Compiler Design (KCS-502)
3rd year (Semester – V)
Session – 2020 - 21
Unit – II
Notes – 9

Anil Singh
Asst. Prof.
CSE Dept.
UCEM, Prayagraj

Constructing SLR Parsing tables

- We construct the SLR parsing ACTION and GOTO functions from the DFA that recognizes viable prefixes.

- It will not produce uniquely-defined parsing action tables for all grammar but it does succeed on many grammars for programming languages.

Constructing SLR Parsing tables

- **Algorithm:**
  1) Construct an augmented grammar G' for the input grammar G.
  2) Construct FOLLOW(A), for all A Є N.
  3) Construct C = {I_0, I_1, …… , I_n}, the collection of sets of LR(0) items for G'.
  4) Construct state I from I_i.
  5) In order to define the entries for state I of the ACTION table and GOTO table, do steps (i) to (v) for each I_i.

Constructing SLR Parsing tables

i. If $[A \rightarrow \alpha.a\beta] \in I_i$ and $GOTO(I_i, a) = I_j$ then $ACTION(I_i, a) = S_j$ which means shift to state j. Here 'a' is the terminal symbol.

ii. If $[A \rightarrow \alpha.] \in I_i$ then $ACTION(I_i, a) = r_j$ for all $a \in FOLLOW(A)$ except $A = S'$.

iii. If $[S' \rightarrow S.] \in I_i$ then $ACTION(I_i, \$) = Accept$. If any conflicting actions are generated by the above rules, we say the grammar is not SLR(1). The algorithm fails to produce a parser in this case.

iv. For all non-terminals A, if $GOTO(I_i, A) = I_j$ then $GOTO(I_i, A) = j$.

v. All the remaining entries in the ACTION table and the GOTO table are marked as error.

6) The initial table of the parser is the one constructed from the set of items containing $[S' \rightarrow .S]$.

Constructing SLR Parsing tables

- The parsing table consisting of the parsing ACTION and GOTO functions determined by this algorithm is called the SLR table for G.
- An LR parser using the SLR table for G is called the SLR parser for G and a grammar having an SLR parsing table is said to be SLR(1).

Constructing SLR Parsing tables

Example 1:

Find the SLR or LR(0) parsing table for the grammar

$$E \rightarrow E+T \mid T$$

$$T \rightarrow T*F \mid F$$

$$F \rightarrow id$$

Constructing SLR Parsing tables

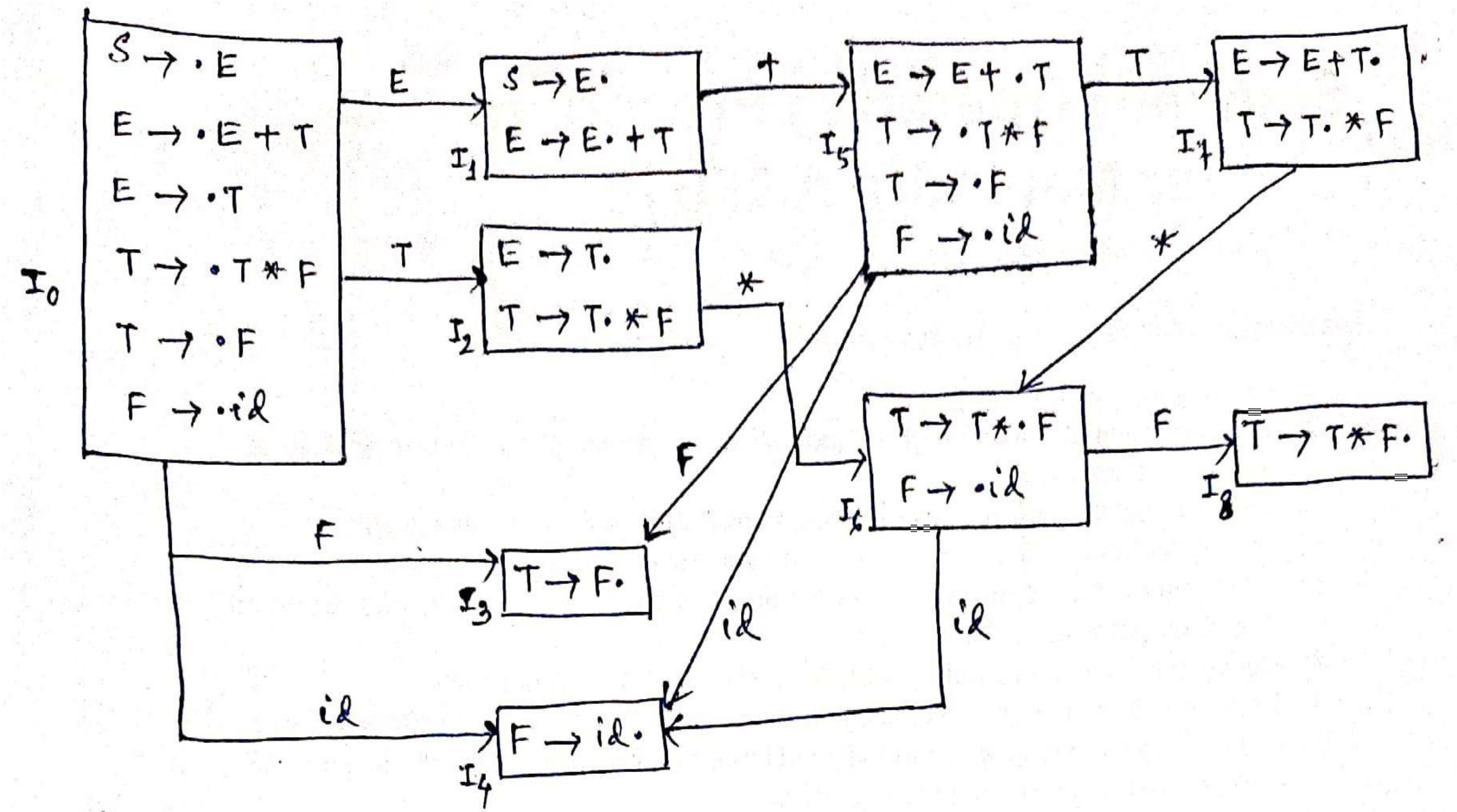
Solution:

The augmented grammar G' is

$$S \rightarrow E$$
$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow id$$

The transition diagram for the given grammar is as follows:

Constructing SLR Parsing tables



Constructing SLR Parsing tables

Now we prepare a FOLLOW set for all non-terminals.

FOLLOW(E) = {$, +}

FOLLOW(T) = {$, *, +}

FOLLOW(F) = {$, *, +}

The productions are numbered as

(1) $E \rightarrow E+T$

(2) $E \rightarrow T$

(3) $T \rightarrow T*F$

(4) $T \rightarrow F$

(5) $F \rightarrow id$

Constructing SLR Parsing tables

Now build SLR parsing table as follows:

Entry in the action table ACTION(I_0, id) = S_4 because GOTO(I_0, id) = I_4

Similarly, ACTION(I_1, +) = S_5 since GOTO(I_1, +) = I_5

ACTION(I_2, *) = S_6

ACTION(I_5, id) = S_4

ACTION(I_6, id) = S_4

ACTION(I_7, *) = S_6

Now we will fill it up with reduce and accept action.

We will also fill up the GOTO table for all the non-terminals.

Constructing SLR Parsing tables

So final SLR table is as follows:

ACTION table					GOTO table		
State	**id**	**+**	*****	**$**	**E**	**T**	**F**
I_0	S_4				1	2	3
I_1		S_5		**Accept**			
I_2		r_2	S_6	r_2			
I_3		r_4	r_4	r_4			
I_4		r_5	r_5	r_5			
I_5	S_4					7	3
I_6	S_4						8
I_7		r_1	S_6	r_1			
I_8		r_3	r_3	r_3			

Constructing SLR Parsing tables

Example 2:

Construct a SLR or LR(0) parsing table for the following grammar:

$$E \rightarrow E+T \mid T$$

$$T \rightarrow TF \mid F$$

$$F \rightarrow F^* \mid a \mid b$$

Constructing SLR Parsing tables

Solution:

The augmented grammar G' is

$$S \rightarrow E$$

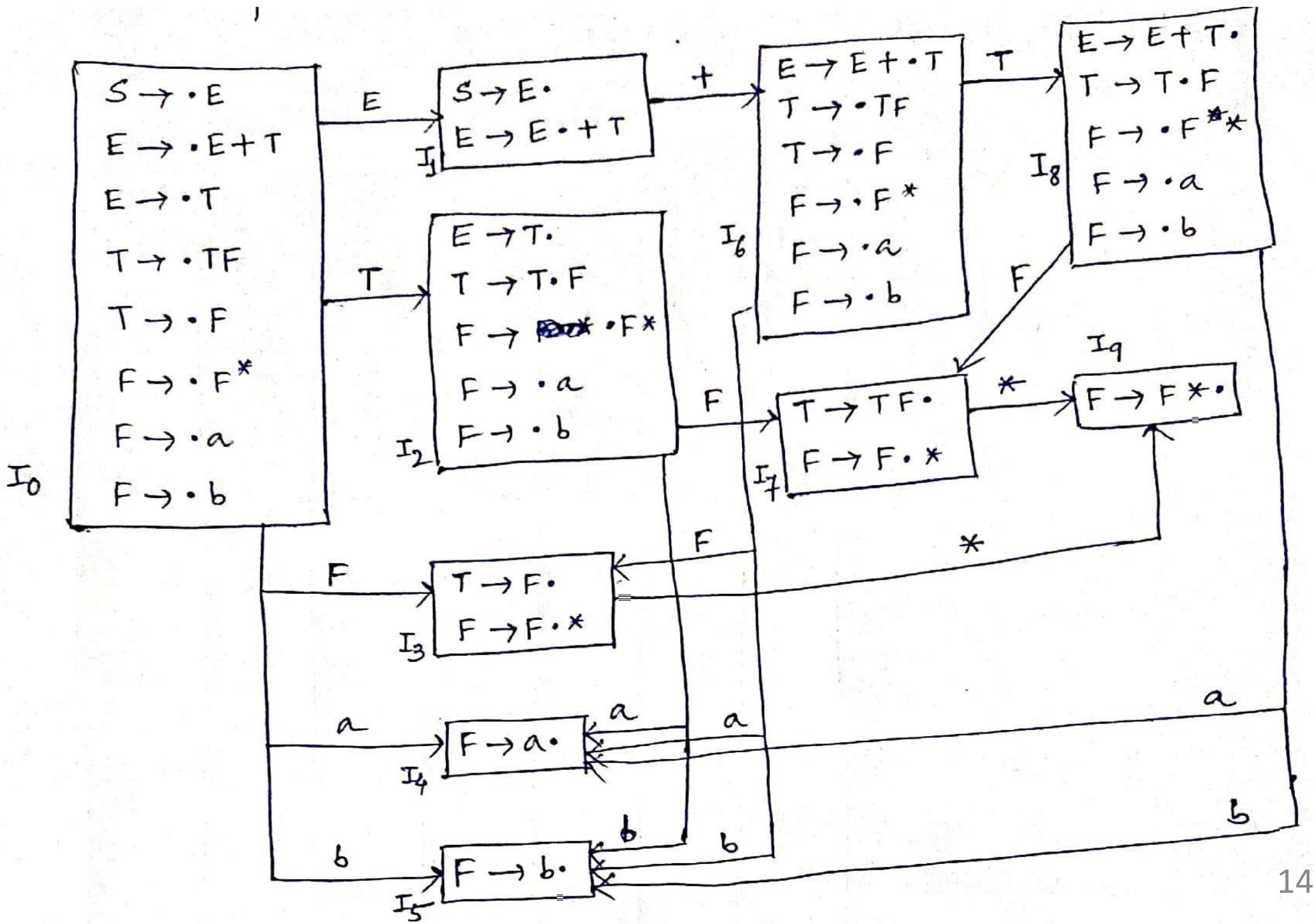
$$E \rightarrow E+T \mid T$$

$$T \rightarrow TF \mid F$$

$$F \rightarrow F^* \mid a \mid b$$

The transition diagram for the given grammar is as follows:

Constructing SLR Parsing tables



Constructing SLR Parsing tables

Now we prepare a FOLLOW set for all non-terminals.

FOLLOW(E) = {$, +}

FOLLOW(T) = {$, +, a, b}

FOLLOW(F) = {$, *, +, a, b}

The productions are numbered as

(1) E → E+T

(2) E → T

(3) T → TF

(4) T → F

(5) F → F*

(6) F → a

(7) F → b

Constructing SLR Parsing tables

So final SLR table is as follows:

ACTION table						GOTO table		
State	**a**	**b**	**+**	*****	**$**	**E**	**T**	**F**
I_0	S_4	S_5				1	2	3
I_1			S_6		Accept			
I_2	S_4	S_5	r_2		r_2			7
I_3	r_4	r_4	r_4	S_9	r_4			
I_4	r_6	r_6	r_6	r_6	r_6			
I_5	r_7	r_7	r_7	r_7	r_7			
I_6	S_4	S_5					8	3
I_7	r_3	r_3	r_3	S_9	r_3			
I_8	S_4	S_5	r_1		r_1			7
I_9	r_5	r_5	r_5	r_5	r_5			

Constructing SLR Parsing tables

Example 3:

Construct a SLR or LR(0) parsing table for the following grammar:

$$S \rightarrow cA \mid ccB$$

$$A \rightarrow cA \mid a$$

$$B \rightarrow ccB \mid b$$

Constructing SLR Parsing tables

Solution:

The augmented grammar G' is

$$S' \rightarrow S$$

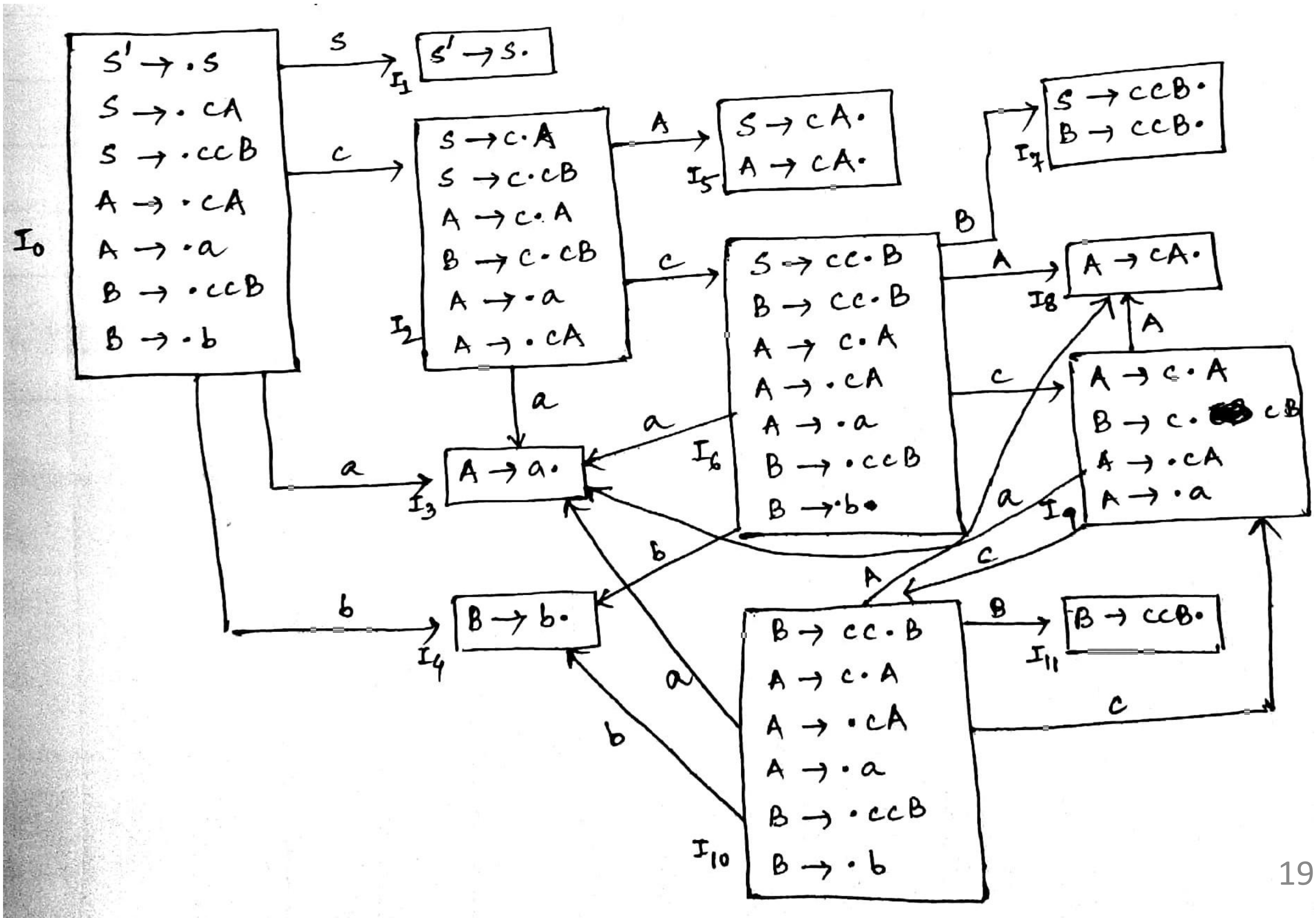
$$S \rightarrow cA \mid ccB$$

$$A \rightarrow cA \mid a$$

$$B \rightarrow ccB \mid b$$

The transition diagram for the given grammar is as follows:

Constructing SLR Parsing tables



Constructing SLR Parsing tables

Now we prepare a FOLLOW set for all non-terminals.

FOLLOW(S) = {$}

FOLLOW(A) = {$}

FOLLOW(B) = {$}

The productions are numbered as

(1) $S \rightarrow cA$

(2) $S \rightarrow ccB$

(3) $A \rightarrow cA$

(4) $A \rightarrow a$

(5) $B \rightarrow ccB$

(6) $B \rightarrow b$

Constructing SLR Parsing tables

So final SLR table is as follows:

ACTION table					GOTO table		
State	**a**	**b**	**c**	**$**	**S**	**A**	**B**
I_0	S_3	S_4	S_2		1		
I_1				**Accept**			
I_2	S_3		S_6			5	
I_3				r_4			
I_4				r_6			
I_5				r_1/r_3			
I_6	S_3	S_4	S_9			8	7
I_7				r_2/r_5			
I_8				r_3			
I_9	S_3		S_{10}			8	
I_{10}	S_3	S_4	S_9			8	11
I_{11}				r_5			

From table, there occurs **reduce-reduce conflict**. So, the given grammar is not LR(0).