Lakhan Kumawat
1906055.

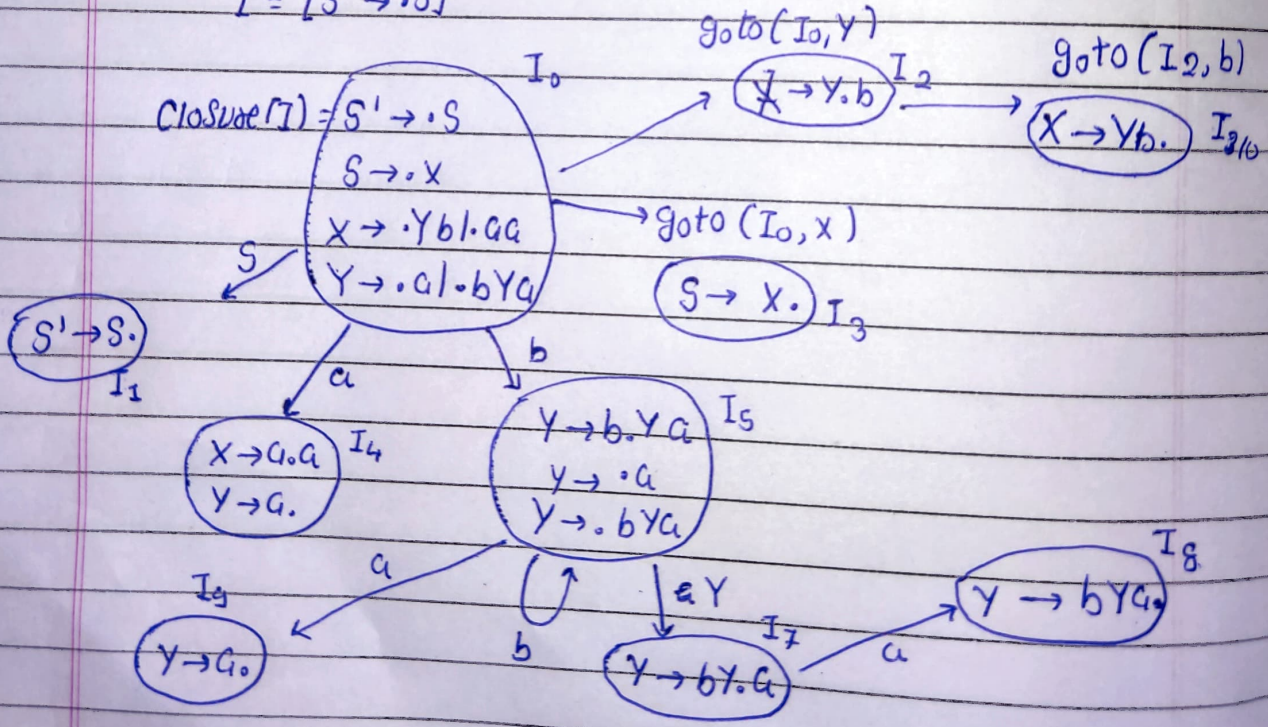
Solution 1) $S \to X$ (1)
$X \to Yb$ (2) $| aa$ (3)
$Y \to a$ (4) $| bYa$ (5)

Step 1: Augmented grammar.

$S' \to S$
$S \to X$
$X \to Yb \mid aa$
$Y \to a \mid bYa$

Step 2: Canonical Collection of LR(0) items.

$I = [S' \to .S]$



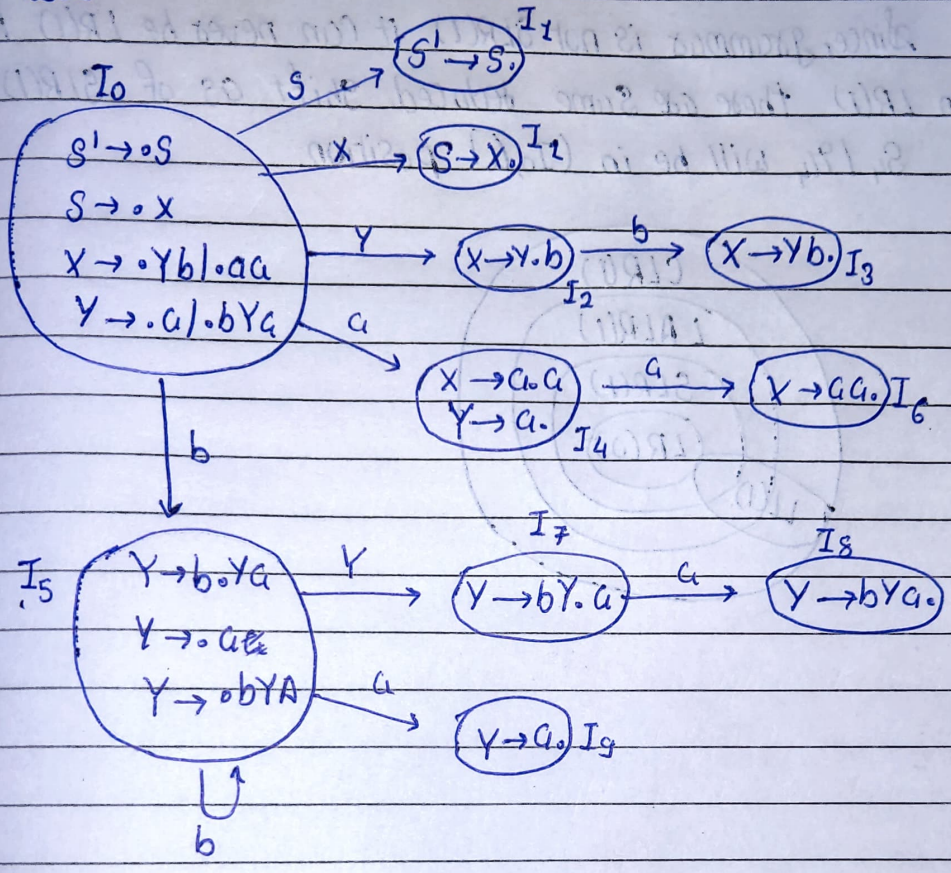
In State I_4, we have shift grammar $(X \to a.a)$ & 1 reduced grammar $(Y \to a.)$ so S-R conflict.

So given grammar not in SLR(1).

Lakhan Kumawat
1906055

Page No.
Date

Solution 17 Continue: DED



Follow (S) = {$}
Follow (X) = {$}
Follow (Y) = {a, b}

	Action			Goto		
	a	b	$	S	X	Y
I_0	S_4 / δ_4	S_5		1	10	2
I_1			accept			
I_2		δ_3				
I_3						
I_4	δ_{11}					
I_5	δ_9	S_5	S_7			7
I_6						
I_7	δ_8					

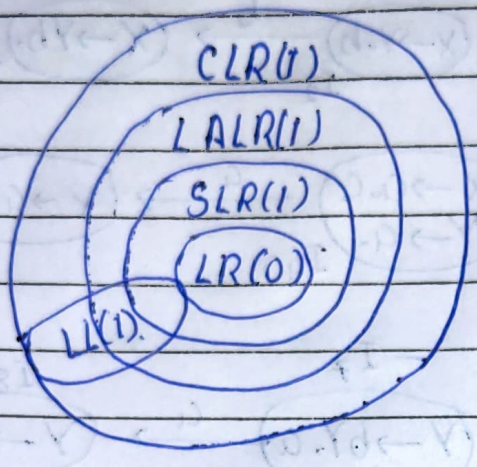
Note: Incomplete table as we already get S-R Conflict

Lakhan Kumawat
2906055.

Page No.
Date

Solution 1> Continued...

Since, grammar is not SLR(1) it can never be LR(1) because in LR(1) these are same reduced shift as of SLR(1). So S_4 / R_4 will be in (I_0, A) position.



Lakhan Kumawat
1906055.

Page No.
Date

Solution 2.) $S \to AaAb \mid BbBa$
$A \to \varepsilon$
$B \to \varepsilon$

1. For LL(1)

First(S) = {a, b}
First(A) = ε
First(B) = ε
Follow(A) = a, b.
Follow(B) = a, b

1. If $A \Rightarrow a$ and $A \Rightarrow b$ are two different rules of grammar, then it should be that First(a) ∩ First(b) = ϕ. Hence two sets haven't any common element.

2. For any non-terminal symbol A you have $A \Rightarrow^* \varepsilon$, then it should be that FIRST(A) ∩ Follow(A) = ϕ. Hence if there is zero production for a non-terminal symbol, then the first and follow sets can't have any common element.

So, for given grammar.

We have FIRST(AaAb) ∩ FIRST(BbBa) = ϕ. Since FIRST(AaAb) = {a} while FIRST(BbBa) = {b} and they don't have any common elements.

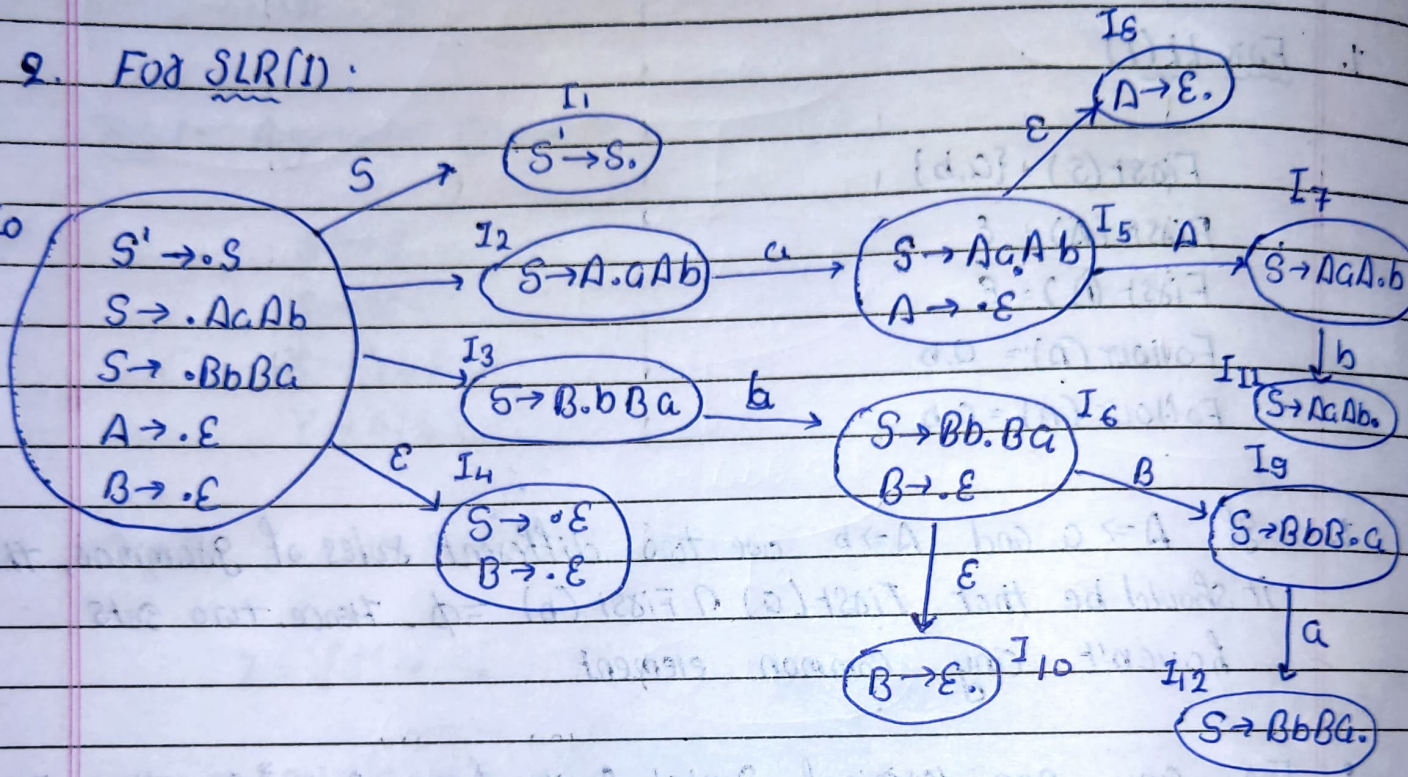
Now let's compute LL(1) table, By definition, if we don't get conflicts grammar is LL(1)

Lakhan kumawat
1906055.

	a	b
S	1	2
A	3	3
B	4	4

** As these are no conflicts, the grammar is LL(1).

2. For SLR(1):



SLR Parsing table

Lakhan Kumawat
1906055

Page No.
Date

SLR Parsing table

State	Action				Goto		
	a	b	ε	$	S	A	B
0			S_4	δ_4	1	2	3
1				Accept			
2	S_5						
3		S_6					
4	δ_3/δ_4	δ_3/δ_4					
5			S_8			7	
6			S_{10}				9
7		S_{11}					
8	δ_3	δ_3					
9	S_{12}						
10	δ_4	δ_4					
11				δ_1			
12				δ_1			

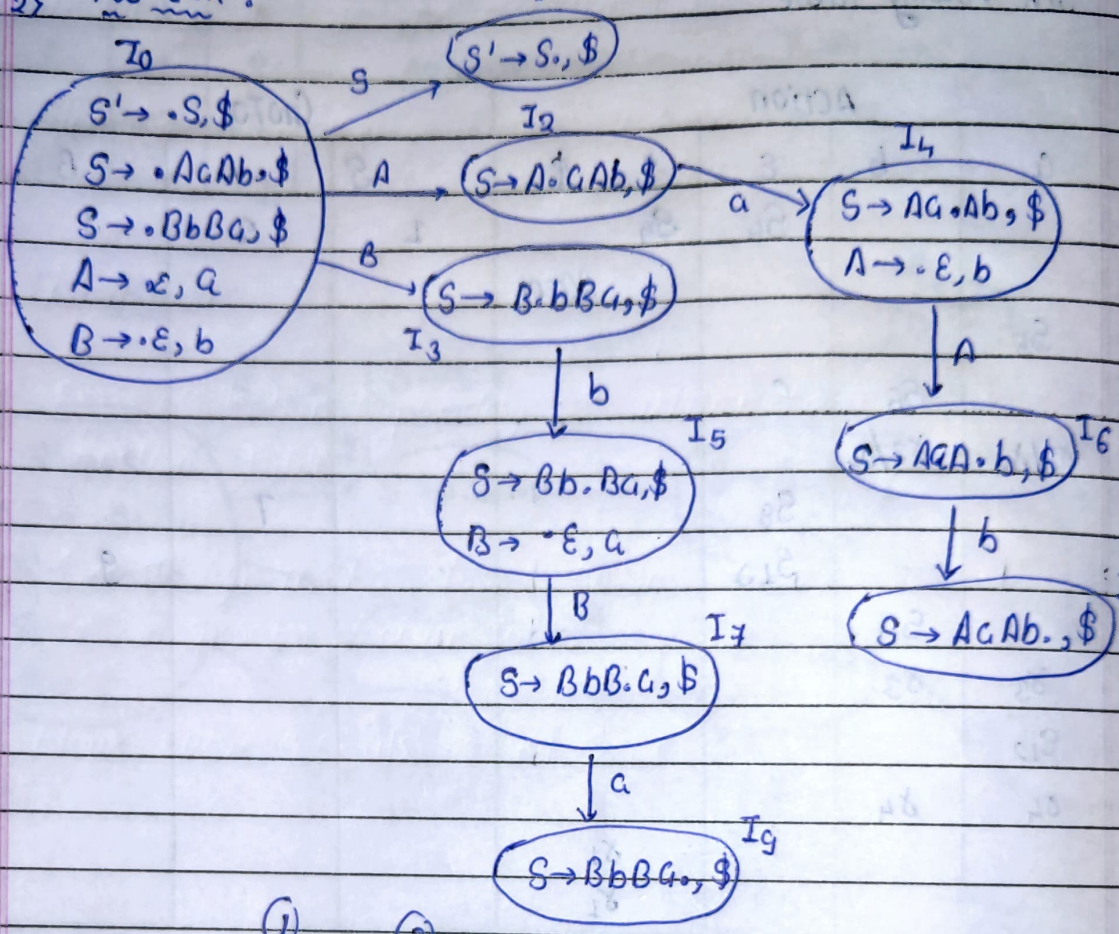
* So, the table has multiply defined entries, so given grammar is not an SLR(1).

* SLR grammars are a superset of all LR(0) grammars.
So the given grammar is not LR(0) grammar as it is not SLR(1).

Lakhan Kumawat
1906055

Page No.
Date

Solution 2> For CLR :



(1) (2)
S → AaAb / BbBa
A → ε (3)
B → ε (4)

Data Flow Diagram for CLR(1) and LALR

State	Action			Goto		
	a	b	$	S	A	B
I_0	γ_3			1	2	3
I_1	γ_4		Accept			
I_2	S_4					
I_3		S_5				
I_4					6	
I_5						7
I_6		S_8				9
I_7						
I_8			γ_1			
I_9			γ_2			

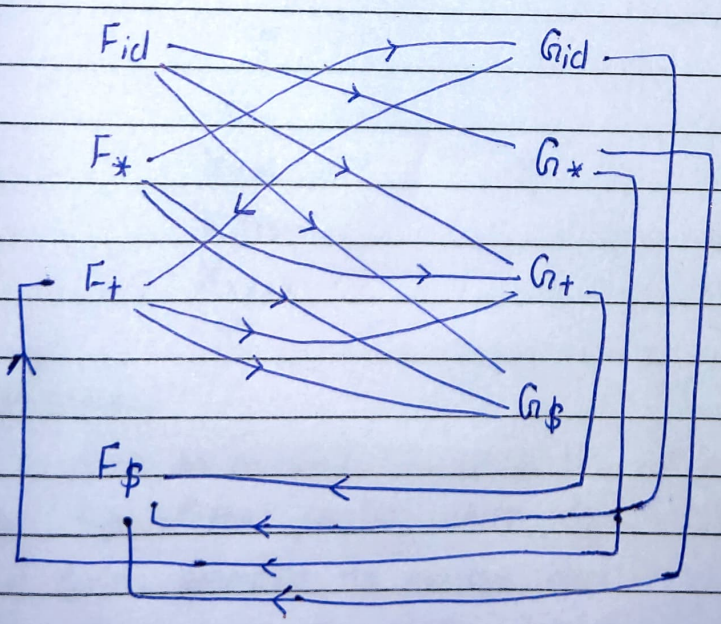
Lakhan Kumawat
1906055

Page No.
Date

Solution 3) $X \to X+Y \mid Y$
$Y \to Y*Z \mid Z$
$Z \to a \mid b \mid c \mid d$

* operator precedence table:-

	+	*	a	b	c	d	$
+	>	<	<	<	<	<	>
*	>	>	<	<	<	<	>
a	>	>	-	-	-	-	>
b	>	>	-	-	-	-	>
c	>	>	-	-	-	-	>
d	>	>	-	-	-	-	>
$	<	<	<	<	<	<	Accept



	id	+	*	$
F	4	2	4	0
G	5	1	3	0

Lakhan Kumawat
1906055

Page No.
Date

Advantage of operator function table:-

⇒ The time Complexity of operation relation table is $O(n^2)$ but the time relation Complexity is $O(n)$

Limitation:-

→ Error detection Capability of relation table is greater than function table.

→ Even though it has blank entries these are non-blank entries also in operator function table.

→ Blank entries are also called error.

Lakhan Kumawat
1906055

Page No.
Date

Solution 4> $S \to A$

(i) $S \to X$.

$X \to Xx \mid \varepsilon$

(ii) $S \to X$

$X \to xX \mid \varepsilon$

Both of these grammars are SLR(1). However, the SLR(1) parsers for one of these grammars will use use O(n) space in its parsing stack when run on string "x*". While the other parsers will only use O(1) stack space.

Solution: An SLR(1) parser traces out a rightmost derivation in reverse. If we try deriving a^* using both grammars. We get the following right most derivations.

S	S
Xx	xX
Xxx	xxX
$Xxxx$	$xxxX$
$Xxxxx$	$xxxxX$
...	...

Justification:

Since in a shift / reduce parse all reductions occur at the top of the parsing stack, this means that in the case of first grammar the parsing stack never has more than two symbols on the parsing stack. We could'nt do this reduction.

Lakhan Kumawat
1906055

Page No.
Date

because deduction performed is $X \to Xx$. If there were more than two symbols on the passing stack, we ~~do~~ couldn't do this deduction. The second grammar, on the other hand, does the deduction $X \to xX$, meaning that there will be a string of $O(n)$ x's on the passing stack, followed by X for which the deduction is done.