Theory of Computation and Compiler Design

Name:- Ricky Sabharwal

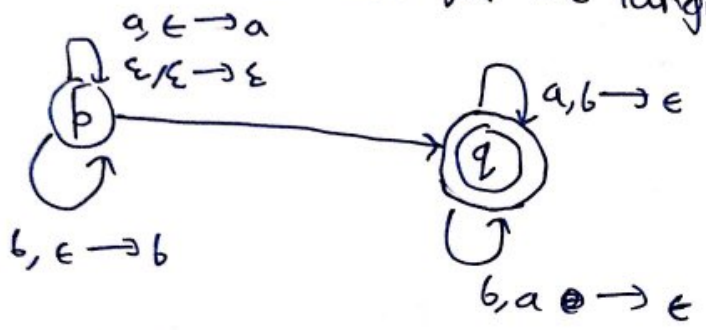
Reg. No:- 17BCE0159

Slot:- A1

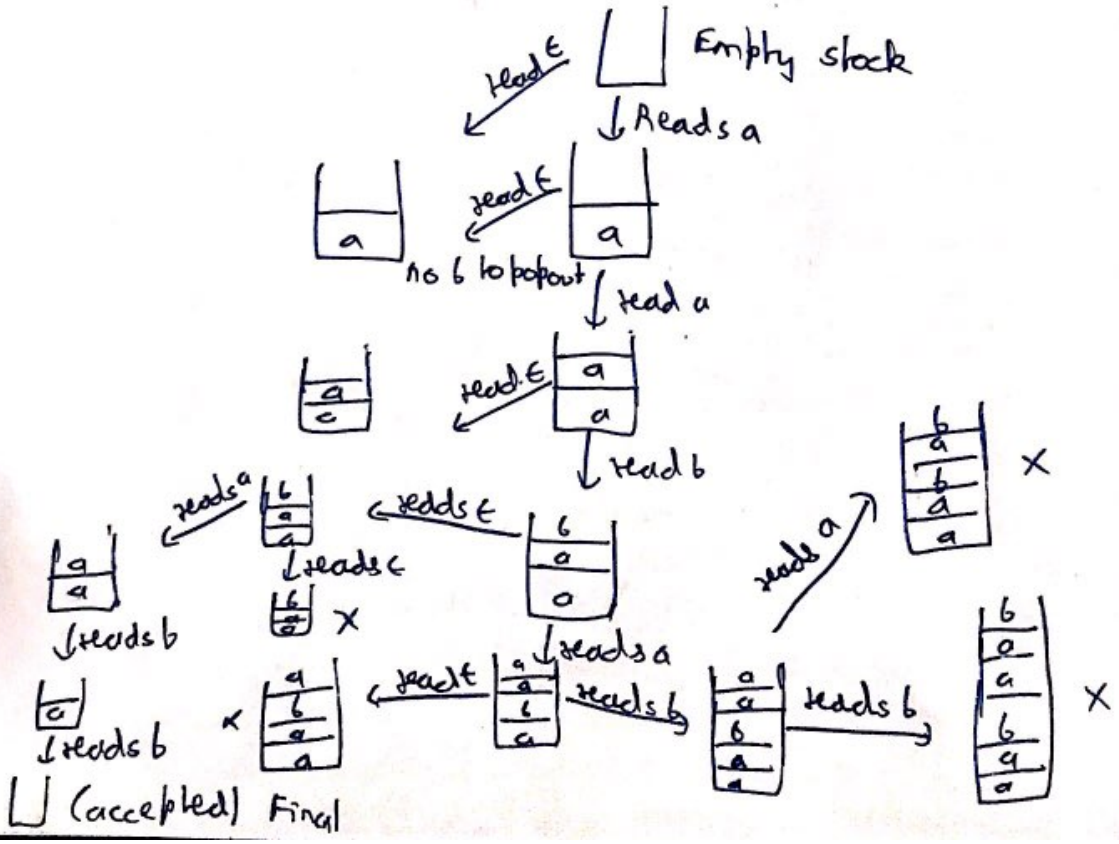
CSE2002

Q1 Consider the language $L = \{w^R w'' / w \in \{a,b\}^*$ and w'' indicates w with each occurence a is replaced by b and vice versa$\}$. Now give a PDA with only two states which accepts L. Is aababb $\in L$? If so show it.

A1 Push down automata for the language is



Yes, aababb $\in L$ as for aababb



Q2 Define recursive sets and recursively enumerable sets. Describe the encoding for TM and an enumeration for TMs. Show that there is a language which is not recursively enumerable and a language which is recursively enumerable but not recursive.

A2 Recursive sets:- A set (or relation) is recursive (or computable or decidable) if it is computable as a total 0-1 valued function.

Recursively Enumerable Sets:- If $A \subseteq N^n$ then A is r.e. (recursively enumerable, or computably enumerable, or semidecidable) if there exists a recursive relation $R \subseteq N^{n+1}$ such that:

$$\vec{x} \in A \iff \exists y R(\vec{x}, y), \text{ for all } \vec{x} \in N^n$$

Encoding TMs

Represent a TM $M = (Q, \Sigma \{0,1\}, \Gamma, \delta, q, L, /, F)$ as a binary string, assigning integers to the states, tape symbols, and directions L and R.

A code for the entire TM M consists of all the codes for the transition functions, in some order, separated by pairs of 1's:

$$C_1 \,||\, C_2 \,||\, \cdots \,||\, C_{n-1} \,||\, C_n$$

where C_i is the code for the i^{th} transition functions of M.

For any non empty Σ, these exists languages that are NOT recursively enumerable. The set of all TMs can be enumerated, so the set of all recursively enumerable language is countable. This implies that these must be some language on Σ that are not recursively enumerable.

Q3 Construct a Turing Machine which will convert a binary number to unary number.

⇒ Using a 2 tape TM

b = blank

$(q_1, [0, b]) \longrightarrow (q_1, [1, b], [L, S])$

$(q_1, [1, b]) \longrightarrow (q_2, [0, b], [R, S])$

$(q_2, [1, b]) \longrightarrow (q_2, [1, b], [R, S])$

$(q_2, [b, b]) \longrightarrow (q_1, [b, 1], [L, L])$

Q4 Consider the grammer

$E \rightarrow E \,'|'\, T / T$

$T \rightarrow T \,\&\, F / F$

$F \rightarrow !F / (E) / 1 / 0$

Hex Non-Terminals = { E, T, F } Terminals = { '|', &, (,), 1, 0 }

$E \rightarrow E \text{ '|' } T / T$

Ans Left Recursion (Removing)

$E \rightarrow TE'$

$E' \rightarrow \text{'|'} TE'$

$T \rightarrow T \& F / F$

Removing left Recursion

$T \rightarrow FT'$

$T' \rightarrow \& FT'$

$F \rightarrow !F / (e) / 1 / 0$

	First	Follow
E	{!, (, 1, 0}	{$,)}
E'	{'|'}	{$,)}
T	{!, F, (, 1, 0}	{1, ,}
T'	{&}	{1, '}
F	{!, (, 1, 0}	{&}

Parse Table

	1	'|'	&	(	)	,	0	1
E	E→TE'			E→TE'		E→TE'	E→TE	
E'		E→'|'TE'						
T	T→FT'			T→FT'		T→FT'	T→FT'	
T'			T'→&FT'					
F	F→!F			F→(E)		F→1	F→0	

Qs Show that the following grammer

$$S \rightarrow AaAb \mid BbBa$$
$$A \rightarrow \epsilon$$
$$B \rightarrow \epsilon$$

is LL(1) but not SLR(1).

Ans First, let's give your productions a number.

1. $S \rightarrow AaAb$
2. $S \rightarrow BbBa$
3. $A \rightarrow \Sigma$
4. $B \rightarrow \Sigma$

Now lets compute the LL(1) table. By definition, if we don't get conflicts, the grammer is LL(1).

	a	b
S	1	2
A	3	3
B	4	4

As there are no conflicts, the grammer is LL(1).

Now for the SLR(1) table. First, the LR(0) automaton.

state 0

$S \rightarrow \cdot AaAb$
$S \rightarrow \cdot BbBa$
$A \rightarrow \cdot$
$B \rightarrow \cdot$

A ⟹ 1

B ⟹ 5

State 1

S ⟶ A . aAb

a ⟹ 2

State 2

S ⟶ Aa . Ab

A ⟶ .

A ⟹ 3

State 3

S ⟶ AaA . b

b ⟹ 4

State 4

S ⟶ AaAb . b

State 5

S ⟶ B . bBa

b ⟹ 6

State 6

S ⟶ Bb . Ba

B ⟶ .

B ⟹ 7

State 7

$S \rightarrow BbB \cdot a$

$a \Rightarrow 8$

State 8

$S \rightarrow BbBa \cdot$

And then the SLR(1) table (I assume S can be followed by anything).

	a	b	A	B
0	R3/R4	R3/R4	1	5
1	S2			
2	R3	R3	3	
3		S4		
4	R1	R1		
5		S4		
6	R4	R4		7
7	S8			
8	R2	R2		

There are conflicts in state 0, so the grammer is not SLR(1).

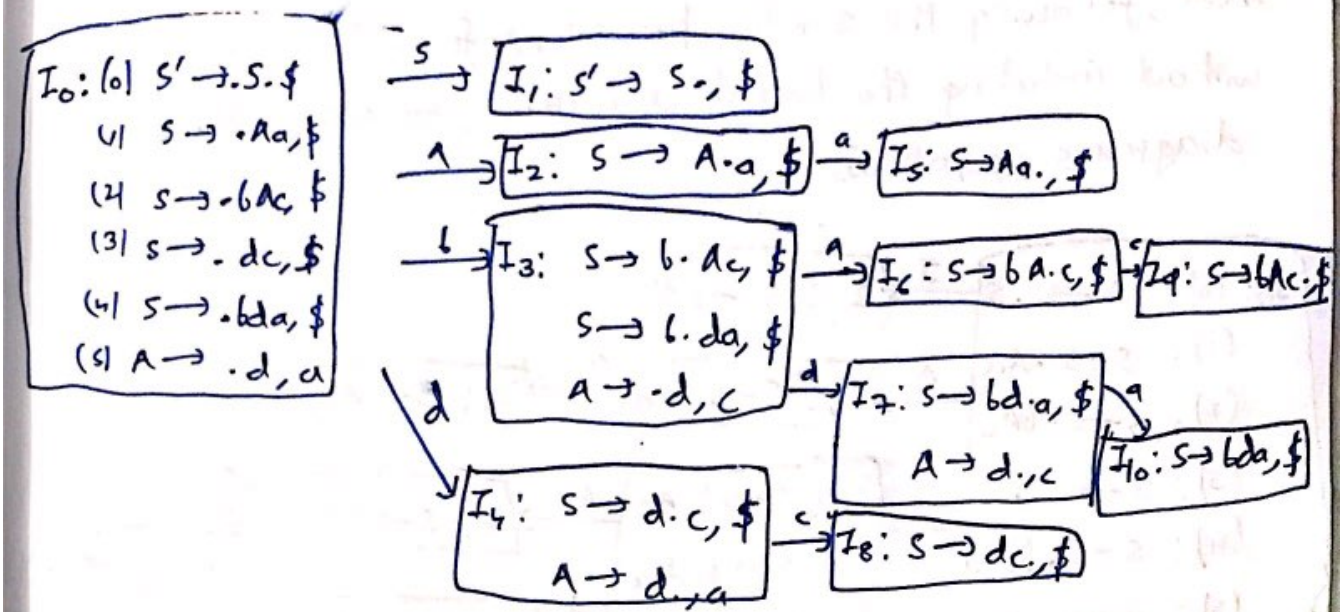
Q6 Show that the following grammer

$S \rightarrow Aa/bAc/dc/bda$

$A \rightarrow d$

is LALR(1) but not SLR(1).

Ans In addition to the rules given above, one extra rule $S' \to S$ as the initial item. Following the procedures for constructing the LR(1) parser, here is the initial state and the resulting state diagram by taking closure.

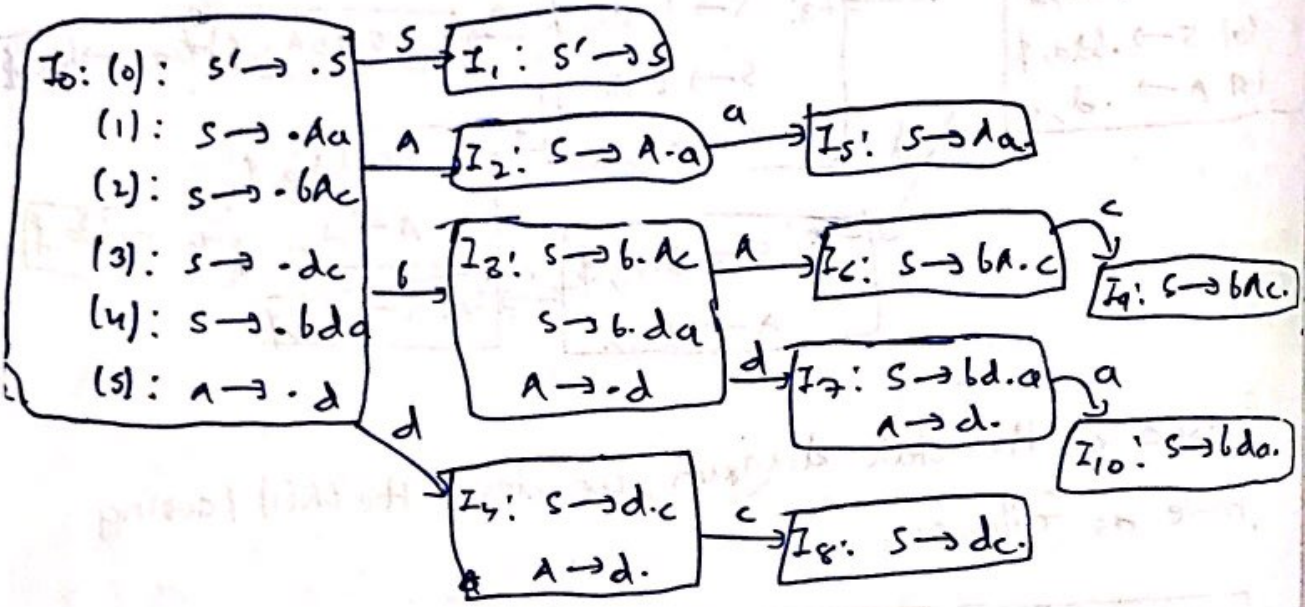


Based on the state diagram, we derive the LR(1) parsing table as follows:-

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

Then, the LALR(1) parsing table can be obtained by merging items with common first components. In this problem, no merging occurs. That is, the final LALR(1) parsing table is the same as the LR(1) one. Thus, the given grammer is LALR(1).

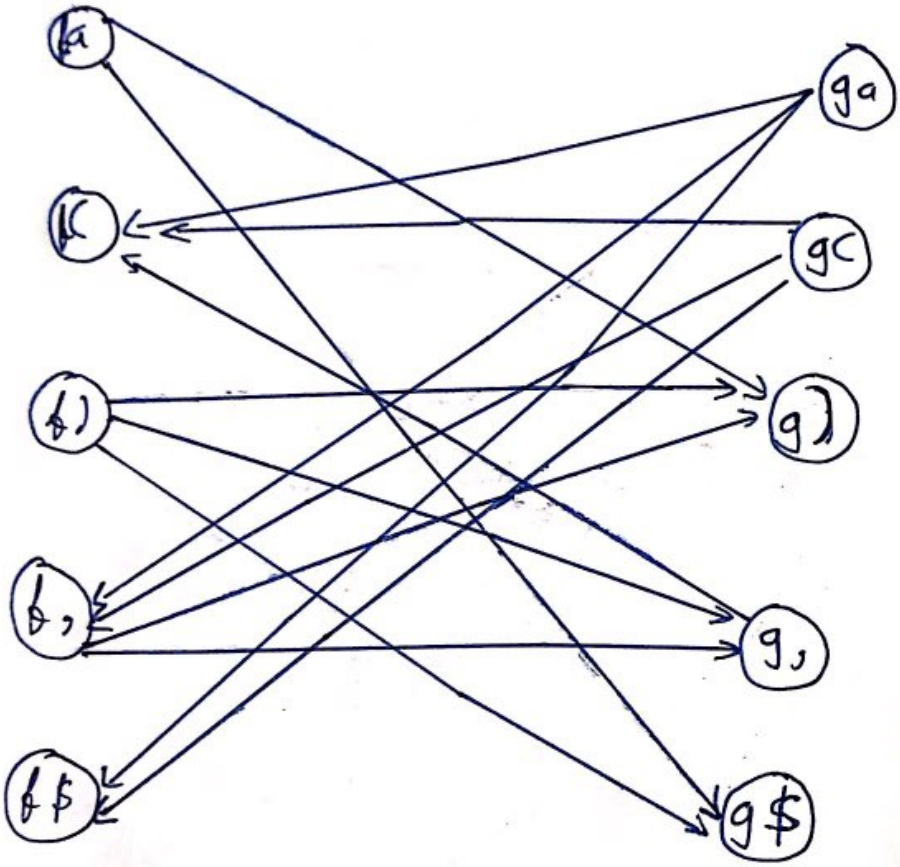
Next, following the similar procedures for taking closure, but without including the lookahead in items, we obtain the state diagrams as follows:-



Lets assume that the parser is in state I_7, and the next symbol is a, since $a \in Follows(A) = \{a, c\}$, it causes a shift-reduce conflict. Same problem also happens to state I_4. Thus, the given grammer is not SLR(1).

Q7 Find the operator-precedence function for the following table.

	a	(	)	,	$
a			>	>	>
(	<	<	=	<	
)			>	>	>
,	<	<	>	>	
$	<	<			



	a	(	)	,	$
f	2	0	2	2	0
g	3	3	0	1	0