CS & IT ENGINEERING

Compiler Design

Lexical and Syntax Analysis

Lecture No. 10

By- DEVA Sir

Relations

LR Tables

LR parsing Algo

LR(0) Table

SLR(1) Table

LALR(1) Table

CLR(1) Table

LR Algorithm

Entries
- Shift Entry
- Reduced Entry
- Goto Entry (State Entry)
- Acceptance Entry
- Blank Entry

No. of Rows X No. of columns

No. of states X $(|T|+1+|V|)$

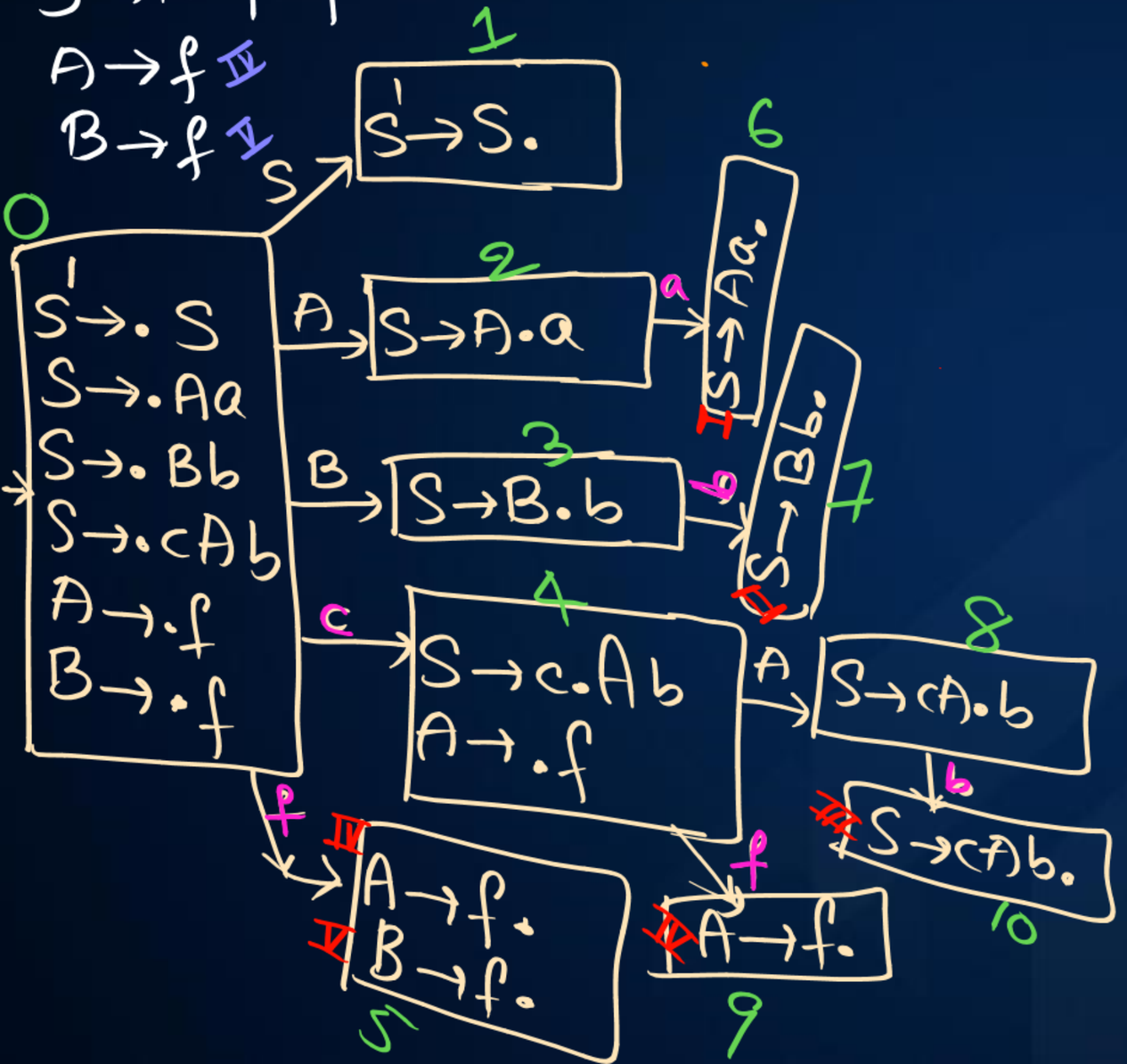
Entries in LL(1) Table :

LR(0) Table construction

$S \to \overset{I}{Aa} \mid \overset{II}{Bb} \mid \overset{III}{cAb}$

$A \to f$ IV

$B \to f$ V



← ACTION Table → ← Goto Table →

	a	b	c	f	$	S	A	B
0			S_4	S_5		1	2	3
1					ACCEPT			
2	S_6							
3		S_7						
4				S_9			8	
5	R_{IV} R_V	R_{IV} R_V	R_{IV} R_V	R_{IV} R_V	R_{IV} R_V			
6	R_I	R_I	R_I	R_I	R_I			
7	R_{II}	R_{II}	R_{II}	R_{II}	R_{II}			
8		S_{10}						
9	R_{III}	R_{III}	R_{III}	R_{III}	R_{III}			
10	R_{III}	R_{III}	R_{III}	R_{III}	R_{III}			

SLR(1) Table construction

$S \to \overset{I}{Aa} \mid \overset{II}{Bb} \mid \overset{III}{cAb}$

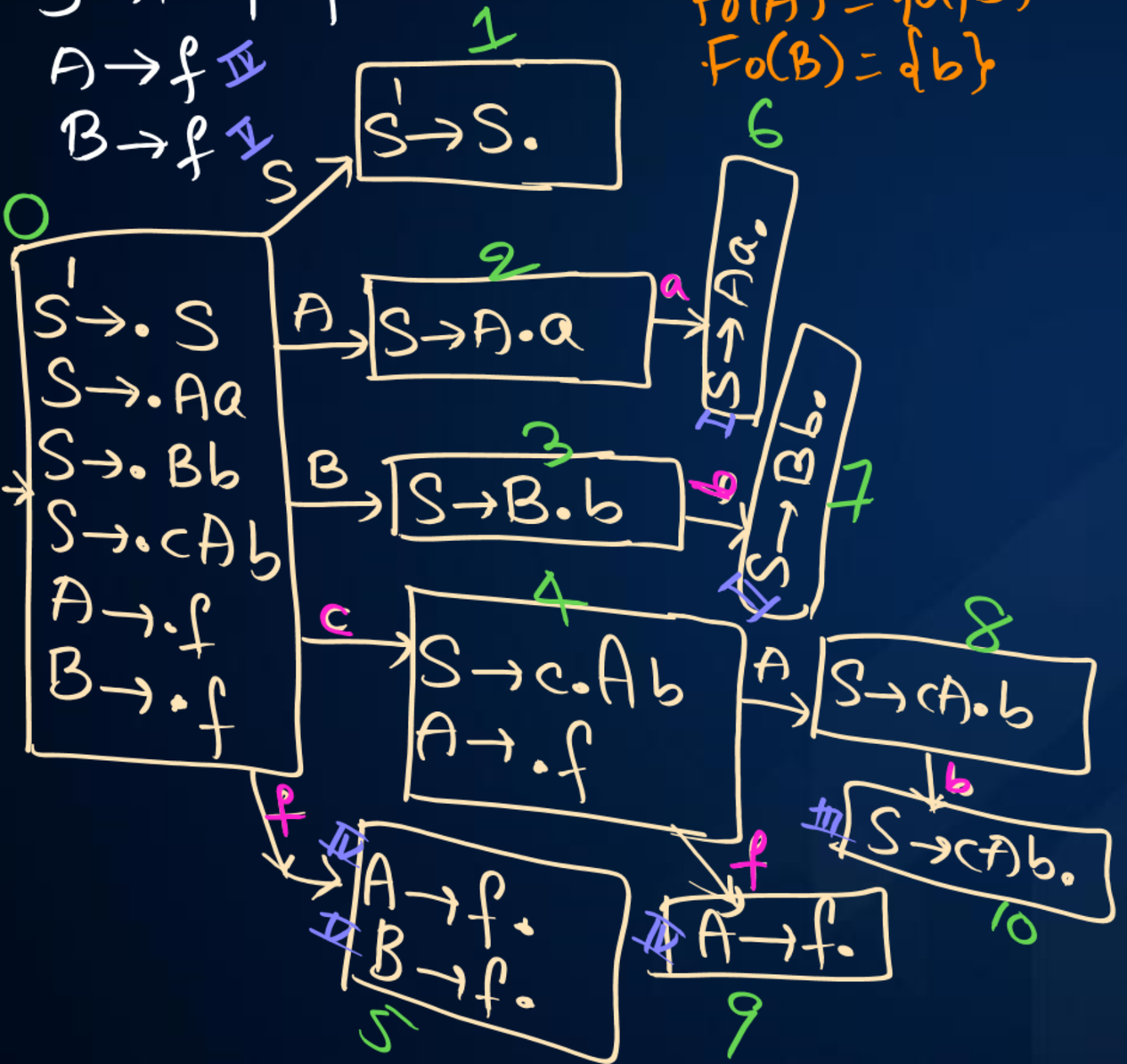
$A \to f$ IV

$B \to f$ V

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

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

$Fo(B) = \{b\}$



← ACTION Table → ← Goto Table →

	a	b	c	f	$	S	A	B
0			S_4	S_5		1	2	3
1					ACCEPT			
2	S_6							
3		S_7						
4				S_9			8	
5	R_{IV}	R_{IV} R_V						
6					R_I			
7					R_{II}			
8		S_{10}						
9	R_{IV}	R_{IV}						
10					R_{III}			

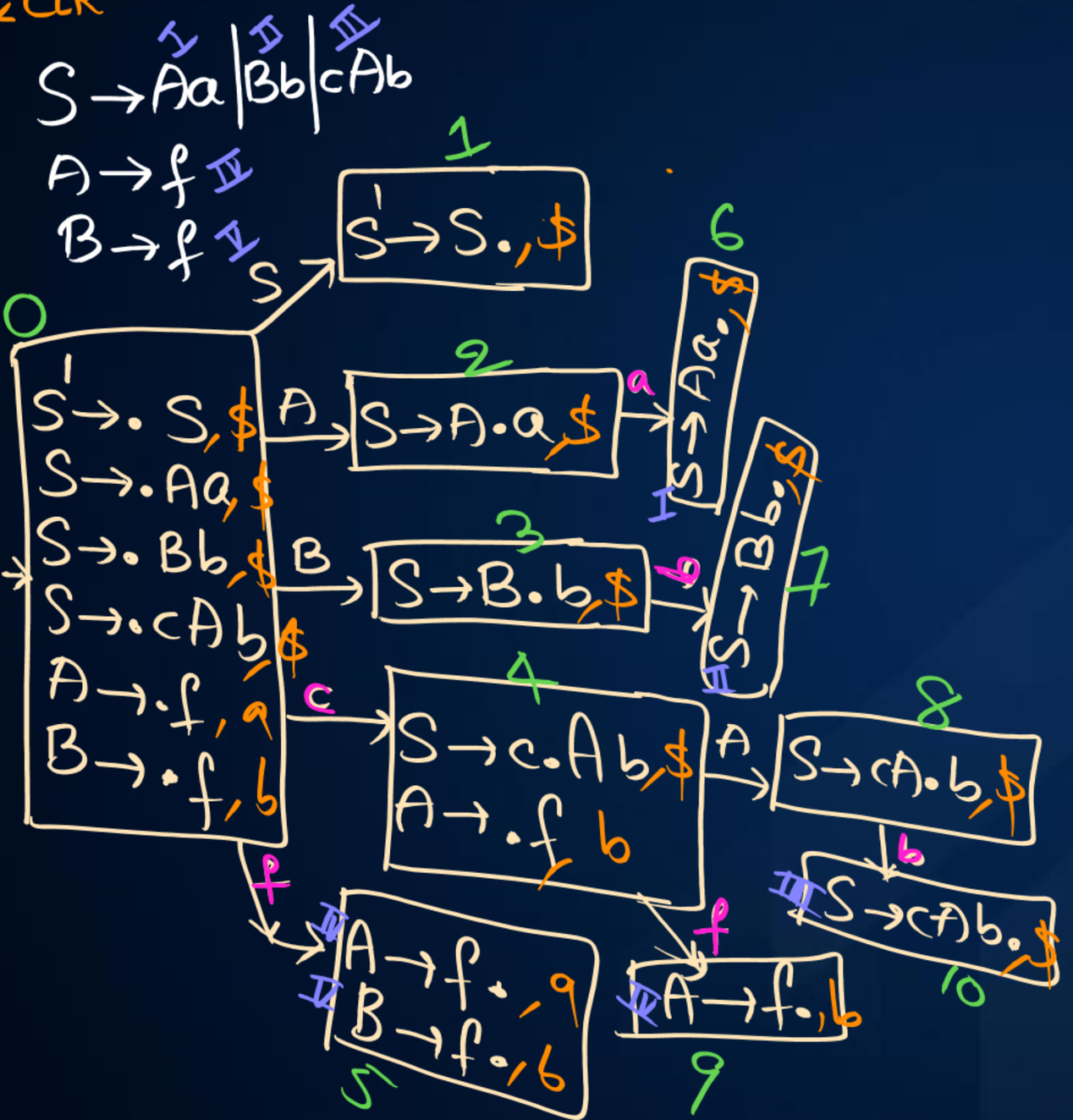
LALR(1) Table construction

↙ CLR

I II III
S → Aa | Bb | cAb

IV
A → f

V
B → f



	ACTION Table					GOTO Table		
	a	b	c	f	$	S	A	B
0			S_4	S_5		1	2	3
1					ACCEPT			
2	S_6							
3		S_7						
4				S_9			8	
5	R_{IV}	R_V						
6					R_I			
7					R_{II}			
8		S_{10}						
9		R_{III}						
10					R_{III}			

No. of states

$$\#\text{states}(LR(0)) = \#\text{states}(SLR) = \#\text{states}(LALR) \leq \#\text{states}(CLR)$$

Table size

$$|LR(0)\ \text{Table}| = |SLR\ \text{Table}| = |LALR\ \text{Table}| \leq |CLR\ \text{Table}|$$

No. of shift Entries

[no. of shift entries in LR(0), SLR, LALR are same] $\leq$ [no. of shift entries in CLR]

No. of goto Entries

[no. of goto entries in LR(0), SLR, LALR are same] $\leq$ [no. of goto entries in CLR]

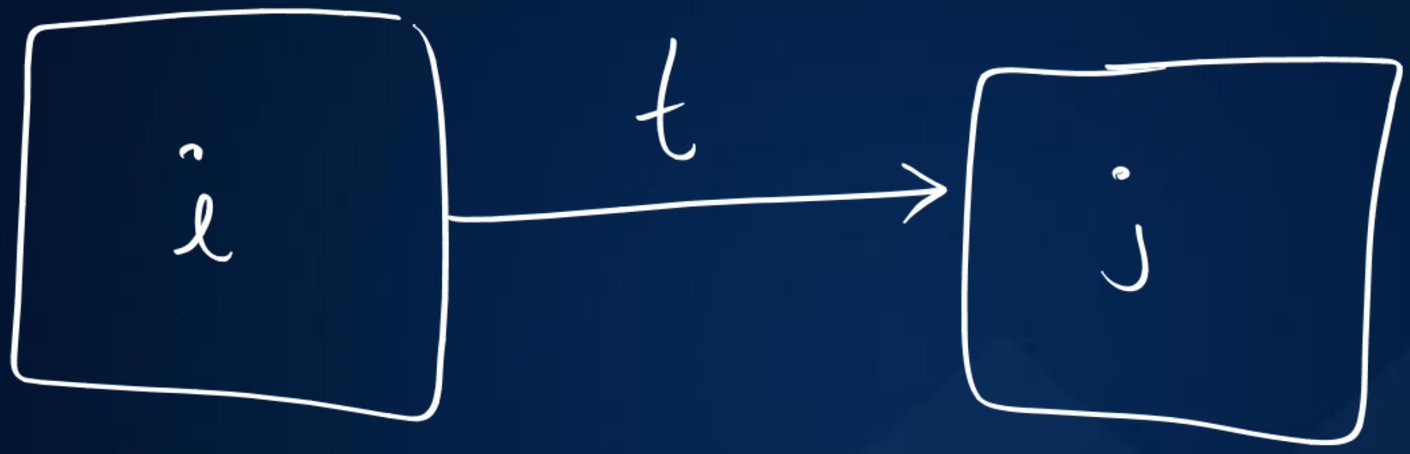
No. of Reduced Entries

[no. of Reduced entries in LR(0)] $\geq$ [in SLR] $\geq$ [in LALR] $\geq$ [in CLR]

No. of conflicts

in LR(0) $\geq$ in SLR $\geq$ in LALR $\geq$ in CLR

Rule for shift Entry in all LR Tables :

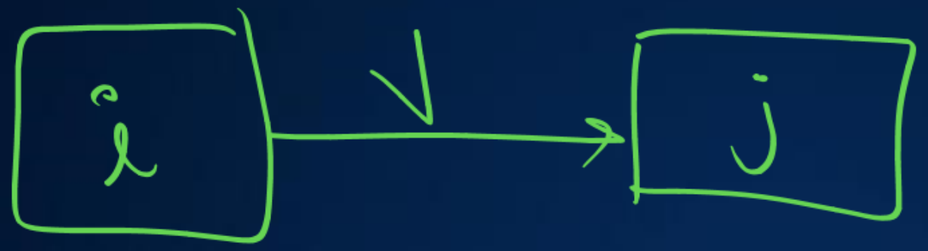


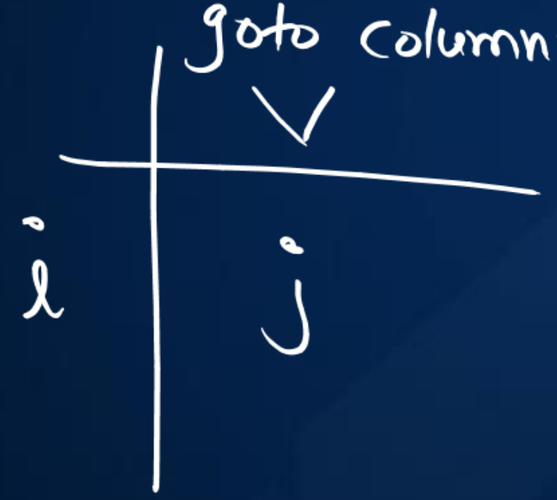
How many shift Entries?

= No of Terminal transitions in DFA

	t
i	S_j

Rule for goto Entry in all LR Tables :



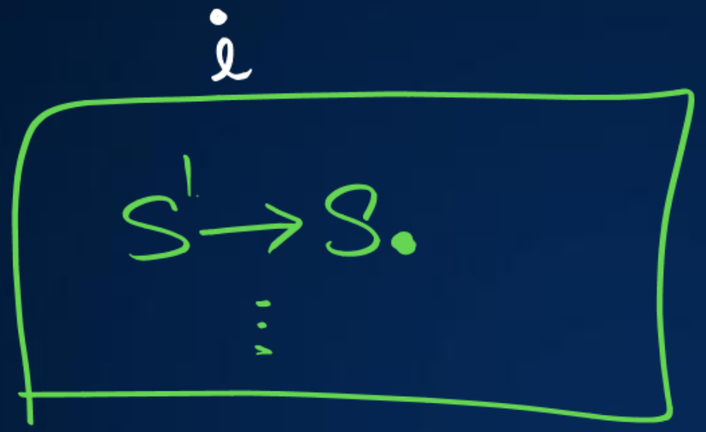


How many goto entries ?

= No. of nonterminal transitions in DFA

= No. of goto entries in Table

Acceptance Entry Rule in all LR Tables :



	$
i	Accept

Reduced Entry Rule in LR(0) Table :

P : $X \to \alpha .$

⋮

i

	ACTION				
i	R_P	R_P	R_P	R_P	R_P

Reduced Entry in SLR Table

P (number) : $X \to \alpha .$ — i (state)

Follow(X) = { t_1, t_3 }

	t	t_3
i	R_P	R_P

Reduced Entry Rule in LALR & CLR

P : $X \to \alpha ., t_1/t_2$

i

	t_1	t_2
i	R_P	R_P

If CFG is CLR and combining states to get LALR then RR Conflicts possible.

(SR conflict is impossible)

In CLR

$X \to \alpha.,\ L_1$ (a)

$Y \to \alpha.,\ L_2$ (b)

no RR Conf

$X \to \alpha.,\ L_3$ (b)

$Y \to \alpha.,\ L_4$ (c)

No RR conf

Combine to get LALR

$X \to \alpha.,\ \{a,b\}\ L_1 \cup L_3$

$Y \to \alpha.,\ L_2 \cup L_4$

$\{b, c\}$

possibility of RR conflict

no SR conf

$X \to \alpha.\ t\beta,\ L_1$

$Y \to \alpha.,\ L_2$

$t \notin L_2$

In CLR

&

no SR conf

$X \to \alpha.\ t\beta,\ L_3$

$Y \to \alpha.,\ L_4$

$t \notin L_4$

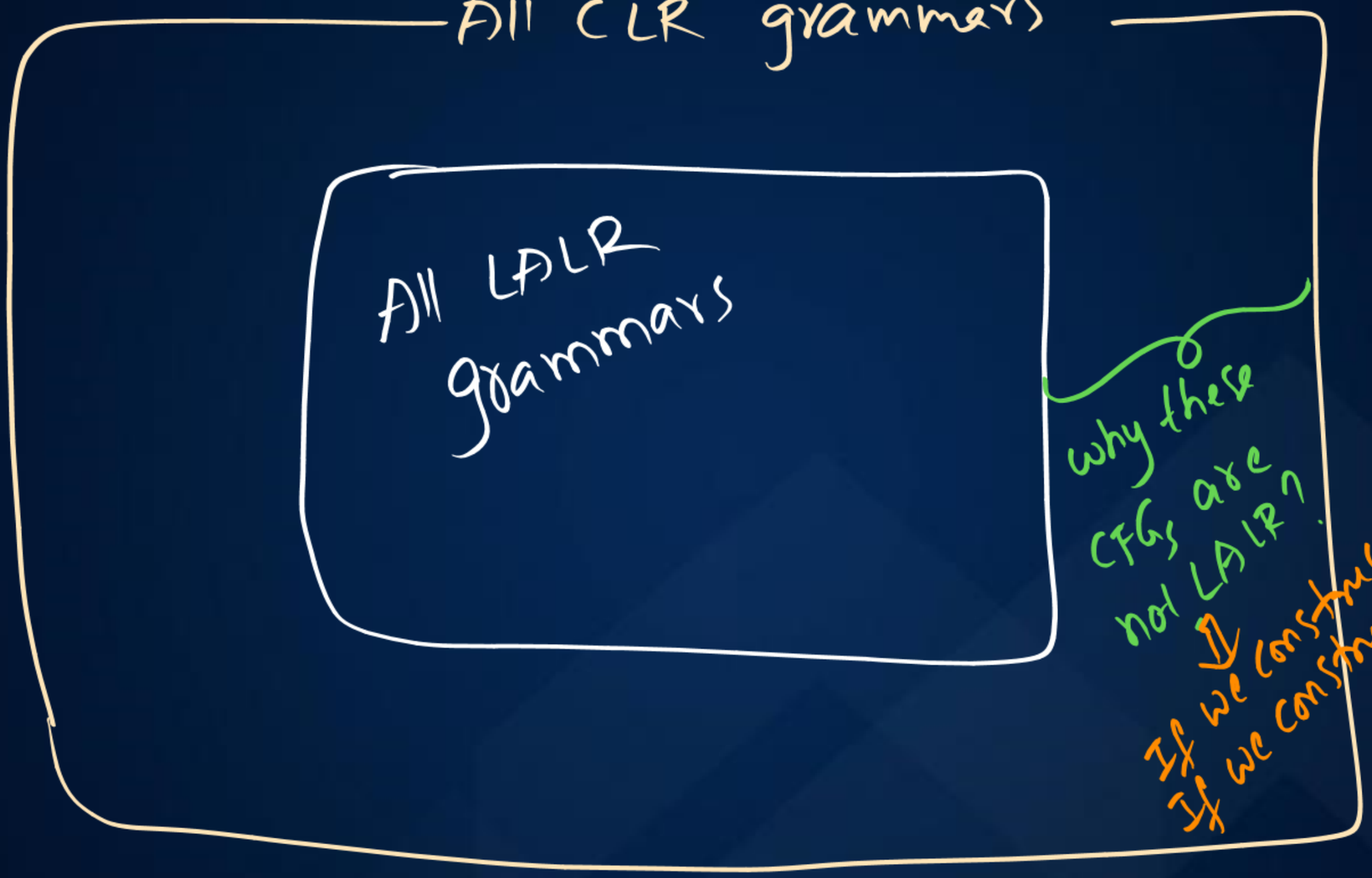
Combine to get LALR

Impossible to get SR conflict

$X \to \alpha.\ t\beta,\ L_1 \cup L_3$

$Y \to \alpha.,\ L_2 \cup L_4$

$t \notin L_2 \cup L_4$


All CLR grammars
All LALR grammars
Why these CFGs are not LALR?
If we construct CLR - no conflicts
If we construct LALR - RR conflicts exist

Input : fa

$S \to \overset{I}{Aa} \mid \overset{II}{Bb} \mid \overset{III}{cAb}$

$A \to f$ IV

$B \to f$ V

Step1 : Input | f | a | $ |

M[0,f] : S_5

Step2 :

M[5,a] : R_{IV}

$A \to f$

i) Pop 2*|f|

ii) Push A

iii) Push M[below A, A] = 2

Step3 :

M[2,a] : S_6

Step4 :

M[6,$] : R_I

$S \to Aa$

Step5 :

M[1,$] : ACCEPT

Shift ⇒ Reduce ⇒ Shift ⇒ Reduce ⇒

← ACTION Table → ← Goto Table →

	a	b	c	f	$	S	A	B
0			S_4	S_5		1	2	3
1					ACCEPT			
2	S_6							
3		S_7						
4				S_9			8	
5	R_{IV}	R_V						
6					R_I			
7					R_{II}			
8		S_{10}						
9		R_{III}						
10					R_{III}			

S

Aa

fa

Production

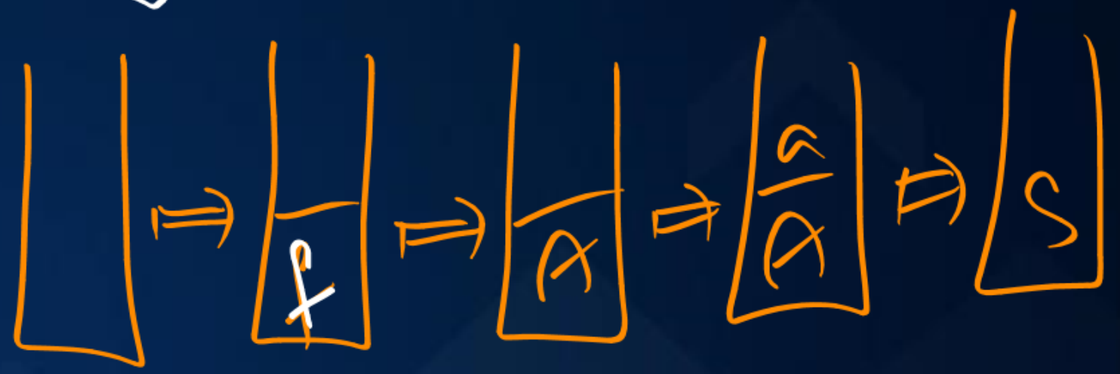
A → f (Handle)

Reverse of RMD

✓ f is viable prefix

✓ A is viable prefix

✓ Aa is viable prefix



$\cdot fa \xrightarrow{\text{Shift}} f \cdot a \xrightarrow{\text{Reduced}} A \cdot a \xrightarrow{\text{Shift}} Aa \cdot \xrightarrow{\text{Reduced}} S.$

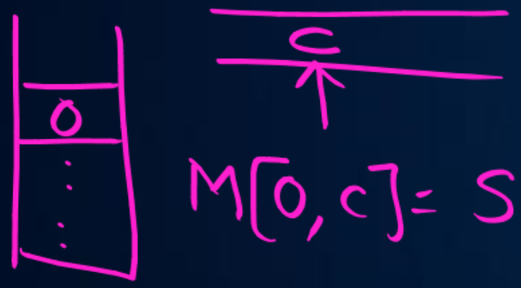
Viable prefix :

↳ Top of stack sequence and prefix of Right Sentential form where dot will not allowed to move further

Set of viable prefixes is Regular language

Shift Action | Reduced Action | Accept | Error

	c
0	S_4



$M[0, c] = S_4$

i) PUSH c (input)

ii) PUSH 4 (next state)

iii) Inc pointer

	a
5	$A \to \beta$

I) POP $2 * |RHS|$

II) PUSH LHS
A

III) PUSH $\leq [\underset{A}{\text{below}}, A]$

Closure( ) ✓

goto( ) ✓

Identify type of LR grammar ✓

Relation among LR grammars ✓

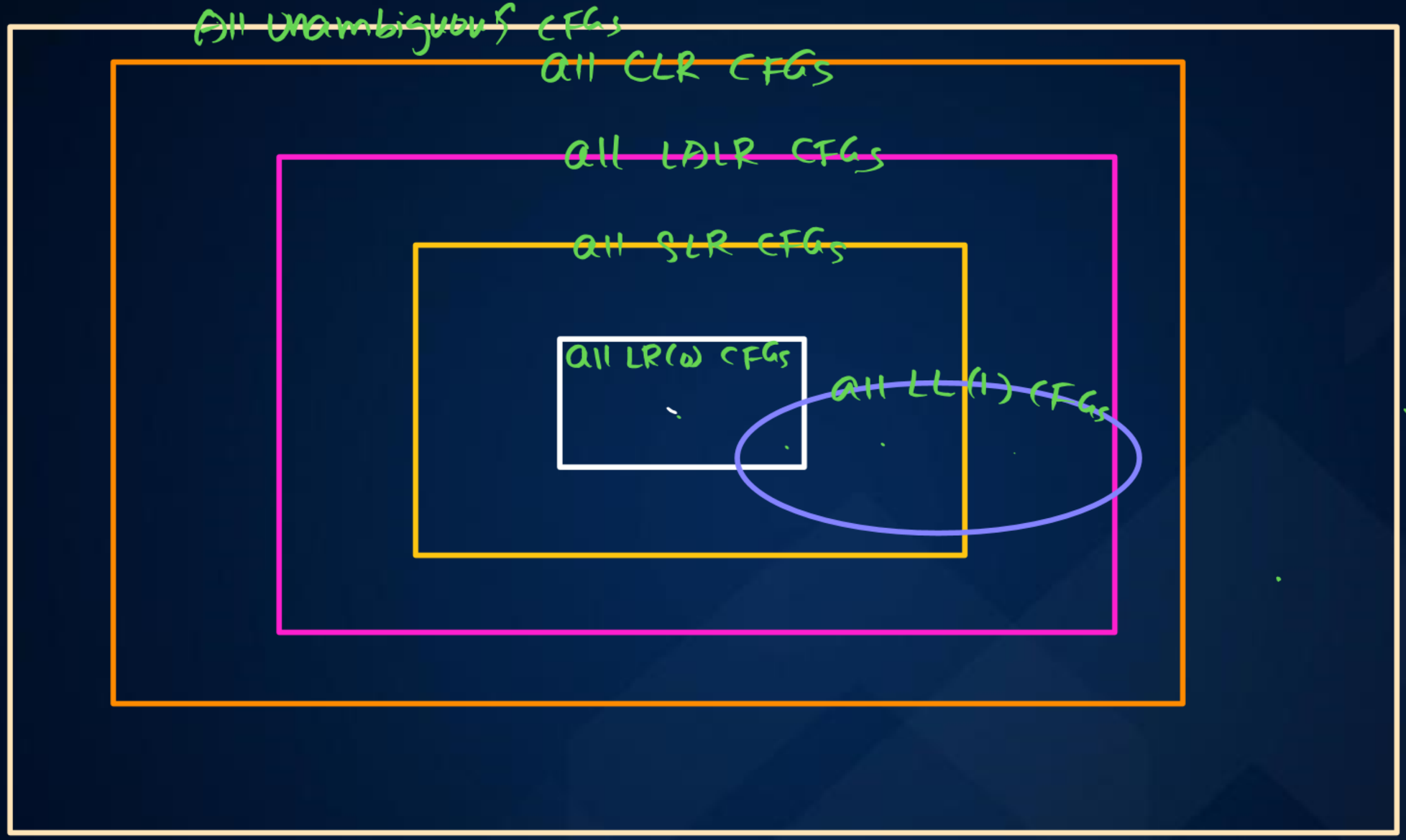
Identify LL(1) CFG ?

FIRST & Follow computation ✓

Elimination of Left Rec ✓

Left Factoring ✓

Ambiguous & unambiguous CFG ✓


All Unambiguous CFGs
All CLR CFGs
All LALR CFGs
All SLR CFGs
All LR(0) CFGs
All LL(1) CFGs
Ambiguous CFGs

I) Every LL(1) is LR(1) / CLR

II) Every LR(0) is SLR

III) Every SLR is LALR

IV) Every LALR is CLR / LR(1)

V) Every Ambiguous CFG is not CLR

VI) Every not CLR is not LALR

VII) Every not LALR is not SLR

VIII) Every not SLR is not LR(0)

IX) Every LR(0) is Unambiguous

X) " SLR "

XI) " CLR "

XII) " LALR "

XIII) " LL(1) "

Summary

- LR parsers ✓

Next: Operator Precedence Parsing


THANK YOU
GW
SOLDIERS !