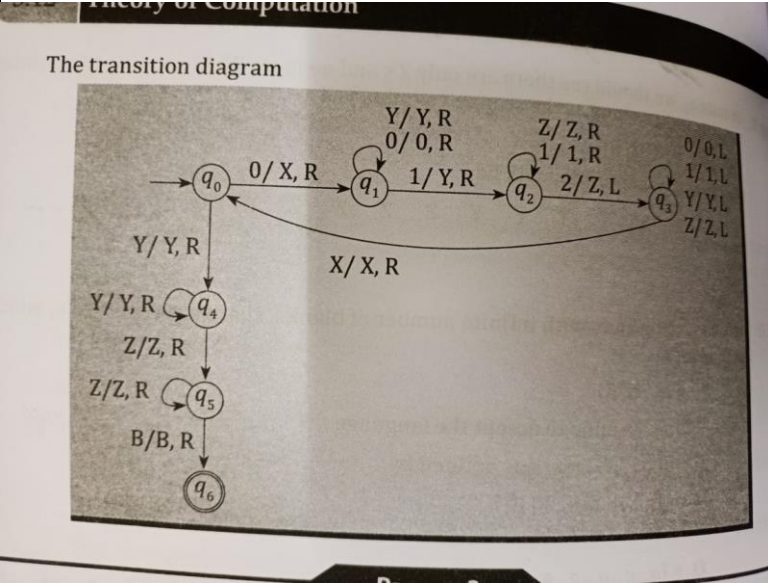
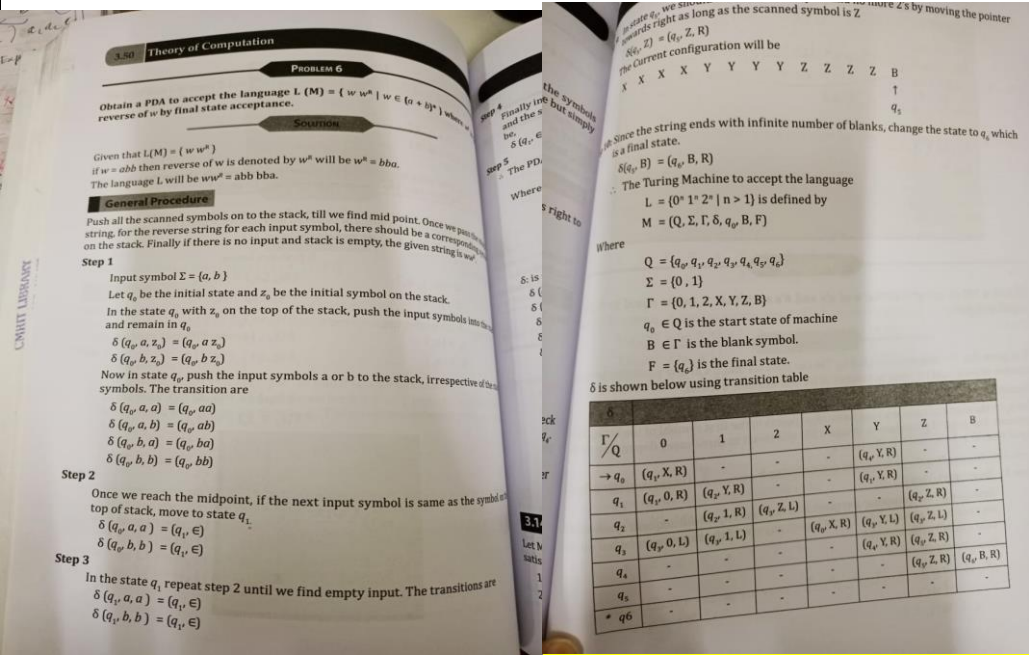


Internal Assessment Test 3– March 2024

Sub:	Automata Theory & Compiler Design	Sub Code:	21CS51	Branch:	AIML
Date:	16/3/2024	Duration:	90 min's	Max Marks:	50
		Sem/Sec:	V / A		OBE

Answer any FIVE FULL Questions

1	<p>What is Turing machine? 2 Marks</p> <p>Construct a Turing machine to accept the language $L(M) = \{ 0^n 1^n 2^n \mid n \geq 1 \}$. 8 Marks</p> <p>SOLUTION:</p> 	10	CO2	L2
---	---	----	-----	----

2	<p>Obtain a PDA to accept the language $L(M) = \{ ww^R \mid w \in (a+b)^* \}$ where w^R is the reverse of w. 10 Marks</p> <p>SOLUTION:</p> 	10	CO2	L2
---	--	----	-----	----

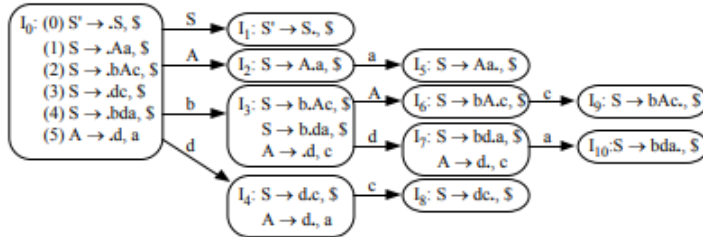
but not SLR(1). 3 Marks

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

$A \rightarrow d$

SOLUTION:

Answer: In addition to the rules given above, one extra rule $S' \rightarrow S$ 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:

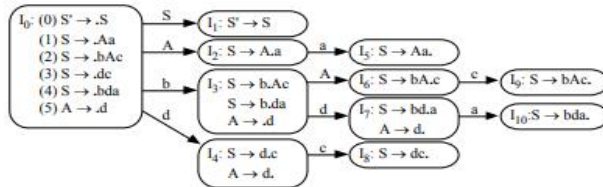


Based on the state diagram, we derive the LR(1) parsing table as follows:

State	Action					Goto	
	a	b	c	d	\$	S	A
0		s3		s4		1	2
1					acc		
2	s5						6
3				s7			
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 grammar is LALR(1).

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



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

4a) Explain with an example to generate the intermediate code for the flow of control Statements. 3 Marks IC and 2 marks example

SOLUTION: Intermediate code is used as a bridge between the high-level language representation of a program and the machine code generated by the compiler or interpreter. Three-address code is a type of intermediate code that uses instructions with at most three operands.

In three-address code, each instruction generally performs a simple operation and can have up to three operands: one result and two operands. The result is usually stored in a temporary variable.

Let's discuss the types of three-address statements and provide an example:

Types of Three-Address Statements:

Assignment Statements: These statements assign a value to a variable.

- Example: $x = y + z$
- Here, $y + z$ is computed, and the result is assigned to variable x .

Arithmetic Expressions: These involve arithmetic operations like addition, subtraction, multiplication, and division.

- Example: $t1 = x + y$
- Here, $x + y$ is computed, and the result is stored in temporary variable $t1$.

Conditional Statements: These statements involve conditionals and control flow, such as if-else statements.

- Example:
if ($x < y$) goto L1
- else goto L2 Here, the program jumps to label $L1$ if $x < y$, otherwise, it jumps to label $L2$.

Jump Statements: These statements change the sequence of execution, such as unconditional jumps.

- Example: goto $L1$
- Here, the program jumps to label $L1$ unconditionally.

Generating Intermediate Code for Flow Control Statements:

Let's consider a simple example of generating intermediate code for a flow control statement, specifically an if-else statement.

```
if ( $x < y$ ) {  
     $z = x + y$ ;  
} else {  
     $z = x - y$ ;  
}
```

We can represent this code using three-address code as follows:

1. if $x < y$ goto L1
2. $t1 = x + y$
3. goto L2
4. L1: $t1 = x - y$

	<p>5. L2: $z = t1$</p> <p>Here, L1 and L2 are labels used for control flow. If $x < y$, the program jumps to L1 and performs $t1 = x - y$. Otherwise, it continues to L2 and performs $t1 = x + y$. Finally, the result is stored in variable z.</p> <p>This demonstrates how control flow statements can be represented using three-address code, providing a simpler and more manageable representation for further optimization and code generation.</p>			
4b)	<p>Also explain three address codes and its types. 1 Mark How would you implement the three Address statements? 2 Marks Explain with suitable examples. 2 Marks</p> <p>SOLUTION:</p> <p>Three-address code (TAC) is a low-level intermediate representation used in compilers to represent statements in a program. Each statement in three-address code typically contains at most three operands and one operator. The main purpose of using three-address code is to simplify complex expressions and control structures into a form that is easier to analyze and optimize.</p> <p>Three-address code consists of the following basic types of statements:</p> <p>Assignment statements: These statements assign the result of an expression to a variable. Example: $t1 = a + b$ $c = t1$</p> <p>Example: if $a < b$ goto L1</p> <p>Unary and binary operation statements: These statements perform arithmetic or logical operations. Example: $t2 = a * b$ $t3 = c - d$</p> <p>Address statements: These statements handle memory addresses, typically in the context of pointers. Example: $t4 = \&a$</p> <p>Function call and return statements: These statements represent function calls and returns. Example: call fun</p> <p>Implementing three-address statements involves creating a data structure to represent each statement and developing algorithms to generate and manipulate these statements during compilation.</p>	5	CO2	L1

5 Construct a canonical parsing table for the grammar given below.

10 CO3 L2

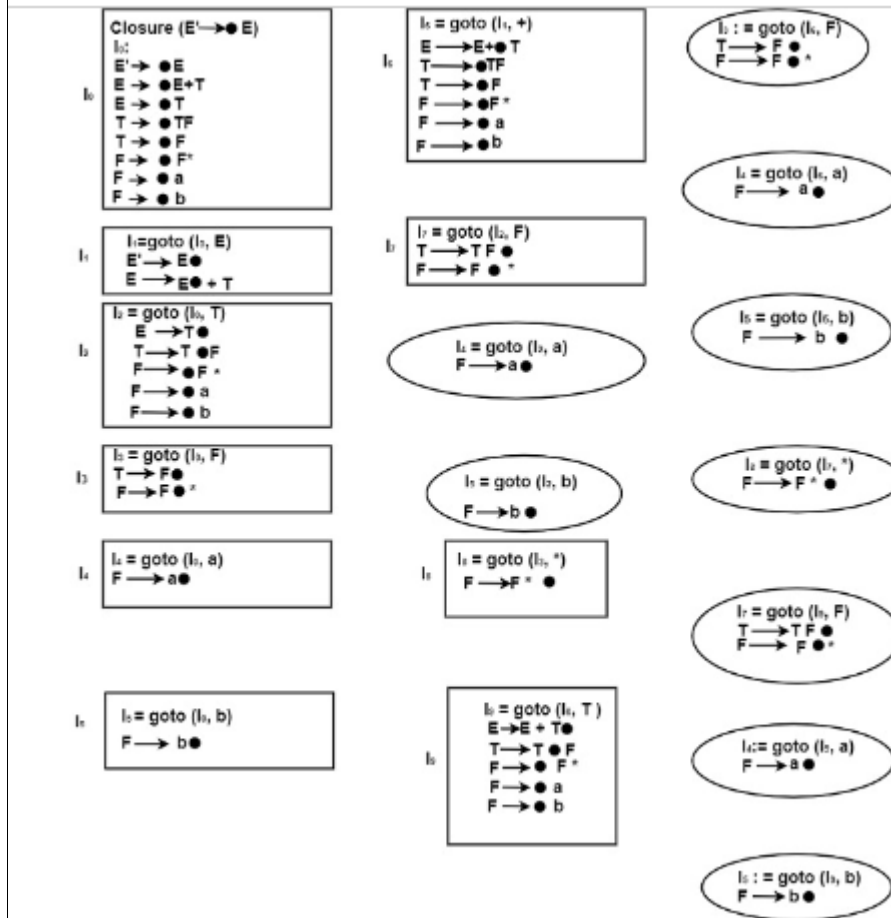
$E \rightarrow E + T | T$

$T \rightarrow TF | F$

$F \rightarrow F * | a | b$ 8 Marks (4 marks GOTO, 4 Marks PT)

and parse any string derived from the grammar. 2 Marks

SOLUTION:



State	Action					goto		
	+	*	a	b	\$	E	T	F
0			s4	s5		1	2	3
1	s6				accept			
2	r2		s4	s5	r2			7
3	r4	s8	r4	r4	r4			
4	r6	r6	r6	r6	r6			
5	r6	r6	r6	r6	r6			
6			s4	s5			9	3
7	r3	s8	r3	r3	r3			
8	r5	r5	r5	r5	r5			
9	r1		s4	s5	r1			7

6 What is a top down parser? Find LL(1) for the given grammar and draw the

10 CO2 L2

parse table.

$S \rightarrow AA$,

$A \rightarrow Aa \mid b$ **8 Marks**

and check for the acceptance of a string $w=abab$ **2 Marks**

SOLUTION:

