Ex: 1) $E \to E+T$

2) $E \to T$

3) $T \to T*F$

4) $T \to F$

5) $F \to (E)$

6) $F \to id$

Augument Grammar

$E' \to E\ \$$

$E \to E+T$

$E \to T$

$T \to T*F$

$T \to F$

$F \to (E)$

$F \to id$

→ Follow(E) → { $, +) }

Follow(T) → { $, +,), * }

Follow(F) → { $, +,), * }

I_0: $E' \to .E$

$E \to .E+T$

$E \to .T$

$T \to .T*F$

$T \to .F$

$F \to .(E)$

$F \to .id$

goto (I_0, E)

I_1: $E' \to E.$

$E \to E.+T$

goto (I_0, T)

I_2: $E \to T.$

$T \to T.*F$

goto (I_0, F)

I_3: $T \to F.$

goto (I_0, ( )

I_4: $F \to (.E)$

$E \to .E+T$

$E \to .T$

$T \to .T*F$

$T \to .F$

$F \to .(E)$

$F \to .id$

goto (I_0, id)

I_5: $F \to id.$

goto (I_1, +)

I_6: $E \to E+.T$

$T \to .T*F$

$T \to .F$

$F \to .(E)$

$F \to .id$

goto (I_2 , *)

I_7 : $T \to T * \cdot F$

$F \to \cdot (E)$

$F \to \cdot id$

goto (I_4 , E)

I_8 : $F \to (E \cdot)$

$E \to E \cdot + T$

goto (I_6, T)

I_9 : $E \to E + T \cdot$

$T \to T \cdot * F$

goto (I_7 , F)

I_{10} : $T \to T * F \cdot$

goto (I_8 ,))

I_{11} $F \to (E) \cdot$

SLR Parsing Table.

state	terminal action						non terminal goto		
	+	*	(	)	id	$	E	T	F
0			S_4		S_5		1	2	3
1	S_6					acc			
2	r_2	S_7		r_2		r_2			
3	r_4	r_4		r_4		r_4			
4			S_4		S_5		8	2	3
5	r_6	r_6		r_6		r_6			
6			S_4		S_5			9	3
7			S_4		S_5				10
8	S_6			S_{11}					
9	r_1	S_7		r_1		r_1			
10	r_3	r_3		r_3		r_3			
11	r_5	r_5		r_5		r_5			

follow (E) = { $, +,) }

follow (T) = { $, +,), *)

follow (F) = { $, +,), *)

to fill the reduce, we need to check the . dots are at the end or not.

SLR Parsing Table

State	action						goto		
	+	*	(	)	id	$	E	T	F
0			S4		S5		1	2	3
1	S6					acc			
2	r2	S7		r2		r2			
3	r4	r4		r4		r4			
4			S4		S5		8	2	3
5	r6	r6		r6		r6			
6			S4		S5			9	3
7			S4		S5				10
8	S6			S11					
9	r1	S7		r1		r1			
10	r3	r3		r3		r3			
11	r5	r5		r5		r5			

STACK	INPUT	ACTION
0	id * id + id $	shift
0 id 5	* id + id $	reduce by F→id
0 F 3	* id + id $	red. by T→F
0 T 2	* id + id $	shift
0 T 2 * 7	id + id $	shift
0 T 2 * 7 id 5	+ id $	red. by F→id
0 T 2 * 7 F 10	+ id $	red. by T→T*F
0 T 2	+ id $	red. by E→T
0 E 1	+ id $	shift
0 E 1 + 6	id $	shift
0 E 1 + 6 id 5	$	red. by F→id
0 E 1 + 6 F 3	$	red. by T→F
0 E 1 + 6 T 9	$	red. by E→E+T
0 E 1	$	Accept