Finding First() & Follow().

Follow A contains set of all terminals present immediate right of A.

Rules:- Follow of start symbol is \$

$Fo(A) = \{ \$ \}$

A is the starting symbol.

3) $S \to aSbS \mid bSaS \mid \epsilon$.

Follow never contain ϵ.

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

Follow of S.

$\to Fo(S) \Rightarrow \{b, a, \$ \}$.

2) $S \to ACD$.
$C \to a \mid b$.

$Fo(A) \Rightarrow First(C) \Rightarrow \{a, b\}$.

Follow of (D) $\Rightarrow$ ~~First of D~~

Follow of $(S) = \{ \$ \}$.

$S \to AaAb \mid BbBa$.
$A \to \epsilon$.
$B \to \epsilon$.

$\Rightarrow Fo(A) \{a, b\}$
$Fo(B) = \{b, a\}$

$S \to \overset{\epsilon}{\cancel{A}}\overset{\epsilon}{\cancel{B}}\overset{\epsilon}{\cancel{C}}$
$A \to DEF$.
$B \to \epsilon$.
$C \to \epsilon$
$D \to \epsilon$.
$E \to \epsilon$.
$F \to \epsilon$.

$\Rightarrow Fo(S) \Rightarrow \{\$\}$
$Fo(A) = First(B)$
$\Rightarrow \boxed{\epsilon}$
$\Rightarrow First(C)$
$\Rightarrow Fo(S)$
$\Rightarrow \{\$\}$

First(A) contains all terminals present in first place of every string derived by A.

(1) $S \to abc \mid def \mid ghi$

(2) First (terminal) = terminal.

(3) First $(\epsilon) \Rightarrow \epsilon$.

$S \to ABc \mid ghi \mid jkl$.
$A \to a \mid b \mid c$
$B \to b$
$D \to d$.

$S \to \overset{\epsilon}{\cancel{A}}\overset{\epsilon}{\cancel{B}}\overset{\epsilon}{\cancel{C}}$
$A \to a \mid b \mid \epsilon$
$B \to c \mid d \mid \epsilon$
$C \to e \mid f \mid \epsilon$.

a, b, c, d, e, f, ϵ.
a, b, ϵ
c, d, ϵ
e, f, ϵ.

First of (S) = Fir(A)
$F(A) \Rightarrow \{a, b, c\}$
and $\{g, j\}$
$Fir(S) = \{a, b, c, g, j\}$

$E \rightarrow TE'$ $\{id, (\}$.

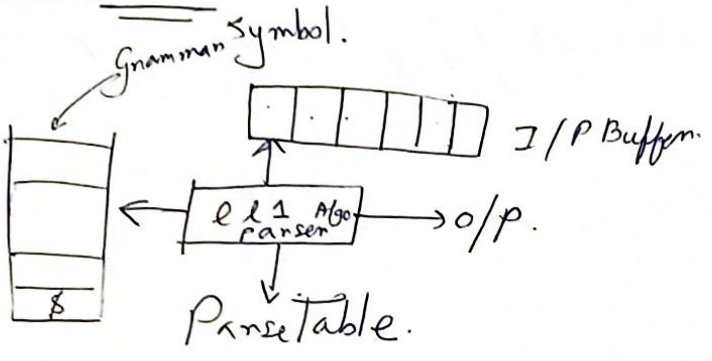
$E' \rightarrow *TE' / \epsilon$. $\{*, \epsilon\}$.

$T \rightarrow FT'$ $\{id, (\}$.

$T' \rightarrow \epsilon / +FT'$ $\{\epsilon, +\}$

$F \rightarrow id / (E$ $\{id, (\}$

ll1 parser.



Bare stack.

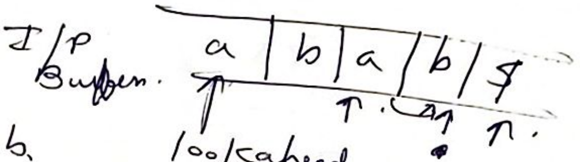
a.

a, b.

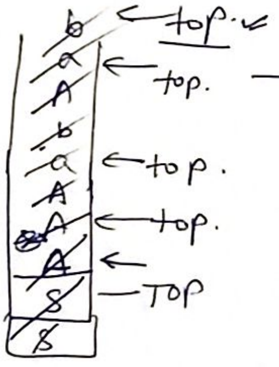
$S \rightarrow AA \rightarrow P1$

$A \rightarrow aA \rightarrow P2$

$A \rightarrow b \rightarrow P3$.



acceptance check of a string.



	a	b.
S	1	1
A	2	3.

acceptable

$S \rightarrow aSbS \mid bSaS / \epsilon$.

	a	b	$.
S	1/3	2/3	3

not ll1 Grammar.

ll1 Grammar Parsing table:-

		First.	Follow.
S	$\rightarrow (L) \mid a$	$\{(, a\}$	$\{\$, ,,)\}$
L	$\rightarrow SL'$	$\{(, a\}$	$\{)\}$
L'	$\rightarrow \epsilon \mid ,SL'$	$\{\epsilon, ,\}$	$\{)\}$

	(	)	a	,	$
S	1		2		
L	3		3		
L'		4		5	

In case of ε, L'-follow.

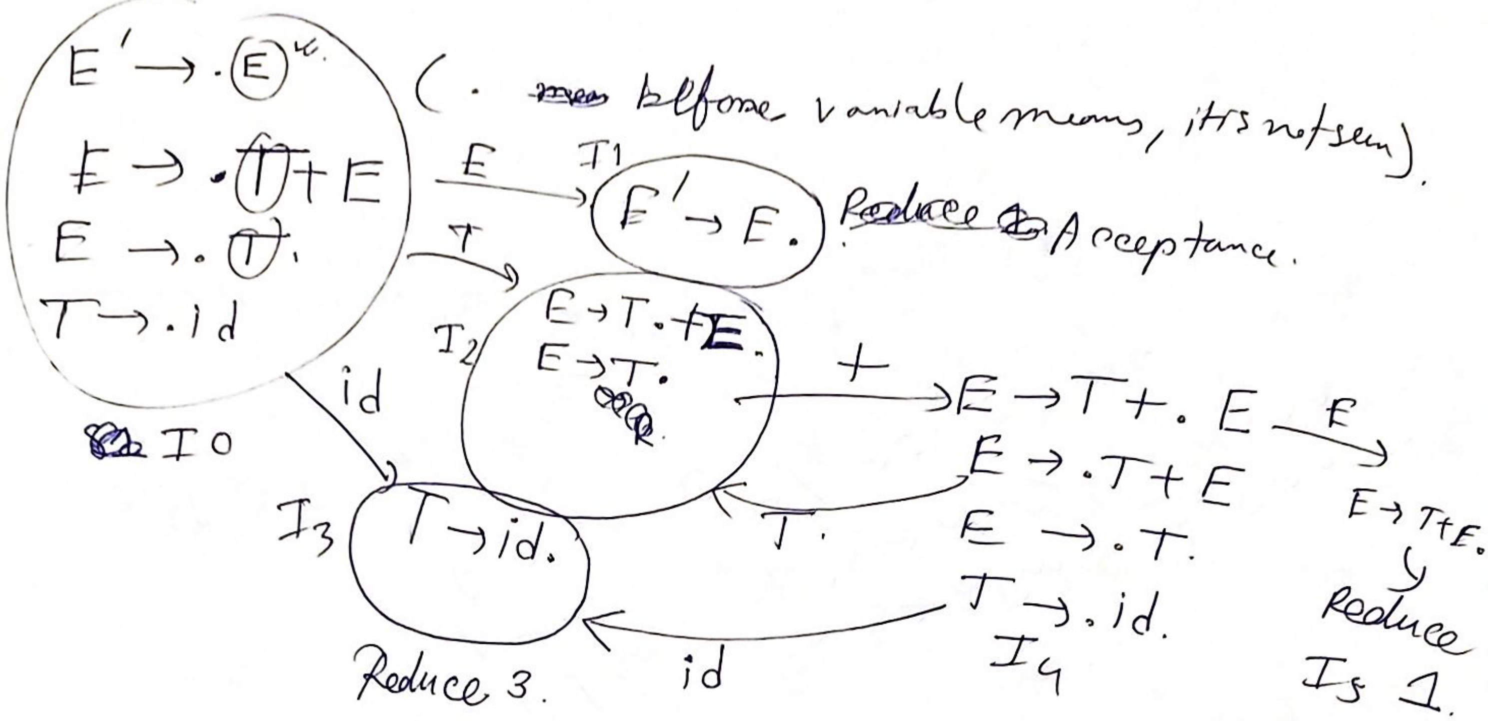
LL' Grammar.

LR-0 parsing table.

LR(0) CI.

E → T+E/T

T → id

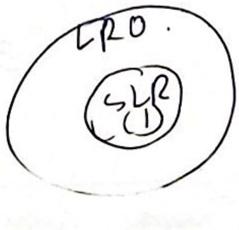


State	Action			Goto	
	id	+	$	E	T
0	S3			1	2
1			acceptance		
2	red2	4/red2	Red2		
3	R3	R3	R3		
4	S3			5	2
5	r1	r1	r1		

→ shift reduce conflict.

// SLR(1) → Here instead of writing reduce step. we will write in the follows of left side variable.

	id	+	$
0	S3		
1			accept
2			red2
3		Red3	Red3
4	S3		
5			R1



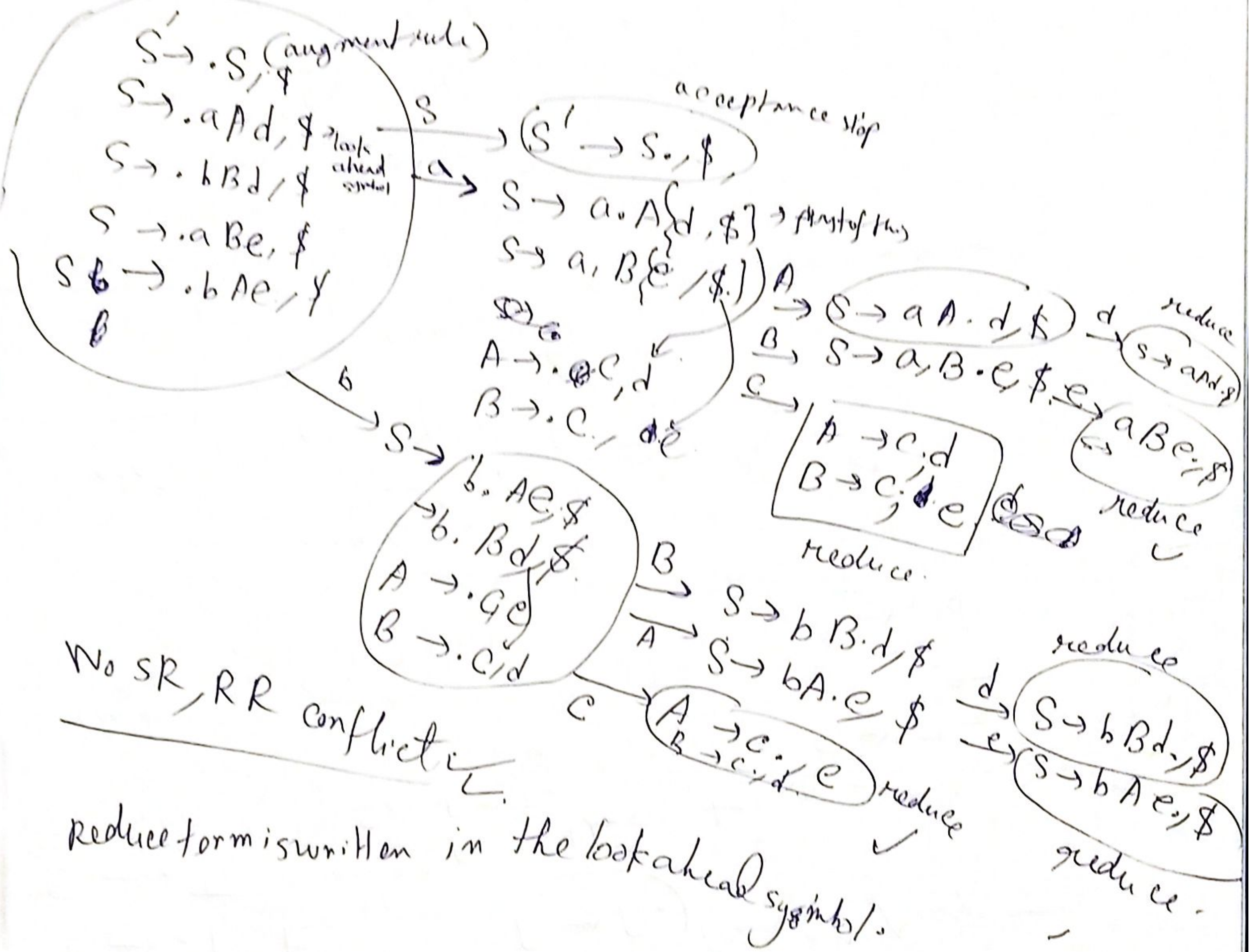
SLR (1) > LR(0)

CLR. Parsing table (LR1 Cannonical items).

S → aAd / bBd / aBe / bAe.

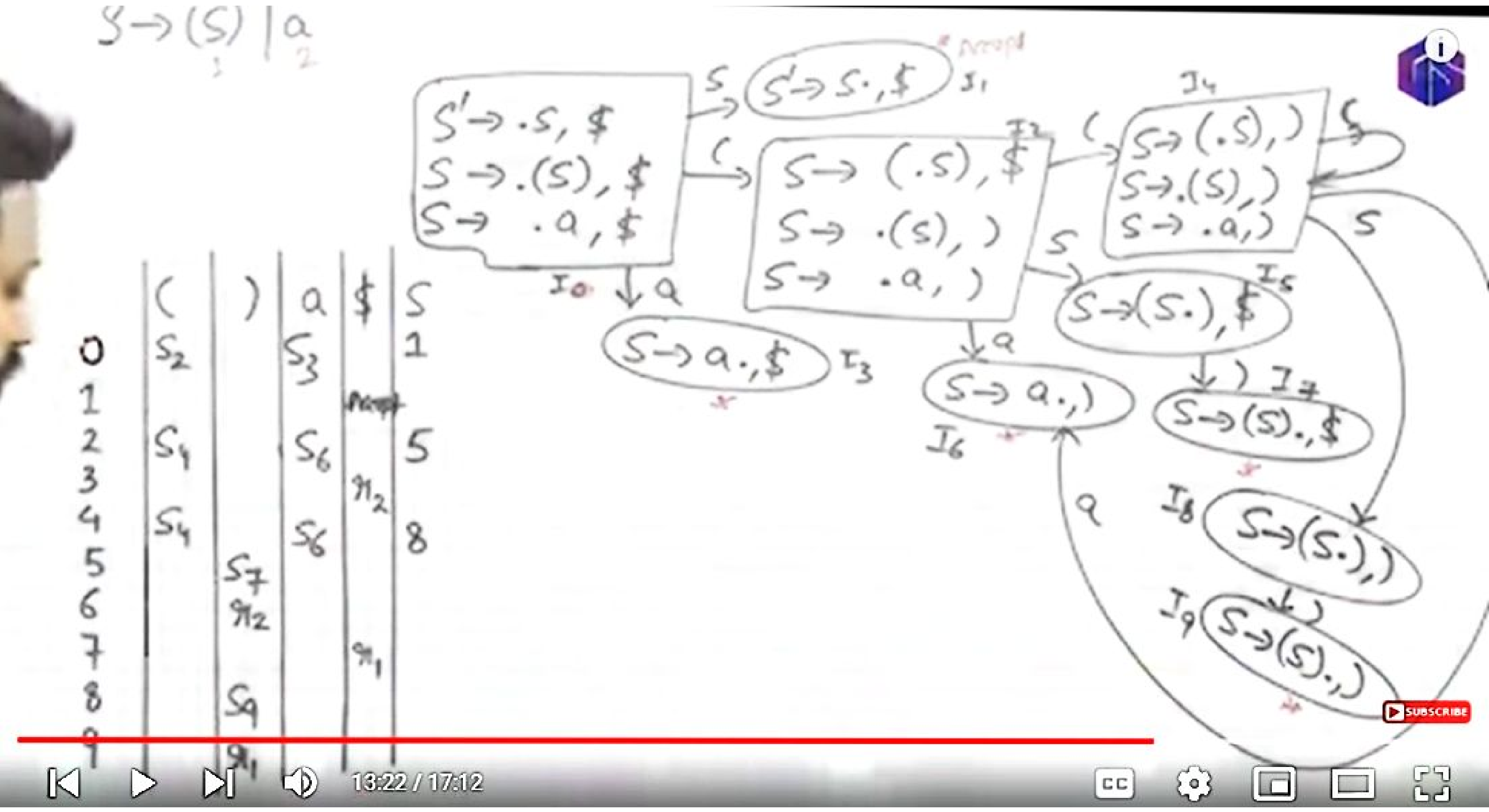
A → c

B → c



No SR, RR conflict.

Reduce form is written in the lookahead symbol.



$S \to (S) \mid a$
1 2

	(	)	a	$	S
0	S_2		S_3		1
1				Accept	
2	S_4		S_6		5
3				r_2	
4	S_4		S_6		8
5		S_7			
6		r_2			
7				r_1	
8		S_9			
9		r_1			

13:22 / 17:12

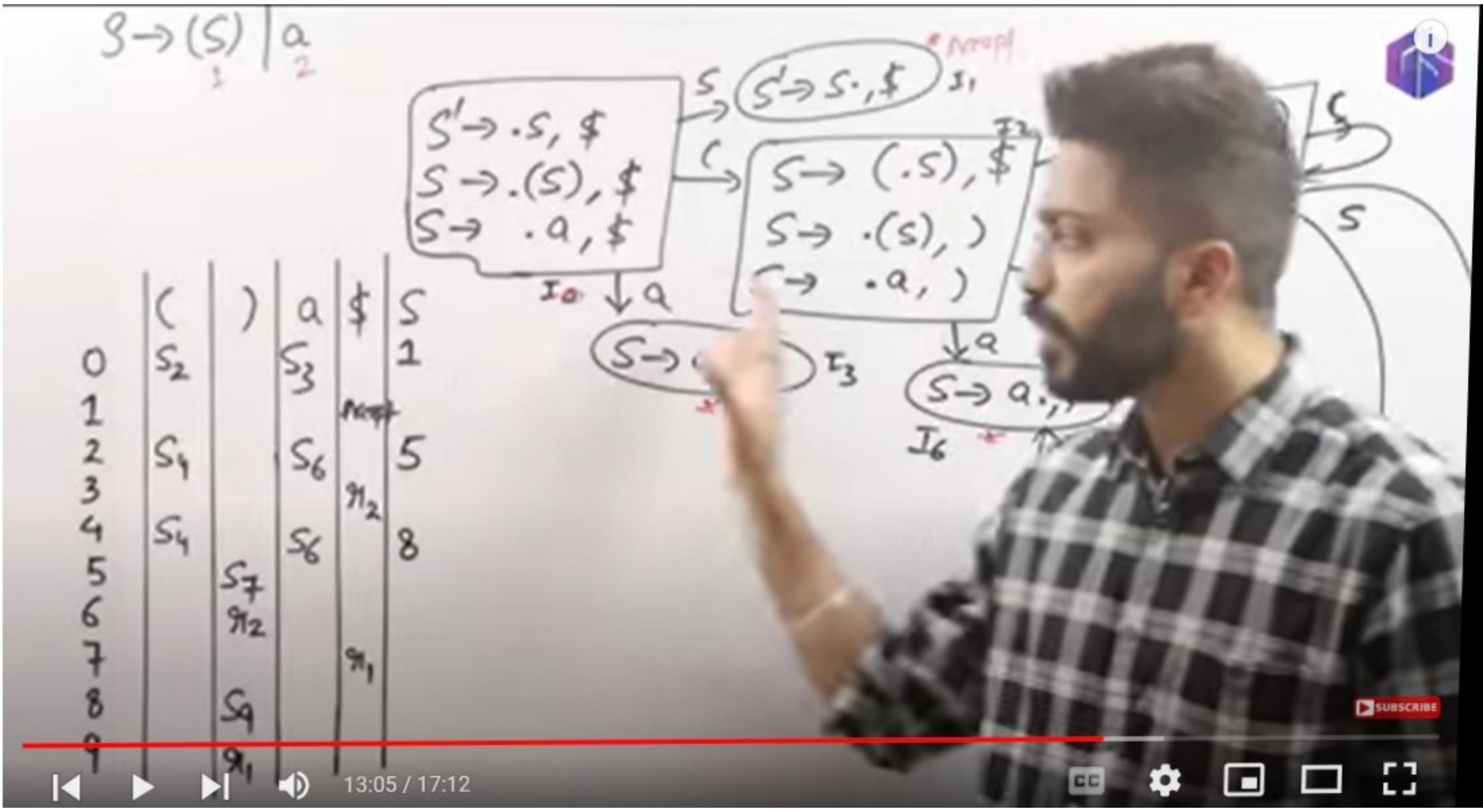
LALR < CLR

CLR → no of states are higher than any parser.

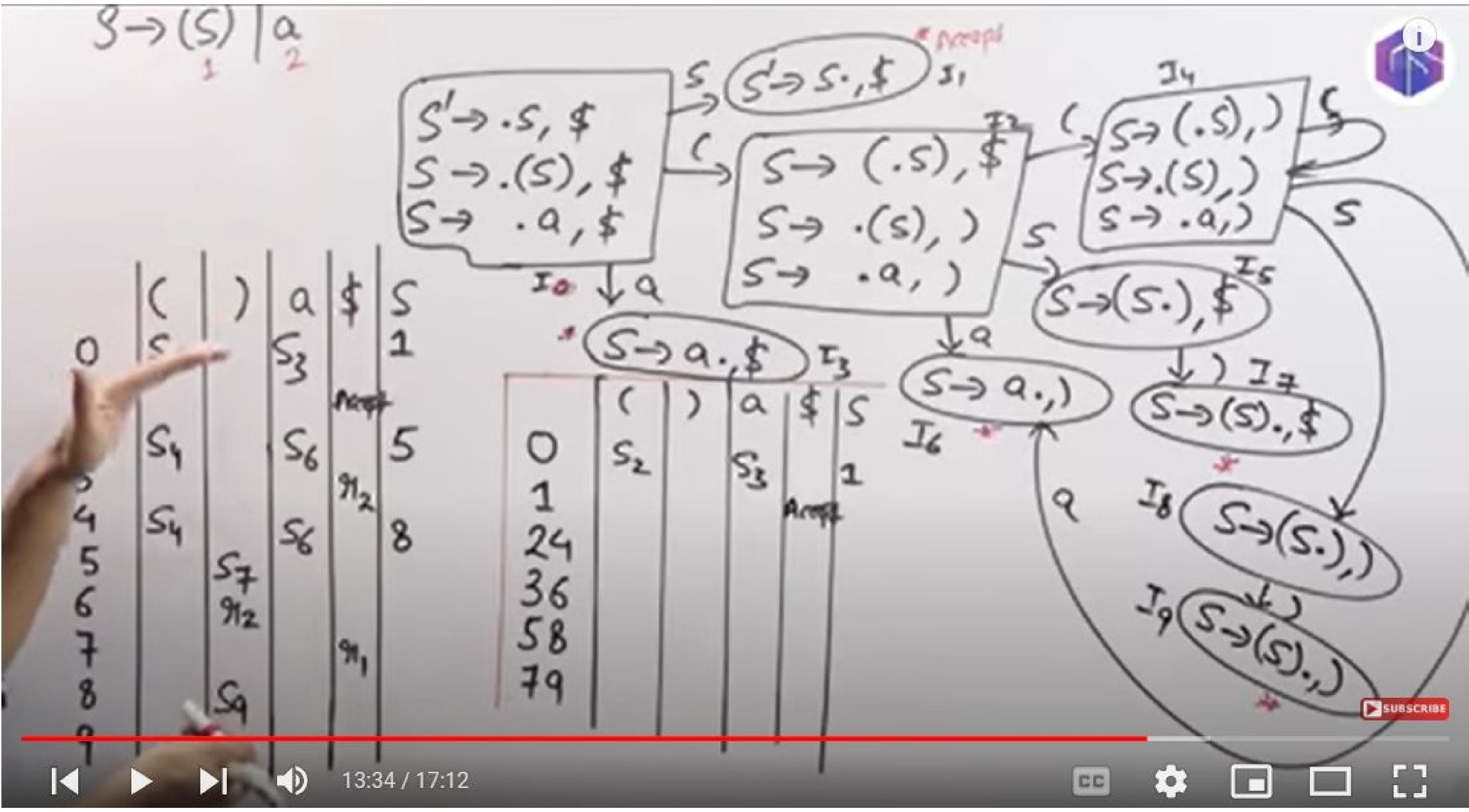
LALR nothing but merging of states.
If possible, then carry on.
If not, → Grammer is not LALR.

CLR → SRX
LALR → SRX.

CLR → RR X.
LALR → may be or may not be.



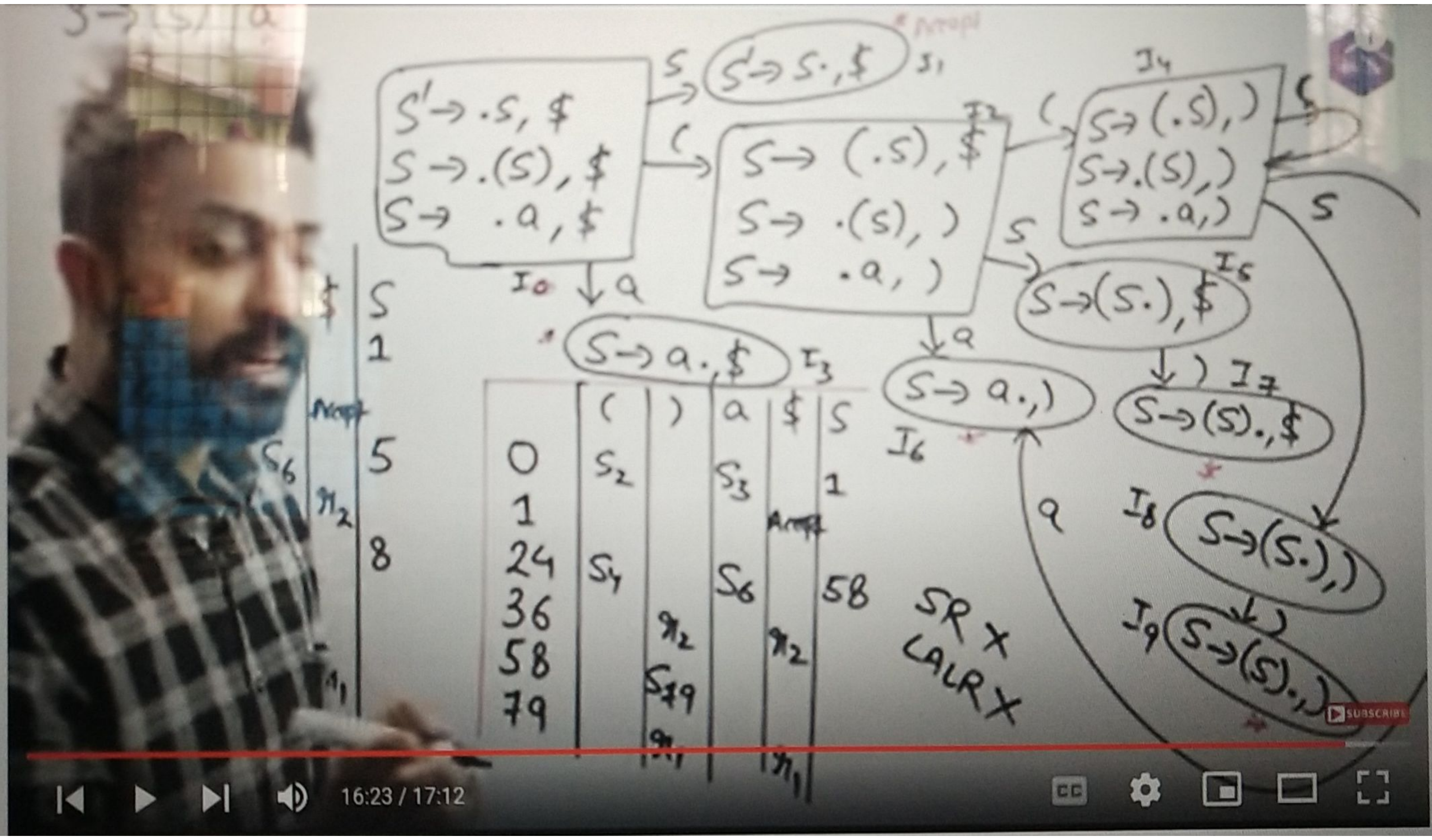
	(	)	a	$	S
0	S2		S3		1
1				accept	
2	S4		S6		5
3				r2	
4	S4		S6		8
5		S7			
6		r2			
7				r1	
8		S9			
9		r1			



$S \to (S) \mid a$
1 2

	(	)	a	$	S
0	S		S_3		1
				Accept	
	S_4		S_6		5
3				r_2	
4	S_4		S_6		8
5		S_7			
6		r_2			
7				r_1	
8		S_9			
9					

	(	)	a	$	S
0	S_2		S_3		1
1				Accept	
24					
36					
58					
79					



	(	)	a	$	S
0	S2		S3		1
1				Accept	
24	S4		S6		58
36		r2		r2	
58		S79			
79		r1		r1	