Page No.
Date : / /20

Compilers Assignment - 2

Ans 1.

$S \rightarrow SS* \mid SS+ \mid a$

a.) Language generated by this grammar is the set of all postfix expression consisting of addition and multiplication over variable, a.

b.) The grammar is certainly unambiguous. This can be justified by the fact that it generates postfix expression which doesn't need any brackets and generates only one binary tree.

c.) Augmented grammar looks like

$S' \rightarrow S$

$S \rightarrow SS*$

$S \rightarrow SS+$

$S \rightarrow a$

Now, let us find the different states.

I_0 : $S' \rightarrow .S$

$S \rightarrow .SS*$

$S \rightarrow .SS+$

$S \rightarrow .a$

I_1 : goto (I_0, S)

$S' \rightarrow S.$

$S \rightarrow S.S*$

$S \rightarrow S.S+$

$S \rightarrow a.$

$S \rightarrow .SS*$

$S \rightarrow .SS+$

$S \rightarrow .a$

* I_2: goto (I_0, a)

$S \rightarrow a.$

* I_3: goto (I_1, S)

$S \rightarrow SS. *$ $S \rightarrow .SS*$

$S \rightarrow SS. +$ $S \rightarrow .SS+$

$S \rightarrow S.S *$ $S \rightarrow .a$

$S \rightarrow S.S +$

* goto $(I_1, a) = I_2$

* goto $(I_3, S) = I_3$

* goto $(I_3, a) = I_2$

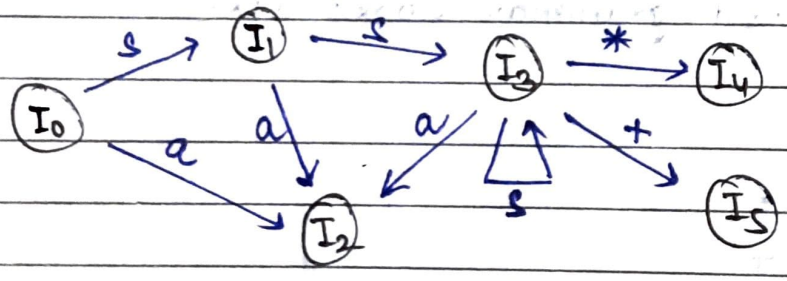
* goto $(I_3, *) = I_4$

$S \rightarrow SS*.$

* goto $(I_3, +) = I_5$

$S \rightarrow SS+.$

Graphically, it looks like



Numbering of rules.

1. $S \rightarrow SS*$ — R1
2. $S \rightarrow SS+$ — R2
3. $S \rightarrow a$ — R3

Date : / /20

Parse table

State	Action *	Action +	Action a	Action $	Goto S
0			s2		1
1			s2	acc.	3
2	r3	r3	r3	r3	
3	s4	s5	s2		3
4	r2	r2	r2	r2	
5.	r1	r1	r1	r1	

rx → reduce using rule x.
sj → shift with goto state j
acc → accept
err → Error (empty cells in table).

Page No.
Date : / /20

Answer 2. Grammar :- Augment it by $S' \rightarrow S$

$S \rightarrow Ma$
$S \rightarrow bMc$
$S \rightarrow dc$
$S \rightarrow bMa$
$M \rightarrow d$
$M \rightarrow \varepsilon$

Now calculate / find out the states.

* I_0

$S' \rightarrow .S$
$S \rightarrow .Ma$
$S \rightarrow .dc$
$S \rightarrow .bMa$
$M \rightarrow .d$
$M \rightarrow .$
$S \rightarrow .bMc$

I_1 : goto (I_0, S)
$S' \rightarrow S.$

I_2 : goto (I_0, M)
$S \rightarrow M.a$

I_3 : goto (I_0, b)
$S \rightarrow b.Ma$
$S \rightarrow b.Mc$
$M \rightarrow .d$
$M \rightarrow .$

I_4 : goto (I_0, d)
$S \rightarrow d.c$
$M \rightarrow d.$

I_5 : goto (I_2, a)
$S \rightarrow Ma.$

I_6 : goto (I_3, M)
$S \rightarrow bM.a$
$S \rightarrow bM.c$

I_7 : goto (I_3, d)
$M \rightarrow d.$

I_8 : goto (I_4, c)
$S \rightarrow dc.$

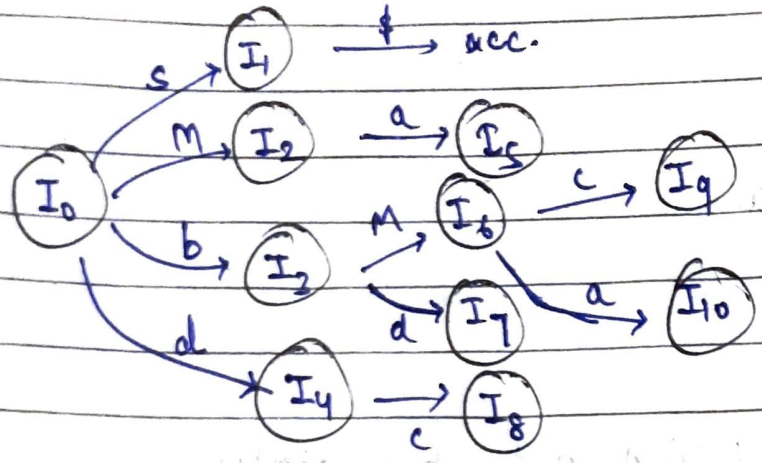
I_9 : goto (I_6, c)
$S \rightarrow bMc.$

I_{10} : goto (I_6, a)
$S \rightarrow bMa.$

Page No.

Date : / /20

Graphically



Rules No:

0	$S' \to S$
1	$S \to Ma$
2	$S \to bMc$
3	$S \to dc$
4	$S \to bMa$
5	$M \to d$
6	$M \to \epsilon$

Parse table.

State	Action					Goto	
	a	b	c	d	$	S	M
0	r6	s3	r6	s4		1	2
1					acc.		
2	s5						
3	r6		r6	s7			6
4	r5		s8, r5				
5					r1		
6	s10		s9				
7	r5		r5				
8					r3		
9					r2		
10					r4		

The conflict occurs in state 4 and type of conflict is shift-reduce conflict.

a.) Since, conflict occur in S4, it look like
$S \to d \cdot c$
$M \to d \cdot$
Now, shift reduce conflict occur due to these LR(0) items.
($S \to d \cdot c$ and $M \to d \cdot$)
↘
req. pair of item

(b.) as string 'dc' $\in L(G)$, consider w_1 = 'dc'. Now, we will show reduction of w_1 until conflict occurs.

Step	Stack	Input	Action
1	0	dc $	shift 4
2.	0 d 4	c $	s8, r5.

↘
conflict.

Hence, shows that for w_1 = 'dc', conflict occurs at step 2.

c.) Take w_2 = 'bc'

Step	Stack	Input	Action
1	0	bc $	s3
2	0 b 3	c $	r6
3	0 b 3 M 6	c $	s9
4	0 b 3 M 6 c 9	$	r2
5	0 S 1	$	accept

Thus w_2 = 'bc', rule 6 ($M \to \epsilon$) is used for the reduction.

Ans 3.

$S \rightarrow id\,[E] := E$

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid id$.

* I_0 :

$S' \rightarrow .S$

$S \rightarrow .id\,[E] := E$

* I_1 : goto (I_0, S)

$S' \rightarrow S.$

* I_2 : goto (I_0, id)

$S \rightarrow id.\,[E] := E$

* I_3 : goto $(I_2, [)$

$S \rightarrow id\,[.E] := E$

$E \rightarrow .E+T$

$E \rightarrow .T$

$T \rightarrow .T*F$

$T \rightarrow .F$

$F \rightarrow .(E)$

$F \rightarrow .id$

* I_4 : goto (I_3, E)

$S \rightarrow id\,[E.] := E$

$E \rightarrow E. + T$

* I_5 : goto (I_3, T)

$E \rightarrow T.$

$T \rightarrow T. * F$

* I_6 : goto (I_3, F)

$T \rightarrow F.$

* I_7 : goto $(I_3, ()$

$F \rightarrow (.E)$

$E \rightarrow .E+T$

$E \rightarrow .T$

$T \rightarrow .T*F$

$T \rightarrow .F$

$F \rightarrow .(E)$

$F \rightarrow .id$.

* goto $(I_3, id) = I_8$

$F \rightarrow id.$

* goto $(I_4,]) = I_9$

$S \rightarrow id\,[E]. := E$

* I_{10} : goto $(I_4, +)$

$E \rightarrow E + .T$ $T \rightarrow .F$

$T \rightarrow .T * F$ $F \rightarrow .(E)$

$F \rightarrow .id$.

Page No.
Date : / /20

a.) We can see that state 7 contains self loop. ∵ goto $(I_7, c) = I_7$.

Now all viable prefixes can be rep. by

Consider the following:

$$A = \left(id[\mid id[E] := \right)$$ (| : or)

$$B = \left((\mid E+(\mid T^*(\mid E+T^*(\right)$$ (| : or, or, or, or)

Final answer can be written as

regex (final-s = I_7) ⇒ $\boxed{ABB^*}$

(b) Now, we know that CLR parser contain max. no. of states. For this grammar, CLR consist of 37 states.

Consider this string w_1 = id [id * (id * id] := id

Clearly this doesn't belong to grammar.
Now, CLR reject this string in 13 steps while SLR reject this in 17 steps.

This is due to the fact that CLR is more powerful considering the fact of one lookup ahead which helps it in detecting the error earlier.

This can be more understood by reducing the strings in resp. parser. First 12 steps are same in both, however, after CLR directly rejects on basis of lookahead while SLR keeps on reducing the stack elements.

Page No.
Date : / /20

c) For the above example, both LALR parse table and SLR parse table look exactly same, the only difference being the definition of states.

Now since to reduce a string through a parse the reduction will be exactly same for a given string via SLR and LALR, as def^n of states doesn't matter only corr. shift and reduce commands matter.

Thus we can consider the same string as part (a.) ie $w = id [id * (id * id] := id$

CLR takes 13 steps to give error while LALR takes 17 steps.