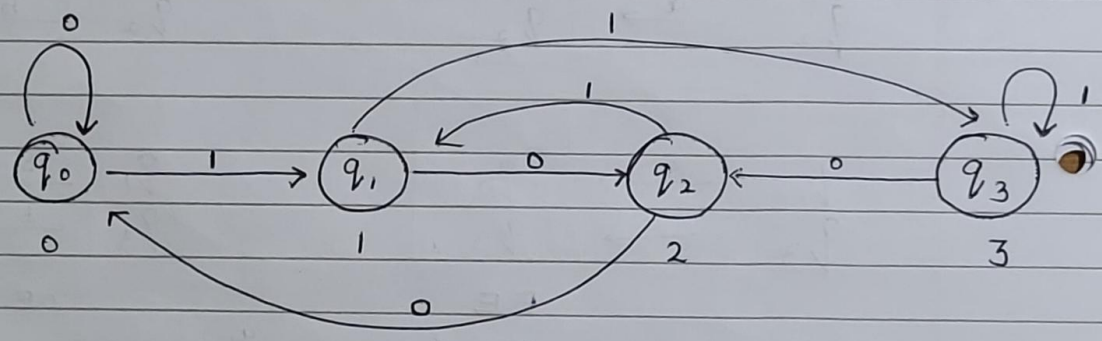


Even no° of a & b

Q Design a DFA such that we check divisibility of a Binary no. with 4

See Procedure

		remainder / 4	
0	0 0 0	0	4 state
1	0 0 1	1	
2	0 1 0	2	
3	0 1 1	3	
4	1 0 0	0	4 state
5	1 0 1	1	
6	1 1 0	2	
7	1 1 1	3	
8	1 0 0 0	0	



aapko Input tab tak execute karne jab tak har state ke liye har ek symbol ka pata na chal jaye.

Ab NFA se hehle DFA KO

3 Representation

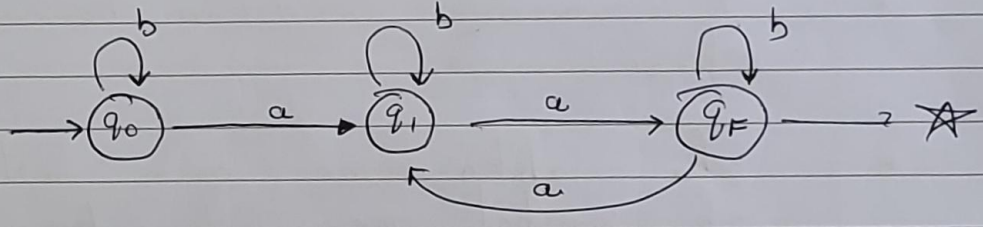
Transition Table

	a	b
State	N.S	N.S
q_0	q_1	q_0
q_1	q_F	q_1
q_F	q_1	q_F

Transition function

$$\delta(q_0, a) \vdash q_1$$
$$\delta(q_0, b) \vdash q_0$$

Diagram

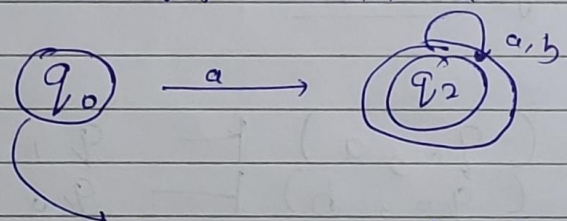


NFA

Jo Complex Soln hota hai uske liye hum pehle NFA design karte hai then usko DFA mein convert karte hain.

Isme less complexity hoti hai sirf min condition ko fulfil karte hai baaki ki Nahi Sochte

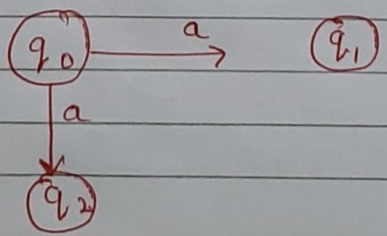
Q String Starts with a



Wo b specify karna.

* Jo minimum condition hai usko solve karo.

* acha abko isme aisa milega ki eh symbol mein 2 ways ho



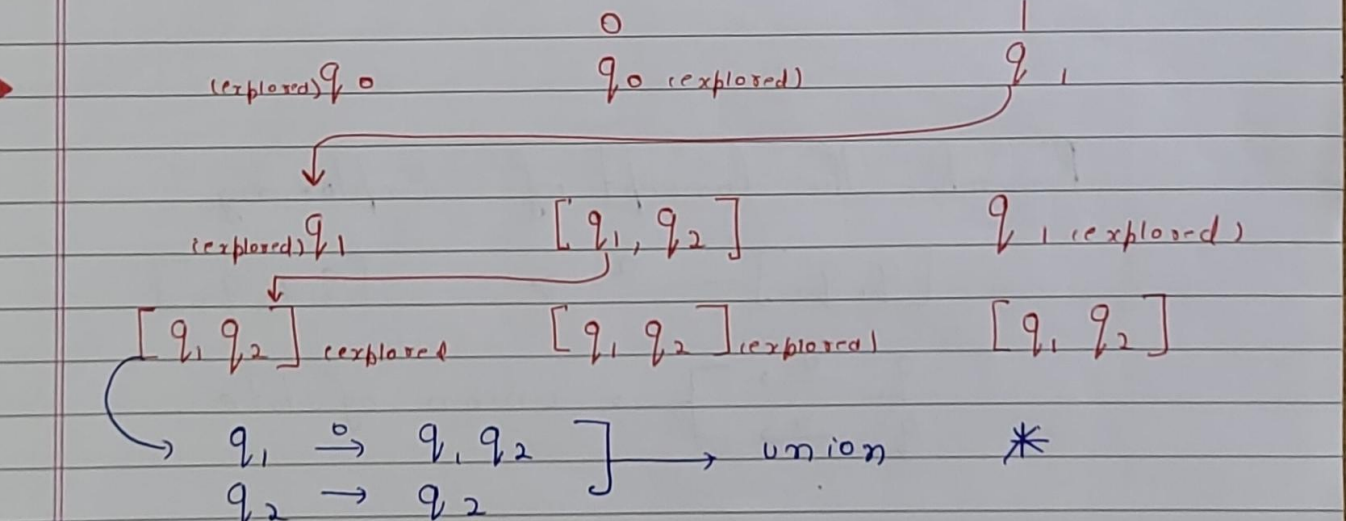
Conversion of NFA to DFA

Hamesha kai bhi soln ho vo hamesha DFA mein hee implement hoga ?

Main agenda Samjo aap transition table ki pehle line likho aur jo states achieve ho rahi hai unko explore karo

Yani unpe har symbol ko execute karo until soare explored ho jaye

	0	1
→ q_0	q_0	q_1
q_1	$\{q_1, q_2\}$	q_1
* q_2	q_2	$\{q_1, q_2\}$

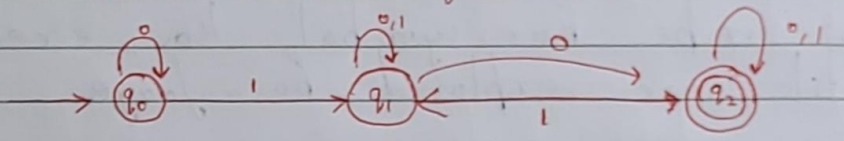


* Ab transition table mein jo final state hai vo jahan be bhi hai vo final

q_0	q_0	q_1
q_1	$[q_1 q_2]$	q_1
* $[q_1 q_2]$	$[q_1 q_2]$	$[q_1 q_2]$

→ Treat it as a single state.

Ques



	0	1
q_0	q_0	q_1
q_1	$\{q_1 q_2\}$	q_1
$\{q_1 q_2\}$	$\{q_1 q_2\}$	$\{q_1 q_2\}$

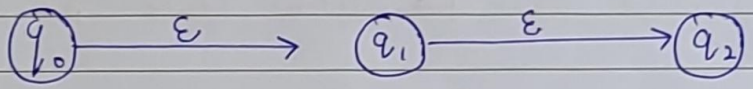
Remember the Concept of Exploring

$[q_1 q_2] \xrightarrow{a} [q_1 q_2 q_3]$

$q_1 \xrightarrow{a}$
$q_2 \xrightarrow{a}$] → union of state

NFA with Epsilon

Ab Epsilon yaani null transition ka matlab bina kuch process kare aap state change karo

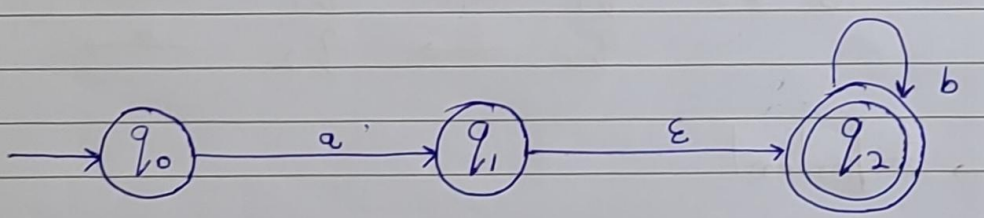


toh hume Null Transition batane hai:

$q_0 \xrightarrow{\varepsilon}$ q_0 hamesha aayega agar movement specify hai toh wo bhi include ho sakta hai

* har ek state ke liye δ' (specify karo) yaani δ' (transition for Epsilon)

Ques



$\delta'(q_0, a) \Rightarrow$ $q_0 \xrightarrow{\varepsilon} \xrightarrow{a} \xrightarrow{\varepsilon}$

↳ it means three transition

$\varepsilon\ a\ \varepsilon = \delta'(q_0, a)$

$\varepsilon\ b\ \varepsilon = \delta'(q_1, b)$

↳ jiski state be aaye vo likho

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

$\delta'(q_0, b) = \phi$ (Agar movement na ho)

$\delta'(q_1, a) = \phi$

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

$\delta'(q_2, d) = \phi$

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

$\delta'(q_1, b) \Rightarrow$

$\varepsilon \Rightarrow q_1 \xrightarrow{\varepsilon} q_1 \xrightarrow{b} \phi \longrightarrow \phi$

$q_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{b} q_2 \xrightarrow{\varepsilon} q_2$

↳ Important

* Then aap usko DFA mein convert mein kar sakte ho.

* ϕ => DFA mein Trap state banado isse.

Minimisation of DFA

DFA ko minimise karna jisey hi vo efficient Bane

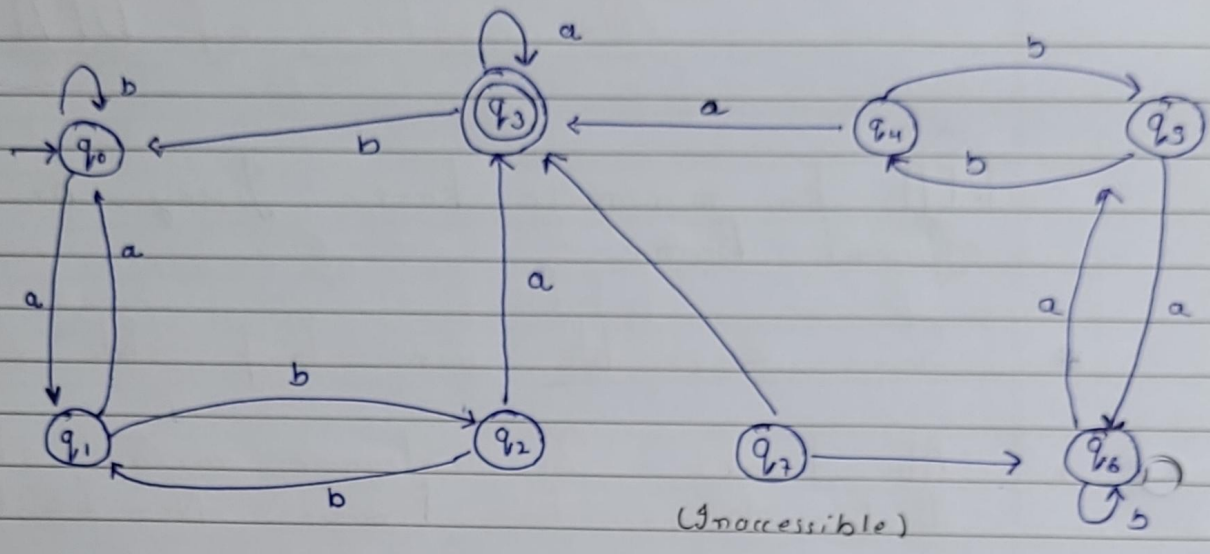
* Saare inaccessible aur trap state remove karlo
* Ab Tumhe 2 categories mein string ko divide karna hai Final Non Final
* Make Transition diagram / table -
* Ab kya karna hai ki aap string ko karo process from K=0 ——— n

Aur kya karo Grp. bana o jinka set same aara hai =

K = 0 (length of String)

* We will stop if set of n-1, n are same.

Ques



Transition Table

	a	b
→ q_0	q_1	q_0
→ q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_3	q_0
q_4	q_3	q_5
q_5	q_6	q_4
q_6	q_5	q_6

$P_i = 0$

F

q_3

N.F

$q_0\ q_1\ q_2\ q_4\ q_5\ q_6$

$P_i = 1$

	a	b
q_0	x	x
q_1	x	x
q_2	✓	x
q_4	✓	x
q_5	x	x
q_6	x	x

$\{q_0\ q_1\ q_5\ q_6\}$ N.F $\{q_2\ q_4\}$ F $\{q_3\}$ F

$P_i = 2$

q_0	x	x	x	x
q_1	x	x	✓	x
q_5	x	x	✓	x
q_6	x	x	x	x

q_2	✓	x	x	x
q_4	✓	x	x	x

$\{q_0\ q_6\}$ N.F $\{q_1\ q_5\}$ F $\{q_2\ q_4\}$ F $\{q_3\}$ F

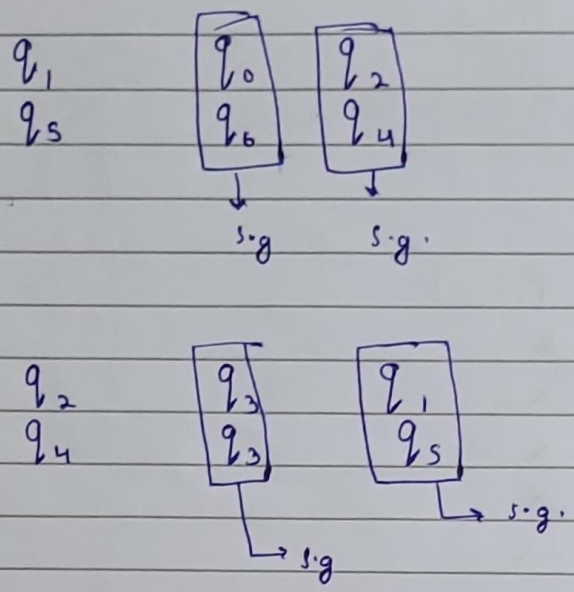
(same pair)

$P_i = 3$

$q_0 \rightarrow q_1\ q_5$

$q_0 \rightarrow q_0\ q_6$

$\{q_0\ q_6\}$ = Same



Regular Expression

Aap jo finite automata banate hai toh uski language aap expression represent kar sakte hai

Symbol => $X^* = \{\varepsilon, X, XX, XXX \cdots\}$
$X^+ = \{X, XX, XXX \cdots\}$

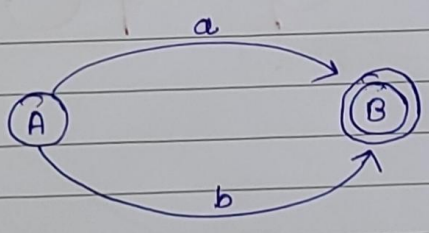
Kleenes Theorem

* RE = a + b ⟶ It will accept a or b

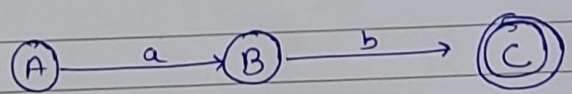
* R.E = (a + b)* ⟶ It will accept any combination of a, b with NULL

* R.E = ab ⟶ It will accept a then b

a + b

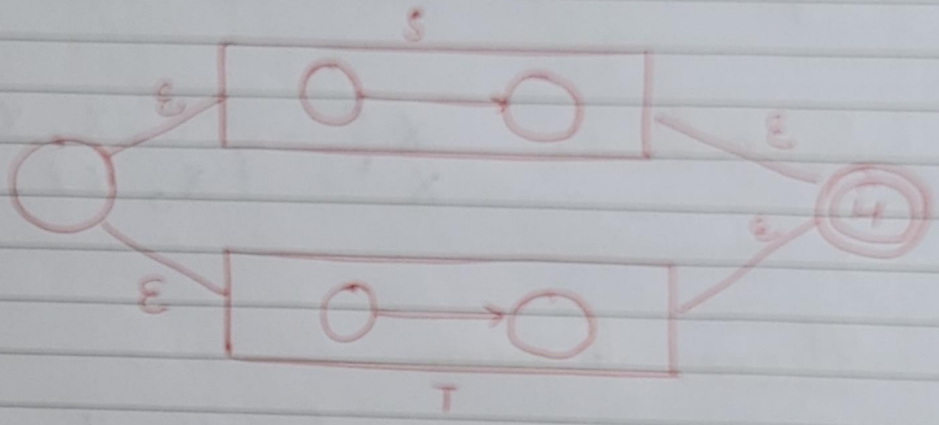


a . b

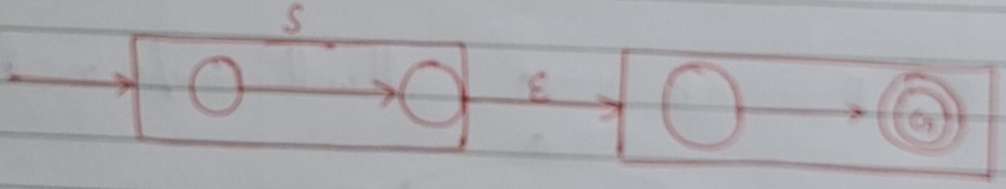


Merge Two Regular Expression S, T they have separate DFA

R = S + T ⟹ connect via null transition



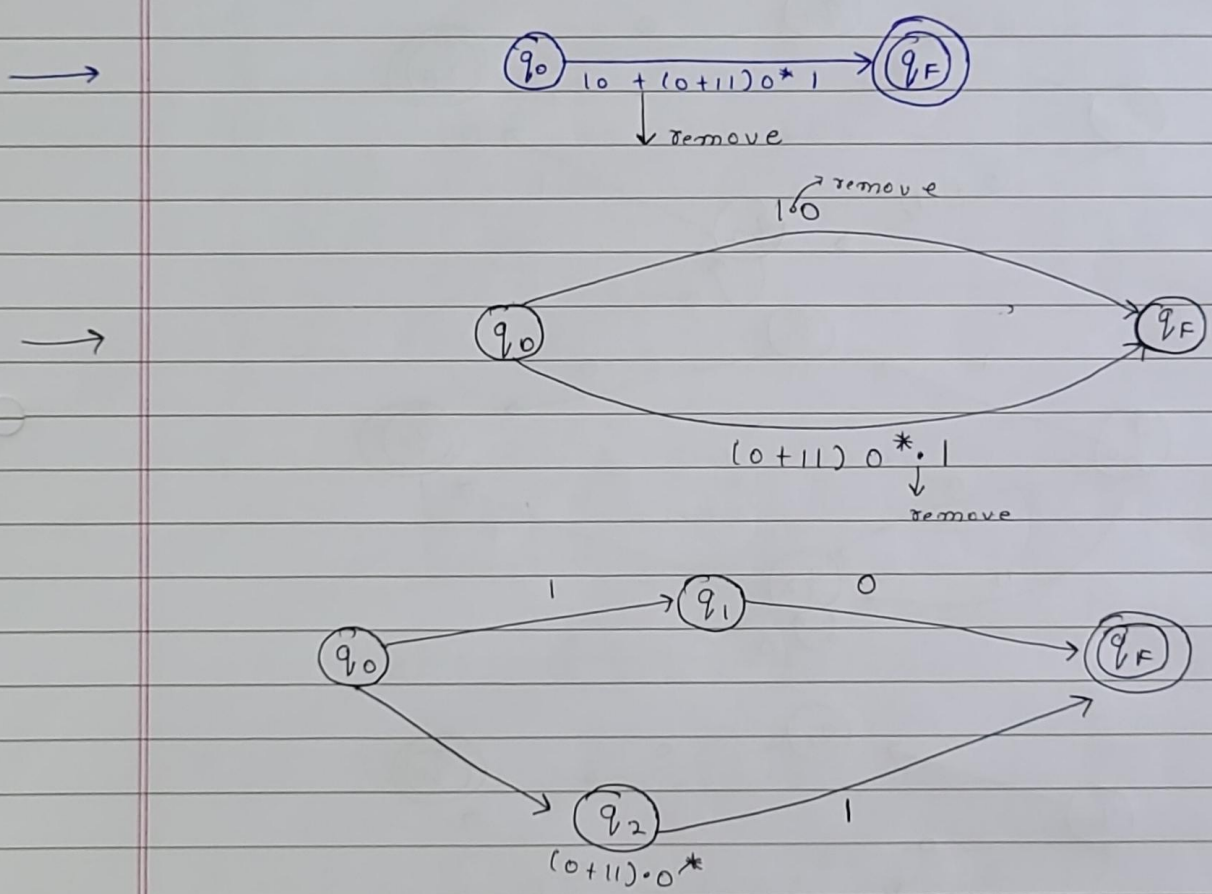
R = S · T

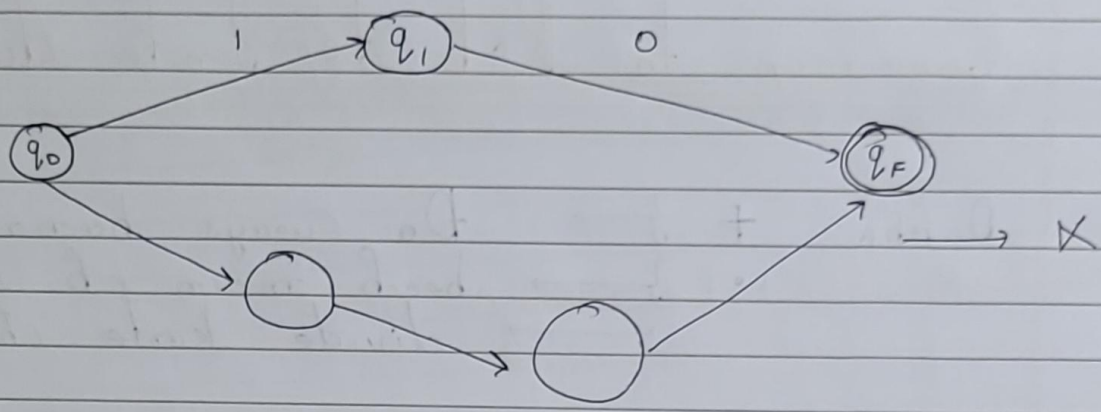


Conversion to R.E to Finite Automata

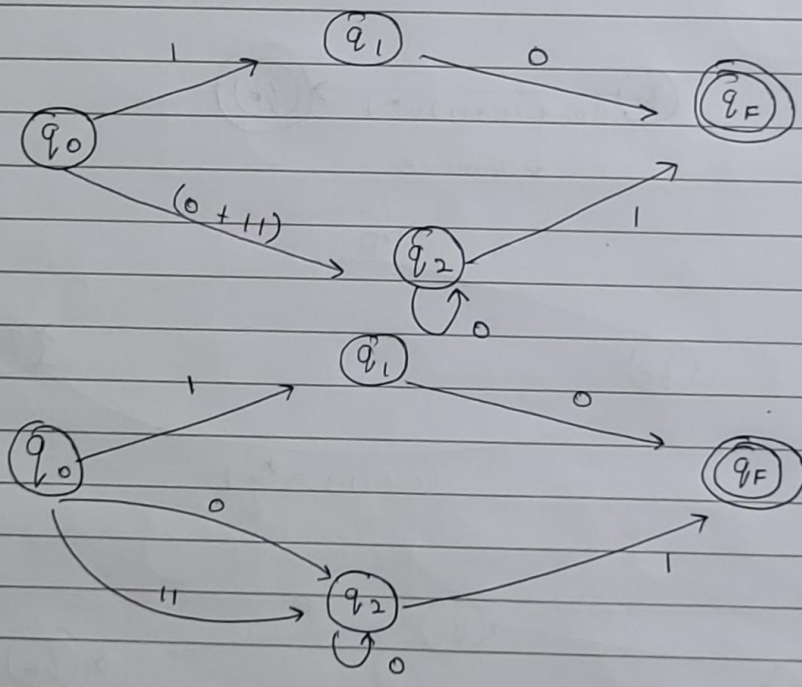
Dekho + ⟶ Do ways banao
. ⟶ beech mein ek state
⟶ divide karte chalo

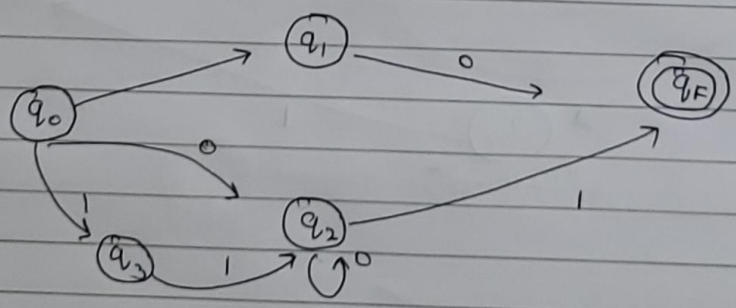
$$|0 + (0+11)0^*|$$





0^* ⟹ same state be self loop





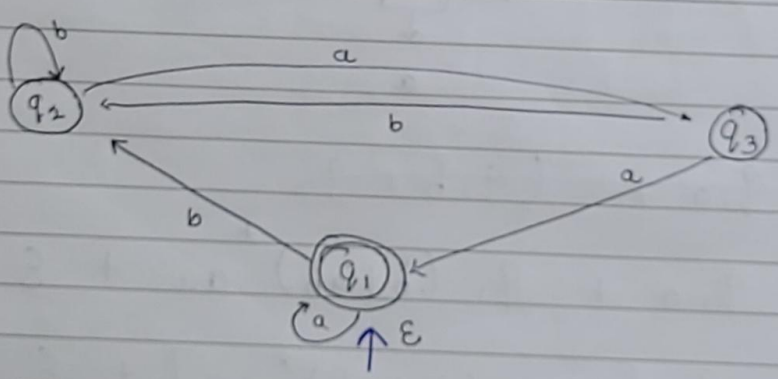
Finite Automata to Regular Expression

$R = \emptyset + RP$
$R = \emptyset P^*$
} → Important

- Har ek state ke liye arrow kahan se aur kya aa raha hai vo likho

Important → Agar Initial final hai toh ε include karo

→ pehle State then Symbol
$q_1 a$



We will solve for final state

$q_2 a a$

F.S

$q_1 = q_1 a + q_3 a + \varepsilon$ (self loop) (Initial + Final)

$q_2 = q_2 b + q_1 b + q_3 b$

$q_3 = q_2 a$

$q_2 = q_2 b + q_1 b + q_2 a b$

or

$q_2 = q_1 b + q_2 (b + ab)$

$(R) \quad (Q) + R \quad (P)$

$q_2 = q_1 b (b + ab)^*$

$q_1 = q_1 a + q_2 a \cdot a + \varepsilon$

a

$= q_1 a + (q_1 b + q_2$

$q_1 = q_1 a + q_1 (b + ab)^* aa + \varepsilon$

$q_1 = q_1 [a + (b + ab)^* aa] + \varepsilon$

$q_1 = [a + (b + ab)^* aa]^* \cdot \varepsilon$

Pumping Lemma for Regular Language

The Language accepted by finite Automata is called regular Language

Ek string jo valid ho vo chuno

then chuno something (jo aapne chuna hai) divide karo xyz mein such that

$i >= 0$, $xy^i z$

$|y| > 0$
$|xy| <= n$ } ⟶ Condition

ab Y ki power increase karo then jo resulted string aaye agar vo Language satisfy nahi karte toh vo Language Regular nahi hai

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

$n = 2$

$\underbrace{a}_{x}\underbrace{a}_{y}\underbrace{b\,b}_{z}$ $\quad |y| > 0$, $|xy| \leq 2$.

$a\,\underbrace{a}_{y'}\,bb \;\; = \;\; \underbrace{a\,a}\,bb$ → ✗ in Language

$y^1 \Rightarrow y^2 \Rightarrow y^3 \Rightarrow y^4 \Rightarrow y^5$ ——

→ pumping length.

Ques $L = \{a^n b^m,\ n \geq 1,\ m > 2\}$ $\quad n = 2,\ m = 3$

$\underbrace{a}_{x}\underbrace{a}_{y}\underbrace{b\,b\,b}_{z}$

Min value = $n = 1$
$m = 3$
$= 4$

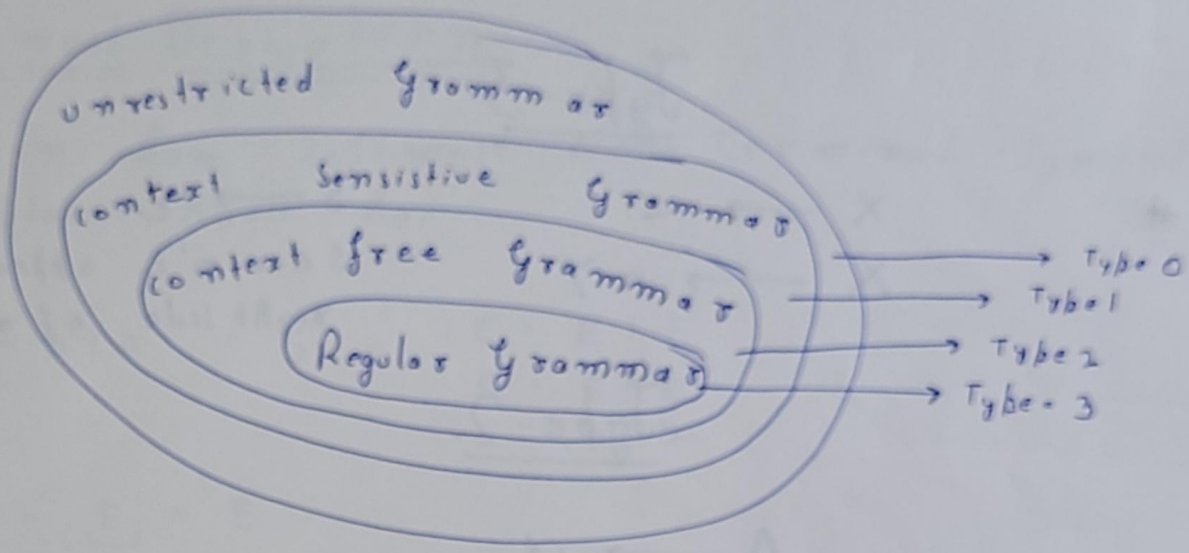
or $= \underbrace{a}_{x}\underbrace{a}_{y}\underbrace{b\,b\,b}_{z}$

$\Rightarrow$ y ki power increase karo
$\Rightarrow$ Saari strings accept hongi

→ Regular Language $\Rightarrow$ ✓.

Grammar

Grammar kya hai set of rules (production) that generates strings ko ek pattern. Yani ye vo strings generate karta hai jo vo Language accept kare



⊙ Grammar G can be formally written as a 4 tuple (V, T, S, P) where

V ⟶ Set of Non Terminal Symbols (A, B)

T ⟶ Set of Terminal Symbols (a, b)

S ⟶ Start Symbol

P ⟶ Production Rules $\alpha \longrightarrow \beta$

Type 0 => Recognized by Turing Machine

Type 1 => Context Sensistve by Liner Bound Automata

Type 2 => Context free Grammar By push down

Type 3 => Finite Automata. (Reg Grammar)

Type-3

$X \longrightarrow a$

$X \longrightarrow aY$

$S \longrightarrow \varepsilon$

S doesn't appear on right side of any rule

Type-2

$A \longrightarrow \alpha$

$A \in$ Non Terminal

$\alpha \in$ Combination of Terminal & Non Terminal

Type 1

$\alpha A \beta \longrightarrow \alpha \gamma \beta$

$A \in N$

$\alpha, \beta, \gamma \in (T \cup N)^*$

↳ can be Empty

$S \longrightarrow \varepsilon$

If S is not allowed in right

Type - 0

$\alpha \longrightarrow \beta$

α = string of Terminal & Non Terminal
=. atleast non terminal

Leftmost Derivation

- mein hum leftmost Non Terminal Symbol replace Karte hai

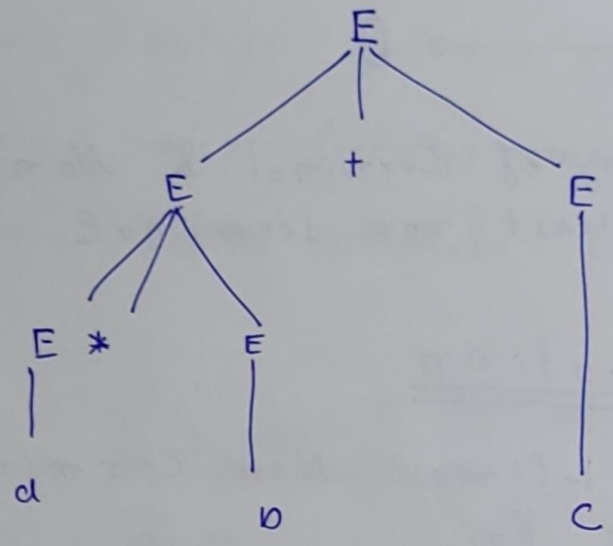
a - b + a ;

E = E + E
E = E - E
E = a | b

E - E + E
↓
Replace

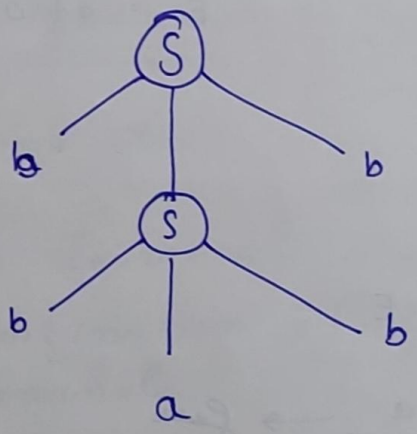
E - E + E
↓
a - b + a → Ans

We Can also represent using Parsed Tree



$S \longrightarrow bSb \mid a \mid b$

Input string bbabb



Ambiguity In Grammar

Agar aap Left most derivation ke through bhi 2 raaste hai uss particular string tak pohoch sakte hai.

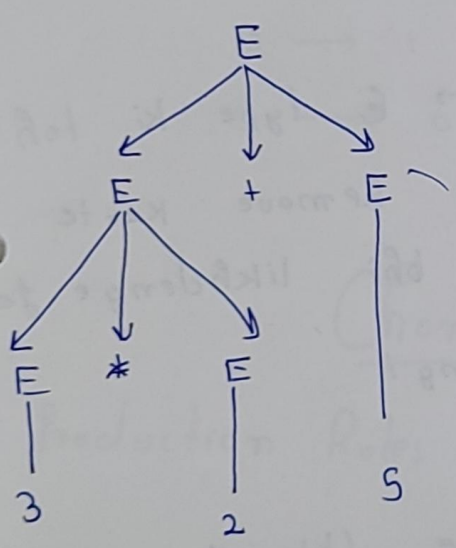
$E \longrightarrow I$

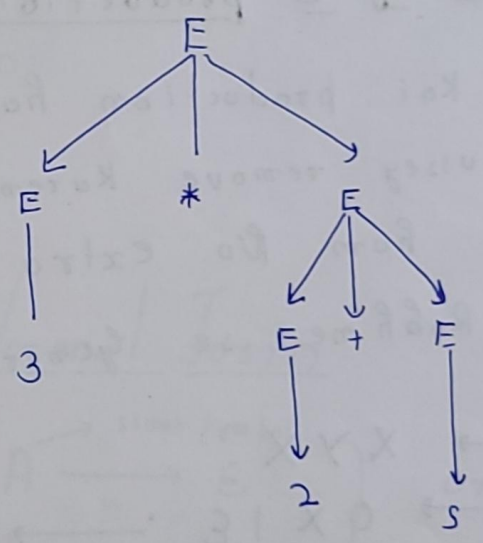
$E \longrightarrow E+E \mid E*E$

$E \longrightarrow (E)$

$I \longrightarrow \varepsilon \mid I \mid 0 \mid 1 \mid 2 \mid 3 \mid \cdots \mid I$

3 * 2 + 5





Grammar Optimize

Useless Symbol

Ek Symbol useless tab hota hai jab vo kisi bhi Production ke right side mein na aaye

Ex

$S \longrightarrow AB$

$A \longrightarrow aA \mid \varepsilon$

$B \longrightarrow bB \mid \varepsilon$

$C \longrightarrow C$ ⟶ useless kyonki C baaki kisi bhi func ke right side mein nahi aata.

Elimination of ε production

Agar koi production hai $X \longrightarrow \varepsilon$ type ki toh hum usse remove karenge but Remove karte samay hum Wo extra values bhi likhdenge jo NULL Rakhne se generate hongi

$S \longrightarrow XYX$

$X \longrightarrow 0X \mid \varepsilon$ ⟶ remove (place X Null)

$Y \longrightarrow 1Y \mid \varepsilon$ ⟶ (remove) (place Y Null)

$S \longrightarrow XYX \mid YX \mid XX \mid X \mid Y$

$X \longrightarrow 0X \mid 0$

$Y \longrightarrow 1Y \mid 1$

Unit production

$X \longrightarrow Y$

(X: Non Terminal)

$Y \longrightarrow$ Symbols.

directly place them

$S \longrightarrow 0A \mid 1B \mid C$

$A \longrightarrow 0S \mid 00$

$B \longrightarrow 1 \mid A$

$C \longrightarrow 01$

$S \longrightarrow C = C \longrightarrow 01$

$B \longrightarrow A - A \rightarrow 0S \mid 00$

$S \longrightarrow 0A \mid 1B \mid 01$

$A \longrightarrow 0S \mid 00$

$B \longrightarrow 1 \mid 0S \mid 00$

$C \longrightarrow 01$

Chomsky Normal Form

Production Rules =>

start symbol

$A \longrightarrow \varepsilon$

$S \longrightarrow AB$

$S \longrightarrow a$

} only these are allowed

Q.

$S \longrightarrow a \mid aA \mid B$

$A \longrightarrow aBB \mid \varepsilon$

$B \longrightarrow Aa \mid b$

$S \longrightarrow aA \Rightarrow S \longrightarrow XA$, $X \longrightarrow a$

$S \longrightarrow B \Rightarrow$ place B value $\Rightarrow S \longrightarrow Aa$, $S \longrightarrow b$

$S \longrightarrow a \mid XA \mid Aa \mid b \Rightarrow S \longrightarrow a \mid XA \mid AX \mid b$

$X \longrightarrow a$ $\quad X \longrightarrow a$

$B \longrightarrow Aa \mid b \quad B \longrightarrow AX \mid b$

$A \longrightarrow aBB \mid \varepsilon \Rightarrow A \longrightarrow \underbrace{XB}_{club} B \mid \varepsilon$

$A \longrightarrow XBB \Rightarrow A \longrightarrow RB$

$R \longrightarrow XB$

$S \longrightarrow a \mid XA \mid AX \mid b$

$X \longrightarrow a$

$B \longrightarrow AX \mid b \mid a$

$A \longrightarrow RB \mid \varepsilon \longrightarrow$ (Remove Null)

$R \longrightarrow XB$

Remove Null

Remove Unit

Remove useless

$\longrightarrow$ ☆☆

$Y \longrightarrow 1Y \mid 1$

Greibach Normal Form

$S \longrightarrow \varepsilon$

$A \longrightarrow a$

$S \longrightarrow a\ A\ S\ B$ ----

First convert the grammar into C.N.F
Left Recursion => Eliminate
Convert to GNF

Left Recursion => Grammar mein agar aisi production hai jaise

$X \longrightarrow X\,a$ (string of Terminal & Non Terminal)

$A \longrightarrow A\alpha \mid \beta$ =>

$A \longrightarrow \beta A'$

$A' \longrightarrow \alpha A' \mid \varepsilon$ → remove

$A \longrightarrow \beta A' \mid \beta$

$A' \longrightarrow \alpha A' \mid \alpha$

$S \longrightarrow X\ B \mid A\ A$

$A \longrightarrow a \mid S\ A$

$B \longrightarrow b$

$X \longrightarrow a$

Dekho Agar $A \longrightarrow SA \longrightarrow$ gnf mein aagaya to automatic AA bhi gnf mein aajayega.

$A \longrightarrow S\ A \quad => \quad A \longrightarrow a \mid X\ B\ A \mid A\ A\ A$

$A \longrightarrow a \mid a\ B\ A \mid A\ A\ A$

$S \longrightarrow a\ B \mid A\ A$

$A \longrightarrow a \mid a\ B\ A \mid A\ A\ A$

$B \longrightarrow b$

$X \longrightarrow a$

$A \longrightarrow A\ A\ A \rightarrow$ remove

$A \longrightarrow \underbrace{A\ A\ A}_{\alpha} \mid a \mid \underbrace{a\ B\ A}_{B_1\ B_2}$

$A \longrightarrow a\ C \mid a \mid a\ B\ A\ C \mid a\ B\ A$

$C \longrightarrow A\ A\ C \mid A\ A$

$Y \longrightarrow 1\ Y \mid 1$

S → aB | AA
A → aC | aBAC | a | aBA
C → AAC | AA
B → b
X → a

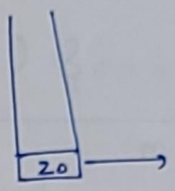
Starting jiska bhi A hai vahan
pe replacement kardo.

S → aB | aCA | aBACA | aA | aBAA
A → aC | aBAC | a | aBA
C → aCAC | aBACAC | aAC | aBAAC
C → aCA | aBACA | aA | aBAA
B → b
X → a

→ ✱.

Push Down Automata

Ab isme hum ek stack add kardete hai jissey hamara kaam aasan ho jaaye



Initial element Agar Ye hai yaani stack empty hai (Tab be)

PDA => collection of 7 components

- Q = finite set of states
- Σ = Input set
- Γ = Stack symbol which can be pushed
- q_0 = Initial state
- Z = Start symbol which is in Γ.
- F = final state set
- δ = Mapping function current to Next State

Language = $a^n b^{2n}$ => 11

$\delta(q_0, a, Z) = (q_0, aaZ)$

mila

state — top element of stack

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

Now when we read b we will change the state from q_0 to q_1

$\delta(q_0, b, a) = (q_1, \varepsilon)$

$\delta(q_1, b, a) = (q_1, \varepsilon)$

Final State => $\delta(q_1, \varepsilon, z) = (q_f, \varepsilon)$

Ques => $0^m 1^m 0^n \mid m, n >= 1$ => ??

↳ Ise hame koi lena dena nahi hai 1 ki agal bagal 0 ki Quantity same honi chahiye.

$(q_0, 0, Z_0) = (q_0, 0Z_0)$

$(q_0, 0, 0) = (q_0, 00_0)$

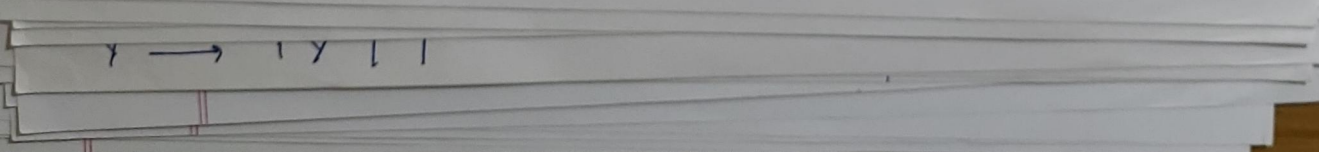
$(q_0, 1, 0) = (q_1, 0)$

$(q_1, 1, 0) = (q_1, 0)$

$(q_1, 0, 0) = (q_2, \varepsilon)$

$(q_2, 0, 0) = (q_2, \varepsilon)$

$(q_2, \varepsilon, Z_0) = q_f$



LL(1) Parsing

Left factoring

$S \longrightarrow a\alpha$, $S \longrightarrow a\beta$

we have to generate a, Ambiguity.

$S \longrightarrow a\,S'$

$S' \longrightarrow \alpha \mid \beta$

Construction of Parsing Table

X = terminal first = {X}

$X \longrightarrow \varepsilon$ => Add Null to first {X}.

X = Non terminal => $X \longrightarrow a\alpha$ → Add

$X \longrightarrow Y_1 Y_2 Y_3$

→ First of Y_1 Add to first of X

Now who...

Grammar mein No Left Recursion

No Left Factoring

$E \longrightarrow TE'$

$E' \longrightarrow +TE' \mid \varepsilon$

$T \longrightarrow FT'$

$T' \longrightarrow *FT' \mid \varepsilon$

$F \longrightarrow (E) \mid id$

First => E - (, id

E' - + ε - blank Null

T - (, id

T' - * ε

F - (, id

Follow => E - $,)

=> E' - $,)

=> T - +, $,)

=> T' - +, $,)

=> ~~F - +, $,)~~

=> F - *, +, $,)

Follow => Non Terminal => Production ki right side mein kahan aaya

Start Symbol - $

E ⟶ iske baad agar terminal hai toh ok

⟶ ε => Follow of left Non Terminal

⟶ Non Terminal (Everything of first of Non Terminal ! Accept ε)

Agar kisi Non Terminal ke first mein Null hai toh ussey place karo. then see what will come.

Tricky

Parsing Table

First ()

E => (, id

E' => +, ε

T => (, id

T' => * ε

F => (id

Follow ()

E => $,)

E' => $,)

T => +, $,)

T' => +, $,)

F => +, *, $,)

Null

Non Terminal	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Kaise

First mein agar Null Nahi hai toh jo terminal symbol vahan pe aap production jo likhi hai ussey likhdo

If Null hai toh uske follow mein Null Transition daal do

hum isme ye dekhte hai ki agar aap kisi Non Terminal pe hai aur aap ko kisi respective terminal pe jaana hai toh which production to use lets see it in action. how Parsing Table works.

Stack	Input	production
$ E	id + id * id $	
	↑	

→ TE' ⟶ reverse karke push kardo

$ E' T	id + id * id $	E ⟶ TE'
$ E' T' F	id + id * id $	T ⟶ FT'
$ E' T' id	id + id * id $	F ⟶ id
$ E' T'	+ id * id $	T' ⟶ ε
$ E'	+ id * id $	E' ⟶ +TE'
$ E' T +	+ id * id $	

⋮

Soon.

$	$	E' ⟶ ε

→ valid.

SLR Parser

Bottom up

Step => Augmented Grammar

Normal => $S \rightarrow AA$, $A \rightarrow aA \mid b$ | Augmented $S' \rightarrow S$ → Added

Canonical Collection of LR (0)

$S' \rightarrow \cdot S$

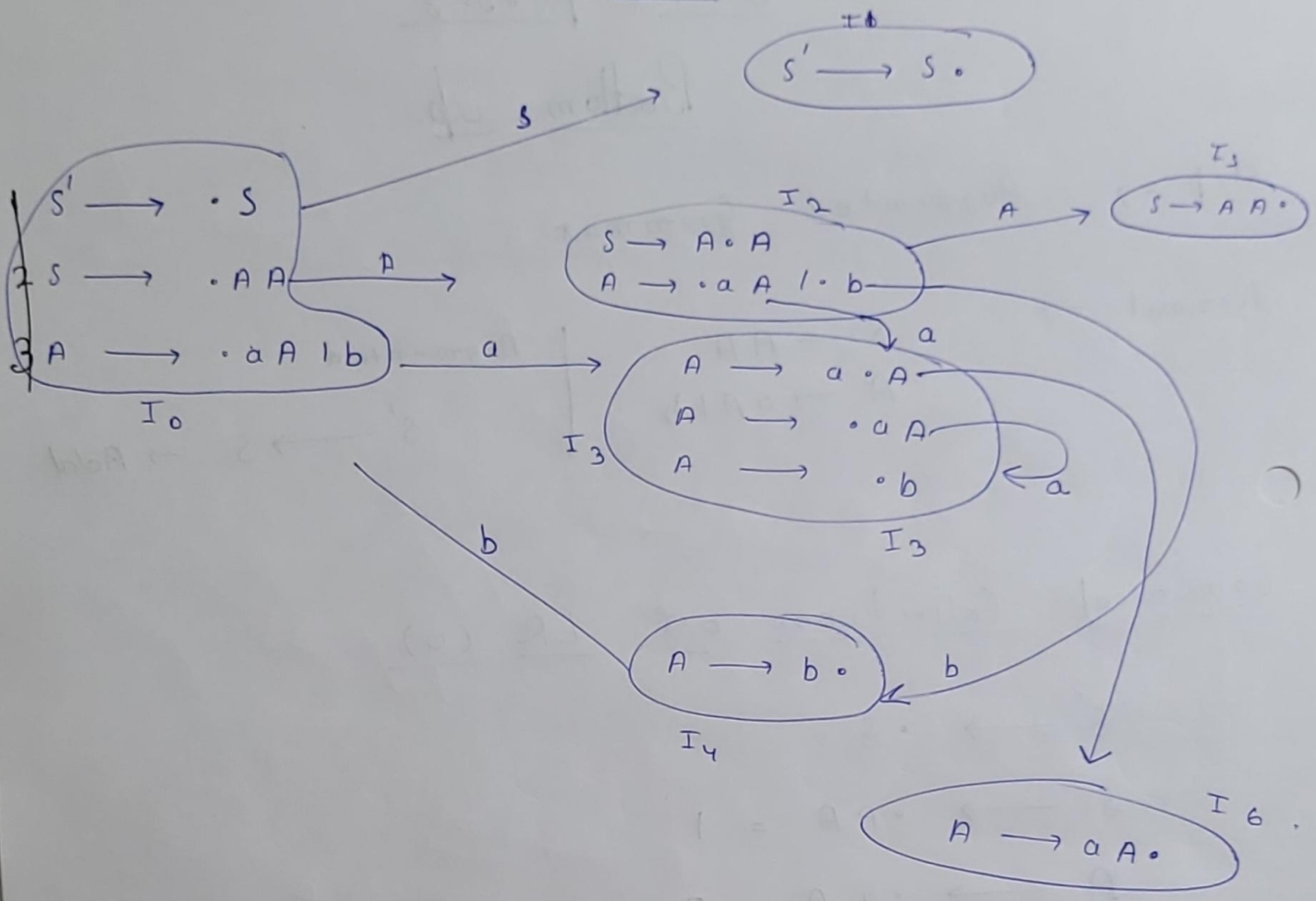
$S \rightarrow \cdot AA$ = 1

$A \rightarrow \cdot aA$ = 2

$A \rightarrow \cdot b$ = 3

- ke baad dekhne kai aage
- • jis bhi Non terminal ke aage kai uska canonical cover of uske Yaani saari production of uski • S

Start



Parsing table mein jo form aa rahe ho through processing I0 pe S ⟶ I1

Aage vo action mein hai write S ⟶ shift

⟶ Parsing table

Explain of Closure

- S ⟶ S ka closure yaani all production like $S \longrightarrow \alpha, \beta$ (↳ whatever)

- ko right karo

$S' \longrightarrow S\cdot \longrightarrow$ Nothing.

$S \longrightarrow A\cdot A$ ⟶ closure of A include

Agar koi same aaj'aye

$I_3 \Rightarrow A \longrightarrow \cdot aA$

$A \longrightarrow a\cdot A$ ⟶ already yahi hai I_3 toh self loop.

- ⟶ right mein jaayegi sabke last mein shift karte jaao.

Parsing Table

State	Action			Goto	
	a	b	$	A	S
I_0	S_3			2	1
I_1			accept		
I_2	S_3	S_4		5	
I_3	S_3	S_4		6	
I_4	γ_3	γ_3	γ_3		
I_5	γ_1	γ_1	γ_1		
I_6	γ_2	γ_2	γ_2		

Agar Augmented production hai to R accepted under $

Agar koi final item hai jo Augmented nahi hai then we use reduce

I_4 => $A \longrightarrow b.$ => reduce production in Grammar γ_3

Working of LR

Initial State Stack = | S | 0 | a | 3 | a | 3 | b | 4 | A | 6 | A | b |
| S | 0 | A | 2 | b | 4 | A | 3 | S | 1 |

Input String => a a b b

[a] => $I_0 + a$ => S 3 => push a & 3 (MOVE)

[a] => I_3 a => S 3 => push a & 3 (MOVE)

[b] => I_3 b => S_4 => push b & 4

[b] => I_4 b => γ_3 => (NOT MOVE) read [b]

A ⟶ [.b]

length - 1 × 2 => Rotodo Add A

Now 3 A -> I_3 A = 6 ⟶ push

I_6 b => γ_2 =>

A ⟶ [.a A]

2 × 2 = 4 => Rotodo add A

I_3 A ⟶ ~~S 3~~ => ~~b & 3~~

6 ⟶ bread => 4 remove

I_0 A ⟶ 2 ⟶ push => I_2 b => push b & 4

I_4 $ ⟶ γ_3 continue

I_1 $ → accept

pointing $

SLR

Jo humari production hoti hai jisme . end mein hota ha $A \longrightarrow aA.$

Usme jo remove hai vo sirf follow of A mein likhna

Item Sets banate samay har ek set se ek Hee arrow nikalte hai

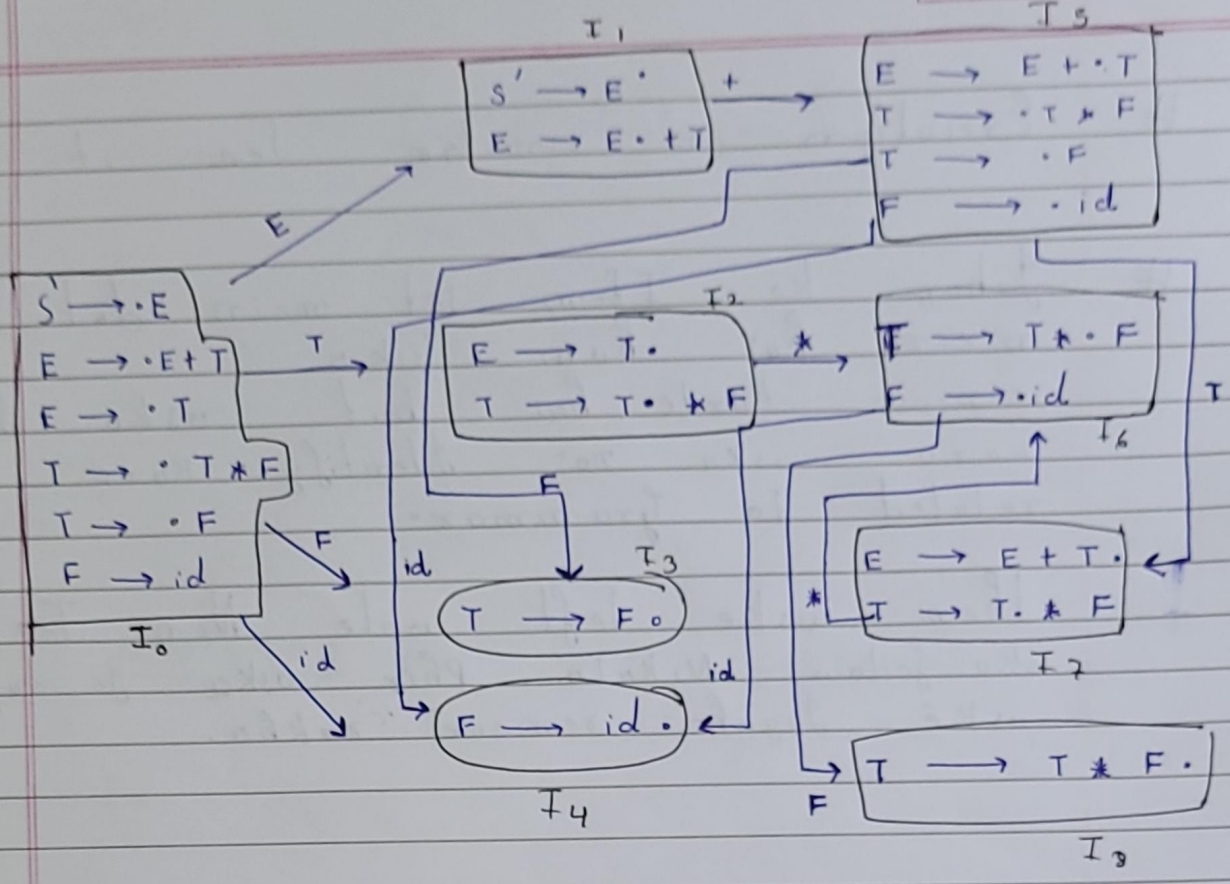
Yaani

$$\left.\begin{array}{l} S' \longrightarrow \cdot E \\ E \longrightarrow \cdot E+T \end{array}\right\} \xrightarrow{E} \begin{array}{l} S' \longrightarrow E \cdot \\ E \longrightarrow E \cdot + T \end{array}$$

Ab dekho Hum . ke baad waale Symbols ka Nikalenge closore Isme . ke baad koi bhi Non Terminal Nahi hai.

Toh Yahi hum stop kardenge.

Har ek . ke baad Waale Non Terminal Ko Canonical cover hoga



State	Action				GoTo		
	id	+	*	$	E	T	F
I_0	S4				1	2	3
I_1		S5		Accept			
I_2							
I_3		R2	S6	R2			
I_4		R5	R5	R5			
I_5	S4					7	3
I_6	S4						8
I_7		R1	S6	R1			
I_8		R3	R3	R3			

Execution is similar See it

Jahan ke Item set mein dot End mein hai hum usko remove karte hai but usko Identify karo usko no. Identify karo related to Grammar.

Then uske left wale Non-Terminal ko follow Nikalo phir usko jo aaye uske jagah remove rakho.

CLR (Parsing)

Ab isme hum kya karenge ki har ek item set ke saath jo function hai usme hoga ek Lookahead Symbol or we can say look ahead.

Compiler Design

Lexical Analyzer

Syntax "

Syntax - - - -

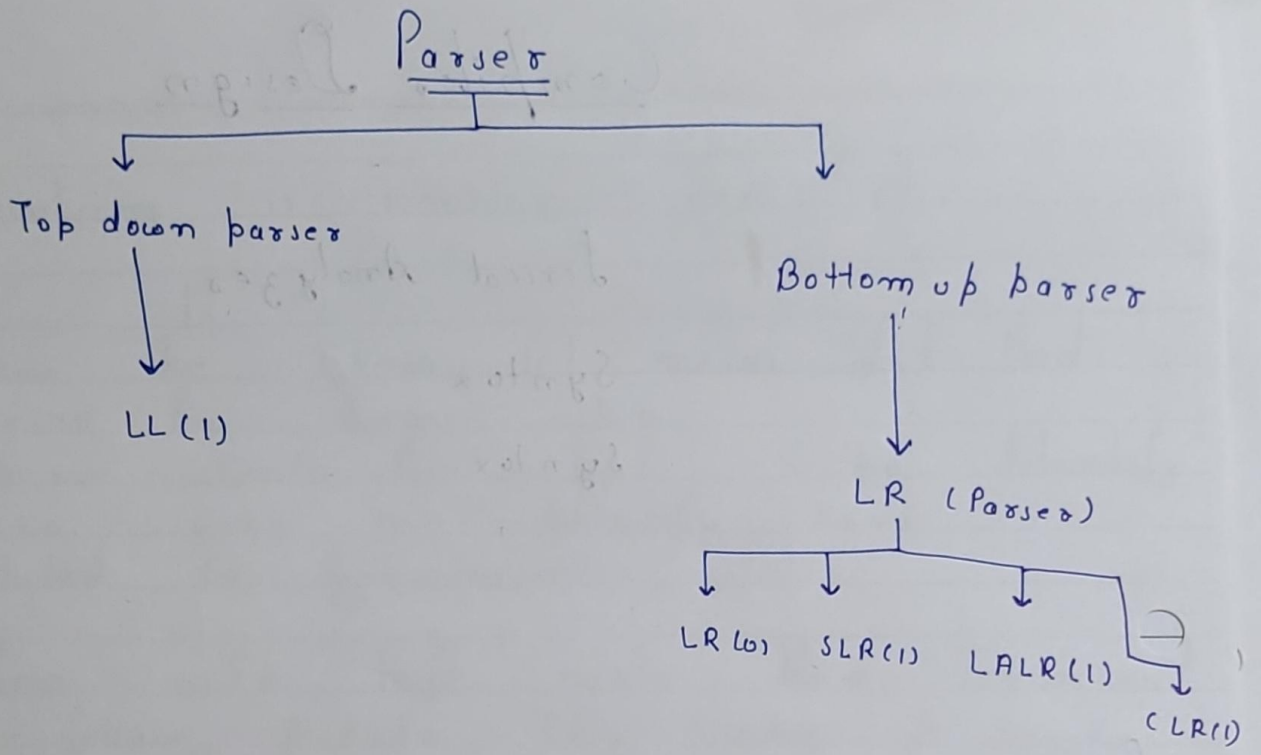
:

:

Lexical Analyzer => Sahi keywoards ke according token Generate karte hai.

Syntax

Ab Jaise Syntax kaa matlab hai ki order of tokens correct hai ya nahi. Uske liye hota hai. Parser



LL(1) => L => Scanning from Left to Right Manner

L => Left most derivation tree

1 => Look Ahead means How many symbols are You going to see when You want to make a decision

Lexical Analyzer

Lexical Analyzer ka wahi kaam hai tokens mein divide karna

Jo aisa karta hai we can say it is as

Lexer

Tokens => Identifiers, key words, operators, seperators, constants
({},)

Count no of Tokens

→

```
int main ( ) {
    /* ____________ */ ⟶ Neglect
    int a = 20 , b = 30 ;
    if (a ≤ b)
        return (b);
    else
        return (a);
}
```

```
printf ("i = %d , &i = %x", i, &i);
```

* " ____ " => 1 (Inside double Quotes only 1 token

#

Augmented production will contain $

If hame likhna hai for a production toh uske baki wali production mein hum dekhte like

$$A \longrightarrow \alpha \cdot B\beta$$
$$B \longrightarrow \cdot\gamma \longrightarrow \underline{\text{Symbol}}$$

B ke baad β hai toh uska first include hoga Agar vo Null hai toh y Copy , otherwise likho First of β.

Jahan vo symbol hai vahi likho

LALR

Jisme Item ho ki uska productions saari same hai but symbols agar different hai toh hum usko saath mein kar dete hai

I67 ;

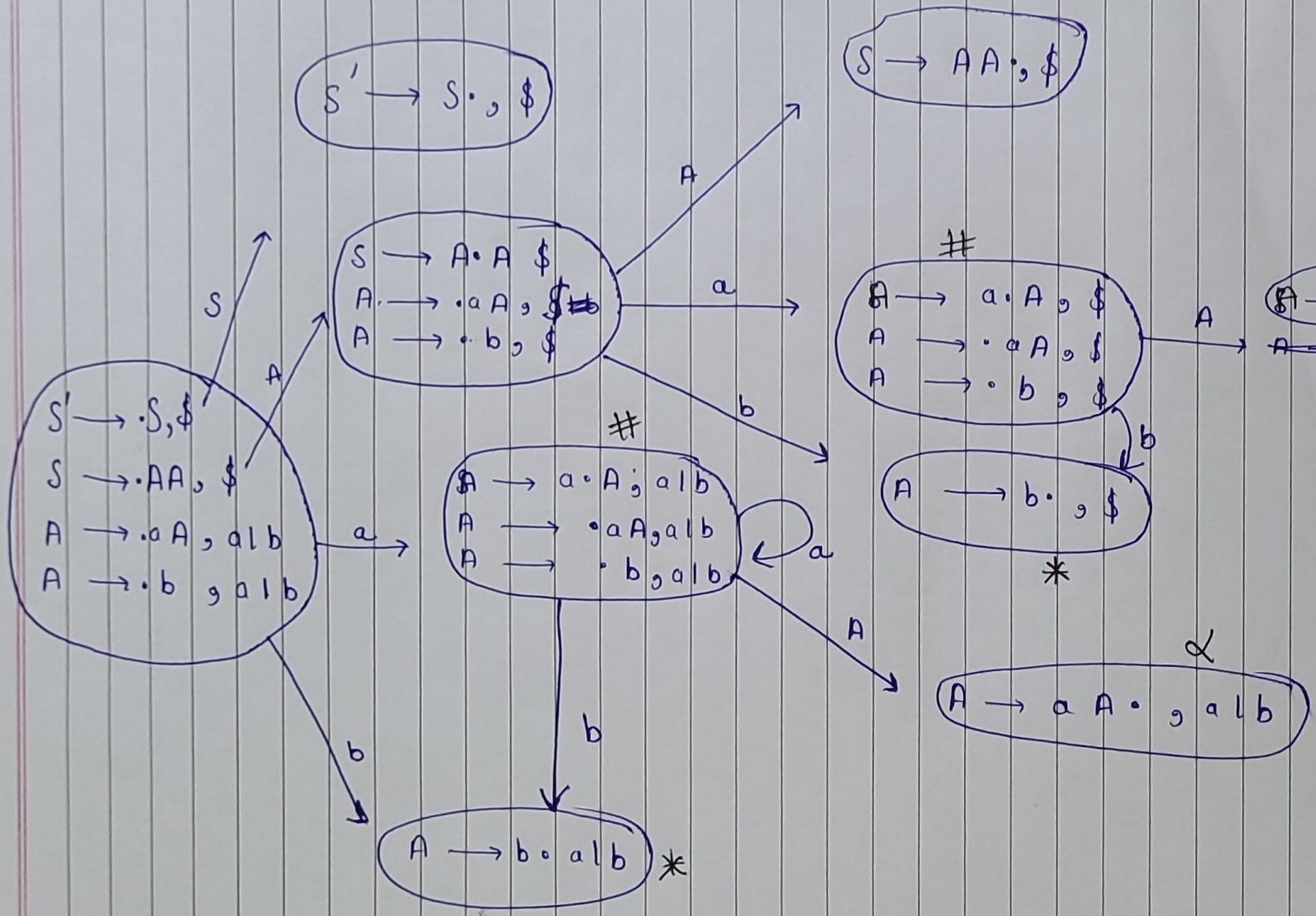
CLASSMATE
Date :
Page :

CLR

$S \rightarrow AA$
$A \rightarrow aA$
$A \rightarrow b$

first of A = a | b

$S' \rightarrow S$
$S \rightarrow AA$
$A \rightarrow aA$
$A \rightarrow b$



Now In LALR We Will See whose Production are same but Symbol diff we will club them.