classmate
Date
Page

SLR Parser

I To determine the set of states of stack, construct LR(0) set of items :

LR(0) set of items performs 2 functions.

1) Closure of a set:
For each production $A \to \alpha \cdot B\beta$ where the dot $(\cdot)$ is followed by a variable 'B', add productions of B in the item set.
ie for each production $B \to \gamma$ add $B \to \cdot\gamma$ in the set.

* No productions are added if dot $(\cdot)$ is followed by a terminal & is at the end of production such as $A \to \alpha\cdot$

2) Move from a set:
For each production $A \to \alpha \cdot a\beta$, perform a move ~~on~~ ~~b~~ on the terminal 'a' to obtain $A \to \alpha a \cdot \beta$ after move (transition)

For each production $A \to \alpha \cdot B\beta$, perform a move on the variable 'B' to obtain $A \to \alpha B \cdot \beta$ after move (goto ~~transition~~ transition)

Consider a grammar : $E \rightarrow E+T / T$
$T \rightarrow T*id / id$

For LR(0) construction; the set of productions are.

0) $S \rightarrow E$ (newly added prod).
1) $E \rightarrow E+T$
2) $E \rightarrow T$
3) $T \rightarrow T*id$
4) $T \rightarrow id$

The first set I_0 (for all grammars) begins with the newly added production which has a new start symbol ~~(S)~~ that derives the old start symbol.
(In above example: $S \rightarrow E$.
Generally : $S' \rightarrow S$)
old start symbol

Hence, for above grammar, the first production in I_0 is $S \rightarrow \cdot E$.
Since dot $(\cdot)$ is followed by E, therefore add productions of E in I_0 as
$E \rightarrow \cdot E+T$ and $E \rightarrow \cdot T$.

After applying closure on I_0 : $S \rightarrow \cdot E$
we get -

I_0 : $S \rightarrow \cdot E$
$E \rightarrow \cdot E+T$
$E \rightarrow \cdot T$
$T \rightarrow \cdot T*id$
$T \rightarrow \cdot id$

Scanned by CamScanner

For move from I_0, consider the variables and terminals that follow the dot (.)

$S \to \cdot E$ move on E to become $S \to E\cdot$

$E \to \cdot E+T$ moves on E to become $E \to E\cdot +T$

$E \to \cdot T$ moves on T to become $E \to T\cdot$

$T \to \cdot T * id$ moves on T to become $T \to T\cdot * id$

$T \to \cdot id$ moves on id to become $T \to id\cdot$

move on same variable/terminal are combined together to obtain the next item set. Hence we get

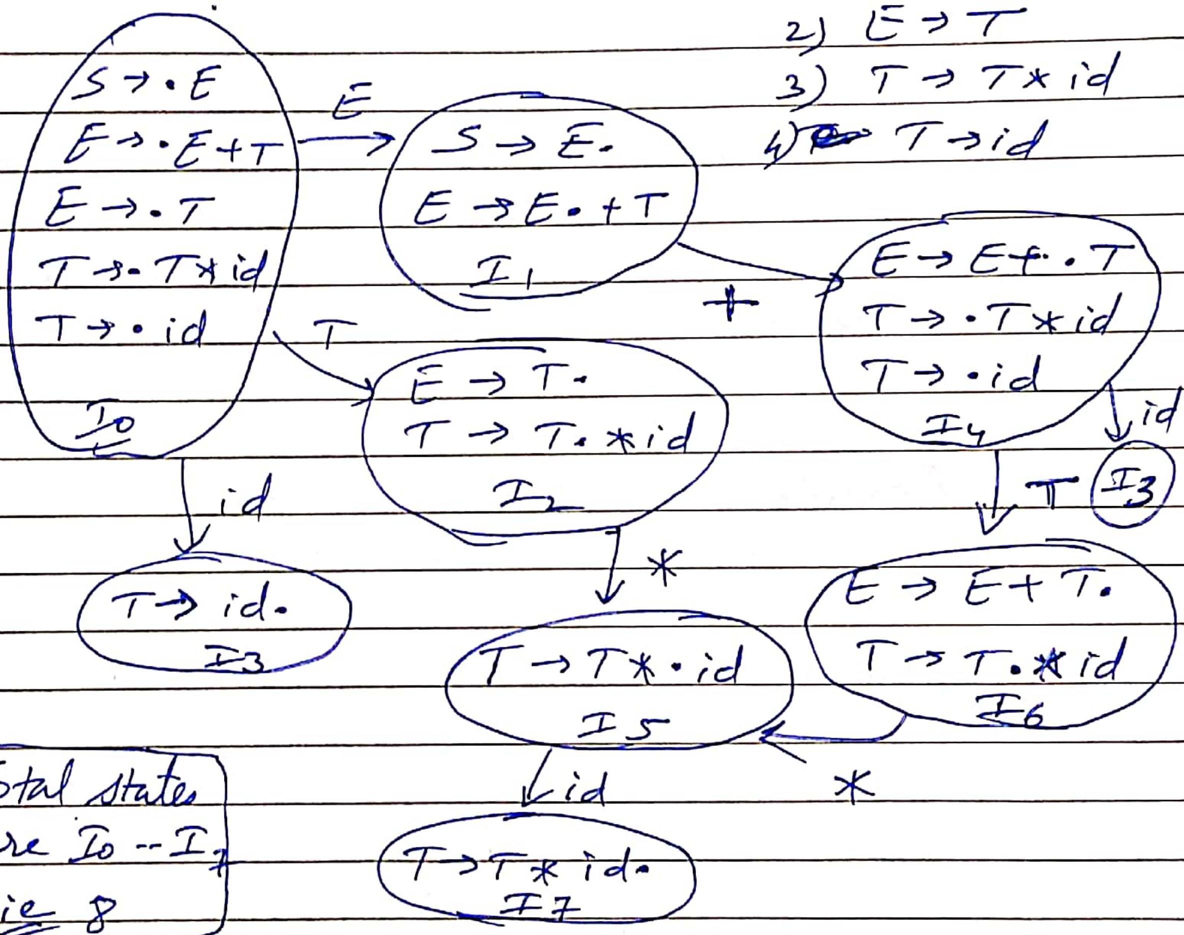
$$\begin{tikzcd}
\begin{array}{l} S \to \cdot E \\ E \to \cdot E+T \\ E \to \cdot T \\ T \to \cdot T*id \\ T \to \cdot id \\ I_0 \end{array} \arrow[r, "E"] \arrow[dr, "T"] \arrow[ddr, "id"] & \left\{ \begin{array}{l} S \to E\cdot \\ E \to E\cdot +T \end{array} \right\} I_1 \\
& \left\{ \begin{array}{l} E \to T\cdot \\ T \to T\cdot * id \end{array} \right\} I_2 \\
& \left\{ T \to id\cdot \right\} I_3 .
\end{tikzcd}$$

Perform closure and move on I_1, I_2, I_3, and so on -- (on every new set) till no new set is obtained).

If the entire ~~set~~ new set (with productions and dot(·) positions) ~~to~~ is identical to an already existing set then it combins with the old set.

LR(0) set of items dev :

0) $S \to E$
1) $E \to E+T$
2) $E \to T$
3) $T \to T*id$
4) $T \to id$



Total states are I0 -- I7 ie 8

Parser Table

State	action (terminals)				goto (Variables)	
	id	+	*	$	E	T
I0						
I1						
I2						
⋮						
I7						

II Obtaining ~~action~~ shift entries (for action part) and goto entries from transitions on terminals and Variables.

A) For each move ~~on~~ of $A \to \alpha \cdot a\beta$ ~~from~~ of set I_x to $A \to \alpha a \cdot \beta$ of set I_y, add entry "shift on a" in I_x state for terminal a along with the next state as I_y.
ie "shift on a and go to set I_y"
~~represe~~ Represented as S_y.

Example: $E \to E \cdot + T$ of I_1
on '+' moves to
$E \to E + \cdot T$ of I_4

Hence

State	action
	a
I_1	S4

S4 ~~mean~~ means shift and goto 4.

B) For each move of ~~$A \to \alpha \cdot B$~~
$A \to \alpha \cdot B\beta$ of set I_x to
$A \to \alpha B \cdot \beta$ of set I_y,
add goto I_y in row of I_x and column of Variable B.

Example: $S \rightarrow \cdot E$ of I_0 on E moves to $S \rightarrow E \cdot$ of I_1

Hence

State	goto B
I_0	1

$\rightarrow$ goto 1 on B

For above example grammar we get.

State	action (terminal)				goto (Variable)	
	id	+	*	\$	E	T
I_0	s3				1	2
I_1		s4				
I_2			s5			
I_3						
I_4	s3					6
I_5	s7					
I_6			s5			
I_7						

III Obtaining reduce entries (for action part).

For each production, $A \rightarrow \alpha \cdot$ of set I_n, add entry 'r_n' on FOLLOW(A)
"r_n" represents "reduce by production n" where n represents production number.

~~Hence~~ Example: $T \rightarrow id \cdot$ of I_3 will get r_4 on FOLLOW(T)
"r_4": because production number of $T \rightarrow id$ is 4.

Since $\text{Follow}(T) = \{*, +, \$\}$

Hence

state	*	+	$
I_3	r4	r4	r4

Adding the following reduce entries in previous table

	Add $\{+, \$\}$
$I_2 : E \to T.$	r2 on Follow(E)
$I_3 : T \to id.$	r4 on Follow(T)
$I_6 : E \to E+T$	r1 on Follow(E)
$I_7 : T \to T * id.$	r3 on Follow(T)
For $I_1 : S \to E.$	Accept (Always).

Hence we get the SLR parser table as

State	action (terminal)				goto (Variable)	
	id	+	*	$	E	T
I_0	s3				1	2
I_1		s4		Accept		
I_2		r2	s5	r2		
I_3		r4	r4	r4		
I_4	s3					6
I_5	s7					
I_6		r1	s5	r1		
I_7		r3	r3	r3		

TASK : Try to parse the i/p string
" id + id * id + id "