Om Sai Ram

consider the grammar

$E \to E+T \mid T$

$T \to TF \mid F$

$F \to F* \mid a \mid b$

construct the SLR parsing table for this grammar

Also parse the i/p a*b+a.

Any No. of production rules in the grammar

① $E \to E+T$
② $E \to T$
③ $T \to TF$
④ $T \to F$
⑤ $F \to F*$
⑥ $F \to a$
⑦ $F \to b$

Now we will build the canonical set of SLR(0) items.

we will first introduce an augmented grammar.

$E' \to \bullet E$ and then the Initial set of items I_0 will be generated.

I_0 :

$E \to \bullet E+T$ (As after • E appears we will add rule of E)

$E \to \bullet T$ (After • T appears so add rule T)

$T \to \bullet TF$

$T \to \bullet F$ (after • F appears so add rule F)

$F \to \bullet F*$

$F \to \bullet a$

$F \to \bullet b$

Now we will goto function for state I_0 goto on E, T, F and a, b will be applied by step by step.

I_1 : goto(I_0, E)

$E' \to E \bullet$

$E \to E \bullet +T$

I_2 : goto(I_0, T)

$E \to T \bullet$

$T \to T \bullet F$

$F \to \bullet F*$

$F \to \bullet a$

$F \to \bullet b$

I_3 : goto(I_0, F)

$T \to F \bullet *$

$F \to F \bullet *$

I_4 : goto(I_0, a)

$F \to a \bullet$

I_5 goto(I_0, b)

$F \to b \bullet$

Now we will start applying goto transitions on state I_1.

From I_1 state it is possible to apply goto transitions only on +. Hence

I_6 : goto(I_1, +)

$E \to E+ \bullet T$

$T \to \bullet TF$

$T \to \bullet F$

$F \to \bullet F*$

$F \to \bullet a$

$F \to \bullet b$

As after '•' the T comes will add T transitions same is true for F.

The goto transitions will be applied on I_2 state now: we will choose F to apply goto transition because there is no point in applying goto on T.

I_7 : goto(I_2, F)

$T \to TF \bullet$

$T \to F \bullet *$

I_8 : goto(I_3, *)

$F \to F* \bullet$

=> No point apply goto on I_4, I_5 we choose I_6

I_9 : goto(I_6, T)

$E \to E+T \bullet$

$T \to T \bullet F$

$F \to \bullet F*$

$F \to \bullet a$

$F \to \bullet b$

Now we will first obtain FOLLOW of E, T and F. As the FOLLOW computation is required when the SLR parsing table is building.

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

As by rule 2 and 4, E→T and T→F
we can state E=T=F
But E is a start symbol then by rule 2 & 4
T and F can act as start symbol.
we have added $ in FOLLOW(E), FOLLOW(T) and FOLLOW(F)

State	Action					Goto		
	+	*	a	b	$	E	T	F
						1	2	3
0			S_4	S_5				
1	S_6				Accept			
2	r_2		S_4	S_5	r_2			
3	r_4	S_8	r_4	r_4	r_4			
4	r_6	r_6	r_6	r_6	r_6			
5	r_6	r_6	r_6	r_6	r_6			
6			S_4	S_5				
7	r_3	S_8	r_3	r_3	r_3			
8	r_5							
9								

Now we will parse the i/p a*b+a using parsing table

Stack	i/p buffer	Action
$0	a*b+a$	shift
$0a4	*b+a$	reduce by F→a
$0F3	*b+a$	shift
$0F3*8	b+a$	reduce by F→F*
$0F3	b+a$	reduce by T→F
$0T2	b+a$	shift
$0T2b5	+a$	reduce by T→TF
$0T2	+a$	reduce by E→T
$0E1	+a$	shift
$0E1+6	a$	shift
$0E1+6a4	$	reduce by F→a
$0E1+6F3	$	reduce by T→
E1+6T9	$	reduce by E→E+T
$0E1	$	Accept

Thus i/p string is parsed.