Automata Theory And Compiler Design

III - B.Tech II - Semester End Examinations (Supply)

Part - A

Subject code: R22IT601PC

1. a) what are the Differences between DFA and NFA?

Ans) A Deterministic Finite Automaton has a single, definite next state for every input symbol, making its behavior predictable. Does not allow transitions without input. Generally more complex to construct for certain languages. Requires more space but has faster execu

A Non-Deterministic Finite Automaton can transition to zero, one, or multiple states for the same input symbols and allow ε(e transitions.

b) Define string. write about concatenation of two strings.

Ans) A string is a finite sequence of characters from a given alphabet.

Concatenation is the operation of joining two strings end-to-end to form a single longer string

Example "snow", "ball" is

"snowball"

6. Regular Expression to accept the language consisting strings of a's and b's of even length.

ans) the regular expression to accept the language consisting of strings of a's and b's of even length is $((a|b)(a|b))^*$

$(a|b)$ matches a single 'a' or a single 'b'

$(a|b)(a|b)$ matches any pair of characters (eg. aa, ab, ba, bb)

) Define regular Grammar.

s) A regular grammar is a formal grammar used to generate and describe regular languages. Its Production rules are highly

restricted and must be one of two forms (either purely right-linear or purely left linear)

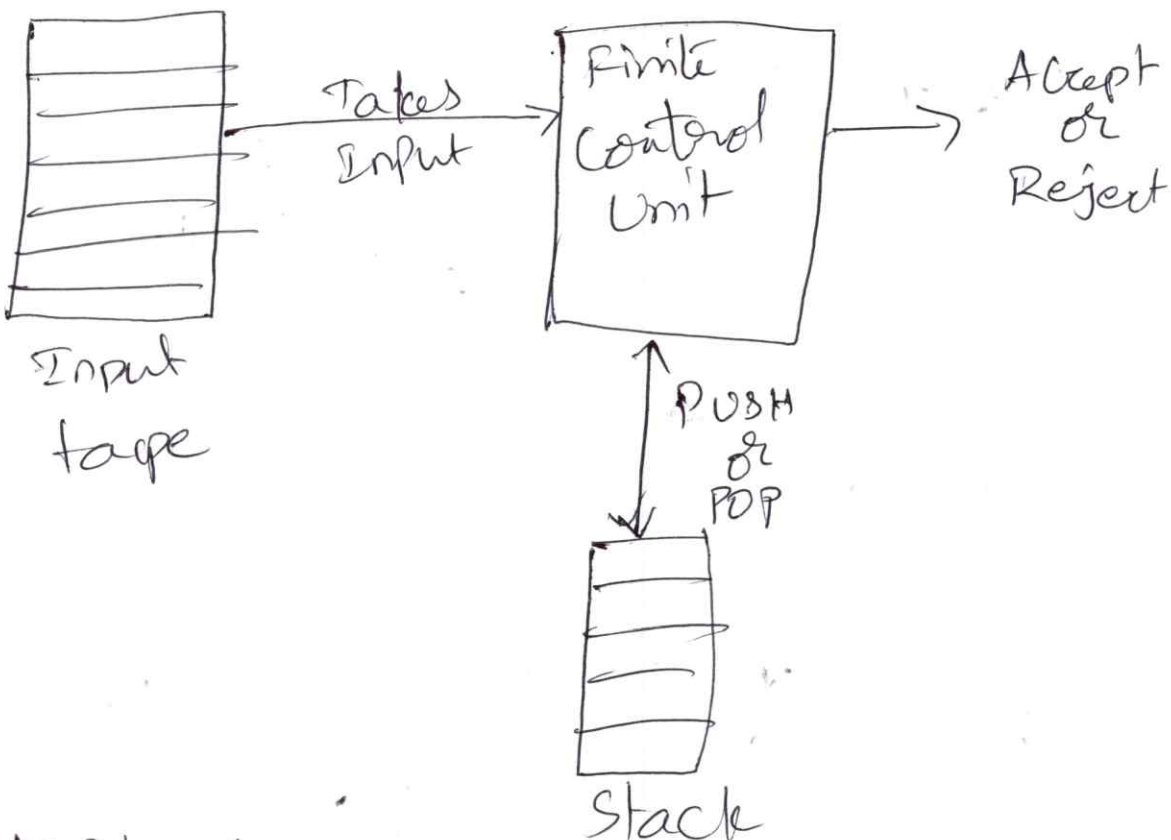
→ Right-linear : $A \to aB$ or $A \to a$

Left-linear : $A \to Ba$ or $A \to a$

where A and B are non-terminals and a is a terminal symbol.

c) Sketch the block diagram of PDA.

Ans)



Input tape: where the input string is read from.

Read Head: Reads one symbol at a time from the input tape.

Finite control: The 'brain' of the PDA, which is in

one of a finite number of stacks

Stack : An infinite storage unit with operations push and pop.

stack Head : points to the top of the stack.

f) What is decidability? Give an example.

Ans) Decidability in the theory of computation refers to whether an algorithm (specifically a Turing machine) exists that can determine for any given input to a problem, if the answer is 'yes' or 'no' in a finite amount of time. A problem is decidable if such a decider algorithm is guaranteed to halt on all inputs.

example the membership problem for regular languages: Given any regular language L and any string w, is w in L? This is decidable because a DFA (or NFA) can process the string in a finite number of steps and definitively determine if it ends in an accepting state

g) Differentiate between Compiler and Interpreter

Ans) Translate the entire source program into machine code (object code) before execution in compiler

Translates and executes the source program line by line ~~or state~~ in interpreter.

h) What is the role of lexical Analyzer?

Ans) The lexical analyzer is the first phase of compiler, role is to read the source program character by character and group them into meaningful sequences called tokens.

i) Define Synthesized and Inherited attributes

Ans) Synthesized Attributes: the value of a synthesized attribute at a parse tree node is determined solely by the attribute values of its children nodes / Information flows up the parse tree.

Inherited Attributes

The values in this at a tree node is determined by the attribute values of its parent and/or sibling nodes. Information flows down or across the parse tree.

=

i) Distinguish between static scope and dynamic scope.

Ans) static scope: The scope of a variable is determined at compile-time by the physical structure of the program text.

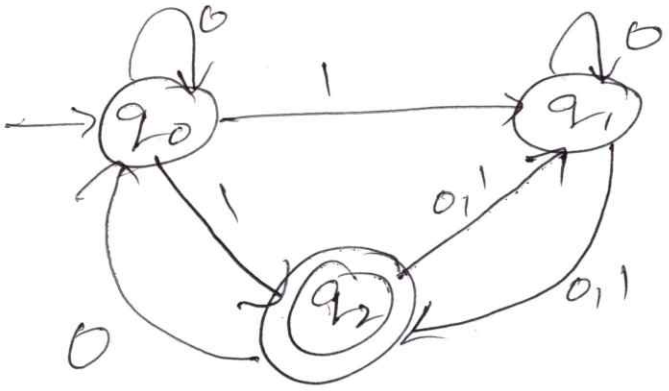
Dynamic scope: The scope of a variable is determined at run-time by the sequence of function calls that led to the current execution point. A variable's visibility depends on the active function calls in the execution stack, not just its textual location.

=

PART–B

(4)

2. Convert the following Non-Deterministic Finite Automata (NFA) to Deterministic Finite Automata (DFA).



Ans) 1) NFA transition table

State / Δ	0	1
→ q_0	q_0	q_1, q_2^*
q_1	$q_1\ q_2^*$	$*q_2$
$*q_2$	q_0, q_1	q_1

Step 1 let Q be the new set the states of the Deterministic Finite Automata T be a new transition table of the DFA.

Step 2: Transition start from q_0 (initial state)

State / Δ	0	1
→ q_0	q_0	$\{q_1, q_2\}$

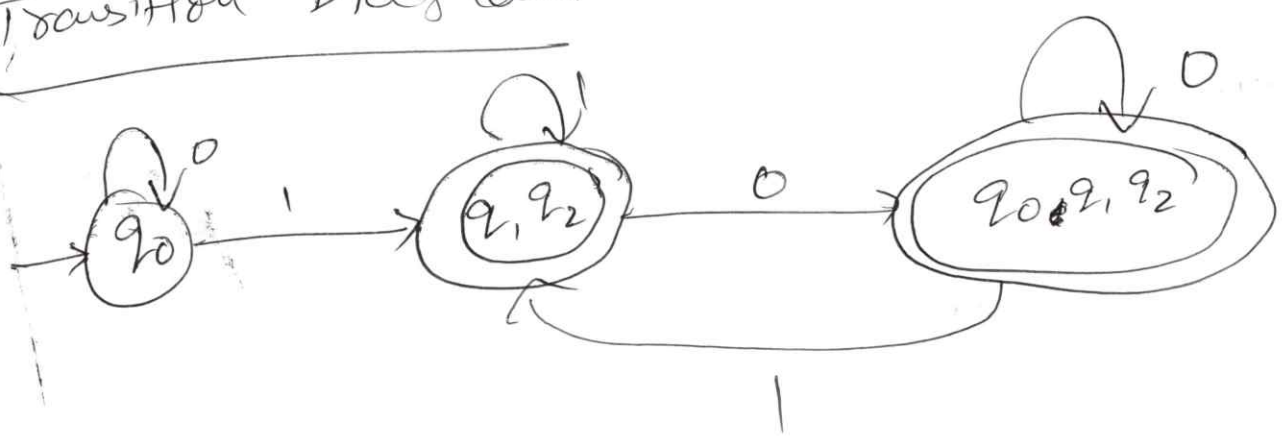
3)

S/A	0	1
q_0	q_0	$\{q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$

4)

S/A	0	1
→ q_0	q_0	$\{q_1, q_2\}$
* $\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$
* $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$

Transition Diagram

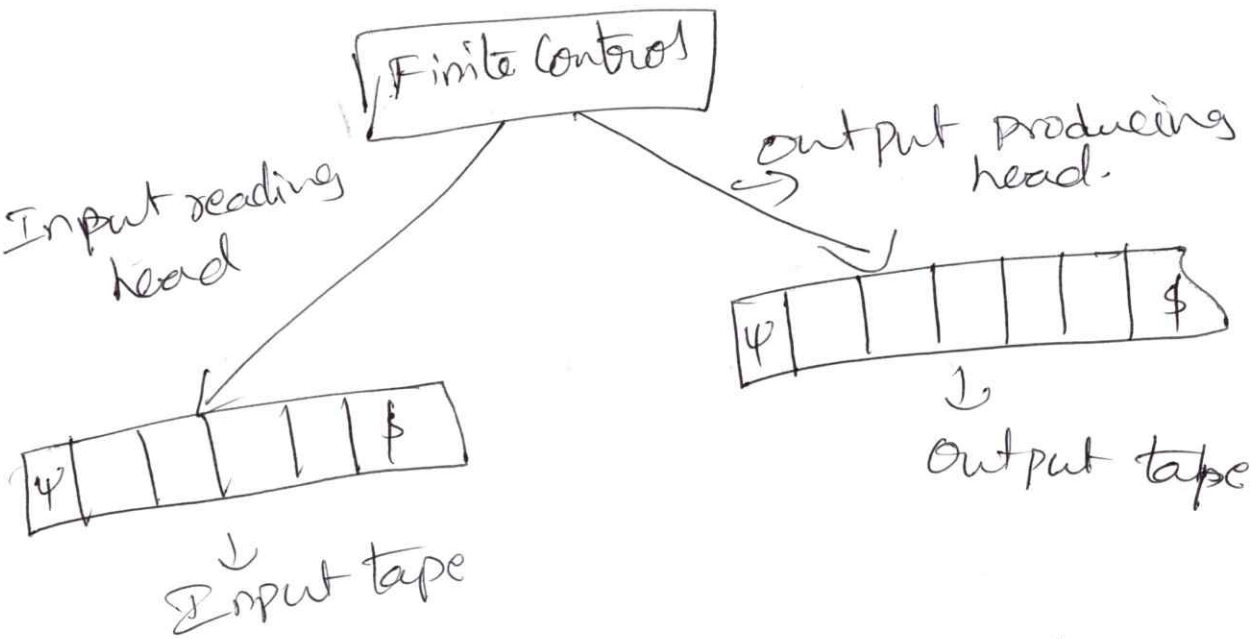


3

a) Explain the Block diagram of finite state machine with an example.

b) Design a DFA which accepts set of all strings which are divisible by 5 for binary alphabet.

a) A finite state machine is a model of computation with a finite number of states, transitions between them based on input.



A finite state machine is represented by 6-tuples $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where

Q - is a finite and non-empty set of states

Σ is a input alphabet

Δ is a output alphabet

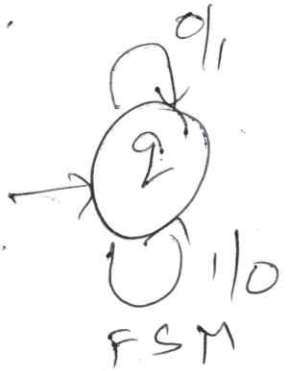
δ is transition function $Q \times \Sigma \to Q$

λ is the output function

$q_0 \in Q$ is initial state.

Example:

Any one example like.

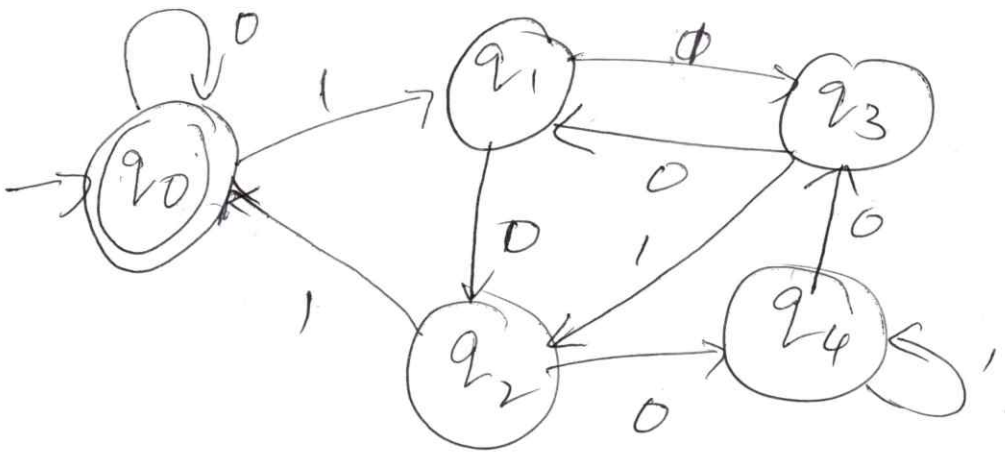


FSM

b)

$$L = \{ w \mid w \bmod 5 = 0 \}$$

i.e $L = \{000, 101, 1010, 1111 \ldots\}$



Example

Consider the input 10011 (decimal 35)

Should go to final state since (35 mod 5)

The about is the example for this.

4) (6)

a) Prove that $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1) = 0^*1(0+10^*1)^*$.

Ans) L.H.S.

$$(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$$

we will take $(1+00^*1)$ as a common factor

$$(1+00^*1)\underbrace{(\epsilon+(0+10^*1)^*(0+10^*1))}$$

$[C\epsilon + R^*R)$.

where $R = (0+10^*1)]$.

i.e $\epsilon + R^*R = \epsilon + RR^* = R^*$

$(1+00^*1)((0+10^*1)^*)$ Out of this $1=0$

$\underbrace{(1+00^*1)}(0+10^*1)^*$

from this taking 1 as common factor.

$1(\epsilon+00^*)(0+10^*1)^*$

Applying $\epsilon + 00^* = 0^*$

ie $0^*1(0+10^*1)^*$.

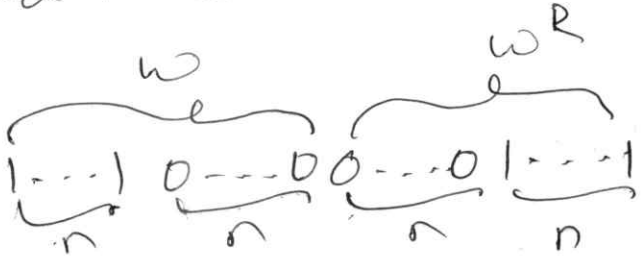
Proved

R.H.S

b) show that $L = \{ww^R / w \in (0+1)^*\}$ is not regular

(or) step 1:

let L is regular and n be the number of states in FA.

string $z =$ 

where n is the number of states of FA.

$w = 1...1\,0...0$ and

reverse of w i.e $w^R = 0..01...1$.

step 2: split the string z into uvw such the $|uv| \le n$ and $|v| \ge 1$ as shown below.

$z =$ 

where $|u| = n-1$

$|v| = 1$

so that $|uv| = |u| + |v| = n-1+1 = n$ which is true.

According to pumping lemma, $uv^iw \in L$ for $i = 0, 1, 2 \ldots$

step 3: if i is 0 i.e v does not appear and the number of 1's on the left of z will be less than the number of 1's on the right of z and so the string is not in the form ww^R.

so it is not regular.

5 a) Discuss closure properties of regular languages.

b) What is meant by ambiguous grammar? Prove that the following grammar is ambiguous.

P: $S \rightarrow E+E \mid E*E \mid id$.

Ans) A set of languages exhibits closure under an operation if applying that operation to one or more languages in the set always results in a language are closed under many operation

key closure properties

1. Union (U) : if L_1, L_2 are regular then $L_1 \cup L_2$ is regular
2. Concatenation ($\cdot$) : if $(L_1 \cdot L_2)$ are regular then $L_1 L_2$ is regular
3. Kleen star : if L is regular L^* regular
4) Intersection ($\cap$)
5) Complement
6) Difference
7) Reversal
8) Homomorphism
9) Inverse Homomorphism.

any 6 with Explanation

b) A context-free Grammar is ambiguous if there exists at least one string in the language it generates that has more than one distinct parse tree (or leftmost / rightmost derivation).

This means the grammar doesn't uniquely define the structure of the input.

P : $S \rightarrow E+E \mid E*E \mid id$.

string id + id * id.

$S \rightarrow E * E$

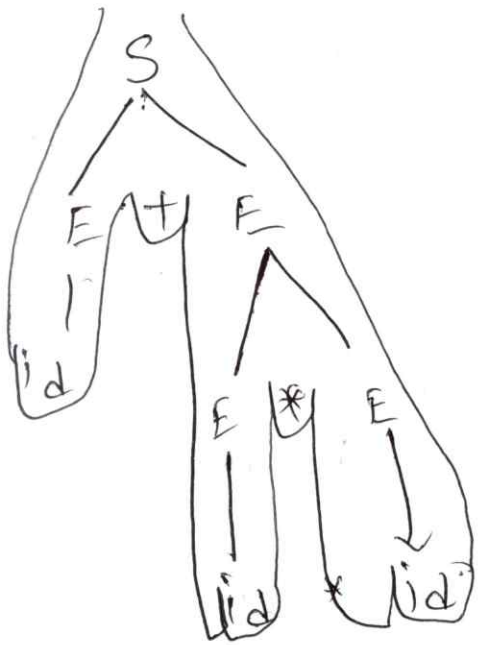
$S \rightarrow (E+E) * E$

$S \rightarrow id + id * id$

$S \rightarrow E + E$

$S \rightarrow E + (E * E)$

$S \rightarrow id + id * id$



Since same string generated with two different parse trees. This is called ambiguity this grammar called ambiguous grammar.

b) Construct a pushdown automaton which accepts the language of words over the alphabet $\{a, b\}$ containing more a's than b's. (8)

Ans) To construct a PDA for language

$L = \{ w \in \{a,b\}^* \mid \#_a(w) > \#_b(w) \}$ use two states q_0 (counting state) and q_1 accepting (final state) Push a for every 'a' input POP a for every b input.

write 7-tuples. Then write Transitions.

like:

1. Read 'a' (push): For every 'a' push another 'a' on to the stack to count the excess 'a's.

$$\delta(q_0, a, z_0) \rightarrow (q_0, az_0)$$

$$\delta(q_0, a, a) \rightarrow (q_0, aa)$$

2. Read b (POP).

$$\delta(q_0, b, a) \rightarrow (q_0, \epsilon) \text{ (POP 'a')}$$

if stack empty on 'b' the PDA crashes/ rejects.

3. End of input: $\delta(q_0, \epsilon, a) \rightarrow (q_1, a)$

$\delta(q_1, \epsilon, a) \rightarrow (q_1, \epsilon)$

$\delta(q_1, \epsilon, z_0) \rightarrow (q_1, z_0)$

7. Construct a Turing Machine to recognize the language

$L = \{a^n b^n c^n / n >= 1\}$.

Ans) Note that n number of a's are followed by n number of b's which inturn are followed by n number of c's. In simple terms

XXaaYY bbzzcc XXaYYYbbzzcc

↑ q0 ↑ q1

Various Configurations.

i.e $\delta(q_0, a) = (q_1, X, R)$

$\delta(q_1, a) = (q_1, 0, R)$

$\delta(q_1, Y) = (q_1, Y, R)$ like this q_b states

TM

$L = \{a^n b^n c^n / n \geq 1\}$ is given by

$M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$.

$Q = \{q_0, q_1, q_2, q_3 q_4, q_5, q_6\}$

$\Sigma = \{a, b, c\}$

$\Gamma = \{a, b, c, X, Y, Z, B\}$;

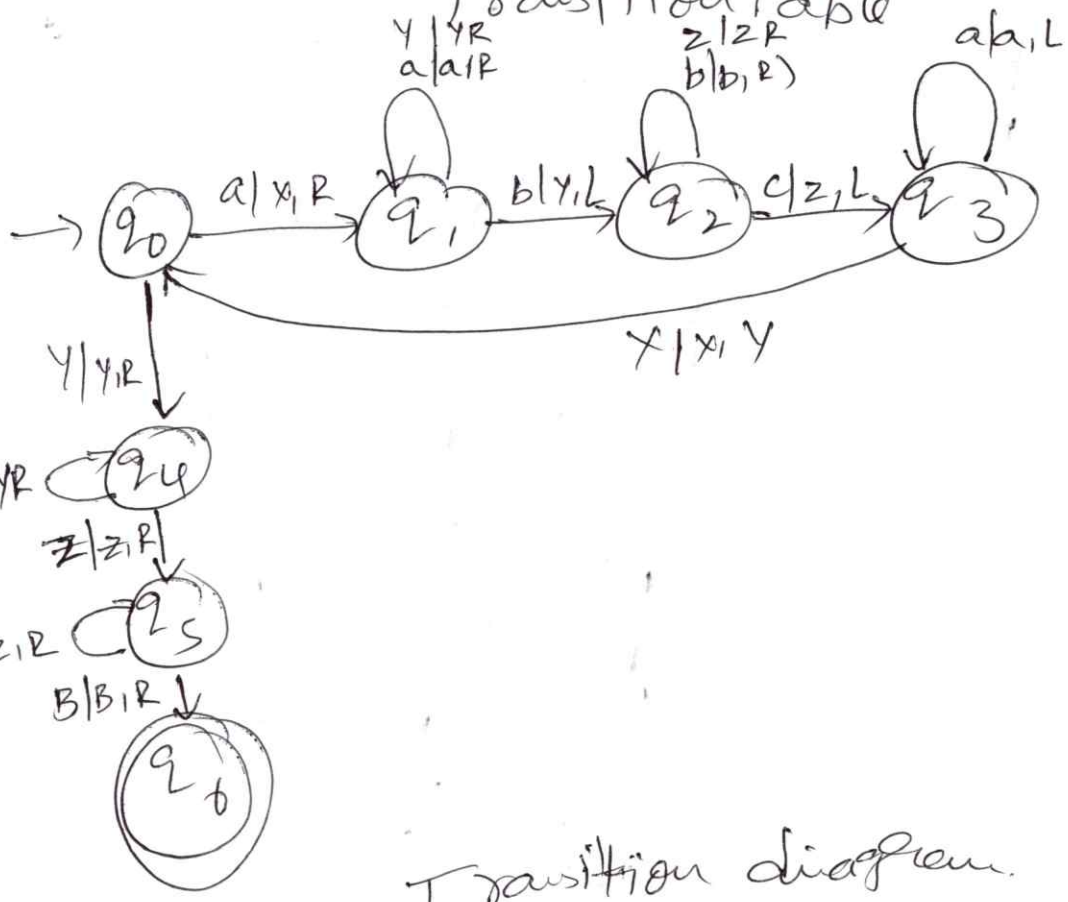
q_0 initial state F: is Final state

B : blank character

δ is transition function.

			r				
state	a	b	c	Z	Y	X	B
q_0	$q_1 X_1 R$				$q_4 Y R$		
q_1	$q_1 a_1 R$	$q_2 Y R$			$q_1 Y R$		
q_2		$q_2 b R$	$q_3 Z L$	$q_2 Z R$			
q_3	$q_3 a_1 L$	$q_3 b L$		$q_3 Z L$	$q_3 Y L$	$q_0 X R$	
q_4				$q_5 Z R$	$q_4 Y R$		
q_5				$q_5 Z R$			$q_6 B R$
q_6							

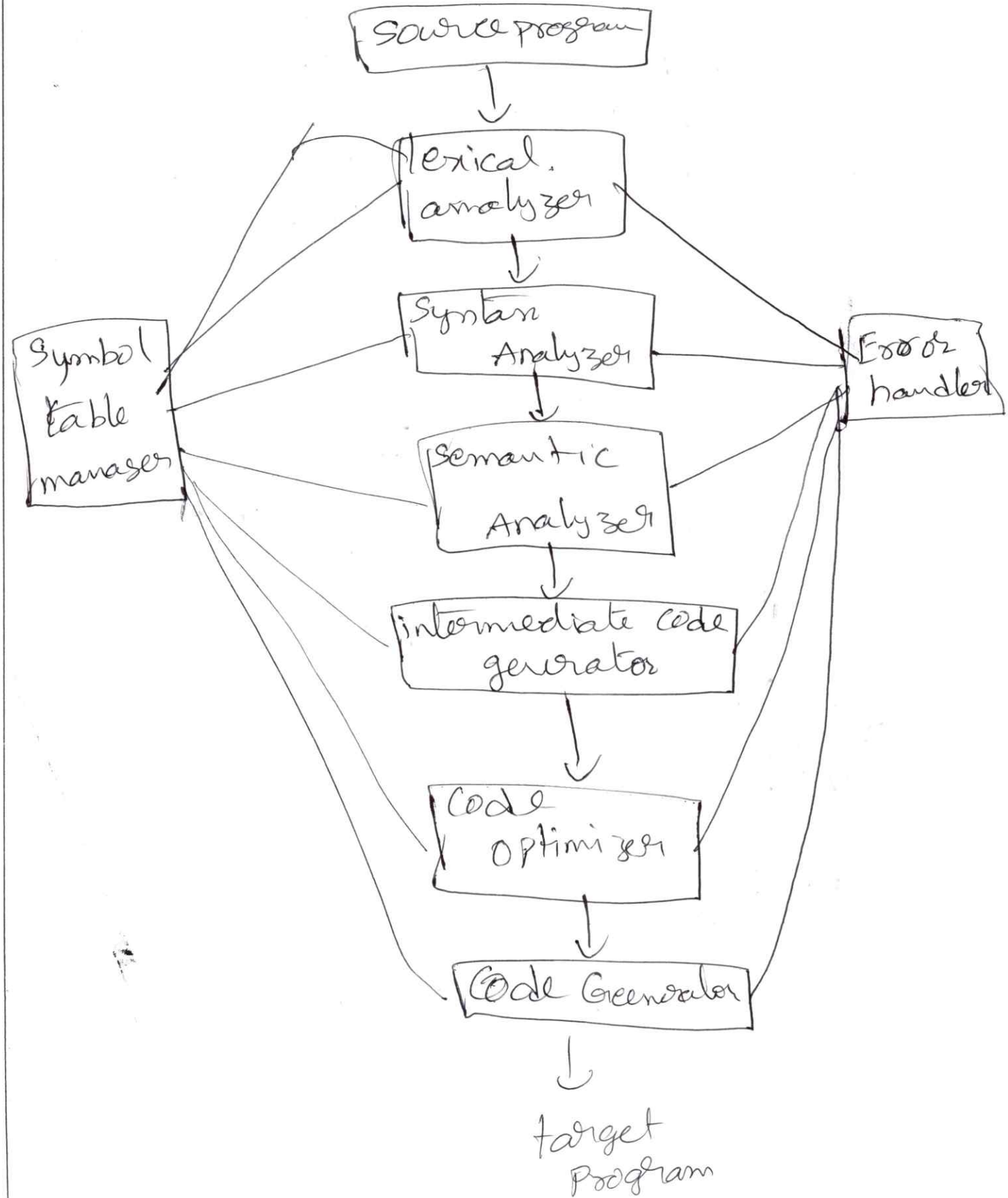
Transition table



Transition diagram

8 Discuss in detail about various phases of Compiler.

Ans)



lexical Analysis: Breaks code into tokens (keywords, identifiers).

syntax Analysis: Checks grammar and builds parse tree.

semantic Analysis: Verifies meaning, like type checks.

Intermediate code generation:
Creates abstract, machine-independent code.

Code optimization - Makes the code run faster / use less memory

Code generation: Produces final machine/target code.

Symbol Table: It is a data structure being used and maintained by the compiler, consisting of all the identifier's name along with their types.

Error handling: This refers to the mechanism in each phase of the compiler to detect, report and recover from error without terminating entire process.

9. Consider the grammar

$E \to E+T$

$E \to T$

$T \to T*F$

$T \to F$

$F \to (E) \mid id$ Construct CLR Parsing table for the above grammar. Give the moves of the CLR Parser on id * id + id.

Ans 1. Augmented Grammar

$E' \to E$

$E \to E+T$

$E \to T$

$T \to T*F$

$T \to F$

$F \to (E) \mid id$

2. LR(0) items (states).

I_0 : $E' \to .E$, $E \to .E+T$, $E \to .T$, $T \to .T*F$, $T \to .F$, $F \to .(E) \mid .id$

I_1 : $E' \to E.$ (Accept).

I_2 : $E \to T.$, $T \to .T*F \mid T \to .F$, $F \to .(E) \mid .id$.

I_3 : $F \to id.$

I_4 : $F \rightarrow (\cdot E)$

I_5 : $F \rightarrow id \cdot$

I_6 : $E \rightarrow E + \cdot T$

I_7 : $T \rightarrow T * \cdot F$, $F \rightarrow \cdot (E)$, $F \rightarrow \cdot id$

I_8 : $T \rightarrow T * F \cdot$

I_9 : $E \rightarrow E + \cdot T$, $T \rightarrow \cdot T * F$, $T \rightarrow \cdot F$, $F \rightarrow \cdot (E) \mid \cdot id$.

3. Find First and Follow sets & Lookaheads.

First (id) = {id}, First (() = {(},

First ()) = {)}, First (E) = {id, (},

First (T) = {id, (}, First (F) = {id, (}.

Follow (E) = {+,), $}

Follow (T) = {+, *,), $}

Follow (F) = {+, *,), $}

Lookaheads:

id for (F → id),

+ (for E → E + T)

* (for T → T * F)

) (for (F → (E))

$ (for E' → E).

4. CLR(1) Table

	Shift							GOTO		
state	id	+	*	(	)	$		E	T	F
0	s_5			s_4				1	2	3
1		s_6				accept				
2		r_3	s_7		r_3	r_3				
3		r_5	r_5		r_5	r_5				
4	s_5			s_4				8	2	3
5		r_7	r_7		r_7	r_7				
6	s_5			s_4					9	3
7	s_5			s_4						10
8		s_6			s_{11}					
9		r_2	s_7		r_2	r_2				
10		r_4	r_4		r_4	r_4				

Step	Stack	Input	Action
1	0	id* id+id$	shift 5
2	05	* id+id$	Reduce by F → id (GOTO(0,F) = 3)
3	03	* id+ id$	Reduce by T → F GOTO (0,T) = 2)
4	02	* id+id$	shift 7
5	027	id+id$	shift 5
6	0275	+id$	Reduce by F → id (GOTO(7,F) = 10)
7	02710	+id$	Reduce by (GOTO (0,T) = 2)
8	02	+id$	Reduce by E → T (GOTO(0,E) = 1)
9	01	+id$	shift 6
10	016	id$	shift 5
11	0165	$	Reduce by F → id (GOTO (6,F) = 3)
12	0163	$	Reduce by T → F (GOTO(6,T) = 9)
13	0169	$	Reduce by E → E+T
14	01	$	Accept.

10

a) Explain heap management mechanism.

b) Explain variats of syntax trees.

Ans) Heap management is the process by which dynamically allocated memory is managed during a program's execution. Unlike the stack, where memory is automatically allocated and deallocated in a last in first out manner, the heap requires explicit mechanisms to allocate memory when requested and deallocate it when it is no longer needed. This ensures efficient use of a finite memory space.

1. Allocation: when a program requests memory in (c malloc, new in c++), the heap manager searches for a sufficiently large, contiguous block of free memory.

2. Deallocation: when the program finishes using a block of memory (using free() or delete), the memory manager marks that block as available. The key challenge here is fragmentation — over time, the heap can become a checkerboard of small, unusable free spaces and occupied blocks.

Garbage collection: In Languages like Java or Python, the heap manager includes an automatic garbage collector that periodically identifies and reclaimes memory that is no longer rechable by the program, relieving the programmer of manual deallocation.

b) Syntax trees are data structures used in compilers to represent the syntactic structure of source code.

Parse Trees

These are full representations of the source code according to the grammar rules. They are typically large and contain every detail, including punctuation and non-terminal symbols from the grammar that have no semantic meaning.

with any Example explain.

Abstract Syntax trees: These are simplified, condensed versions of parse trees. They only represent the essential structural and

semantic information needed for the compiler's later stages. Redundant details like parantheses, semicolons, and certain intermediate production rules are omitted making them more efficient to traverse and manipulate.

11. Explain how to implement 3 address code using Quadruples triples and indirect triples with an Example.

Three address code is an intermediate representation used by optimizing compilers. It breaks down complex expressions into a sequence of simple operations, each involving at most three addresses.

Ex: $a = b * -c + b * d$.

Quadruples

A quadruple is a structure with four fields (op, arg1, arg2, result)

#	OP	Arg1	Arg2	Result
0	Uminus	c		t_1
1	*	b	t_1	t_2
2	*	b	d	t_3
3	+	t_2	t_3	t_4
4	=	t_4		a

Triples:

A triple uses only three fields: (OP, arg1, arg2). Instead of storing results in named temporary variables (t_1, t_2), the index of the triple that computed the value is used as an operand reference. This saves memory by avoiding the need for a symbol table entry for every temporary variable.

#	op	Arg1	Arg2
0	Uminus	c	
1	*	b	0
2	*	b	d
3	+	(1)	2
4	=	a	3

Indirect triples:

Indirect triples use a separate instructio list that stores pointers to the triples in a main table. The triple table itself is never modified during optimization: Instead, optimizations modify the instruction list (the pointer array) by reordering or removing pointers. This makes optimizations cleaner and easier to implement.

- Instruction list

#	Instruction Pointer
0	10
1	11
2	12
3	13
4	14

Triple Table.

Index	op	Arg1	Arg2
(10)	Uminus	c	
(11)	*	b	10
(12)	*	b	d
(13)	+	(11)	(12)
(14)	=	a	(13)