CS6109 - COMPILER DESIGN

NAME: ASHWIN MUTHURAMAN A

REGISTER NUMBER: 2020103005

SUBJECT CODE: CS6109

SUBJECT TITLE: COMPILER DESIGN

ASSIGNMENT-I

1) Convert the following regular expressions to deterministic finite automata.

   (i) $(a|b)^*$

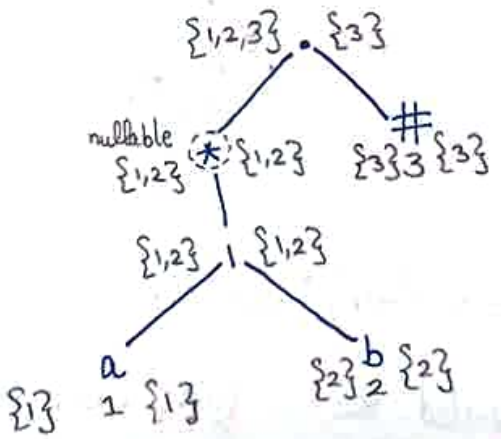
   (ii) $(a|b)^*abb(a|b)^*$

   (iii) $(a|b)^*a(a|b)$

Ans: (i) Given RE: $(a|b)^*$

⇒ Augmented RE: $(a|b)^*.\#$

⇒ Syntax Tree for augmented RE:

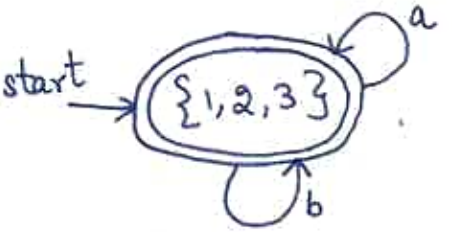


The firstpos and lastpos for each node is represented to the left and right of each node in syntax tree respectively.

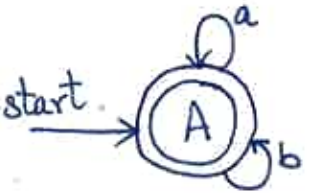
⇒ Now, let us calculate/compute followpos table:

Symbol	Node number	followpos
a	1	1, 2, 3
b	2	1, 2, 3
#	3	-

⇒ Construction of DFA:



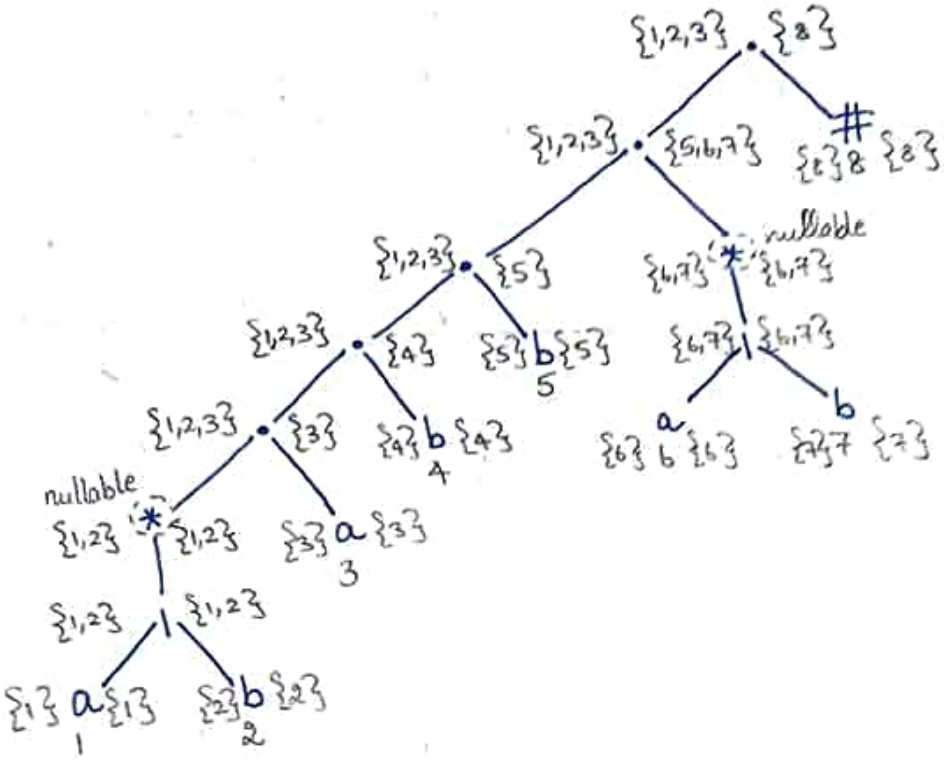
⇒ Renaming the states, we get the DFA as,



(ii) Given RE: $(a|b)^*abb(a|b)^*$

⇒ Augmented RE: $(a|b)^*.a.b.b.(a|b)^*.\#$

⇒ Syntax Tree for augmented RE:

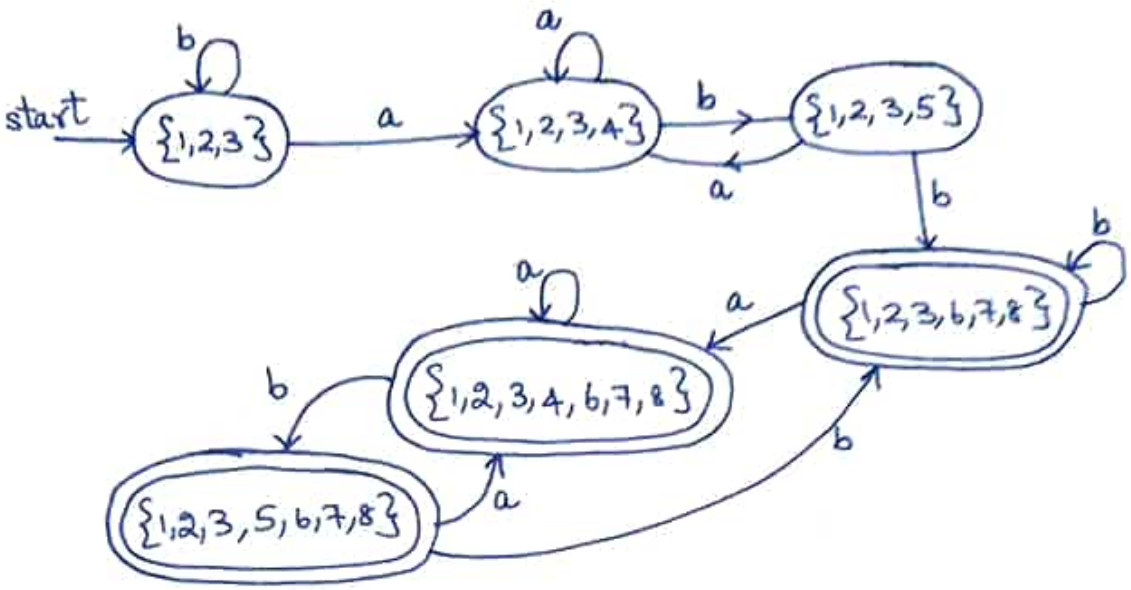


The firstpos and lastpos for each node is represented to the left and right of each node in syntax tree respectively.

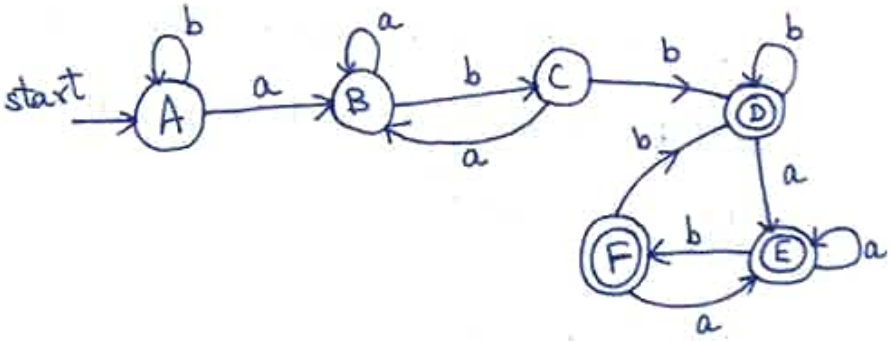
⇒ Computing followpos table:

Symbol	Node number	followpos
a	1	1, 2, 3
b	2	1, 2, 3
a	3	4
b	4	5
b	5	6, 7, 8
a	6	6, 7, 8
b	7	6, 7, 8
#	8	–

⇒ Construction of DFA:



⇒ Renaming the states, we get the DFA as:

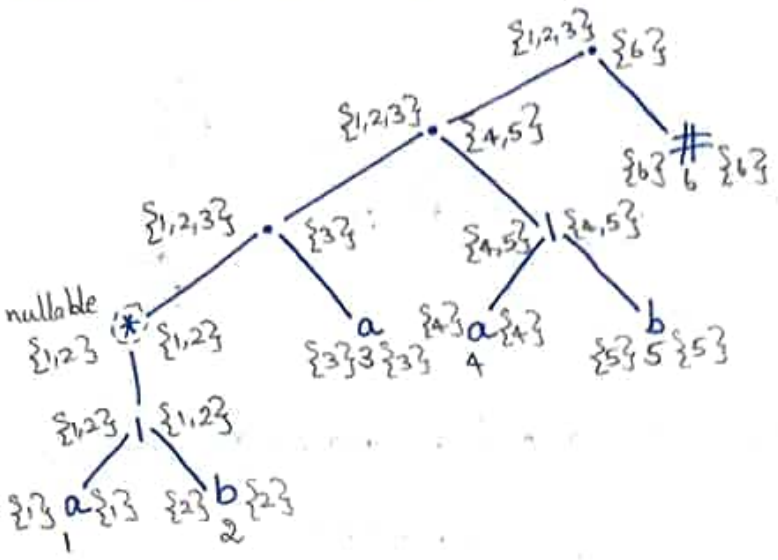


(iii) Given RE: $(a|b)^*a(a|b)$

⇒ Augmented RE: $(a|b)^*.a.(a|b).\#$

The firstpos and lastpos for each node is represented to the left and right of each node in syntax tree respectively.

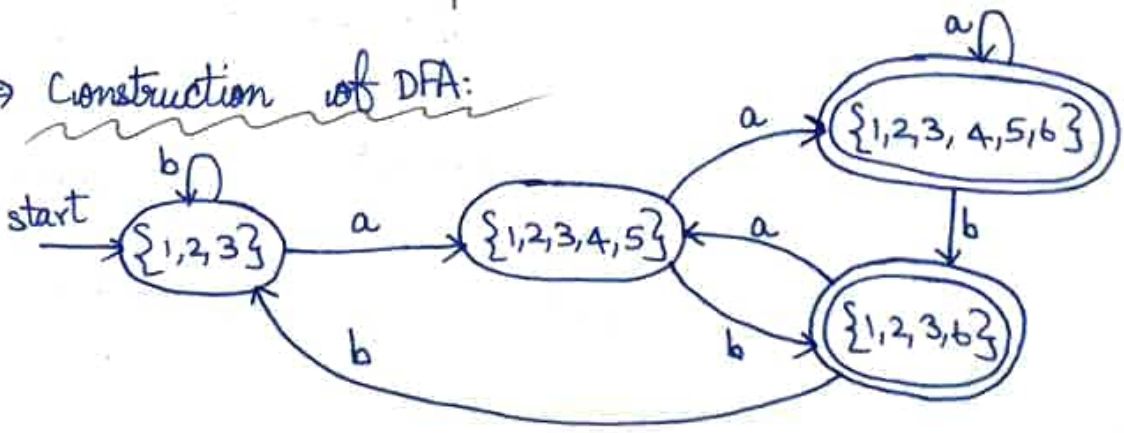
⇒ Syntax Tree for augmented RE:



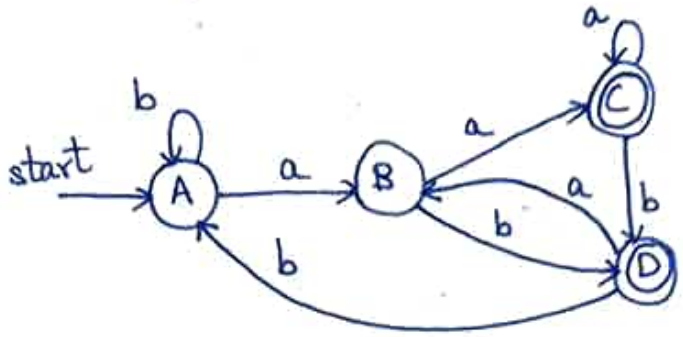
⇒ Computing followpos table:

Symbol	Node number	followpos
a	1	1,2,3
b	2	1,2,3
a	3	4,5
a	4	6
b	5	6
#	6	–

⇒ Construction of DFA:



⇒ Renaming the states, we get the DFA as:



2) Show that the following grammar is LR(1) and not LALR(1).

$S \to Aa \mid bAc \mid Bc \mid bBa$

$A \to d$

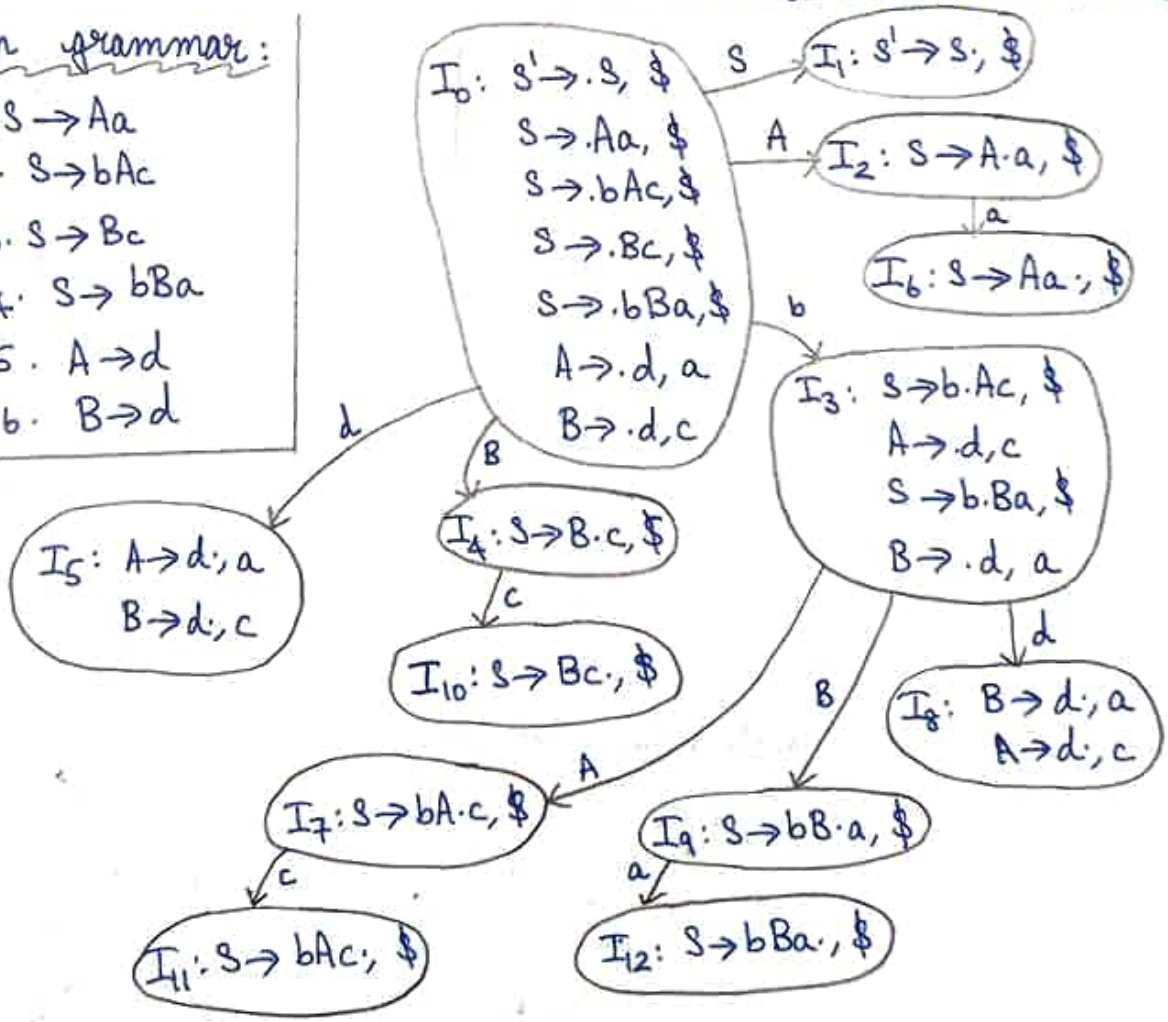
$B \to d$

Ans:

Given grammar:

1. $S \to Aa$
2. $S \to bAc$
3. $S \to Bc$
4. $S \to bBa$
5. $A \to d$
6. $B \to d$

The canonical LR(1) collection is:



LR(1) Parsing Table:

State	Action					Goto		
	a	b	c	d	$	S	A	B
0		S_3		S_5		1	2	4
1					accept			
2	S_6							
3				S_8			7	9
4			S_{10}					
5	r_5		r_6					
6					r_1			
7			S_{11}					
8	r_6		r_5					
9	S_{12}							
10					r_3			
11					r_2			
12					r_4			

Since, there is no conflict in LR(1) Parsing Table, the given grammar is in LR(1).

For checking LALR(1), we need to combine similar cores. Here, we observe that item sets 5 and 8 are similar and hence we combine those states and produce the LALR(1) Parsing Table as follows:

LALR(1) Parsing Table:

State	Action					Goto		
	a	b	c	d	$	S	A	B
0		S_3		$S_{5,8}$		1	2	4
1					accept			
2	S_6							
3				$S_{5,8}$			7	9
4			S_{10}					
5,8	r_5, r_6		r_6, r_5					
6					r_1			
7			S_{11}					
9	S_{12}							
10					r_3			
11					r_2			
12					r_4			

As we can see, there is a reduce-reduce conflict in Action[58, a] and in Action[58, c]. Therefore, the given grammar is not in LALR(1).

∴ Hence shown that the given grammar is LR(1) and not LALR(1).

3) Convert the following Regular Expression to Deterministic Finite Automata:

(i) $0^*10^*|11^*$

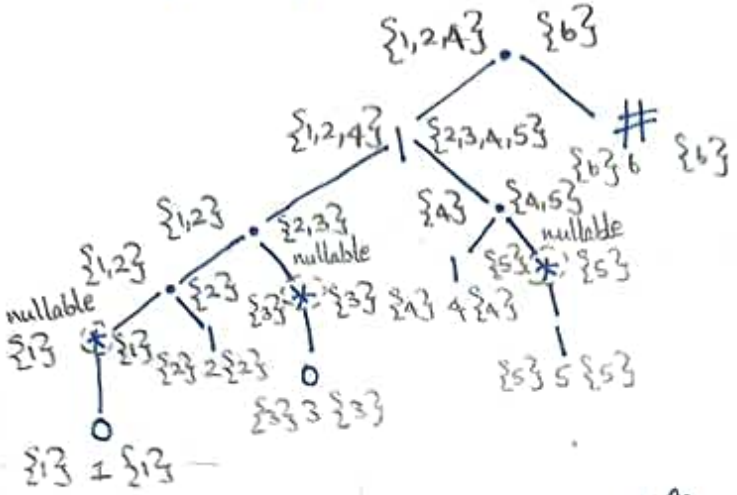
(ii) $((\epsilon|a)b^*)^*$

Ans: (i) Given RE: $((0^*10^*)|(11^*))$

⇒ Augmented RE: $((0^*.1.0^*)|(1.1^*)).\#$

⇒ Syntax Tree for Augmented RE:

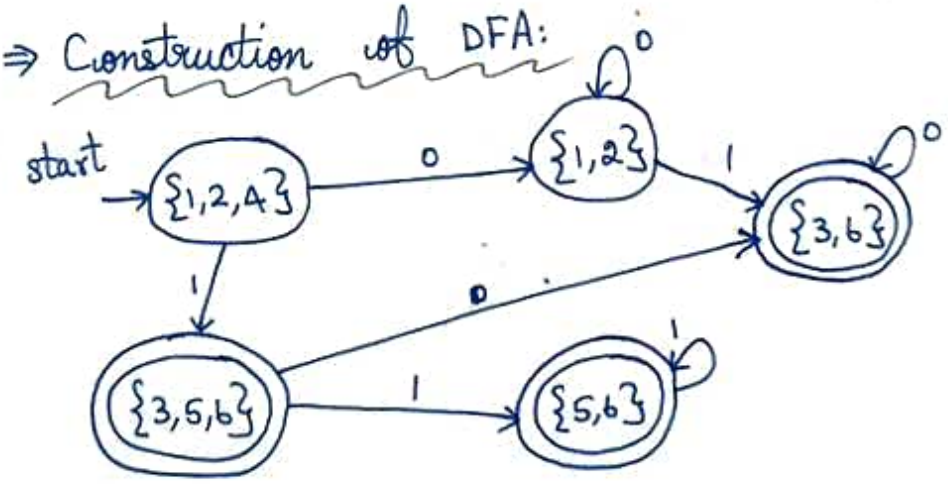
The firstpos and lastpos for each node is represented to the left and right of each node in syntax tree respectively.



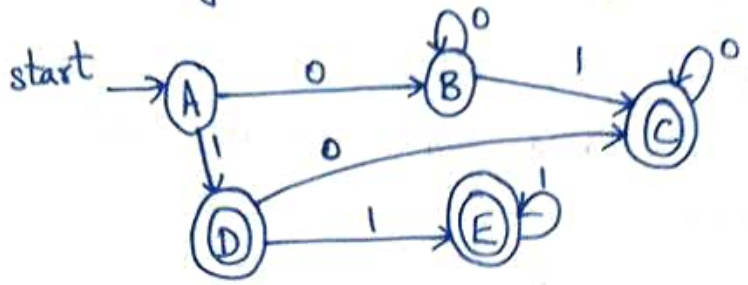
Now, let us compute followpos table:

Symbol	Node number	followpos
0	1	1, 2
1	2	3, 6
0	3	3, 6
1	4	5, 6
1	5	5, 6
#	6	–

⇒ Construction of DFA:



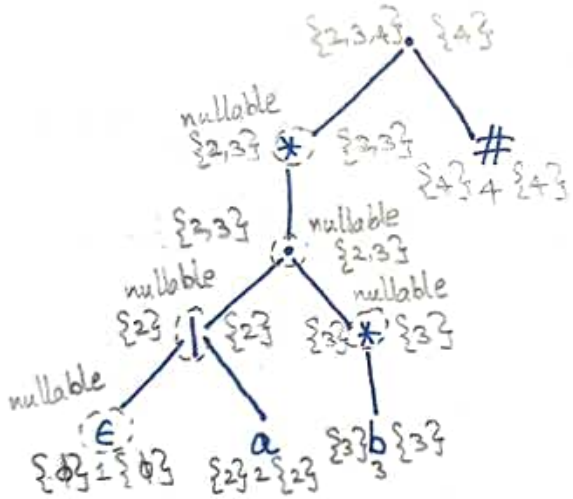
⇒ Renaming the states, we get the DFA as,



(ii) Given RE: $((\epsilon|a)b^*)^*$

⇒ Augmented RE: $((\epsilon|a).b^*)^*.\#$

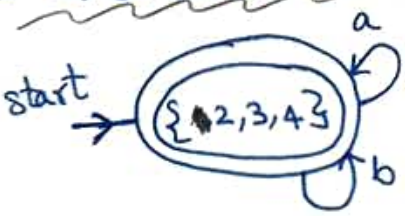
⇒ Syntax Tree:



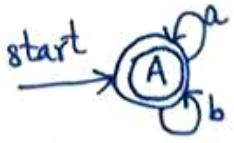
Followpos Table:

Symbol	Node	followpos
#	4	-
a	2	2,3,4
b	3	2,3,4

⇒ Construction of DFA:



⇒ Renaming the states, we get the DFA as,



4) Find whether the given grammar is LR(1) or LALR.

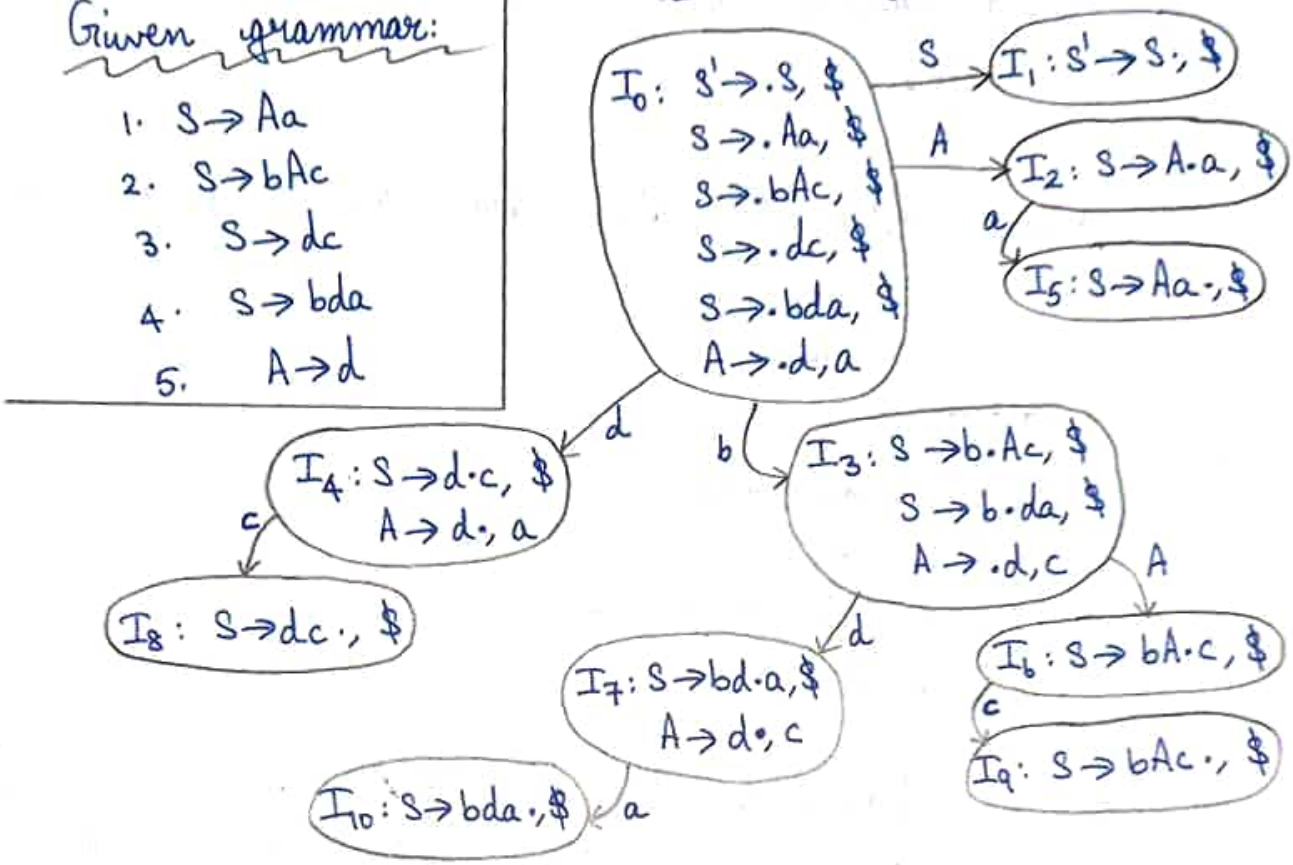
$S \to Aa \mid bAc \mid dc \mid bda$
$A \to d$

Ans:

Given grammar:

1. $S \to Aa$
2. $S \to bAc$
3. $S \to dc$
4. $S \to bda$
5. $A \to d$

The canonical LR(1) collection:



The LR(1) Parsing Table is as follows:

State	Action					Goto	
	a	b	c	d	$	S	A
0		S_3		S_4		1	2
1					accept		
2	S_5						
3				S_7			6
4	r_5		S_8				
5					r_1		
6			S_9				
7	S_{10}		r_5				
8					r_3		
9					r_2		
10					r_4		

Since, the LR(1) Parsing table does not have any conflicts, the given grammar is LR(1).

Now, for LALR(1), there are no redundant cores to combine them. Thus, the given grammar is not LALR(1).

5) Find whether the given grammar is LL or SLR.

$S \rightarrow SA \mid A$
$A \rightarrow d$

Ans:

Given Grammar:

1. $S \rightarrow SA$
2. $S \rightarrow A$
3. $A \rightarrow d$

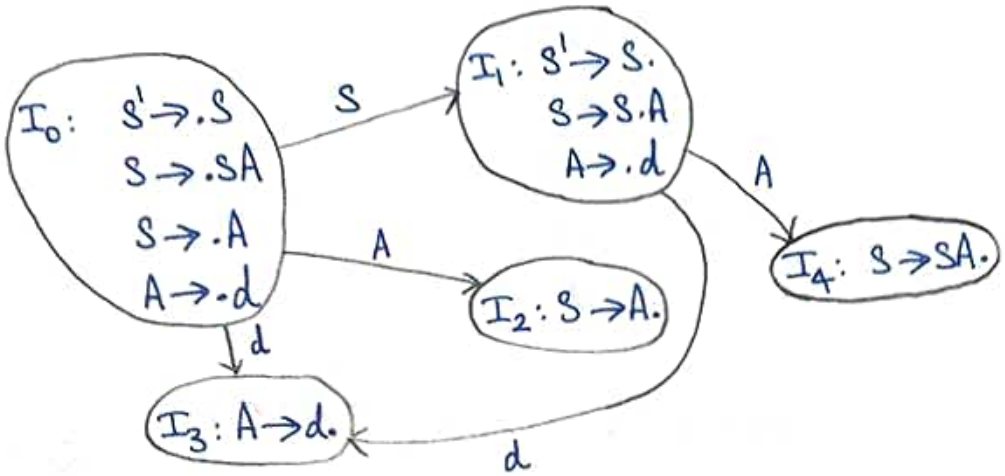
Clearly, the rule 1 in given grammar has a left recursion. Thus, the given grammar is not in LL.

Now, let us compute FIRST and FOLLOW sets:

$FIRST(d) = \{d\}$
$FIRST(S) = \{d\}$
$FIRST(A) = \{d\}$

$FOLLOW(S) = \{\$, d\}$
$FOLLOW(A) = \{\$, d\}$

The cannonical LR(0) collection:



SLR(1) Parsing Table:

state	Action		Goto	
	d	$	S	A
0	S_3		1	2
1	S_3	accept		4
2	r_2	r_2		
3	r_3	r_3		
4	r_1	r_1		

Since, there is no conflict in SLR Parsing Table, the given grammar is SLR.

6) Consider the following grammar:

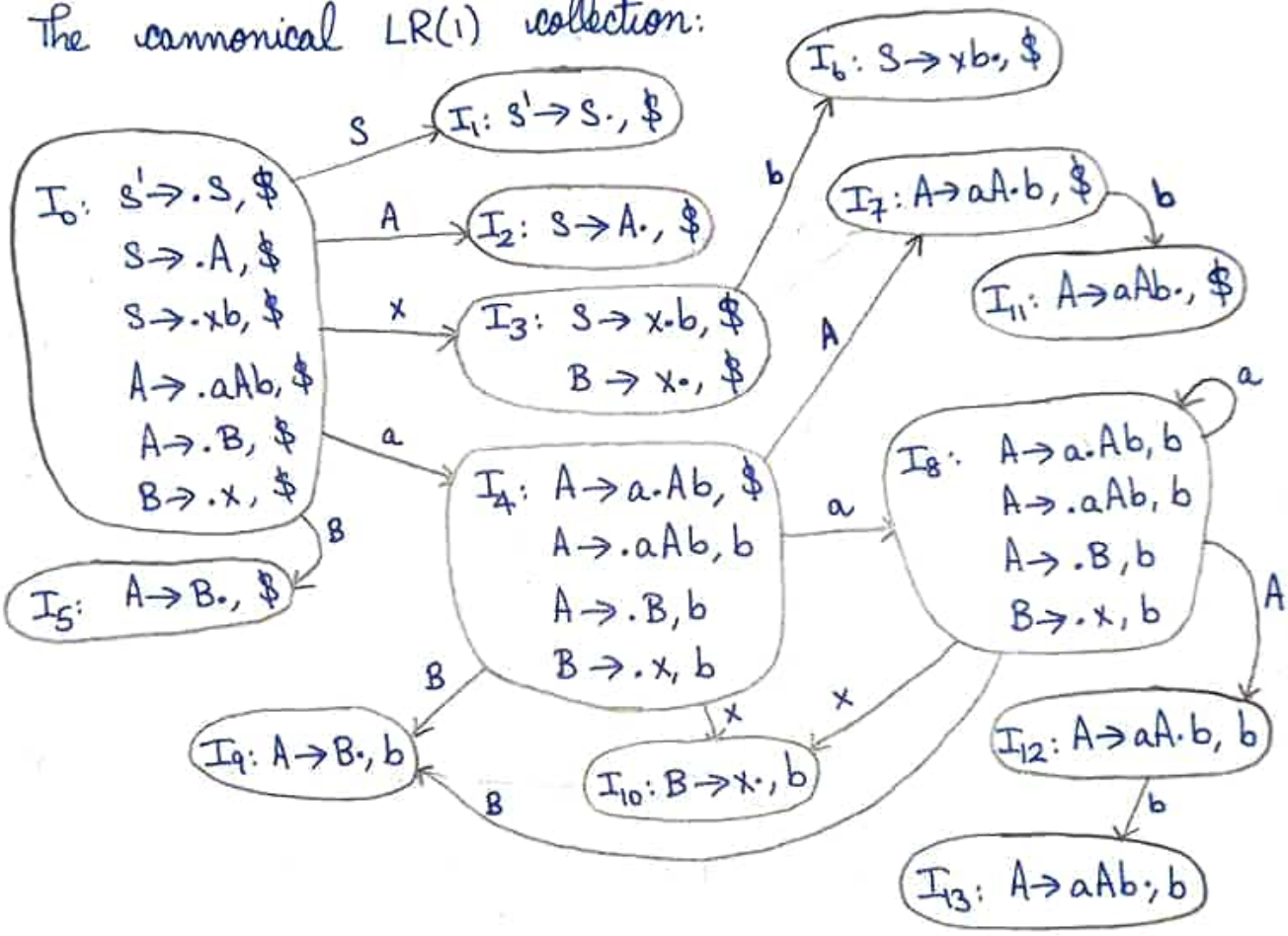
$S \rightarrow A \mid xb$
$A \rightarrow aAb \mid B$
$B \rightarrow x$

(i) Compute LR(1) and create the parsing table.
(ii) Show stack trace for input $w = axb\$$

Ans: (i) Given Grammar:

1. $S \rightarrow A$
2. $S \rightarrow xb$
3. $A \rightarrow aAb$
4. $A \rightarrow B$
5. $B \rightarrow x$

The cannonical LR(1) collection:



The LR(1) Parsing Table is as follows:

State	Action				Goto		
	x	b	a	$	S	A	B
0	S_3		S_4		1	2	5
1				accept			
2				r_1			
3		S_6		r_5			
4	S_{10}		S_8			7	9
5				r_4			
6				r_2			
7		S_{11}					
8	S_{10}		S_8			12	9
9		r_4					
10		r_5					
11				r_3			
12		S_{13}					
13		r_3					

(ii) The stack trace for the input $w = axb\$$ is as follows:

Line	Stack	Symbols	Input	Action
(1)	0	$	axb$	S_4
(2)	04	$a	xb$	S_{10}
(3)	04 10	$ax	b$	r_5
(4)	049	$aB	b$	r_4
(5)	047	$aA	b$	S_{11}
(6)	047 11	$aAb	$	r_3
(7)	02	$A	$	r_1
(8)	01	$S	$	accept