SLR PARSER (Bottom-up Parsing)

- SLR Stands for Simple LR Parsers. It is the easiest to implement.
- The parsing can be done as follows.

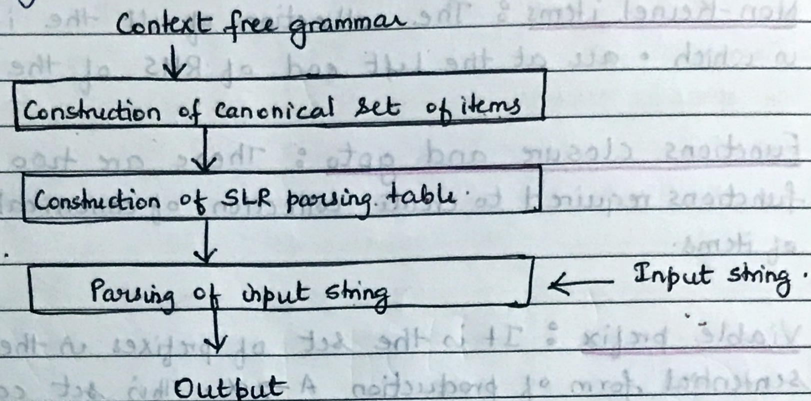


fig:- Working of SLR(1)

- A grammar for which SLR parser can be constructed is called SLR grammar.

Definition of LR(0) items and related terms:-

1) The LR(0) item for grammar G is production rule in which symbol • is inserted at some position in RHS of the rule.

Eg:-
$S \rightarrow \cdot ABC$
$S \rightarrow A \cdot BC$
$S \rightarrow AB \cdot C$
$S \rightarrow ABC \cdot$

The production $S \rightarrow \epsilon$ generates only one item $S \rightarrow \cdot$

2) Augmented grammar: If a grammar G is having start symbol S then augmented grammar is a new grammar G' in which S' is a new start symbol such that $S' \rightarrow S$.

The purpose of this grammar is to indicate the acceptance of input. That is when parser is about to reduce $S' \rightarrow S$ it reaches to acceptance state.

3) Kernel items : It is collection of items $S' \rightarrow \cdot S$ and all the items whose dots are not at the leftmost end of RHS of the rule.

Non-Kernel items : The collection of all the items in which $\cdot$ are at the left end of RHS of the rule.

4) Functions closure and goto : These are two important functions required to create collection of canonical set of items.

5) Viable prefix : It is the set of prefixes in the right sentential form of production $A \rightarrow \alpha$. This set can appear on the stack during shift/reduce action.

Closure operation :

For a CFG G, if I is the set of items then the function closure (I) can be constructed using following rules.

1) Consider I is a set of canonical items and initially every item I is added to closure (I).

2) If rule $A \rightarrow \alpha \cdot B\beta$ is a rule in closure (I) and there is another rule for B such as $B \rightarrow \gamma$ then

closure (I) : $A \rightarrow \alpha \cdot B\beta$

$B \rightarrow \cdot \gamma$

This rule has to be applied until no more new items can be added to closure (I)

goto operation :

- The function goto can be defines as follows :

If there is a production $A \rightarrow \alpha \cdot B\beta$ then goto $(A \rightarrow \alpha \cdot B\beta, B)$ $= A \rightarrow \alpha B \cdot \beta$. That means simply shifting ~~simply~~ of $\cdot$ one position ahead over the grammar symbol (may be terminal or non-terminal). The rule $A \rightarrow \alpha \cdot B\beta$ is in I then the same goto function can be written as goto (I, B)

Example:

Consider the grammar

$E' \to E$ | 1) $E' \to E$ | 1) $E' \to E$ | 6) $F \to (E)$

$E \to E+T \mid T$ | 2) $E \to E+T$ | 2) $E \to E+T$ | 7) $F \to id$.

$T \to T*F \mid F$ | 3) $E \to T$ | 3) $E \to T$

$F \to (E) \mid id$. | 4) $T \to T*F$ | 4) $T \to T*F$

5) $T \to F$

compute closure(I) and goto(I)

Solⁿ:- I is the set of one item $\{E' \to \cdot E\}$ then closure(I) contains

$$\left.\begin{array}{l} E' \to \cdot E \;\rightarrow \text{Kernel items.} \\ E \to \cdot E+T \\ E \to \cdot T \\ T \to \cdot T*F \\ T \to \cdot F \\ F \to \cdot (E) \\ F \to \cdot id. \end{array}\right\} I_0$$

Let us call this as state I_0

Now we apply goto on each grammar symbol in I_0 ie E, T, F, (, id.

GOTO operation

goto (I_0, E) = $E' \to E\cdot$
$E \to E\cdot+T$

goto (I_0, T) = $E \to T\cdot$
$T \to T\cdot*F$

goto (I_0, F) = $T \to F\cdot$

goto $(I_0, ()$ = $F \to (\cdot E)$
$E \to \cdot E+T$
$E \to \cdot T$
$T \to \cdot T*F$
$T \to \cdot F$
$F \to \cdot (E)$
$F \to \cdot id$

goto (I_0, id) = $F \to id\cdot$

Construction of Canonical LR(0) collection of set of item:-

1) For the grammar G initially add $S' \to \cdot S$ in the set of item C.

2) For each set of items I_i in C and for each grammar symbol X (may be terminal or non-terminal) add closure (I_i, X).

This process should be repeated by applying goto (I_i, X) for each X in I_i such that goto (I_i, X) is not empty and not in C. The set of items has to be constructed until no more set of items can be added to C.

Eg:

$$\left.\begin{array}{l} E' \to \cdot E \\ E \to \cdot E + T \\ E \to \cdot T \\ T \to \cdot T * F \\ T \to \cdot F \\ F \to \cdot (E) \\ F \to \cdot id \end{array}\right\} I_0$$

Grammar symbols for I_0 are E, T, F, (, id.

$$goto(I_0, E) = \left.\begin{array}{l} E' \to E \cdot \\ E \to E \cdot + T \end{array}\right\} I_1$$

$$goto(I_0, T) = \left.\begin{array}{l} E \to T \cdot \\ T \to T \cdot * F \end{array}\right\} I_2$$

$$goto(I_0, F) = T \to F \cdot \quad —— \quad I_3$$

$$goto(I_0, () = \left.\begin{array}{l} F \to (\cdot E) \\ E \to \cdot E + T \\ E \to \cdot T \\ T \to \cdot T * F \\ T \to \cdot F \\ F \to \cdot (E) \\ F \to \cdot id \end{array}\right\} I_4$$

goto (I_0, id) = $F \rightarrow id \cdot$ — I_5

Grammar symbols for I_1 is +

goto (I_1, +) = $E \rightarrow E + \cdot T$
$T \rightarrow \cdot T * F$
$T \rightarrow \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot id$
} I_6

Grammar symbols for I_2 are *

goto (I_2, *) = $T \rightarrow T * \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot id$
} I_7

Grammar symbols for I_3 ~~are~~ is ϵ

Grammar symbols for I_4 are E, T, F, (, id

goto (I_4, E) = $F \rightarrow (E \cdot)$
$E \rightarrow E \cdot + T$
} I_8

goto (I_4, T) = $E \rightarrow T \cdot$
$T \rightarrow T \cdot * F$
} I_2

goto (I_4, F) = $T \rightarrow F \cdot$ — I_3

goto (I_4, ( ) = $F \rightarrow (\cdot E)$
$E \rightarrow \cdot E + T$
$E \rightarrow \cdot T$
$T \rightarrow \cdot T * F$
$T \rightarrow \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot id$
} I_4

goto (I_4, id) = $F \rightarrow id \cdot$ — I_5

Grammar symbols for I_5 ~~are~~ is ϵ

Grammar symbols for I_6 are T, F, (, id.

goto (I_6, T) = $\left.\begin{array}{l} E \rightarrow E+T \cdot \\ T \rightarrow T \cdot * F \end{array}\right\}$ I_9

goto (I_6, F) = $T \rightarrow F \cdot$ — I_3

goto (I_6, ( ) = $\left.\begin{array}{l} F \rightarrow (\cdot E) \\ E \rightarrow \cdot E+T \\ E \rightarrow \cdot T \\ T \rightarrow \cdot T * F \\ T \rightarrow \cdot F \\ F \rightarrow \cdot (E) \\ F \rightarrow \cdot id \end{array}\right\}$ I_4

goto (I_6, id) = $F \rightarrow id \cdot$ — I_5

Grammar symbols for I_7 are F, (, id.

goto (I_7, F) = $T \rightarrow T * F \cdot$ — I_{10}

goto (I_7, ( ) = $\left.\begin{array}{l} F \rightarrow (\cdot E) \\ E \rightarrow \cdot E+T \\ E \rightarrow \cdot T \\ T \rightarrow \cdot T * F \\ T \rightarrow \cdot F \\ F \rightarrow \cdot (E) \\ F \rightarrow \cdot id \end{array}\right\}$ I_4

goto (I_7, id) = $F \rightarrow id \cdot$ — I_5

Grammar symbols for I_8 are), +

goto (I_8,)) = $F \rightarrow (E) \cdot \longrightarrow I_{11}$

$$goto(I_8, +) = \left.\begin{array}{l} E \rightarrow E + \cdot T \\ T \rightarrow \cdot T * F \\ T \rightarrow \cdot F \\ F \rightarrow \cdot (E) \\ F \rightarrow \cdot id \end{array}\right\} I_6$$

Grammar symbols for I_9 are $*$

$$goto(I_9, *) = \left.\begin{array}{l} T \rightarrow T * \cdot F \\ F \rightarrow \cdot (E) \\ F \rightarrow \cdot id \end{array}\right\} I_7$$

Grammar symbols for I_{10} is ϵ

Grammar symbols for I_{11} is ϵ.

Construction SLR Parsing Table :-

Input : An augmented grammar G'

Output : SLR Parsing table.

Algorithm :

1) Initially construct set of items $C = \{I_0, I_1, \cdots I_n\}$ where C is a collection of set of LR(0) items for the input grammar G'.

2) The parsing actions are based on each item I_i. The actions are as given below.

   a) If $\boxed{A \to \alpha \cdot a \beta}$ is in I_i and $\boxed{goto(I_i, a) = I_j}$, then set action[i, a] as "shift j". Note that a must be a terminal symbol.

   b) If there is a rule $\boxed{A \to \alpha \cdot}$ is in $\boxed{I_i}$ then set $\boxed{action[i, a] \text{ to "reduce } A \to \alpha\text{"}}$ for all symbols a, where $a \in$ FOLLOW(A). Note that A must not be an augmented grammar S'.

   c) If $\boxed{S' \to S \cdot}$ is in I_i then the entry in the action table $\boxed{action[i, \$] = \text{"accept"}}$

3) The goto part of the SLR table can be filled as: The goto transitions for state i is considered for non-terminals only.

   If $\boxed{goto(I_i, A) = I_j}$ then $\boxed{goto(i, A) = j}$

4) All the entries not defined by rule 2 and 3 are considered to be "error"

FOLLOW(E') = {$}
FOLLOW(E) = {+,), $}
FOLLOW(T) = {+, *,), $}
FOLLOW(F) = {+, *,), $}

SLR Parsing Table

State	Action						GOTO		
	+	*	(	)	id	$	E	T	F
0			s4		s5		1	2	3
1	s6					accept			
2	r3	s7		r3		r3			
3	r5	r5		r5		r5			
4			s4		s5		8	2	3
5	r7	r7		r7		r7			
6			s4		s5			9	3
7			s4		s5				10
8	s6			s11					
9	r2	s7		r2		r2			
10	r4	r4		r4		r4			
11	r6	r6		r6		r6			

$\boxed{I_0}$: $A \to \alpha \cdot a \beta \to I_i$, goto $(I_i, a) = I_j \Rightarrow$

set action $[i, a] = S_j$

✓ $\boxed{F \to \cdot (E) \to I_0}$, goto $(I_0, () = I_4$

$\alpha \cdot a \beta$

set action $[0, (] = \boxed{S4}$

✓ $\boxed{F \to \cdot id \to I_0}$, goto $(I_0, id) = I_5$

$A \to \alpha \cdot a \beta$

set action $[0, id] = \boxed{S5}$

goto $(I_0, E) \Rightarrow I_1$ set ~~ad~~ goto $[0, E] = 1$.
goto $(I_0, T) \Rightarrow I_2$ " " $[0, T] = 2$
goto $(I_0, F) \Rightarrow I_3$ " " $[0, F] = 3$

I_1 : $E' \to E\cdot \Rightarrow I_1$

action [1, $] = accept

$E \to E\cdot + T \Rightarrow I_1$

$A \to \alpha \cdot a \beta$

goto $(I1, +) = I_6 \Rightarrow$ action $(1, +) = S6$.

I_2 : $E \to T\cdot \Rightarrow I_2$

$A \to \alpha \cdot$

set action $[2, a]$ = reduce $E \to T$ ie $r3$ for all a in follow(E) ie follow(E) = {+,), $}.

set action [2, +] = r3

" [2,)] = r3

" [2, $] = r3.

$T \to T \cdot * F \Rightarrow I_2$.

$A \to \alpha \cdot a \beta \Rightarrow I_2$.

goto $(I_2, *) = I_7$

action [2, *] = S7

I_3 : $T \to F \cdot \Rightarrow I_3$.

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

action [3, +] = r5

I_4 :- $F \to \cdot (E)$

$A \to \alpha \cdot a \beta$

goto (4, () = I_4.

action [4, (] = S4.

$F \to \cdot id$.

goto (4, id) = I_5

action [4, id] = S5

I_5 :- $F \to id \cdot$

$A \to \alpha \cdot$

foll(F) = +, *,), $

action [5, +] = r7

I6 : $F \rightarrow \cdot(E)$ goto(6, ( ) = I_4.
$A \rightarrow \alpha \cdot a \beta$

action [6, ( ] = S4

$F \rightarrow \cdot id$ goto(6, id) = I_5.

action [6, id] = S5

I7 :- $F \rightarrow \cdot(E)$ goto(7, ( ) = I_4.
$A \rightarrow \alpha \cdot a \beta$

action [7, ( ] = S4.

$F \rightarrow \cdot id$.   goto(7, id] = I_5.

action [7, id] = S5

I8 :- $F \rightarrow (E \cdot)$ goto(8,)) = I_{11}
$A \rightarrow \alpha \cdot a \beta$

action [8,)] = S11

$E \rightarrow E \cdot + T$ goto(8, +) = I_6
$A \rightarrow \alpha \cdot a \beta$

action [8, +] = S6.

I9 : $T \rightarrow T \cdot * F$
$A \rightarrow \alpha \cdot a \beta$ goto(9, *) = I_7

action [9, *] = S7

$E \rightarrow E + T \cdot$
$A \rightarrow \alpha \cdot$ {+,), $}

action [9, +] = r2

I10 :- $T \rightarrow T * F \cdot$
$A \rightarrow \alpha \cdot$ +, *,), $

action [10, +] = r4

I11 $F \rightarrow (E) \cdot$ +, *,), $.
$A \rightarrow \alpha \cdot$
action [11, +] = r6

$G' - S'$ $\quad L \to *R \mid id$

*

eg Consider the grammar

$S \to L = R$

$S \to R$

$L \to *R$

$L \to id$

$R \to L$

$S \to L = R \mid R$

$G' \quad S'$

Construct SLR Parsing table.

Soln Given grammar is

1) $S \to L = R$
2) $S \to R$
3) $L \to *R$
4) $L \to id$
5) $R \to L$

- Now compute FIRST and FOLLOW

First(S) = {*, id} $\quad$ Follow(S) = { \$ }

First(L) = {*, id} $\quad$ Follow(L) = { =, \$ }

First(R) = {*, id} $\quad$ Follow(R) = { =, \$ }

- Augmented Grammar is

$S' \to S$

- Item of Augmented grammar. LR(0) item of a grammar G is a production of G with a dot at some position of the right side.

Closure :-

$S' \to \cdot S$

$S \to \cdot L = R$

$S \to \cdot R$

$L \to \cdot *R$

$L \to \cdot id$

$R \to \cdot L$

I_0

- ~~[illegible]~~ Grammar symbols for I_0 are S, L, R, * id

goto $(I_0, S) = S' \to S\cdot$ — I_1

$goto(I_0, L) = \left.\begin{array}{l} S \rightarrow L \cdot = R \\ R \rightarrow L \cdot \end{array}\right\} I_2$

$goto(I_0, R) = S \rightarrow R \cdot$ — I_3

$goto(I_0, *) = \left.\begin{array}{l} L \rightarrow * \cdot R \\ R \rightarrow \cdot L \\ L \rightarrow \cdot * R \\ L \rightarrow \cdot id \end{array}\right\} I_4$

$goto(I_0, id) = L \rightarrow id \cdot$ — I_5

Grammar symbols for I_1 is ϵ

Grammar symbols for I_2 is =

$goto(I_2, =) = \left.\begin{array}{l} S \rightarrow L = \cdot R \\ R \rightarrow \cdot L \\ L \rightarrow \cdot * R \\ L \rightarrow \cdot id \end{array}\right\} I_6$

GS for I_3 is ϵ

GS for I_4 is R, L, *, id

$goto(I_4, R) = L \rightarrow * R \cdot$ — I_7

$goto(I_4, L) = R \rightarrow L \cdot$ — I_8

$goto(I_4, *) = \left.\begin{array}{l} L \rightarrow * \cdot R \\ R \rightarrow \cdot L \\ L \rightarrow \cdot * R \\ L \rightarrow \cdot id \end{array}\right\} I_4$

$goto(I_4, id) = L \rightarrow id \cdot$ — I_5

GS for I_5 is ϵ

Grammar symbols for I_6 is R, L, *, id.

goto (I_6, R) = $S \to L = R\cdot$ — I_9

goto (I_6, L) = $R \to L\cdot$ — I_8

goto (I_6, *) = $L \to *\cdot R$
$R \to \cdot L$
$L \to \cdot * R$
$L \to \cdot id$
} I_4

goto (I_6, id) = $L \to id\cdot$ — I_5

GS for I_7 is ϵ

GS for I_8 is ϵ

GS for I_9 is ϵ

SLR Parsing Table

States	Action				GOTO		
	=	*	id	$	S	L	R
0		S4	S5		1	2	3
1				accept			
2	S6/r5			r5			
3				r2			
4		S4	S5			8	7
5	r4			r4			
6		S4	S5			8	9
7	r3			r3			
8	r5			r5			
9				r1			

I_0: $L \to \cdot \overset{a}{*} R$
$A \to \alpha \cdot a \beta$

goto(0, *) = I_4

$\boxed{A \to \alpha \cdot a \beta}$

action [0, *] = S4.

$L \to \cdot id$
$A \to \alpha \cdot a \beta$

goto(0, id) = I_5
action [0, id] = S5

$A \to \alpha \cdot a\beta$ - I_i
goto $(I_i, a) = I_j$
set action $[i, a]$ = shift j

$A \to \alpha \cdot$ - I_i # Augment gr. S'
action $[i, a]$ = Reduce $\vec{A} \to \alpha$ for all $a \in$ FOLLOW(A)

If $S' \to S \cdot$ - I_i
action $[i, \$]$ = "accept"

(I_1) $S' \to S \cdot$

action $[1, \$]$ = accept

(I_2) $S \to L \cdot = R$

$A \to \alpha \cdot a\beta$ goto$(2, =) = I6$.

action $[2, =] = S6$.

$\boxed{A \to \alpha \cdot}$

(I_2) $R \to L \cdot$

$A \to \alpha \cdot$

action $[2, =] = r5$ follow$(R) = \{=, \$\}$

action $[2, \$] = r5$

(I_3) $S \to R \cdot$

$A \to \alpha \cdot$ follow$(S) = \{\$\}$.

action $[3, \$] = r2$

(I_4) $L \to \cdot * R$ goto$(4, *) = I_4$.

$A \to \alpha \cdot a\beta$

$S' \to S \cdot \to I_1$

action$[4, *] = S4$.

$1, \$$ = ac

$L \to \cdot id$ goto$(4, id) = I_5$.

$A \to \alpha \cdot a\beta$ action $[4, id] = S5$

(I_5) $L \to id \cdot$

$A \to \alpha \cdot$ follow$(L) = \{=, \$\}$

action $[5, =] = r4$

action $[5, \$] = r4$

(I_6) $L \to \cdot * R$. goto$(6, *) = I_4$.

$A \to \alpha \cdot a\beta$

action $[6, *] = S4$.

$L \to \cdot id$. goto$(6, id) = I_5$

$A \to \alpha \cdot a\beta$

action $[6, id] = S5$

(I_7) $L \to * R \cdot$

$A \to \alpha \cdot$ follow$(L) = \{=, \$\}$.

action $[7, =] = r3$

action $[7, \$] = r3$

I_8 :- $R \to L \cdot$

$A \to \alpha \cdot$ follow(R) = {=, $}.

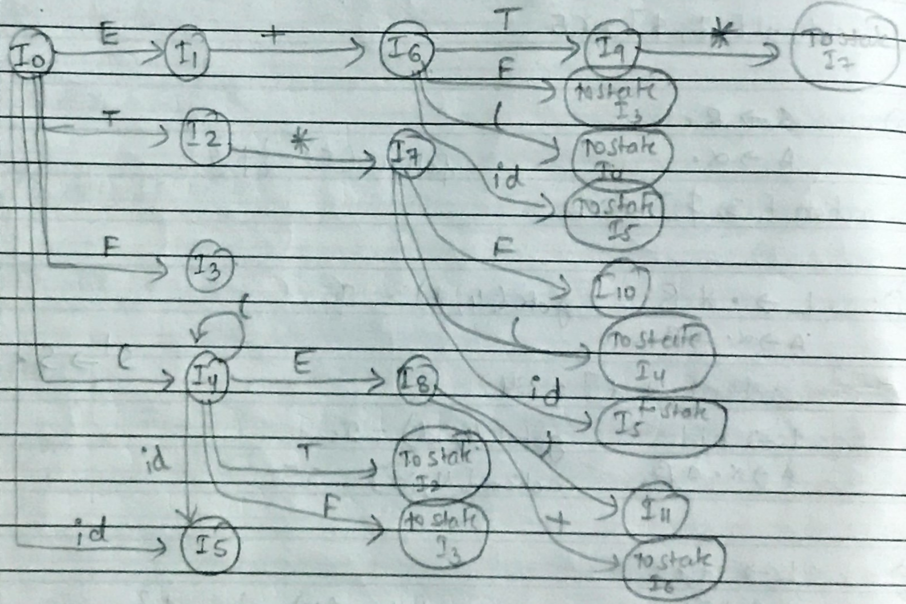
action [8, =] = r5

" [8, $] = r5

I_9 $S \to L = R \cdot$ follow(S) = {$}

$A \to \alpha \cdot$

action [9, $] = r1



DFA for set of items

Eg. Consider the following grammar : (10mks)

$P \rightarrow D ; D$

$D \rightarrow id : T$

$T \rightarrow int | char$

i) obtain collection of LR(0) itemsets

ii) Construct SLR Parsing table.

Solution

$\cancel{A}$ $\cancel{\epsilon P}$.