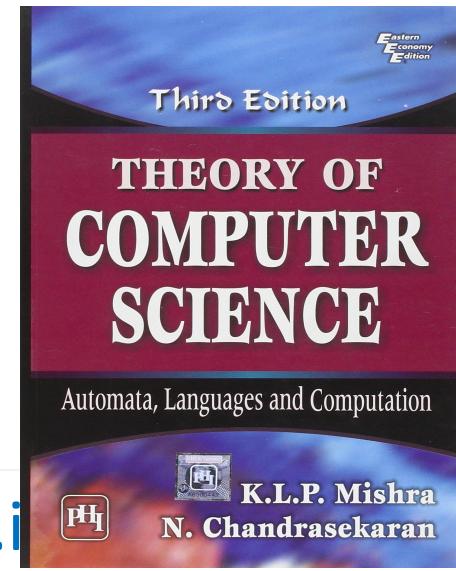
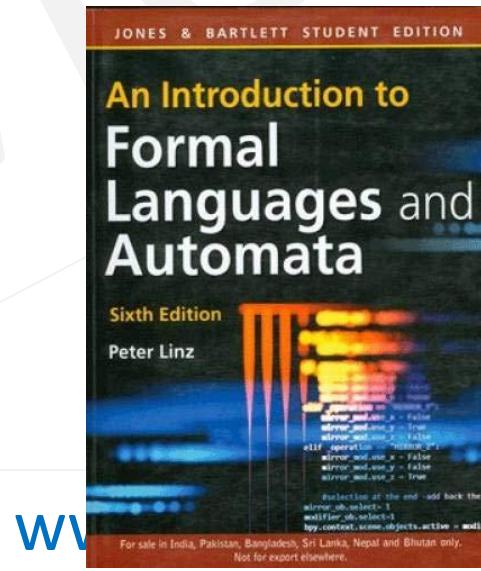
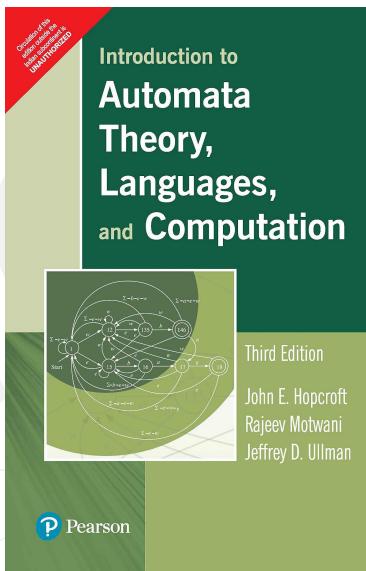


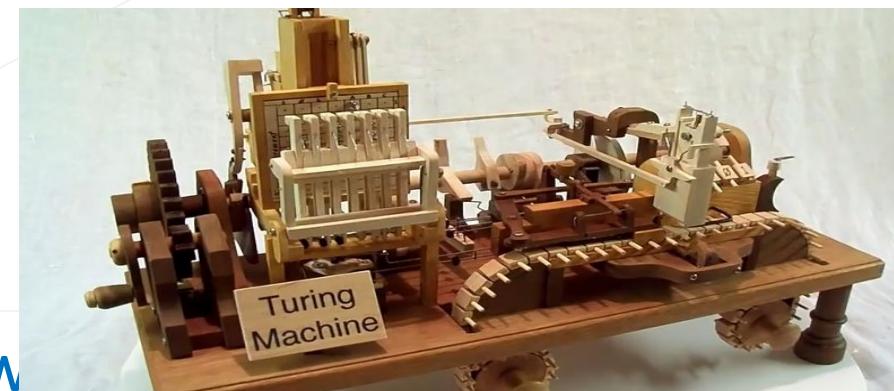
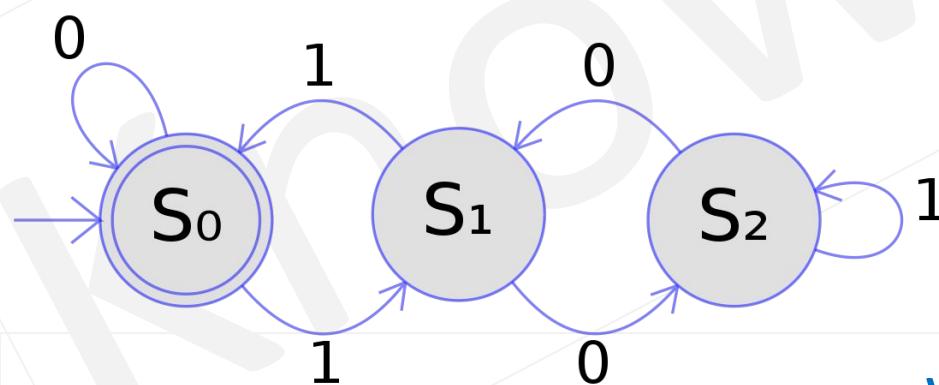
Theory of Computation

- Core subjects for CS/IT Students in university and Competitive Exams.
- In GATE 8-10 Marks out of 100 Marks, and 6-7 questions on an average
- In NET 18-20 Marks out of 200 marks and 7-8 questions
- Both parts are important Theory and Numerical
- Needs less time, good scoring
- Not Applied in Industry
- **Syllabus:** Regular expressions and finite automata. Context-free grammars and push-down automata. Regular and context - free languages. Pumping lemma. Turing machines and undecidability.



INTRODUCTION TO THEORY OF COMPUTATIONS

- **Theory of Computation (TOC)** is a branch of theoretical computer science that addresses:
- **What can be computed, How efficiently can it be computed, What limitations exist in computation?**
- It involves the study of abstract machines, also known as **automata**, and mathematical models that define what problems can be solved using these machines.
- TOC focuses on different computational models, their capabilities, and the complexity of solving problems. The main areas it deals with include:
 - **Automata Theory**: Studies different types of automata (e.g., finite automata, pushdown automata) and their abilities to solve problems.
 - **Formal Languages**: Explores the classification of languages and grammars that machines can recognize.
 - **Computability Theory**: Investigates which problems can or cannot be solved by any computational model.
 - **Complexity Theory**: Considers the resources (time, space) needed to solve problems and categorizes them based on difficulty (e.g., P, NP).



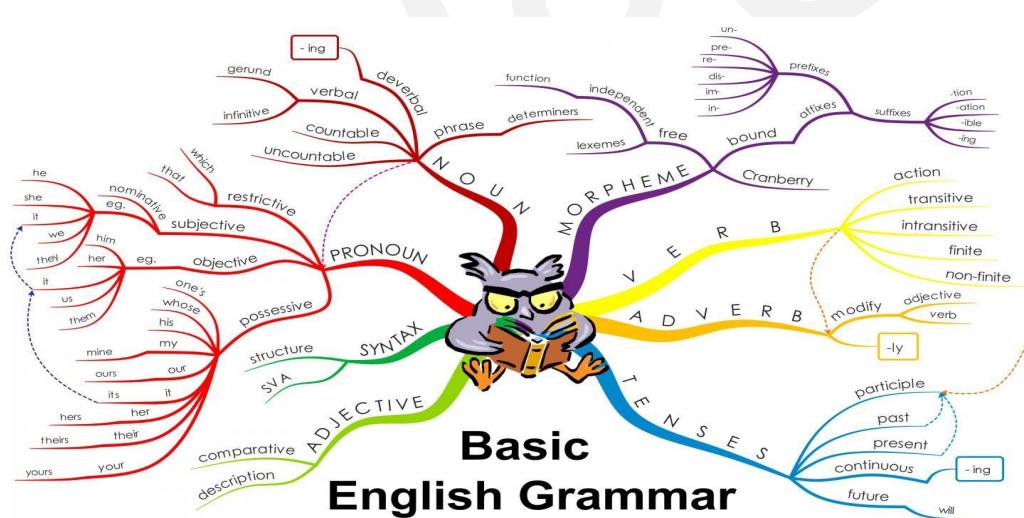
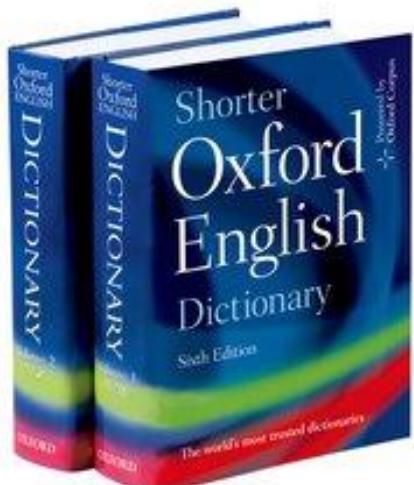
PROBLEM

- **Machine Communication:**
 - Machines play a crucial role in human development, and we need a language to communicate with them. However, this language need not be as complex as natural languages.
- **Formal vs. Informal Languages:**
 - Machines use **formal languages**, which are structured and concise, unlike **informal languages** used in daily human interaction.
- **Language Definition:**
 - A language is "a system suitable for expressing ideas, facts, or concepts" using a set of symbols.



METHODS TO DEFINE LANGUAGE

- In natural language, we can list a finite and predefined set of words in a dictionary, but the sentences formed using these words are infinite. To determine which sentences are valid or invalid, we use **grammar** or rules as a mechanism.



MATHEMATICL DEFINATION OF LANGUAGE

- **SYMBOL**- Symbols are the basic building blocks, which can be any character/token. (cow, sheep, , white flag, , ,  etc.) (in English we called them as letters).
- **ALPHABET**- An alphabet is a finite non empty set of symbols, (every language has its own alphabet). here in toc, we use symbol Σ for depicting alphabet. e.g. $\Sigma = \{0,1\}$. for English $\Sigma = \{a, b, c, \dots, z\}$ (in English also alphabet is a set of letters, thought in general we called them as alphabet).
- **STRING** - It is a finite sequence of symbols (which are the member of set alphabet). E.g. $\Sigma = \{a, b\}$ String- aabb, aa, b, so on. (in English we called them as words).
- **LANGUAGE** - A language is defined as a set of strings. In the next level we consider programs as a string and programming constructs/tokens like int, floats as letters/symbols.

- Similarly, in our system we have finite number of symbols/letters but using those letters we can generate infinite strings/words.
- So, we may have languages that have infinite number of words, so it is not possible for us to list them, we have to use some framework, which can somehow represent the same language. There are mainly two methods to represent a language
 - by a grammar that generates a language [RG generate RL]
 - by a machine that accepts a language [FA accept RL language]

Machine

Grammar

SOME BASIC OPERATIONS ON STRINGS

- So, before we proceed further let's do a little home work on string, which will help us throughout the subject.
- **Length of a string** - It is defined as number symbol in the string. Denoted like $|W|$, e.g. length of string $|00110| = 5$.
- $|aaba| =$
- $|010| =$

Concatenation of string- Let x and y be two strings, then concatenation is defined as the string formed by making a copy of string x followed by a copy of string y .
(NOTE- It's not commutative)

E.g. $w = ab$, $x = ba$

$wx =$

$xw =$

$w = w_1 w_2 w_3 \dots \dots w_m$

$x = x_1 x_2 x_3 \dots \dots x_n$

$wx =$

$|wx| = |w| + |x|$

Reverse of a string – If there is a string w then reverse of a string a denoted by w^r it is just the same string but written in reverse order.

$$W = w_1 \ w_2 \ w_3 \ \dots \ \dots \ w_n$$

$$w^r =$$

$$|w| \boxed{} |w^r|$$

Empty/Null String- The string with zero occurrence of symbols. It is denoted by \in , $|\in|=0$. If there is a string w , then w^n stands for the strings obtained by repeating w , n times.

$$w^3 = www$$

$$w^2 = ww$$

$$w^1 = w$$

$$w^0 =$$

$$w\in = \in w =$$

Substring- Any string of consecutive symbols in some string ‘w’ can be collectively said as a substring. E.g. w= abab its substrings can be ab, a, ba...etc.

- | | | | | | | | | | |
|----|-----|-----|-----|-----|-----|---|---|---|---|
| | S | u | b | s | t | r | i | n | g |
| 1. | utg | | | | | | | | |
| 2. | | sbr | | | | | | | |
| 3. | | | rts | | | | | | |
| 4. | | | | str | | | | | |
| 5. | | | | | sub | | | | |
| 6. | | | | | | ∈ | | | |

Q Consider a sting 'GATE' find the total number of substring possible?

	Substring of length 0	Substring of length 1	Substring of length 2	Substring of length 3	Substring of length 4
Number of sting possible	1	4	3	2	1
SUBSTRINGS	\in	G, A, T, E	GA, AT, TE	GAT, ATE	GATE

Q Consider a sting 'GGGE' find the total number of substring possible?

	Substring of length 0	Substring of length 1	Substring of length 2	Substring of length 3	Substring of length 4
Number of sting possible					
SUBSTRINGS					

If a string has 'n' distinct symbols then total number of different sub string will be $[n(n+1)/2] + 1$. If w is any string than empty string \in and the string w itself is called a trivial substring and the remaining of the other are the non-trivial sub string $[(n(n+1)/2)-1]$

Q Consider a sting 'GATE' find the total number of prefix and suffix possible?

Prefix(GATE) =

Suffix(GATE) =

If there is a string of length n then no of prefix or suffix will be $n+1$

Q || $\Sigma = \{a, b\}$ then, find the following?

$$\Sigma^0 =$$

$$\Sigma^1 =$$

$$\Sigma^2 =$$

$$\Sigma^3 =$$

- Σ^K is the set of all the strings from the alphabet Σ of length exactly K.
- $\Sigma^k = \{w \mid |w| = k\}$ (using the symbols from the alphabet Σ)

Kleene closure- If Σ is a set of symbols, then we use Σ^* to denote the set of strings obtained by concatenating zero or more symbols from Σ of any length, in general any string of any length which can have only symbols specified in Σ .

- $\Sigma^* = \bigcup_{i=0}^{i=\infty} \{w \mid |w| = i\}$ (using the symbols from the alphabet Σ)

Positive closure – If Σ is a set of symbols, then we use Σ^+ to denote the set of strings obtained by concatenating one or more symbols from Σ of any length, in general any string of any length which can have only symbols specified in Σ (except ϵ).

- $\Sigma^+ = \bigcup_{i=1}^{i=\infty} \{w \mid |w| = i\}$ (using the symbols from the alphabet Σ)

LANGUAGES

- Since languages are sets, the union, intersection and difference of two languages are immediately defined.
- The complement of a language is defined with respect of Σ^*

- $L^c =$

- The reverse of a language is the set of all the strings after reversal

- $L^r =$

- The concatenation of two languages L_1 and L_2 is the set of all the strings obtained by concatenating any elements of L_1 and L_2 .

- $L_1 L_2 =$

- Let L^n is defined as L is concatenated with itself n times.

- $L^n =$

- $L^* =$

Q Given the language $L = \{ab, aa, baa\}$, which of the following strings are in L^* ? (GATE-2012) (1 Marks)

1) abaabaaaabaa

2) aaaabaaaaa

3) baaaaabaaaab

4) baaaaabaa

- (A) 1, 2 and 3**
- (B) 2, 3 and 4**
- (C) 1, 2 and 4**
- (D) 1, 3 and 4**

Q In a string of length n , how many proper prefixes can be generated

- a) 2^n
- b) n
- c) $n(n + 1)/2$
- d) $n - 1$

Q The number of substrings (of all lengths inclusive) that can be formed from a character string of length n is

a) $n!$

b) n^2

c) $\frac{n(n-1)}{2} - 1$

d) $\frac{n(n+1)}{2} + 1$

Q Suppose $L_1 = \{10, 1\}$ and $L_2 = \{011, 11\}$. How many elements are there in $L = L_1 L_2$

- a) 4
- b) 3
- c) 2
- d) None of these

Q Let x_n denote the number of binary strings of length n that contain no consecutive 0's, which of the following recurrences does x_n satisfy?

a) $x_n = 2x_{n-1}$

b) $x_n = x_{\left[\frac{n}{2}\right]} + 1$

c) $x_n = x_{\left[\frac{n}{2}\right]} + n$

d) $x_n = x_{n-1} + x_{n-2}$

Q The value of x_5 is?

a) 5

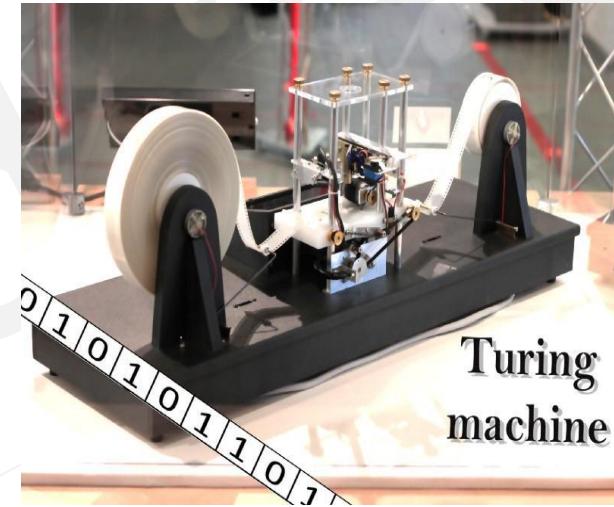
b) 7

c) 13

d) 16

Automaton

- An automaton is a self-operating system that transforms, transmits, and utilizes energy, materials, and information to perform functions without human involvement. The term often refers to a machine that follows predefined rules to execute complex tasks autonomously. Examples include automatic machine tools, packing machines, and photo printing devices.



FINITE AUTOMATA

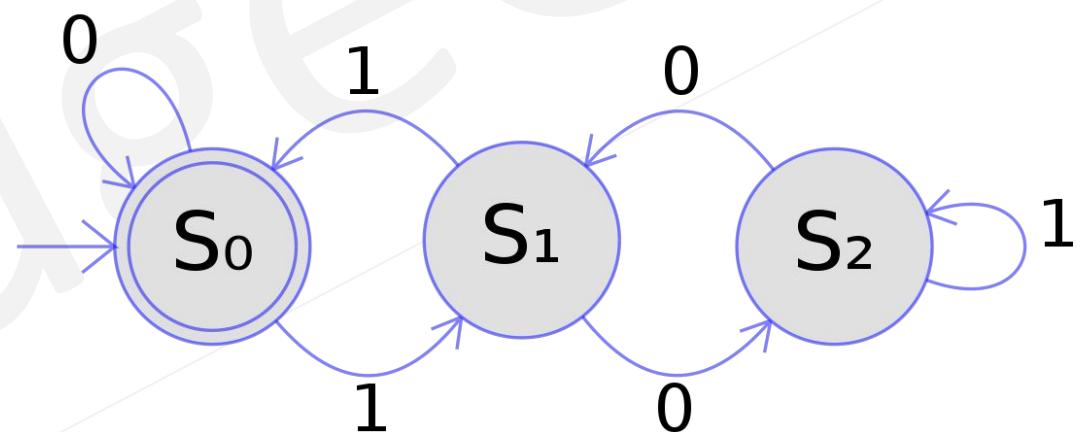
- A finite automaton is a computational model with a limited set of states, where control shifts between states based on external inputs. It is commonly used as an abstract machine in computer science for recognizing patterns in symbol strings. Finite automata have applications in pattern matching, lexical analysis, parsing, and more. Finite automata can be broadly classified into two types-

1. Finite automata without output

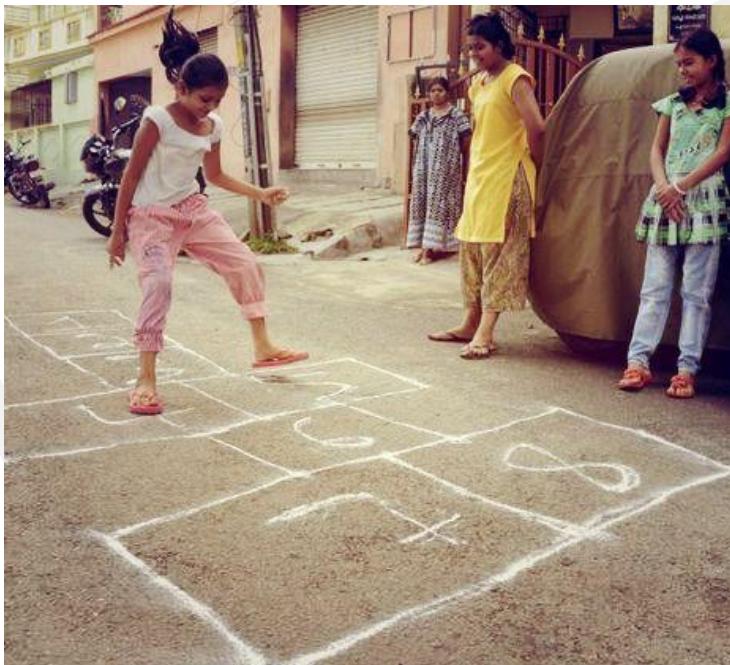
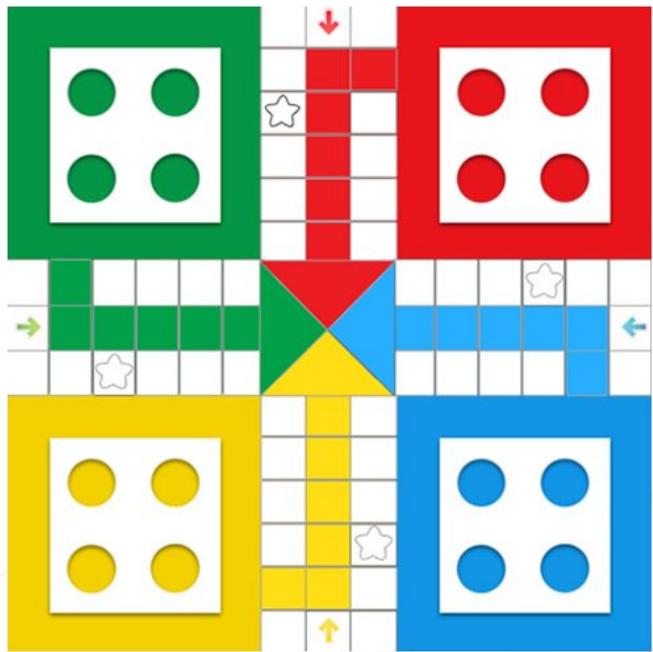
1. Deterministic finite automata.
2. Non deterministic finite automata.
3. Non deterministic finite automata with \in

2. Finite automata with output

1. Moore machine
2. Mealy machine



- A finite automaton is limited in its ability to store information during computation because it lacks temporary storage. The machine can only store a finite amount of information by switching between a limited number of states. As a result, it can only handle scenarios where the amount of information to be stored is strictly bounded.



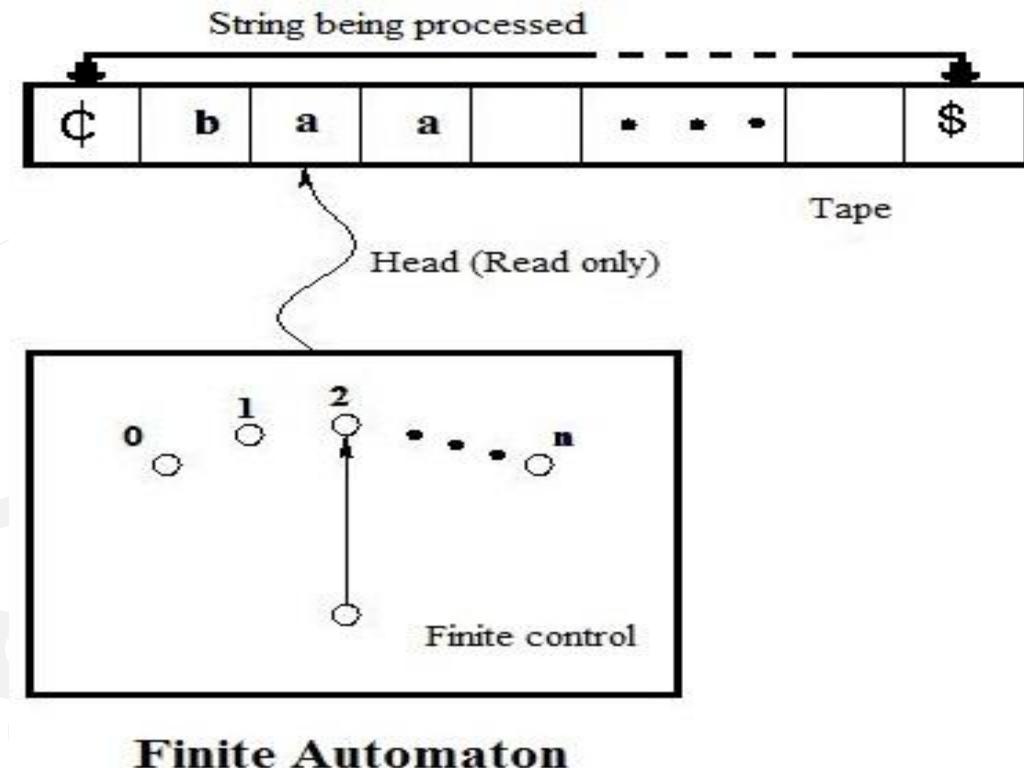
DETERMINISTIC FINITE AUTOMATA

A **deterministic finite automaton (DFA)** is defined by 5-tuple $(Q, \Sigma, \delta, S, F)$ where:

- Q is a finite and non-empty set of states
- Σ is a finite non-empty set of finite input alphabet
- δ is a transition function, ($\delta: Q \times \Sigma \rightarrow Q$)
- S is **initial state** (always one) ($S \in Q$)
- F is a set of final states ($F \subseteq Q$) ($0 \leq |F| \leq N$, where n is the number of states)

A deterministic finite automaton (DFA) operates through various components:

- **Input Tape:** Divided into squares, each containing a symbol from the input alphabet. It has end markers (¢ at the left and \$ at the right) or can be infinite. The input string lies between these markers.
- **Reading Head:** This head reads one square at a time and moves to the right.
- **Finite Control:** Acts as the inference engine and manages transitions between states.
- A DFA ensures a unique computation for each input string, making it deterministic. DFAs are commonly used in tasks like lexical analysis (spell checking) in compiler design.



Representation

TRANSITION STATE DIAGRAM- A graphical representation where each circle represents a state, and directed edges show transitions between states. The initial state has an arrow pointing to it, and the final state is represented by two concentric circles.

δ	Σ	
Q	a	b
	q_0	
	q_1	

TRANSITION TABLE- A two-dimensional table where rows represent states, and columns represent input symbols. The table shows how the machine transitions between states based on inputs.

TRANSITION ID- $\delta \{q_i, a\} = q_j$, This notation indicates that when the machine is in state q_i and reads input symbol a , it transitions to state q_j .

ACCEPTANCE BY DFA

- Let 'w' be any string designed from the alphabet Σ , corresponding to w, if there exist a transition for which it starts at the initial state and ends in any One of the final states, then the string 'w' is said to be accepted by the finite automata. $\delta^*(q_0, w) = q_f$ for some $q_f \in F$.
- Mathematically, it can be represented as: - $L(M) = \{w \in \Sigma^* \mid \delta^*(S, w) \in F\}$

Q. Consider the following table of an FA
If the final state is q₄, to which of the following string is accepted?

1. aaaaa
 2. aabbaabbhhh
 3. bbabababbb
-
- a) 1 & 2
 - b) 2 & 3
 - c) 3 & 1
 - d) all of the above

δ	a	b
start	q ₁	q ₀
q ₀	q ₁	q ₀
q ₁	q ₂	q ₁
q ₂	q ₃	q ₂
q ₃	q ₄	q ₃
q ₄	q ₄	q ₄

Q In the automaton below, s is the start state and t are the only final state. (GATE-2006) (2 Marks)

Consider the strings $u = ababab$, $v = bab$, and $w = aabb$.

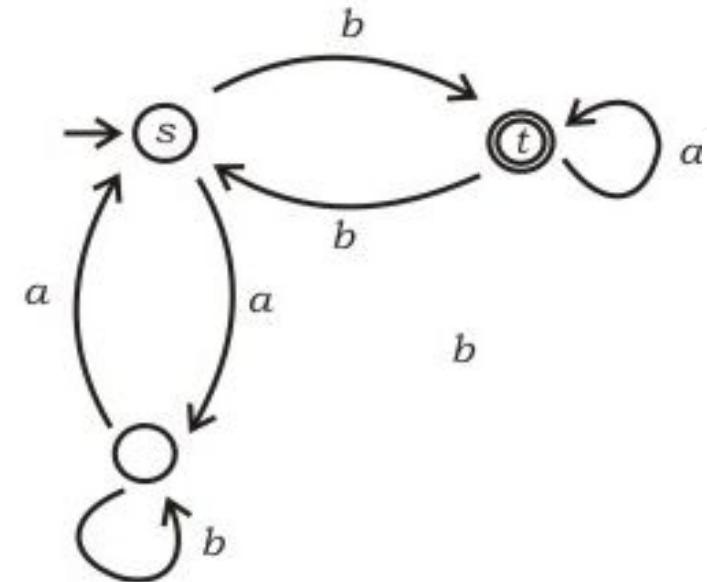
Which of the following statements is true?

(A) The automaton accepts u and v but not w

(B) The automaton accepts each of u, v, and w

(C) The automaton rejects each of u, v, and w

(D) The automaton accepts u but rejects v and w



- A language is considered a regular language if it can be accepted by a Deterministic Finite Automaton (DFA). The best way to understand DFAs and regular languages is by designing multiple DFAs, which helps in grasping both the DFA design process and the concept of regular languages.

Q Design a minimal DFA that accepts a language ‘L’, where $L=\{a\}$ over the alphabet $\Sigma=\{a\}$.

Q design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, where every accepted string 'w' starts with substring s

i) $s = b$

ii) $s = ab$

iii) $s = abb$

Conclusion

- if $w = sx$, $|s| = m$, then no of states in the DFA is $m+2$
- Dead state is required
- Loop on final state

Q design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$. where every accepted string 'w' ends with substring 's'.

i) $s = ab$

ii) $s = aa$

iii) $s = bab$

Conclusion

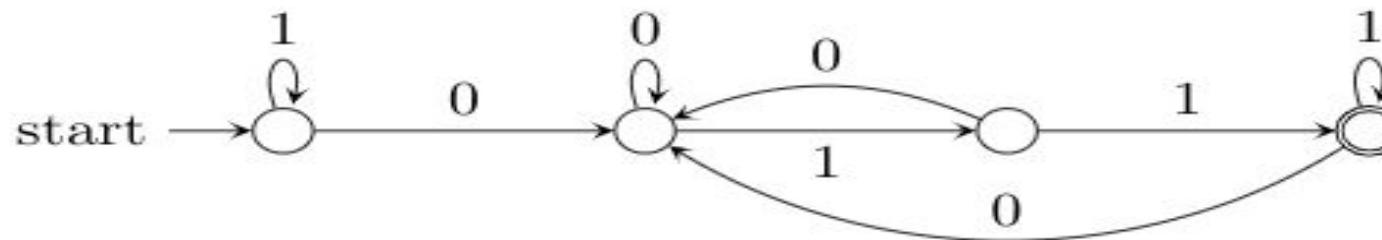
- if $w = xs$, $|s| = m$, then no of states in the DFA is $m+1$
- No need of dead state
- Can not stay of final state for ever

Q Consider the following language:

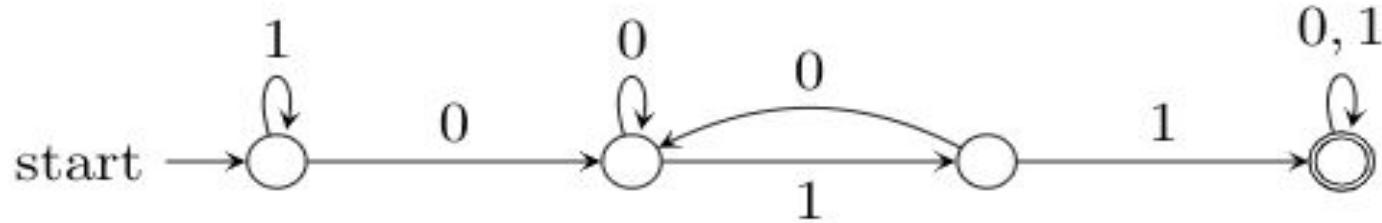
$$L = \{ w \in \{0,1\}^* \mid w \text{ ends with the substring } 011 \}$$

Which one of the following deterministic finite automata accepts L? **(GATE 2021) (2 MARKS)**

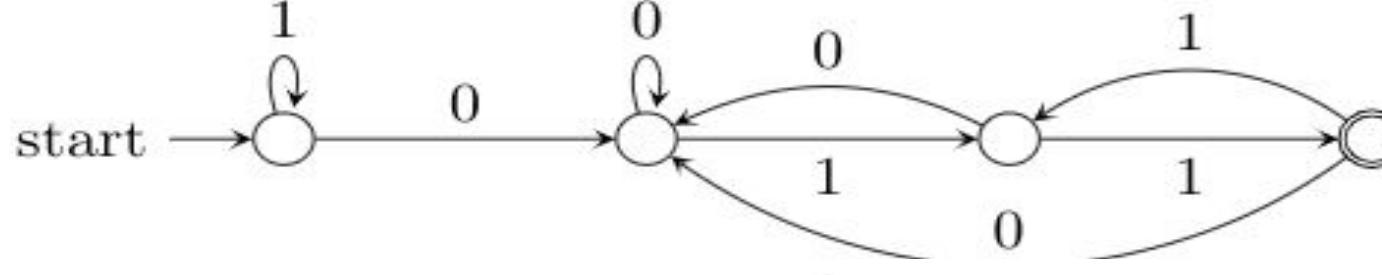
(A)



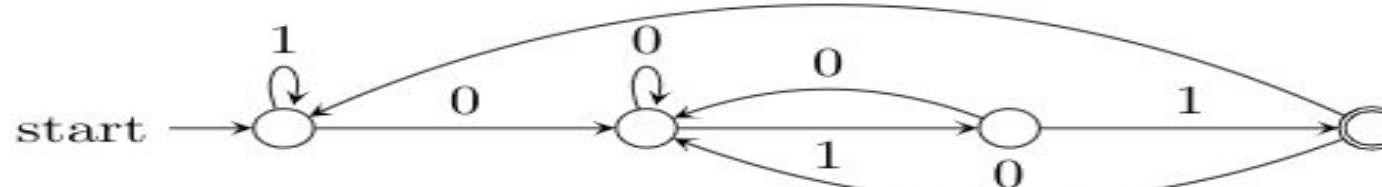
(B)



(C)



(D)



Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$. where every accepted string 'w' contains sub string s.

i) abb

ii) aba

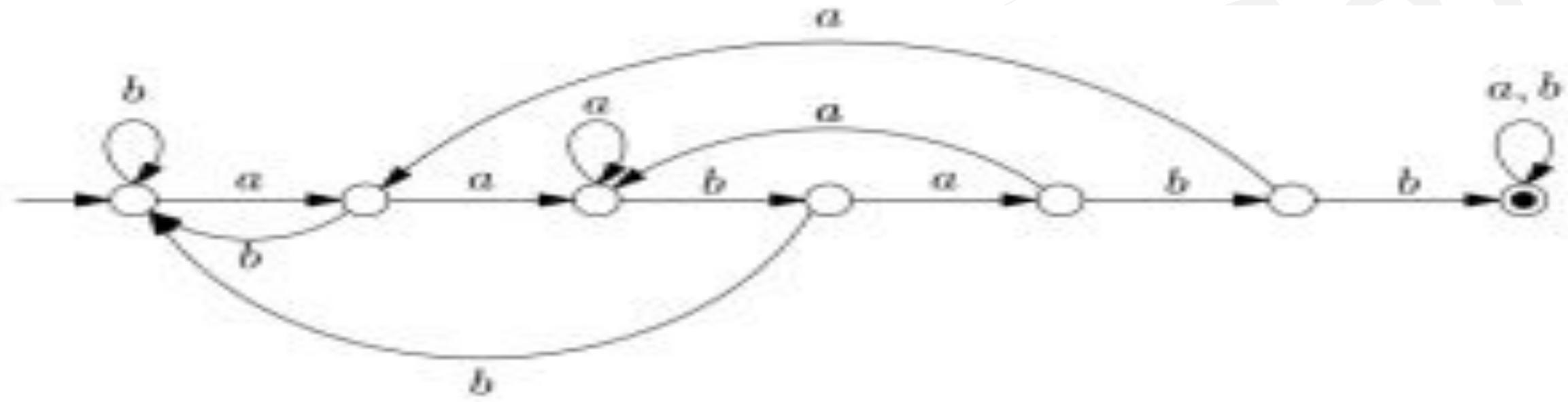
Conclusion

- if $w = xsx$, $|s| = m$, then no of states in the DFA is $m+1$
- No need of dead state
- Can loop on final state

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with a.

Q Consider the following Deterministic Finite Automata (GATE-2015) (2 Marks)

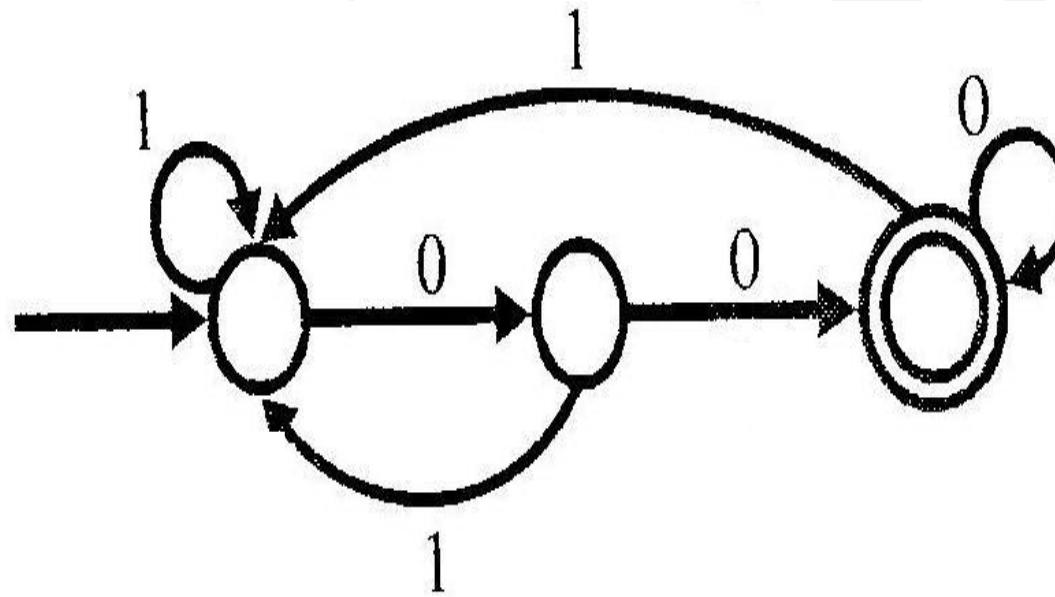
Which of the following is true?



- (A) It accepts all strings with prefix as “aababb”
- (B) It accepts all strings with substring as “aababb”
- (C) It accepts all strings with suffix as “aababb”
- (D) None of the above

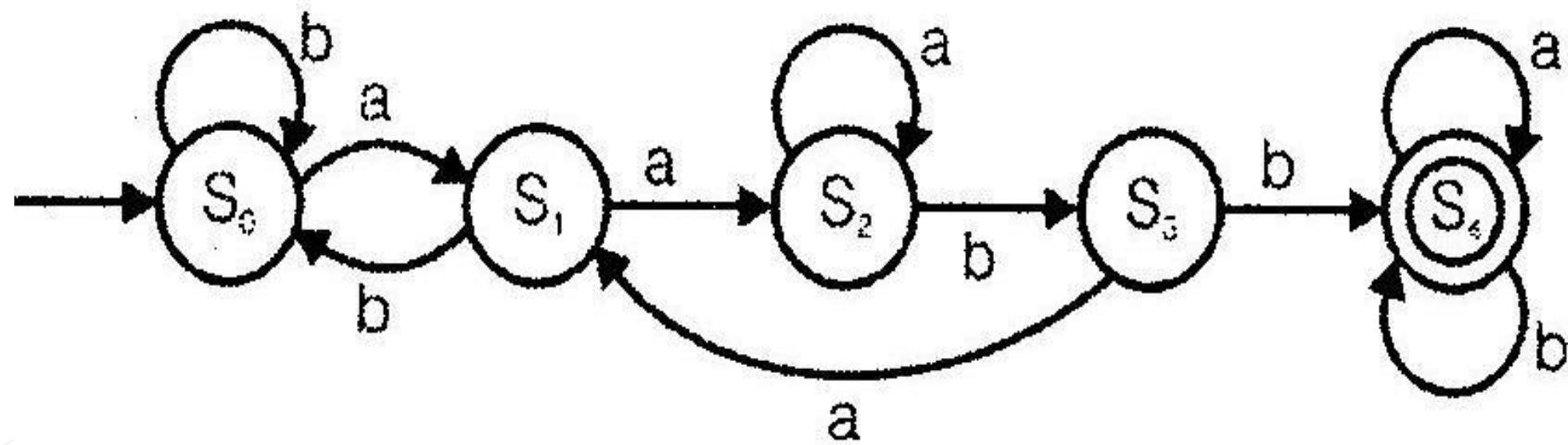
Q The following DFA accepts the set of all strings over {0, 1} that
(GATE-2009) (1 Marks)

- a) Begin either with 0 or 1
- b) End with 0
- c) End with 00
- d) Contain the substring 00



Q Consider the following machine M

What is the language $L(M)$ accepted by this machine?



- a) $L(M) = \{\text{Set of all words starting with aabb}\}$
- b) $L(M) = \{\text{Set of all words having aabb as a sub word}\}$
- c) $L(M) = \{\text{Set of all words ending with aabb}\}$
- d) $L(M) = \{\text{Set of all words with exactly one occurrence of aabb}\}$

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with same symbol.

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with different symbol.

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w , is like $w=SX$.

i) $s= aa/bb$

ii) $s=aaa/bbb$

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w , is like $w=XS$.

i) $s= aa/bb$

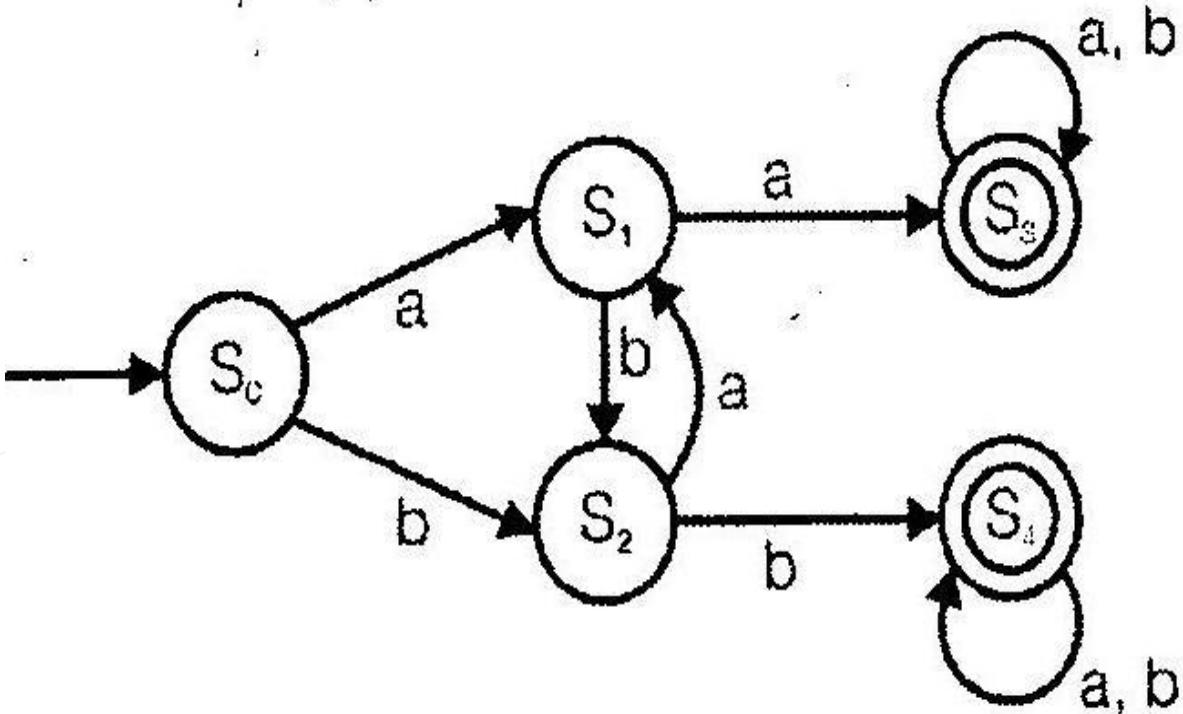
ii) $s=aaa/bbb$

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w , is like $w=XSX$.

i) $s= aa/bb$

ii) $s=aaa/bbb$

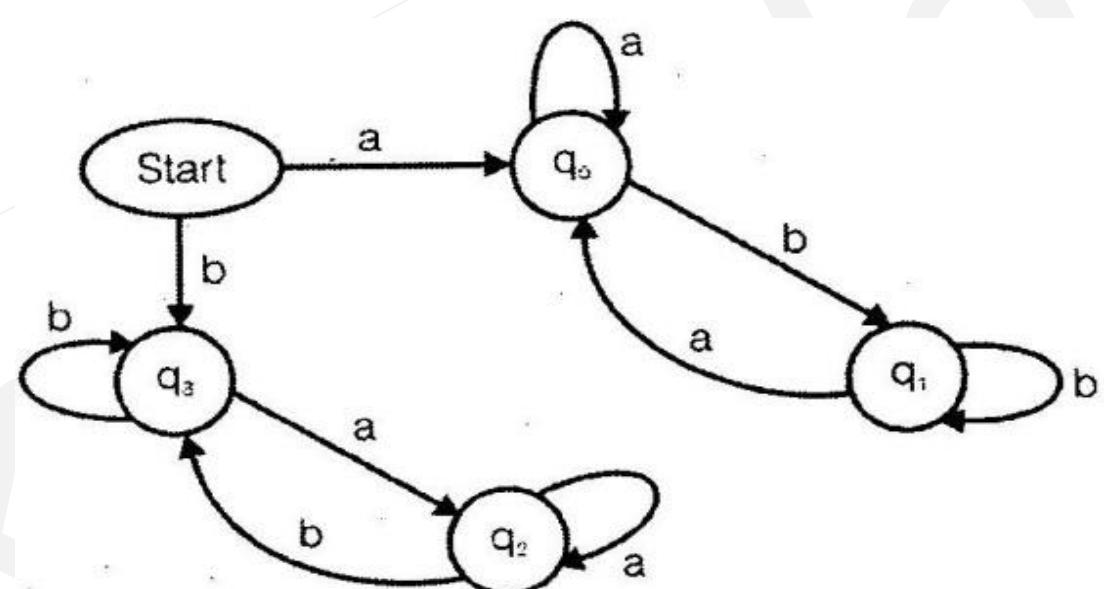
Q Consider the machine M shown below $L(M)$?



- a) $L(M) = \{\text{words starting with aa or bb}\}$
- b) $L(M) = \{\text{words ending with aa or bb}\}$
- c) $L(M) = \{\text{words containing aa or bb as a sub word}\}$
- d) None of these

Q Let L be the set of all binary strings who's last two symbols are the same. The number of states in the minimum state deterministic finite state automation accepting L is **(GATE-1998) (1 Marks)**

- a) 2
- b) 5
- c) 8
- d) 3



Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' accepted must be like

i) $|w| = 3$

ii) $|w| \leq 3$

iii) $|w| \geq 3$

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain exactly two a's, $|w|_a = 2$.

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain at least two a's,
 $|w|_a \geq 2$

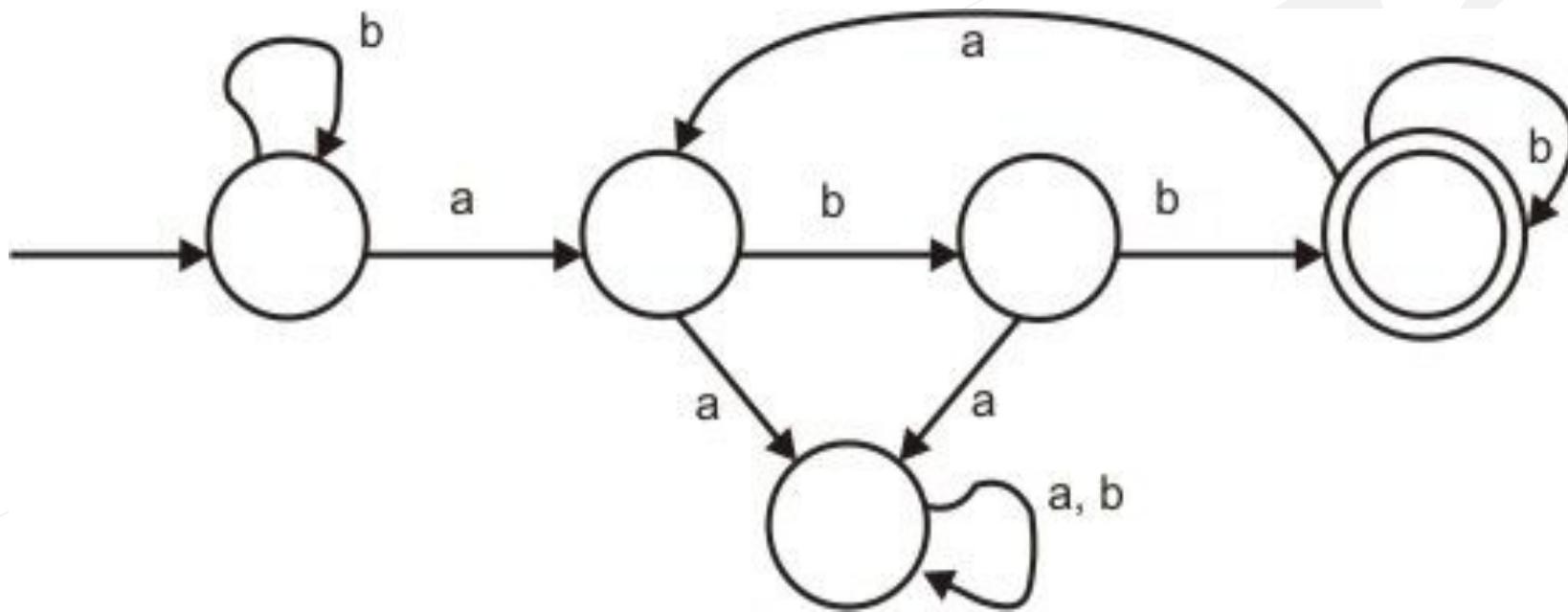
Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain at most two a's,
 $|w|_a \leq 2$

Q The minimum possible number of states of a deterministic finite automaton that accepts a regular language L is _____ (GATE-2017) (2 Marks)

$$L = \{w_1aw_2 \mid w_1, w_2 \in \{a, b\}^*, |w_1| = 2, |w_2| \geq 3\}$$

Q Consider the machine M (GATE-2005) (2 Marks)

The language recognized by M is :



- (A) $\{w \in \{a, b\}^* / \text{every } a \text{ in } w \text{ is followed by exactly two } b's\}$
- (B) $\{w \in \{a, b\}^* \text{ every } a \text{ in } w \text{ is followed by at least two } b'\}$
- (C) $\{w \in \{a, b\}^* w \text{ contains the substring 'abb'}\}$
- (D) $\{w \in \{a, b\}^* w \text{ does not contain 'aa' as a substring}\}$

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' accepted must be like

i) $|w| = 0(\text{mod } 3)$

ii) $|w| = 1(\text{mod } 4)$

Conclusion

No of states will be n, if the equation is on format $r \pmod{n}$

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain

- i) number of a is, $|w|_a = 0(\text{mod } 2)$
- ii) number of b is, $|w|_b = 1(\text{mod } 2)$

- iii) number of a is $|w|_a = 2(\text{mod } 3)$
- iv) number of a is $|w|_a = 1(\text{mod } 4)$

Q How many minimum states are required in a DFA to find whether a given binary string has odd number of 0's, there can be any number of 1's. (GATE-2015) (1 Marks)

- (A) 1 (B) 2 (C) 3 (D) 4

Q The smallest finite automation which accepts the language { $x \mid \text{length of } x \text{ is divisible by 3}\}$ has (GATE-2002) (2 Marks)

- (A) 2 states (B) 3 states (C) 4 states (D) 5 states

Q Let $\Sigma = \{0, 1\}$, then an automaton A accepting only those words from Σ having an odd number of 1's requires _____ states including the start state

- a) 2
- b) 3
- c) 4
- d) 5

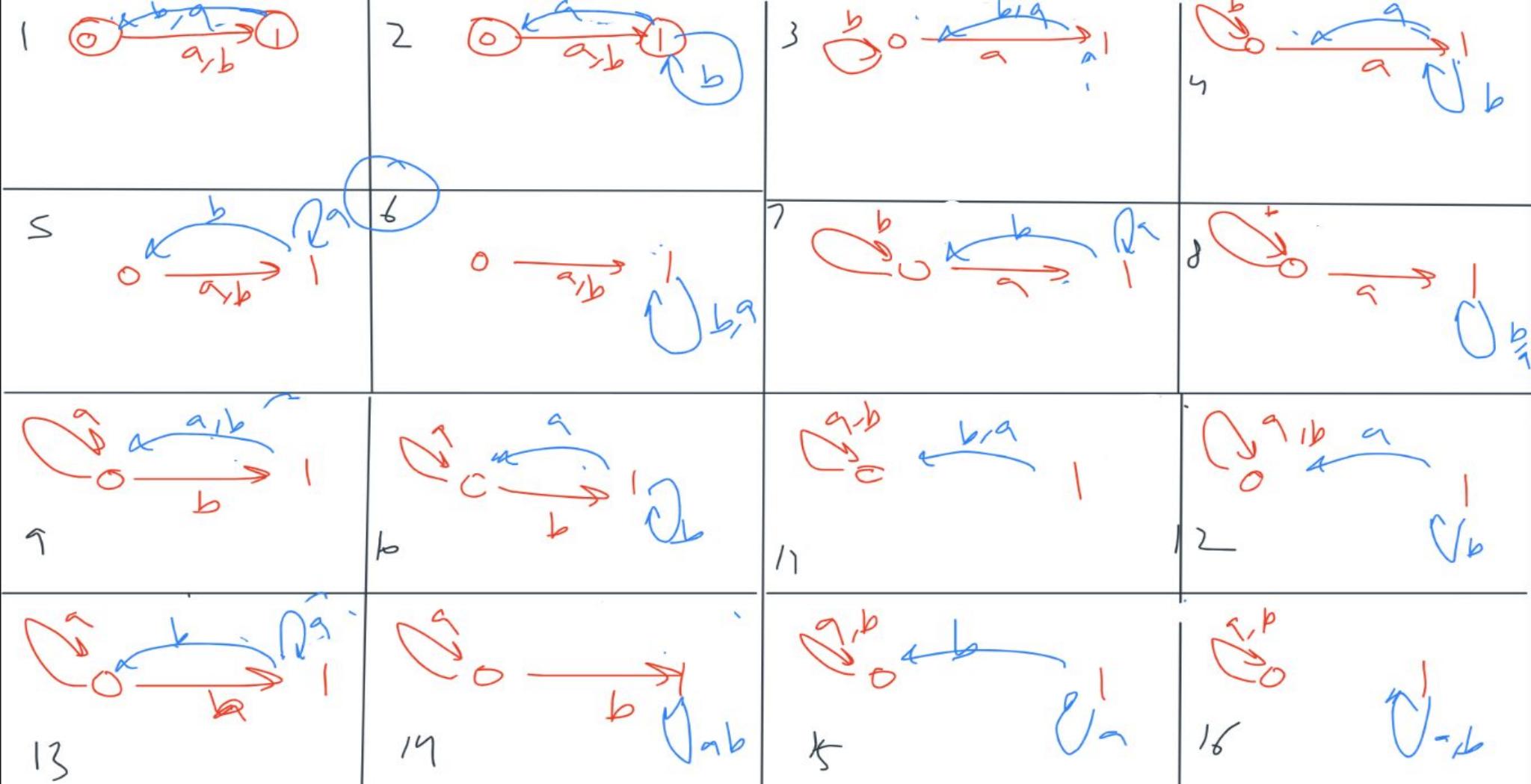
Q The smallest finite automaton which accepts the language {x | length of x is divisible by 500} has

- a) 200 states
- b) 300 states
- c) 400 states
- d) 500 states

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain odd occurrence of the substring 'ab'.

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain even occurrence of the substring ‘baa’.

Q how many DFA can be designed, with a fixed initial state, over an alphabet $\Sigma = \{a, b\}$, and the number of states be two.



- If no of states is $|Q| = n$, and no of input alphabet $|\Sigma| = m$
- Then no of states will be $n^{n*m} * 2^n$

Q how many DFA can be constructed over the alphabet, such that no of states $|Q| = 2$ and size of input alphabet is $|\Sigma| = 2$, that accepts empty language(Φ)?

Q how many DFA can be constructed over the alphabet, such that no of states $|Q| = 2$ and size of input alphabet is $|\Sigma| = 2$, that accepts universal language (Σ^*)?

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that for every accepted string 2nd from left end is always b.

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that for every accepted string 2nd from right end is always b.

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' which is accepted starts with 'a' & length is divisible by 3, i.e. $0(\text{mod } 3)$?

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' which is accepted starts with 'ab' & length is divisible by $2(\text{mod } 4)$?

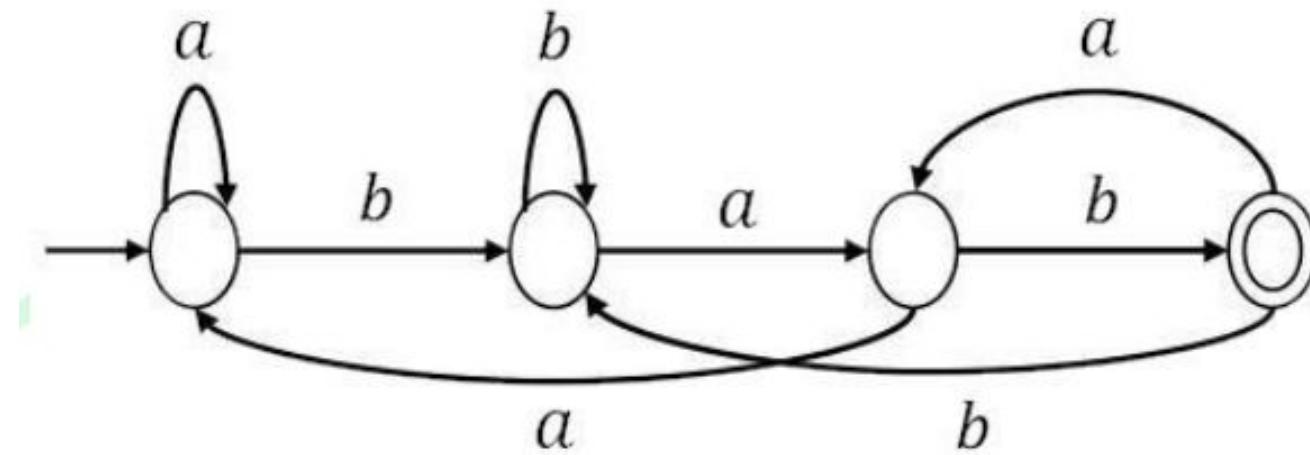
- No of states will be n, $n = \text{first condition} + \text{second condition} - 1$

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain be like,
no of $a = 0(\text{mod } 2)$ || no of $b = 0(\text{mod } 2)$?

cross product method(one more approach , where concentrate on even and odd logic)

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain be like, no of $a = 0(\text{mod } 2)$ & no of $b = 0(\text{mod } 2)$?

Q. Consider the following deterministic finite automaton (DFA) defined over the alphabet, $\Sigma = \{a, b\}$. Identify which of the following language(s) is/are accepted by the given DFA. (Gate 2025)



- A) The set of all strings containing an even number of b 's.
- B) The set of all strings containing the pattern bab .
- C) The set of all strings ending with the pattern bab .
- D) The set of all strings not containing the pattern aba .

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b, c\}$, such that every string accepted must contain be like, $a = 0(\text{mod } 2) \& \& \text{ no of } b = 0(\text{mod } 3) \& \& \text{ no of } c = 0(\text{mod } 5)$?

- if, $n(\text{mod } m) \& \& p(\text{mod } r)$, then the no of states will be $m * r$, if m & r are relatively prime to each other www.knowledgategate.in

Q A minimum state deterministic finite automaton accepting the language $L = \{w \mid w \in \{0,1\}^*, \text{ number of } 0^s \text{ and } 1^s \text{ in } w \text{ are divisible by } 3 \text{ and } 5, \text{ respectively}\}$ has
(GATE-2007) (2 Marks)

- (A)** 15 states **(B)** 11 states **(C)** 10 states **(D)** 9 states

Q Consider a DFA over $\Sigma = \{a, b\}$ accepting all strings which have number of a's divisible by 6 and number of b's divisible by 8. What is the minimum number of states that the DFA will have? **(GATE-2001) (2 Marks)**

- (A) 8 (B) 14 (C) 15 (D) 48

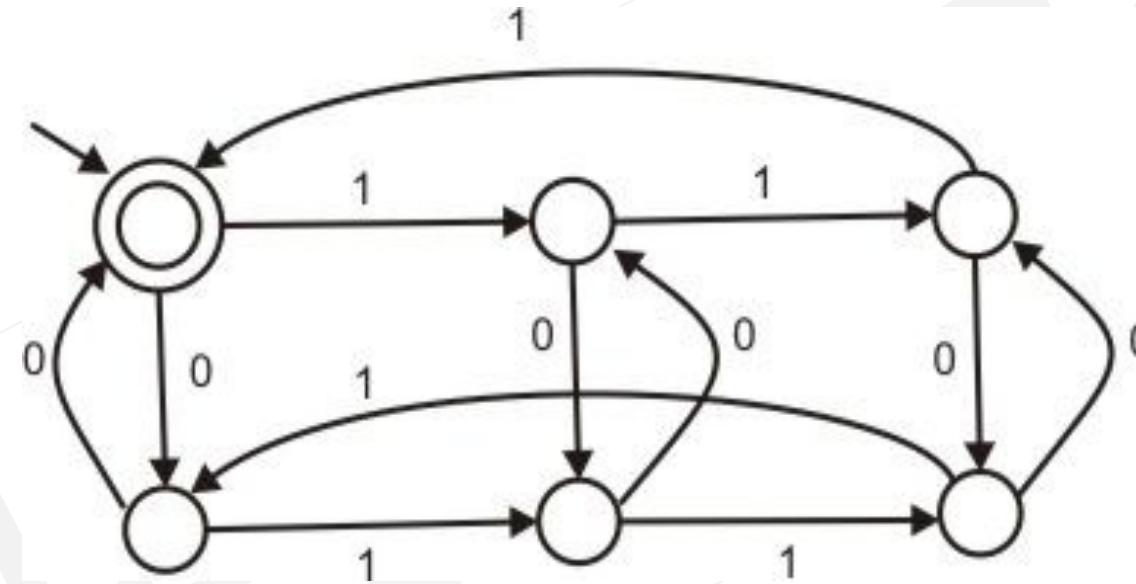
Q The following finite state machine accepts all those binary strings in which the number of 1's and 0's is respectively. **(GATE-2004) (2 Marks)**

(A) divisible by 3 and 2

(B) odd and even

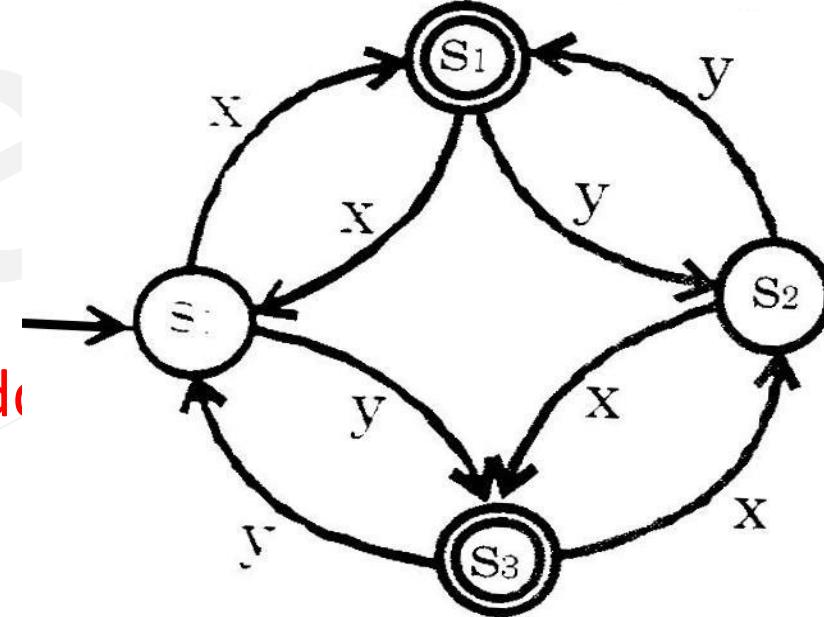
(C) even and odd

(D) divisible by 2 and 3



Q. Consider the following DFA in which S_0 is the start and S_1, S_3 are the final states. What language does this DFA recognize?(GATE 2007, 2 Marks)

- a) All strings of x and y
- b) All strings of x and y and even number of x and even number of y or odd no of x and odd number of y
- c) All strings of x and y which have equal number of x and y
- d) All strings of x and y with either even number of x and odd number of y or odd number of x and even number of y



Q Given below are two finite state automata (\rightarrow indicates the start state and F indicates a final state) Which of the following represents the product automaton $Z \times Y$? (GATE-2008) (2 Marks)

Y:

	a	b
$\rightarrow 1$	1	2
2(F)	2	1

Z:

	a	b
$\rightarrow 1$	2	2
2(F)	1	1

(A)

	a	b
$\rightarrow P$	S	R
Q	R	S
R(F)	Q	P
S	Q	P

(B)

	a	b
$\rightarrow P$	S	Q
Q	R	S
R(F)	Q	P
S	P	Q

(C)

	a	b
$\rightarrow P$	Q	S
Q	R	S
R(F)	Q	P
S	Q	P

(D)

	a	b
$\rightarrow P$	S	Q
Q	S	R
R(F)	Q	P
S	Q	P

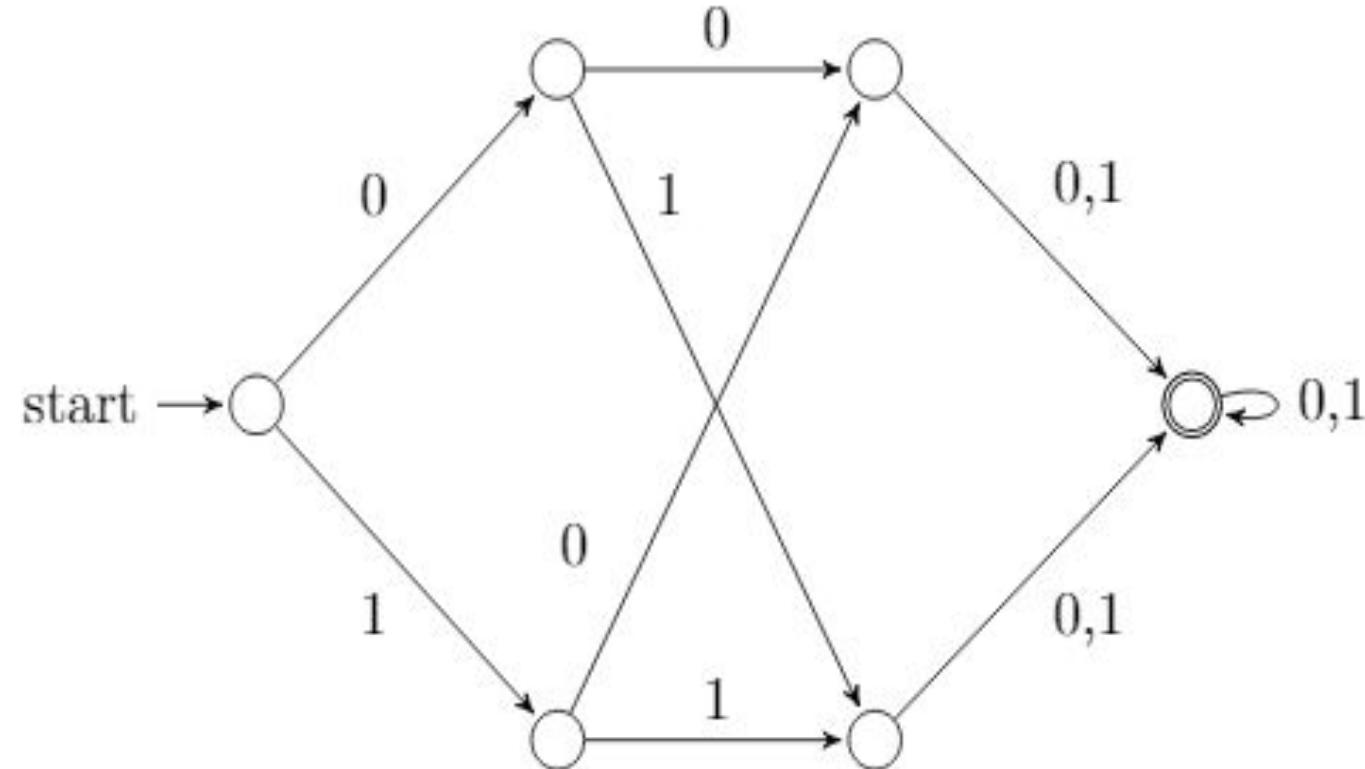
Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{0,1\}$, such that every string ‘w’ which is accepted has a decimal equivalent

- i) $0(\text{mod } 3)$
- ii) $2(\text{mod } 4)$
- iii) $0(\text{mod } 5)$
- iv) $2(\text{mod } 6)$
- v) $3(\text{mod } 8)$

on a format of $m (\text{mod } n)$

- 1) if n is odd, number of states will be n
- 2) if n is even and $n=2^m$, then number of states will be $m+1$
- 3) if n is even and $n \neq 2^m$, then for www.knowledgigate.in no direct formula

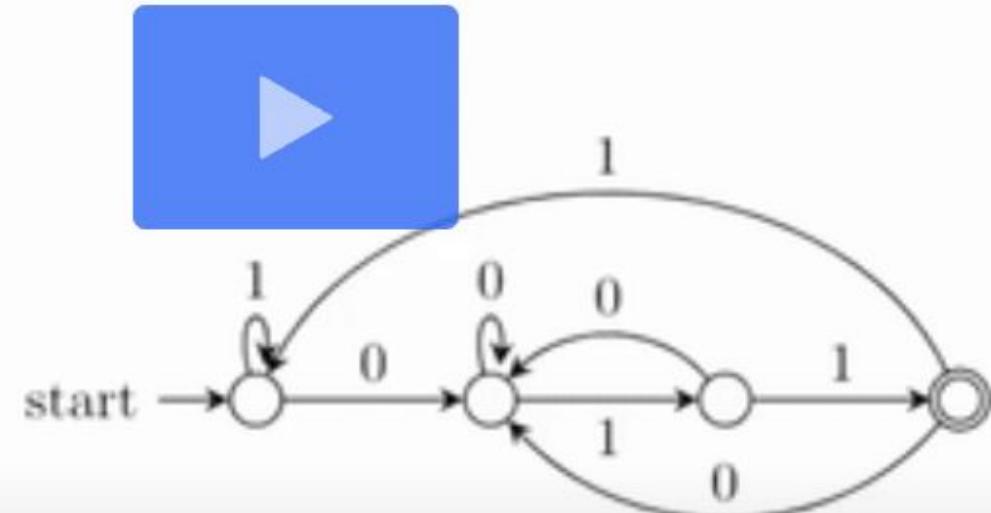
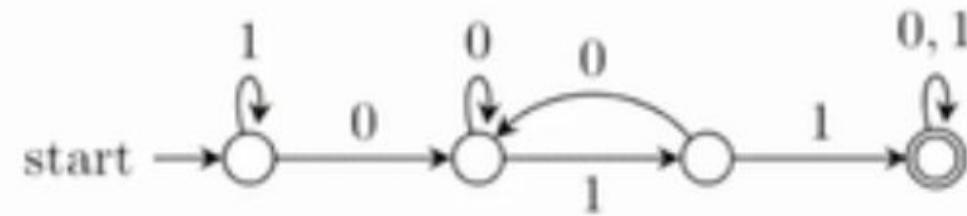
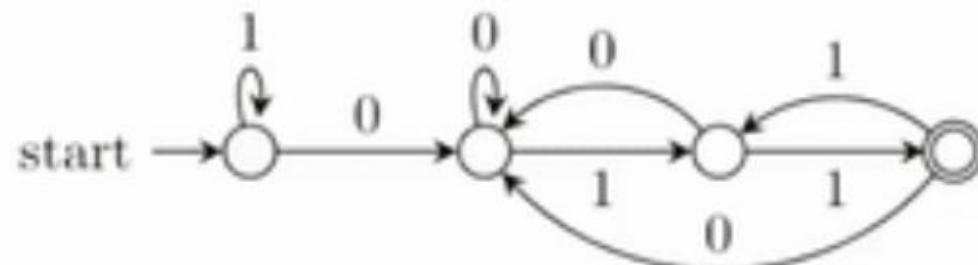
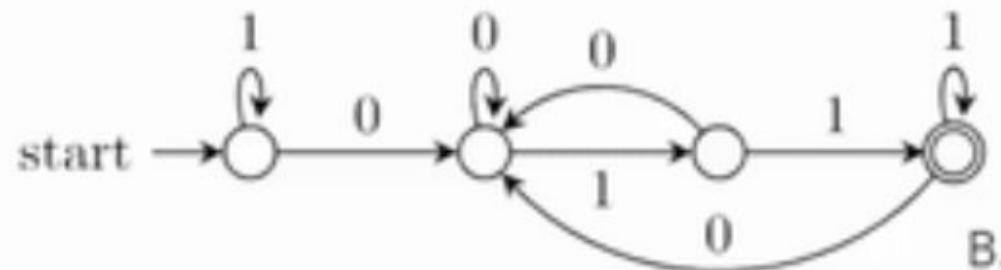
Q Consider the following deterministic finite automaton (DFA). The number of strings of length 8 accepted by the above automaton is _____. (GATE 2021) (1 MARKS)



Consider the following language:

$$L = \{w \in \{0, 1\}^* \mid w \text{ ends with the substring } 011\}$$

Which one of the following deterministic finite automata accepts L?



COMPLIMENT OF DFA

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must not contain a substring aaa?

Note

- The finite automata which is obtained by interchanging final & non-final states is known as complement of the finite automata.
- $L(FA) \cup L(FA^C) = \Sigma^*$ $L(FA) \cap L(FA^C) = \Phi$
- $L(FA)$ subset of Σ^* and $L(FA)^C$ subset of Σ^*
- No of states in FA = no of states in complement of FA
- No of final states in a FA and it's Complement FA may have different final states
- Complement always exist for DFA (as it is a complete system)

Q construct the DFA for the following languages:

i) $L = \{a^m b^n \mid m, n \geq 0\}$

ii) $L = \{a^m b^n c^p \mid m, n, p \geq 0\}$

- If type is $L = \{a^m b^n \mid m \geq 0, n \geq 0\}$, then no of states is no of alphabet +1

Q construct the DFA for the following languages:

- i) $L = \{a^m b^n \mid m \geq 0, n \geq 1\}$ ii) $L = \{a^m b^n \mid m \geq 0, n \geq 2\}$

- If type is $L = \{a^m b^n \mid m \geq 0, n \geq j\}$, then no. of states is $j+2$

Q construct the DFA for the following languages:

- i) $L = \{a^m b^n \mid m \geq 1, n \geq 0\}$ ii) $L = \{a^m b^n \mid m \geq 2, n \geq 0\}$

- If type is $L = \{a^m b^n \mid m \geq i, n \geq 0\}$, then no. of states is $i+3$

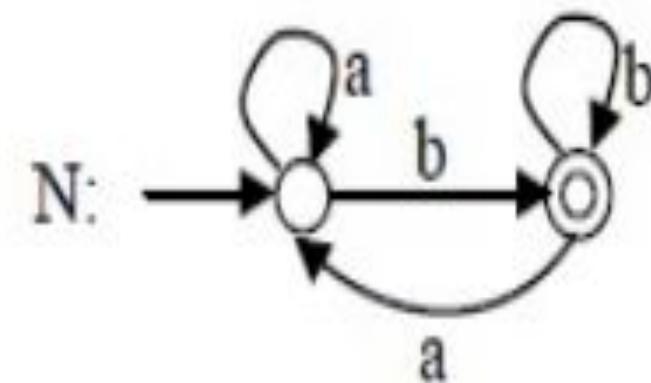
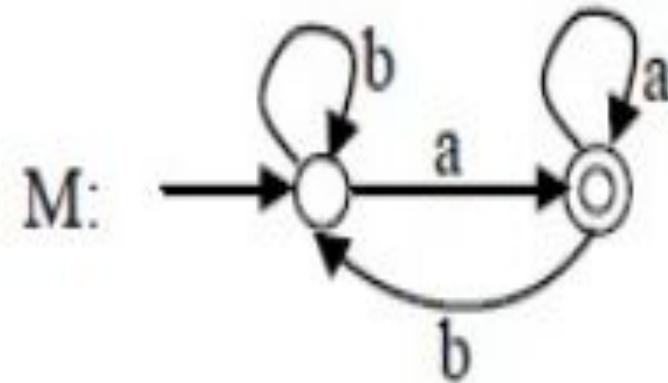
Q construct the DFA for the following languages:

$$L = \{a^m b^n \mid m \geq 1, n \geq 2\}$$

- If type is $L = \{a^m b^n \mid m \geq i, n \geq j\}$, then no of states is $i+j+2$

Q Consider the DFAs M and N given above. The number of states in a minimal DFA that accepts the language $L(M) \cap L(N)$ is _____. (GATE-2015) (2 Marks)

(A) 0



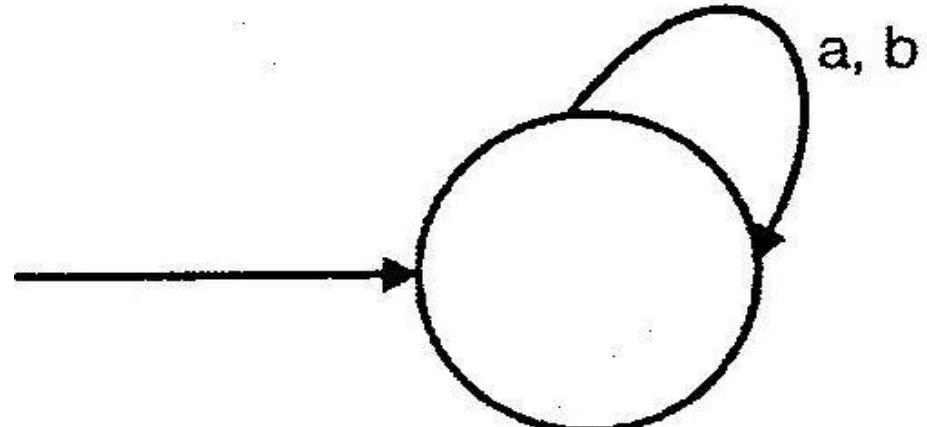
(B) 1

(C) 2

(D) 3

Q The FSM over an alphabet {a, b} shown in the figure accepts

- a) all strings
- c) ϵ - alone



- d) None of these

Q Definition of a language L with alphabet {a} is given as following.

$L = \{a^{nk} \mid k > 0\}$, and n is a positive integer constant}

What is the minimum number of states needed in DFA to recognize L?

(NET-DEC-2018) (2 Marks) (GATE-2011)

- (A) $k+1$
- (B) $n+1$
- (C) $2^{(n+1)}$
- (D) $2^{(k+1)}$

Q. Which of the following can be recognized by a deterministic finite automation? (GATE 1998, 2 Marks)

- a) The numbers $1, 2, 4, 2^n$, written in unary
- b) The set of binary string in which the number of zeroes is the same as the number of ones
- c) The numbers $1, 2, 4, 8, 2^n$, written in binary
- d) The set $\{1, 101, 11011, 1110111, \dots\}$

Q Consider the set of strings on $\{0,1\}$ in which, every substring of 3 symbols has at most two zeros. For example, 001110 and 011001 are in the language, but 100010 is not. All strings of length less than 3 are also in the language. A partially completed DFA that accepts this language is shown below. (GATE-2012) (2 Marks)

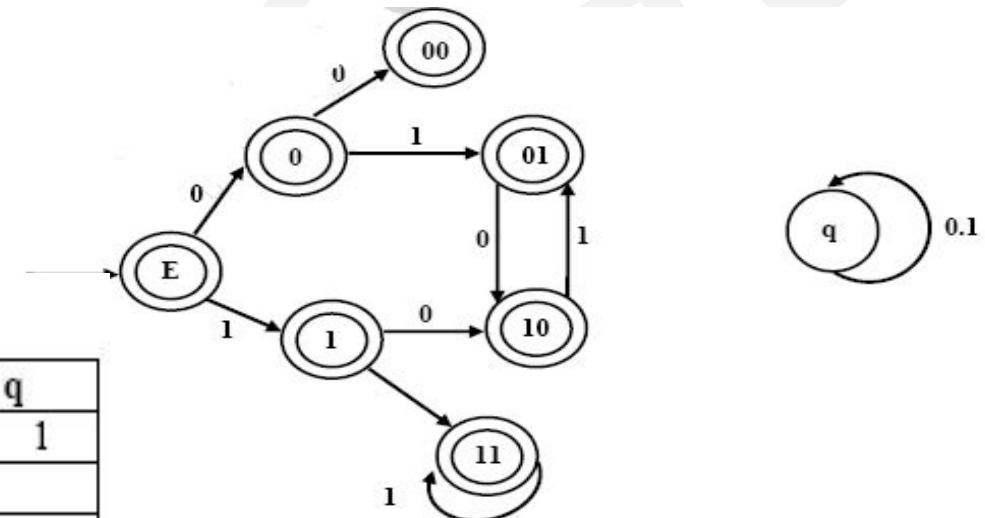
The missing arcs in the DFA are

(A)

	00	01	10	11	q
00	1	0			
01				1	
10	0				
11			0		

(B)

	00	01	10	11	q
00		0			1
01		1			
10				0	
11		0			



(C)

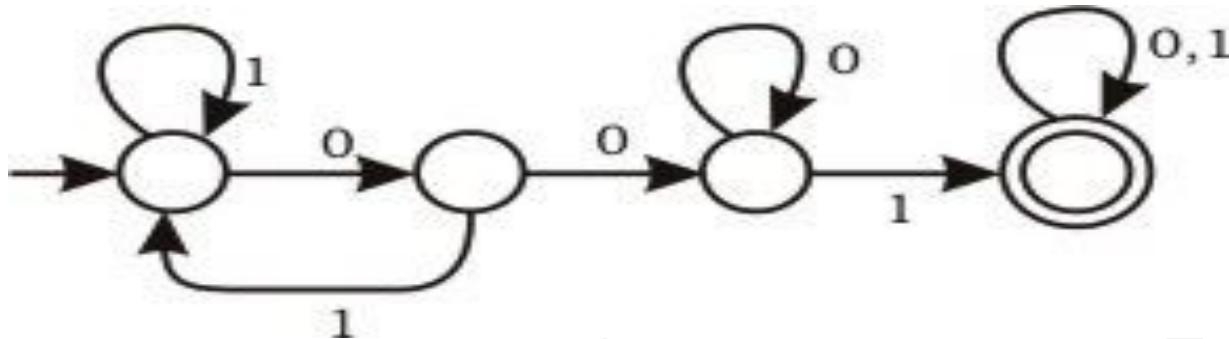
	00	01	10	11	q
00		1			0
01		1			
10			0		
11		0			

(D)

	00	01	10	11	q
00		1			0
01				1	
10	0				
11		0			

Q Consider the following deterministic finite state automaton M. Let S denote the set of seven-bit binary strings in which the first, the fourth, and the last bits are 1. The number of strings in S that are accepted by M is **(GATE-2003) (2 Marks)**

- (A) 1 (B) 5 (C) 7 (D) 8



Q Let $L \subseteq \{0,1\}^*$ be an arbitrary regular language accepted by a minimal DFA with k states. Which one of the following languages must necessarily be accepted by a minimal DFA with k states? **(GATE 2021) (1 MARKS)**

(A) $L - \{01\}$

(B) $L \cup \{01\}$

(C) $\{0,1\}^* - L$

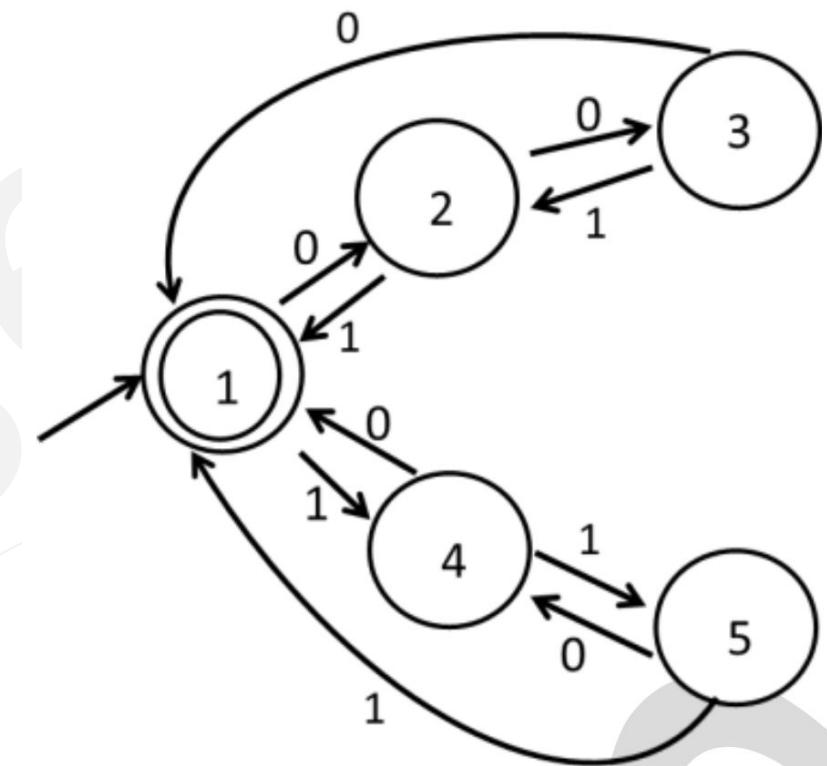
(D) $L \cdot L$

Q. Consider the 5-state DFA M accepting the language $L(M) \subset (0+1)^*$ shown below.

For any string $w \in (0+1)^*$ let $n_0(w)$ be the number of 0's in w and $n_1(w)$ be the number of 1's in W. **(Gate 2024,CS) (2 Marks) (MSQ)**

Which of the following statements is/are FALSE?

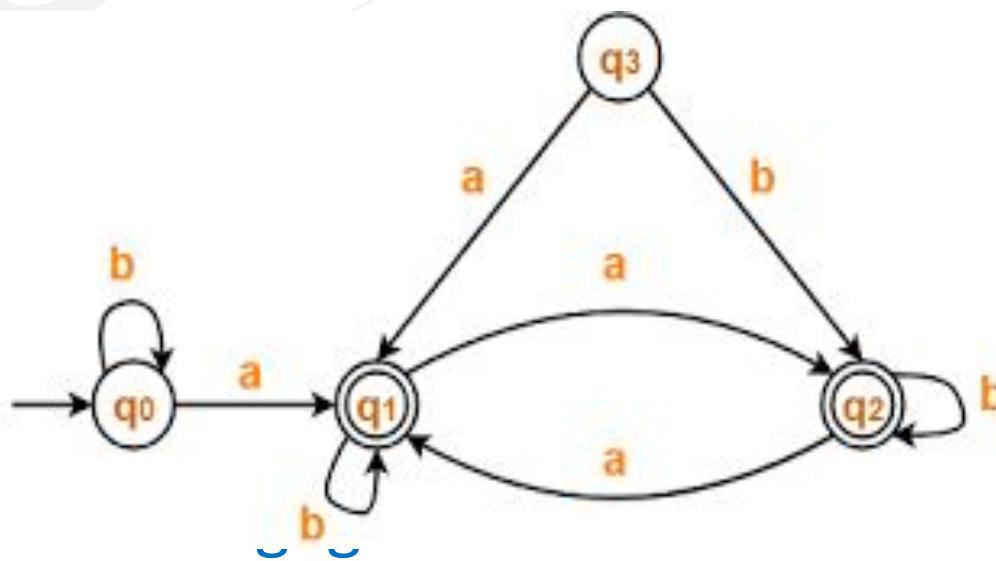
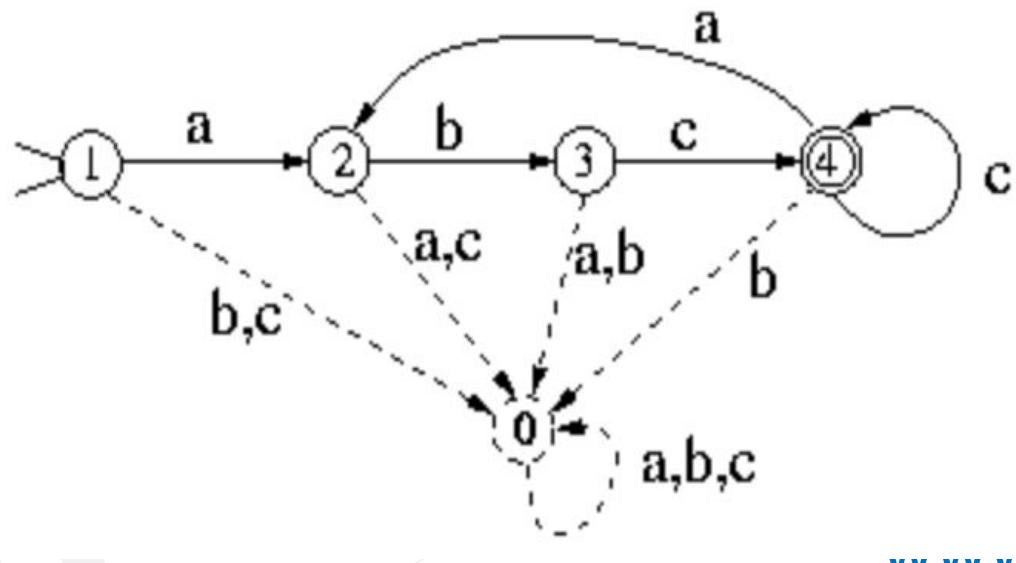
- (a) States 2 and 4 are distinguishable in M
- (b) States 3 and 4 are distinguishable in M
- (c) States 2 and 5 are distinguishable in M
- (d) Any string W with $n_0(w)=n_1(w)$ is in $L(M)$



MINIMIZATION OF FINITE AUTOMATA

- The process of removing states that do not affect the language-accepting capability of a deterministic finite automaton (DFA). The result is a minimal deterministic finite automaton (MFA), which is unique for a given language.
- Designing a minimal DFA directly can be difficult, so the recommended approach is to first design a DFA and then minimize it.
- States in DFA can be classified as:
 - **Productive States:** These states contribute to the machine's ability to accept a language. Their presence affects the language-accepting power.
 - **Non-Productive States:** These states do not add any value to the language-accepting capability of the machine and can be removed during minimization.
- **NOTE- MFA is always unique for a language.**

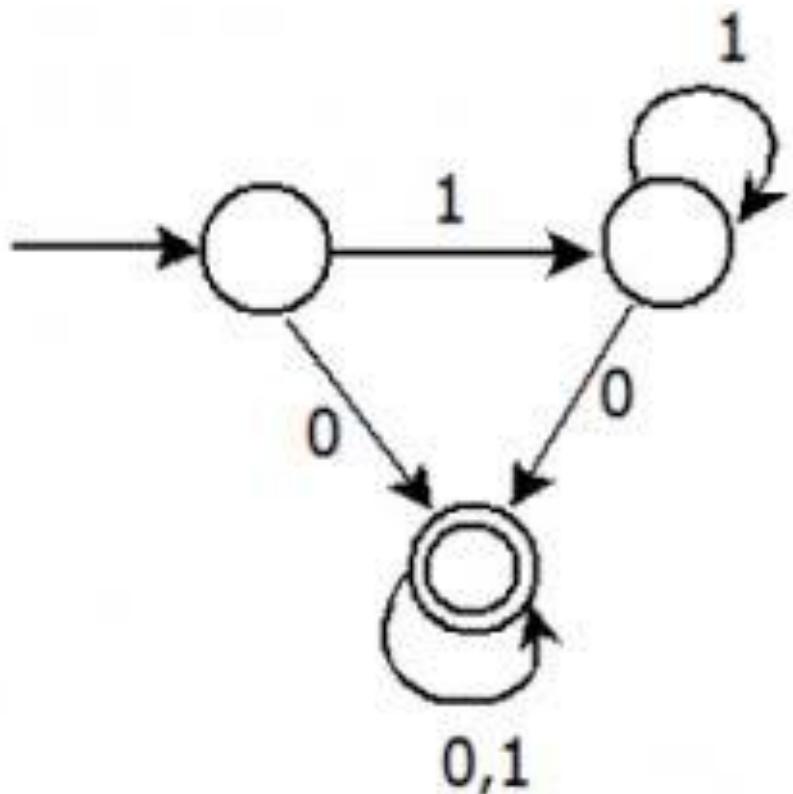
- **NON- PRODUCTIVE STATES**- These states don't add anything to the language accepting power to the machine. They can further be divided into three types, Dead State, Unreachable State, Equal State
- **Dead State**- It is basically created to make the system complete, can be defined as a state from which there is no transition possible to the final state. In a DFA there can be more than one dead state but logically always one dead state is sufficient to complete the functionality.
- **Unreachable State**- It is that state which cannot be reached starting from initial state by parsing any input string.

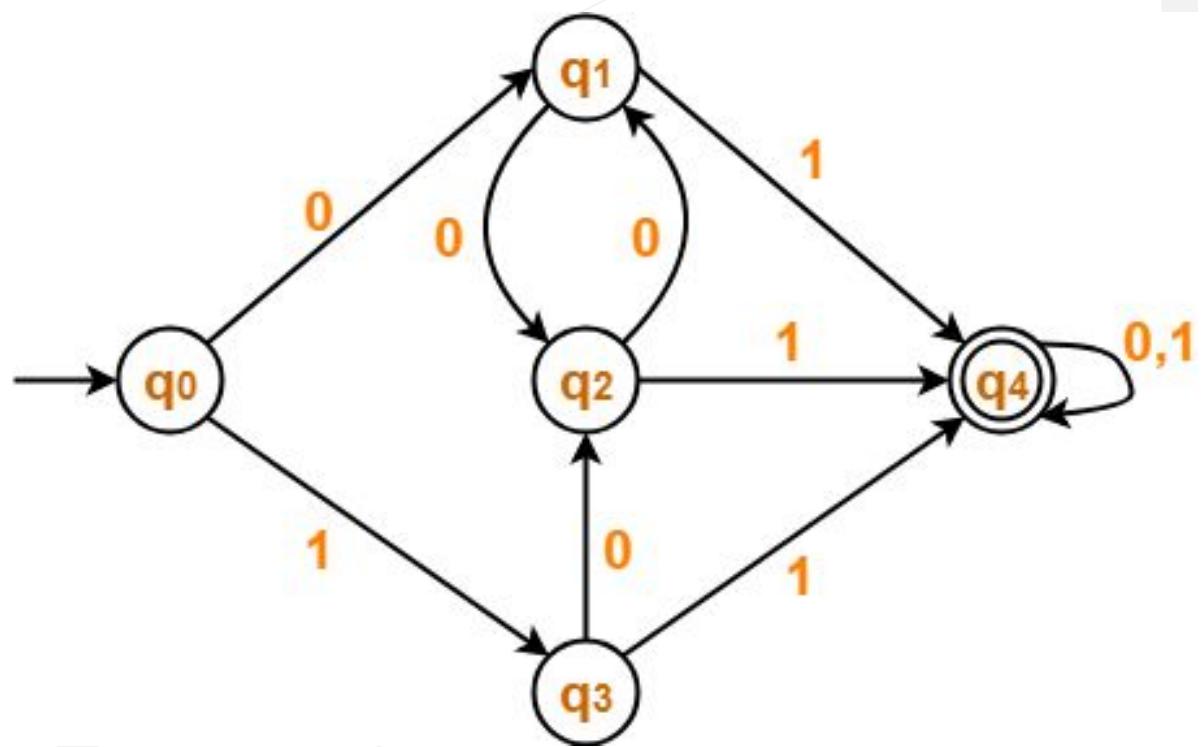
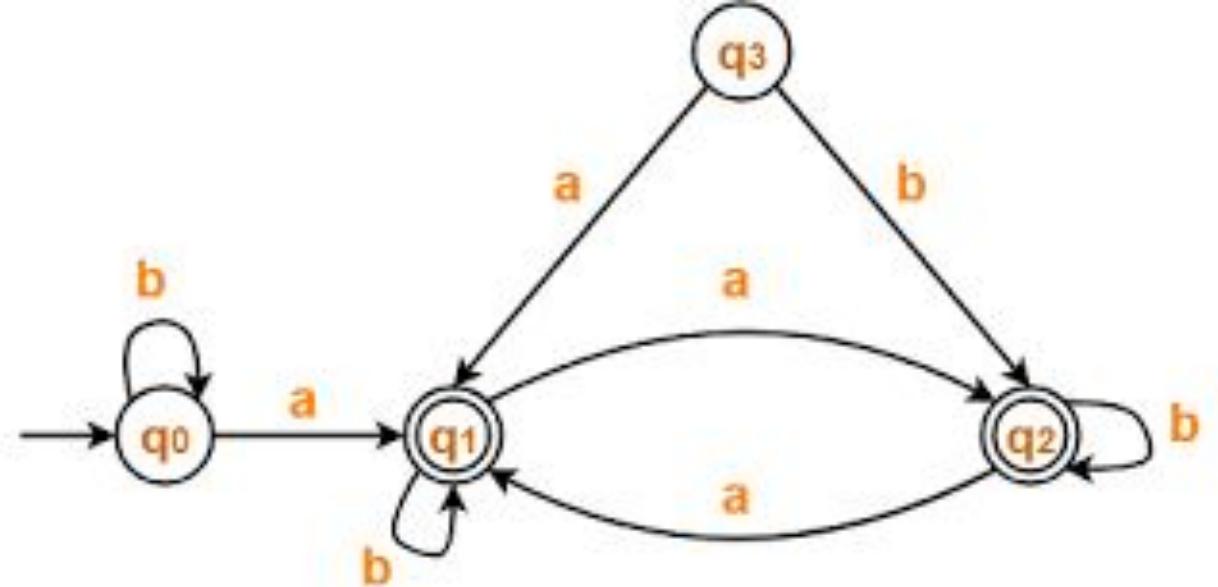


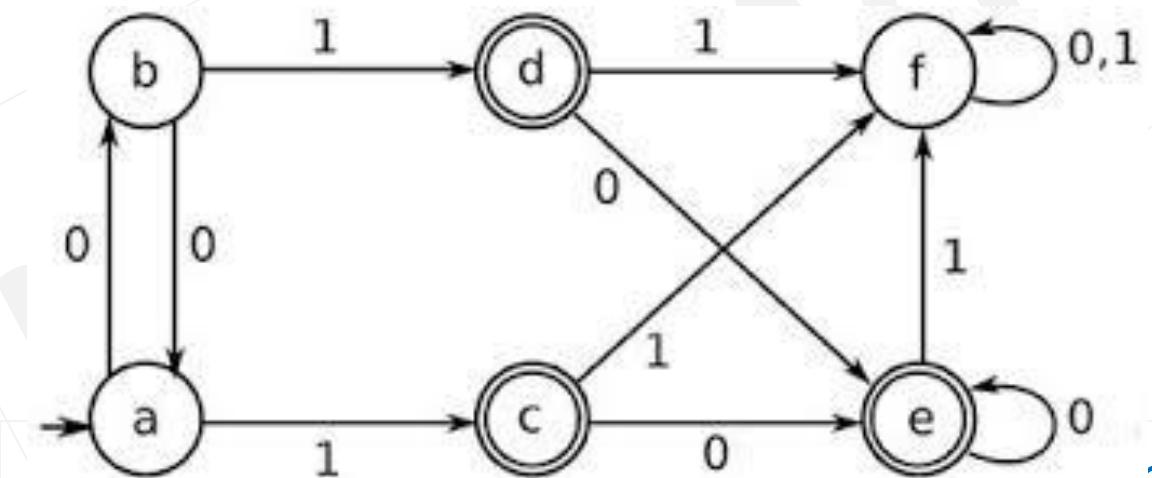
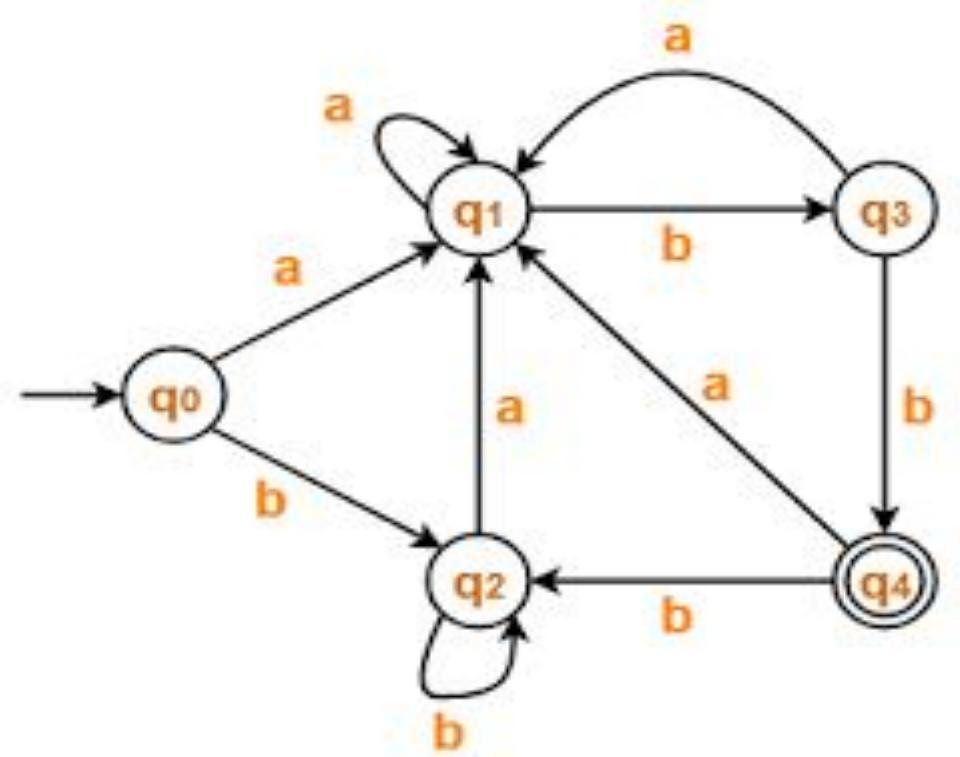
- **Equal State**-These are those states that behave in same manner on each and every input string. That is for any string w where $w \in \Sigma^*$ either both of the states will go to final state or both will go to non-final state. (remember the example of an equal state DFA).
- More formally, two states q_1 and q_2 are equivalent (denoted by $q_1 \cong q_2$) if both $\delta(q_1, x)$ and $\delta(q_2, x)$ are final states or both of them are non-final states for all $x \in \Sigma^*$. If q_1 and q_2 are k -equivalent for all $k \geq 0$, then they are k -equivalent.

Procedure of Minimization

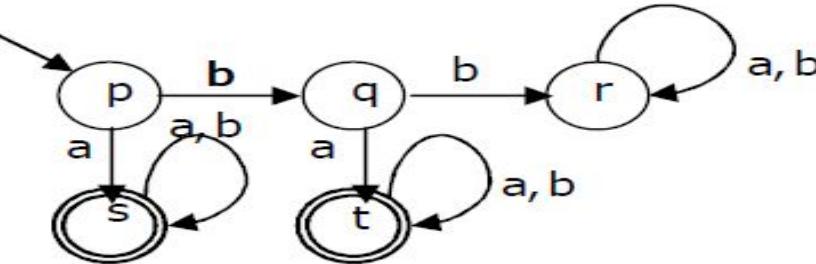
- For this first of all, group all the non-final states in one set and all final states in another set.
- Now, on both the sets, individually check, whether any of the underlying elements (states) of that particular set are behaving in the same way, that is are they having same transition(to same set) on each input alphabet.
- if the answer is yes, then these two states are equal, otherwise not.







Q A deterministic finite automation (DFA) D with alphabet {a, b} is given below (GATE-2011) (2 Marks)



Which of the following finite state machines is a valid minimal DFA which accepts the same language as D?

- (A) A DFA with three states: p, q, and s. State p is the start state. State s is the accept state. Transitions: p to q on b, p to s on a, q to r on b, q to s on a, and s to s on a/b.
- (B) A DFA with three states: p, q, and s. State p is the start state. State s is the accept state. Transitions: p to q on a/b, q to r on a, q to s on b, and s to s on a/b.
- (C) A DFA with four states: p, q, r, and s. State p is the start state. State r is the accept state. Transitions: p to q on a/b, q to r on b, q to s on a, r to r on a/b, s to s on a/b, and s to q on a/b.
- (D) A DFA with three states: p, q, and s. State p is the start state. State s is the accept state. Transitions: p to q on b, p to s on a, q to r on a, q to s on b, and s to s on a/b.

Q. Let $\Sigma = \{1,2,3,4\}$. For $x \in \Sigma^*$, let $prod(x)$ be the product of symbols in x modulo 7. We take $prod(\epsilon) = 1$, where ϵ is the null string.

For example, $prod(124) = (1 \times 2 \times 4) \text{ mod } 7 = 1$.

Define $L = \{x \in \Sigma^* \mid prod(x) = 2\}$.

The number of states in a minimum state DFA for L is _____. (Answer in integer) **(Gate 2025)**

NON DETERMINISTIC FINITE AUTOMATA

- Non-determinism in finite automata means that an automaton has multiple possible moves in a given state for a particular input. Unlike deterministic automata, where the next move is uniquely defined, non-deterministic automata allow for a set of possible transitions.
- Key points:
 - **Choice of Moves:** At any given step, the automaton can choose from multiple transitions for the same input.
 - **Theoretical Concept:** Non-deterministic machines are theoretical models, not meant to be physically implemented. They help simplify the design process but are not directly used in practice.
 - **Purpose:** NFAs are easier to design compared to deterministic automata (DFA), and every NFA can be converted into an equivalent DFA for practical applications.

FORMAL DESCRIPTION OF NDFA

- A Non-Deterministic finite automaton (NDFA) is a 5-tuple $(Q, \Sigma, \delta, S, F)$ where:
 - Q is a finite and non-empty set of states
 - Σ is a finite non-empty set of finite input alphabet
 - δ is a transition function $\delta: Q \times \Sigma \rightarrow 2^Q$
 - q_0 is **initial state** (always one) ($q_0 \in Q$)
 - F is a set of final states ($F \subseteq Q$) ($0 \leq |F| \leq N$), where n is the number of states

Some points to remember

- Every DFA is also an NFA. Every NFA can be translated to an equivalent DFA, so their language accepting capability is same Both only recognize ***regular languages***. Accepting power of NDFA= Accepting power of DFA.
- NFA need not to be a complete system. There can be a state that doesn't have any transition on some input symbol. A null transition is also possible for NFA, such special NFA are called Null-NFA. We will discuss it later.
- NDFA is a theoretical engine and is not implementable, but it is very easy to design compare to DFA. No concept of dead state, therefore complementation of NFA is also not possible. NDFA will respond for only valid strings and no need to respond for invalid strings. (it is a Incomplete system)

ACCEPTANCE BY NDFA

- Let 'w' be any string defined over the alphabet Σ , corresponding to w, there can be multiple transitions for NFA starting from initial state, if there exist at least one transition for which we start at the initial state and ends in any One of the final state, then the string 'w' is said to be accepted by the non-deterministic finite automata, otherwise not.
- Mathematically, it can be represented as, $L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$

Q design a NDFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, where every accepted string 'w' starts with substring s, Where s = aba

conclusion if $w = sx$, $|s| = m$, then no of states in the NDFA is $m+1$

Q Design a NDFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$. where every accepted string 'w' ends with substring 's', Where s = bab

conclusion if $w = sx$, $|s| = m$, then no of states in the NDFA is $m+1$

Q Design a minimal DFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$. where every accepted string 'w' contains sub string s, Where s = aba

conclusion if $w = xsx$, $|s| = m$, then no of states in the DFA is $m+1$

Q Let w be any string of length n in $\{0,1\}^*$. Let L be the set of all substrings of w . What is the minimum number of states in a non-deterministic finite automaton that accepts L ? **(GATE-2010) (1 Marks)**

- (A) $n-1$
- (B) n
- (C) $n+1$
- (D) $2n-1$

Q Design a NDFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with same symbol.

Q Design a NDFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with different symbol.

Q Design a minimal NDFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w , is like $w=SX$, Where $s = aaa/bbb$

Q Design a minimal NDFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w , is like $w=XS$, Where $s = aaa/bbb$

Q Design a NFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' accepted must be like

i) $|w| = 3$

ii) $|w| \leq 3$

iii) $|w| \geq 3$

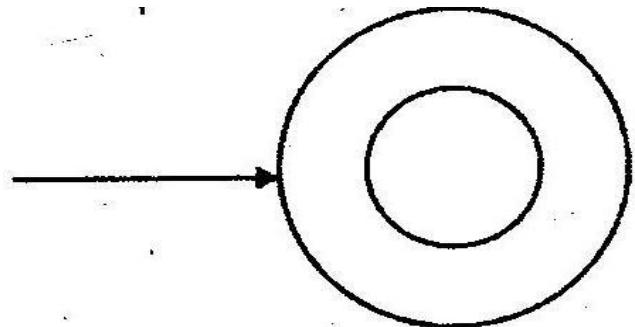
Q Design a NFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain exactly two a's.

Q Design a NFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain at least two a's.

Q Design a NFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain at most two a's.

Q Design a NDFA that accepts all strings over the alphabet $\Sigma = \{a, b\}$ such that for every accepted string 3rd from right end is always a.

Q The FSM shown in the figure accepts



- a) all strings
- b) no strings
- c) ϵ - alone
- d) None of these

Q (GATE-2014) (2 Marks)

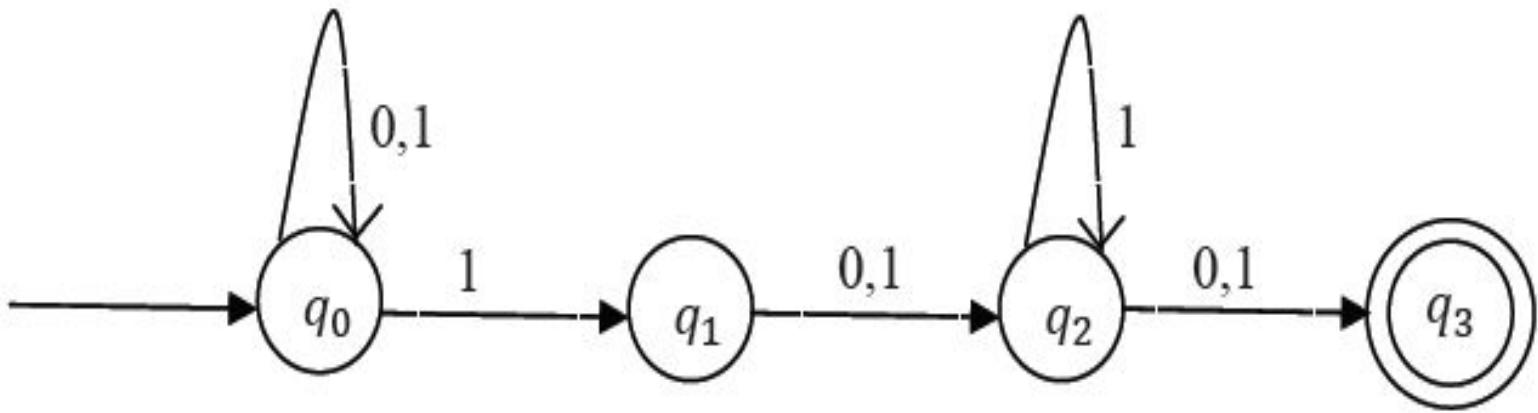
Consider the finite automaton in the following figure.

(A) $\{q_0, q_1, q_2\}$

(B) $\{q_0, q_1\}$

(C) $\{q_0, q_1, q_2, q_3\}$

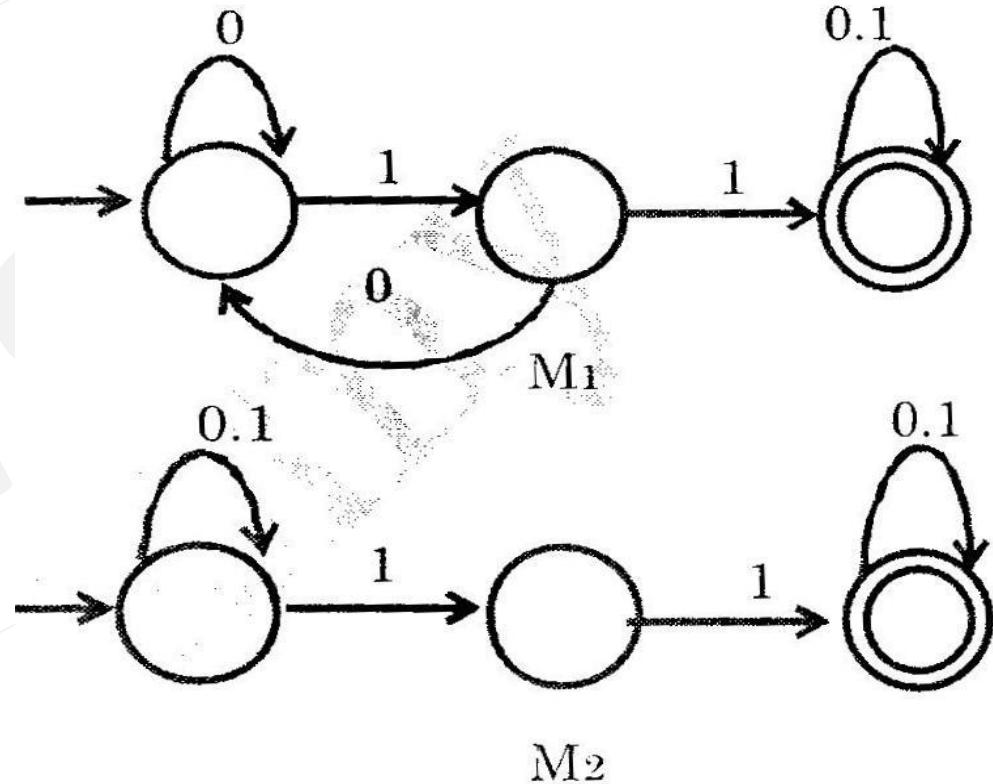
(D) $\{q_3\}$



What is the set of reachable states for the input string 0011?

Q. Consider the following two finite automata. M_1 accepts L_1 and M_2 accepts L_2 which one of the following is True? (GATE 2008, 2 Marks)

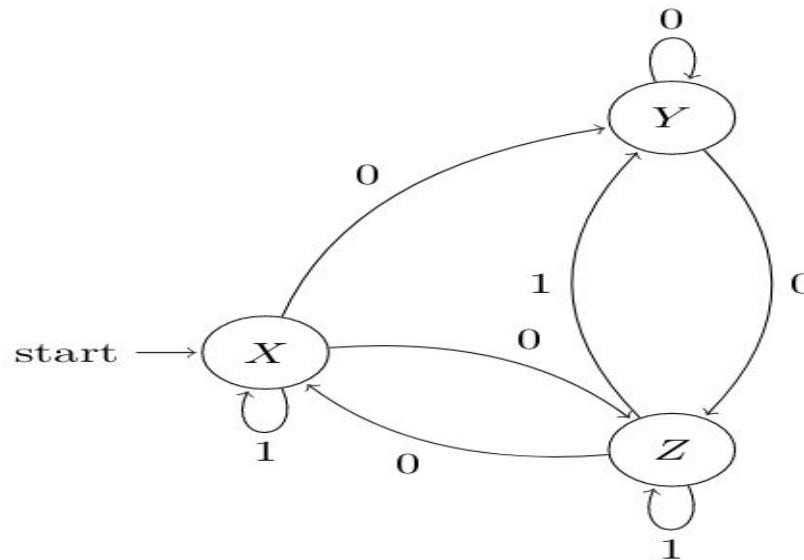
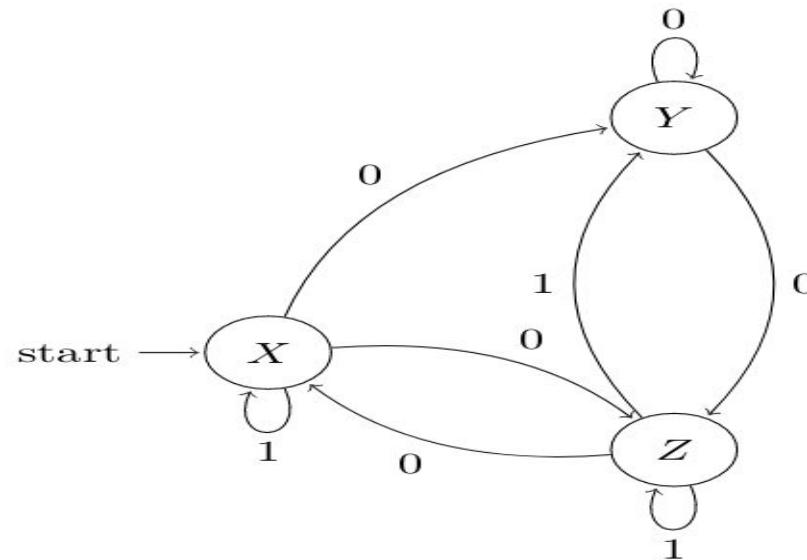
- a) $L_1 = L_2$
- b) $L_1 \subset L_2$
- c) $L_1 \cap L_2' = \emptyset$
- d) $L_1 \cup L_2 \neq L_1$



Q Consider the non-deterministic finite automaton (NFA) shown in the figure.
(GATE-2005) (2 Marks)

State X is the starting state of the automaton. Let the language accepted by the NFA with Y as the only accepting state be L_1 . Similarly, let the language accepted by the NFA with Z as the only accepting state be L_2 . Which of the following statements about L_1 and L_2 is TRUE?

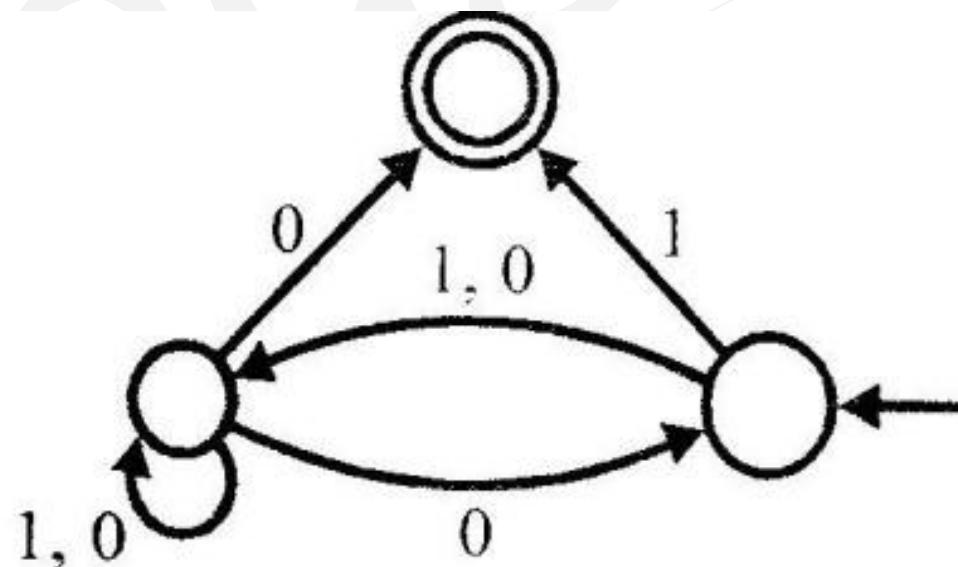
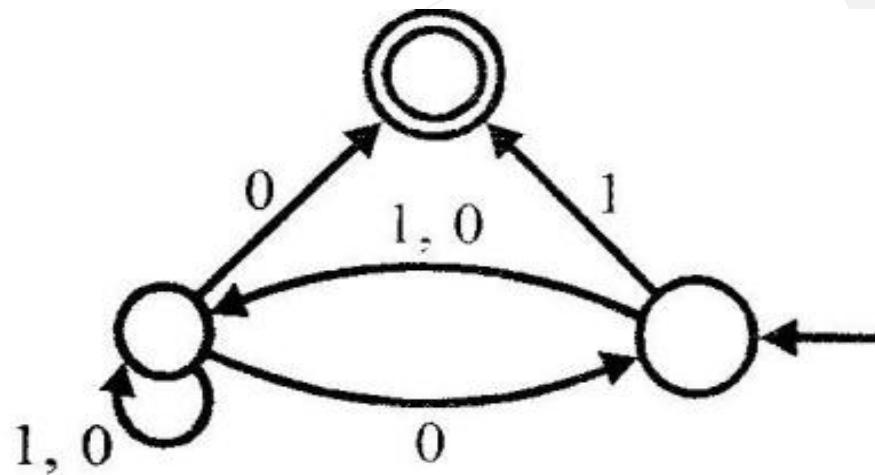
- (A) $L_1 = L_2$** **(B) $L_1 \subset L_2$** **(C) $L_2 \subset L_1$** **(D) None of the above**



Q. Consider the NFAM shown below. (GATE 2003, 2 Marks)

Let the language accepted by M be L. Let L_1 be the language accepted by the NFA M_1 , obtained by changing the non-accepting state and by changing the non-accepting state of M to accepting states. Which of the following statements are True?

- a) $L_1 = \{0,1\}^* - L$
- b) $L_1 = \{0,1\}^*$
- c) $L_1 \subset L$
- d) $L_1 = L$



NFA and DFA Equivalence

- In this topic we will be learning about the equivalence of NFA and DFA and how an NFA can be converted to equivalent DFA. Let us take an example and understand the conversion. Since every NFA and DFA has equal power that means, for every language if a NFA is possible, then DFA is also possible. So, every NFA can be converted to DFA. The process of conversion of an NFA into a DFA is called Subset Construction.
- If NFA have ‘n’ states which is converted into DFA which ‘m’ states than the relationship between n and m will be $1 \leq m \leq 2^n$

Q Let N be an NFA with n states and let M be the minimized DFA with m states recognizing the same language. Which of the following is NECESSARILY true? (GATE-2008) (2 Marks)

- (A) $m \leq 2^n$ (B) $n \leq m$
(C) M has one accept state (D) $m = 2^n$

Q Given an arbitrary non-deterministic finite automaton (NFA) with N states, the maximum number of states in an equivalent minimized DFA is at least **(GATE-2001) (2 Marks)**

- (A) N^2 (B) 2^N (C) $2N$ (D) $N!$

Q Let N be an NFA with n states. Let k be the number of states of a minimal DFA which is equivalent to N . Which one of the following is necessarily true? (GATE-2018) (1 Marks)

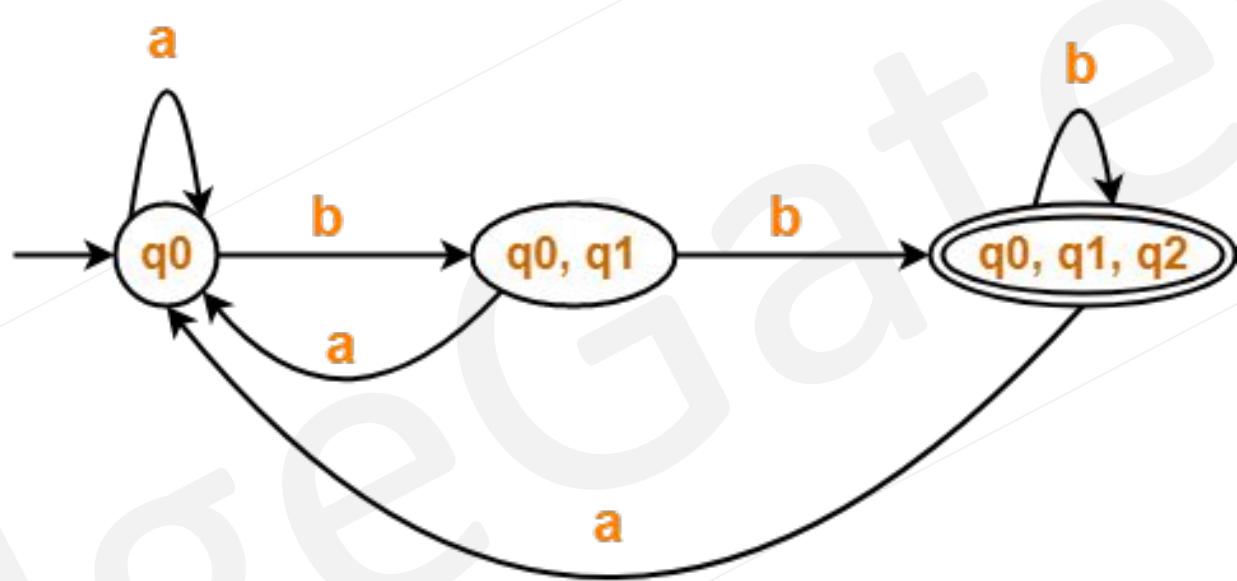
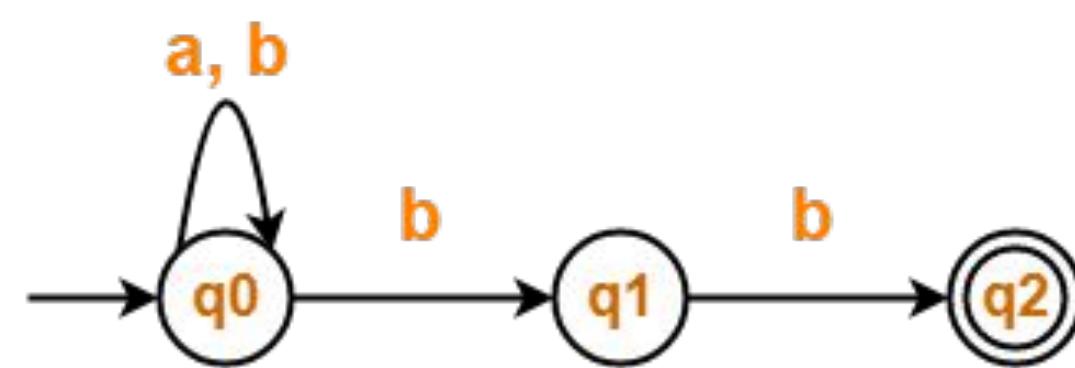
- a) $k \geq 2^n$ b) $k \geq n$ c) $k \leq n^2$ d) $k \leq 2^n$

Q. A regular language L is accepted by a non-deterministic finite automaton (NFA) with n states. Which of the following statement(s) is/are FALSE? (Gate 2025)

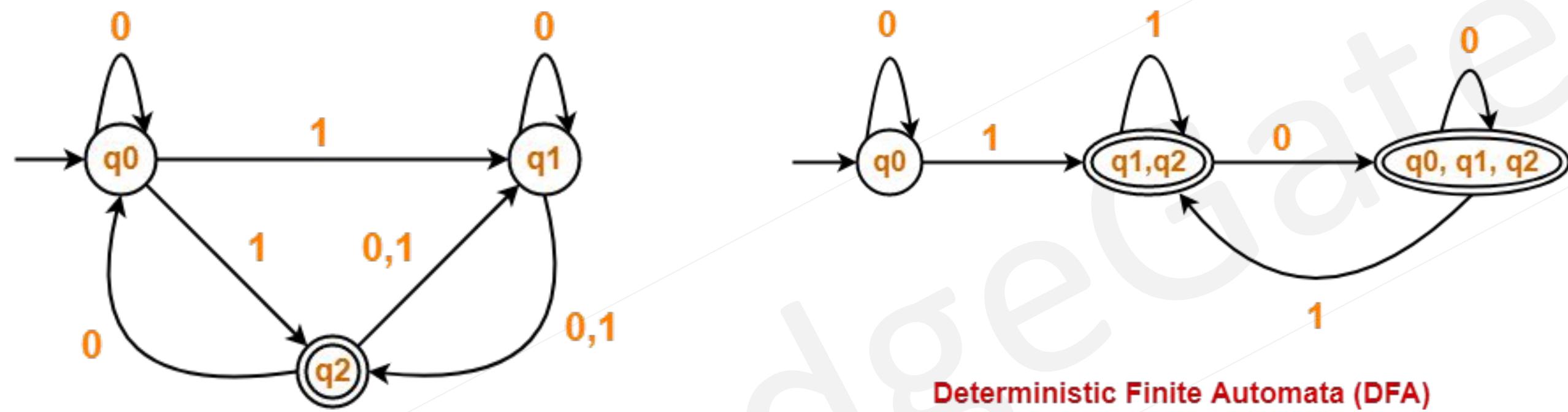
- A) L may have an accepting NFA with $< n$ states.
- B) L may have an accepting DFA with $< n$ states.
- C) There exists a DFA with $\leq 2^n$ states that accepts L .
- D) Every DFA that accepts L has $> 2^n$ states.

Procedure for Conversion

- There lies a fixed algorithm for the NFA and DFA conversion. Following things must be considered
 - Initial state will always remain same.
 - Start the construction of δ' with the initial state & continue for every new state that comes under the input column and terminate the process whenever no new state appears under the input column.
 - Every subset of states that contain the final state of the NFA is a final state in the resulting DFA.
 - $\delta'(q_0, q_1, q_2, q_3, \dots, q_{n-1}, a) = \bigcup_{i=0}^{i=n-1} \delta(q_i, a)$



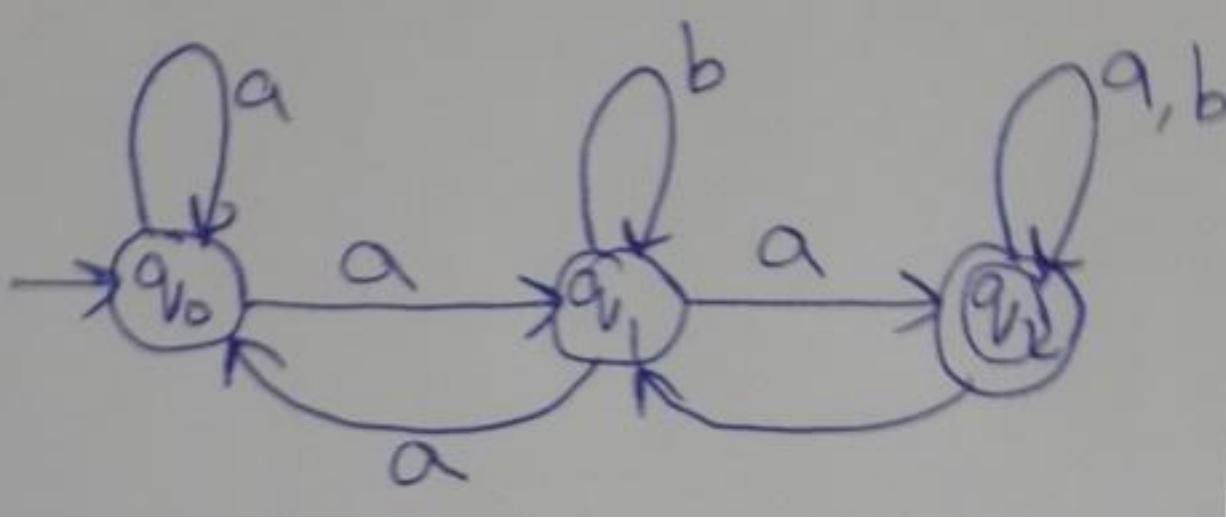
Deterministic Finite Automata (DFA)



Deterministic Finite Automata (DFA)



Deterministic Finite Automata (DFA)

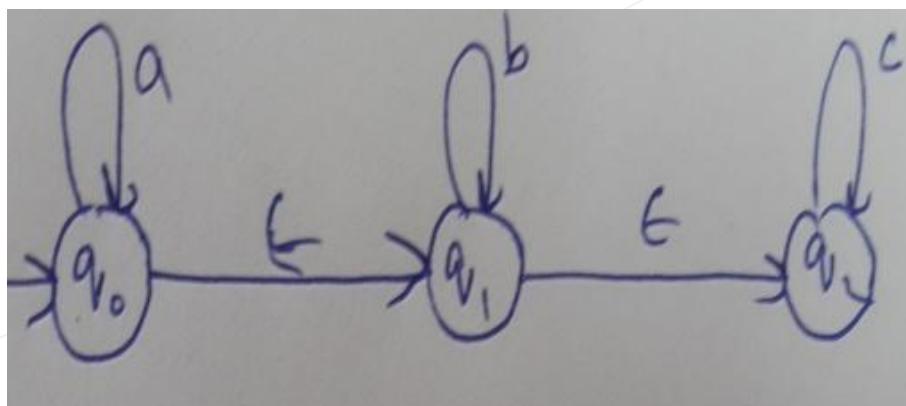


NFA WITH EPSILON MOVES (ϵ -NFA)

- An automaton that consists of null transitions is called a Null- NFA i.e. we allow a transition on null means empty string.
- ϵ -NFA is a 5-tuple $(Q, \Sigma, \delta, S, F)$ where:
 - Q is a finite and non-empty set of states
 - Σ is a finite non-empty set of finite input alphabet
 - δ is a transition function $\delta: (Q \times (\Sigma \cup \{\epsilon\})) \rightarrow 2^Q$
 - S is **initial state** (always one) ($S \in Q$)
 - F is a set of final states ($F \subseteq Q$) ($0 \leq |F| \leq n$, where n is the number of states)

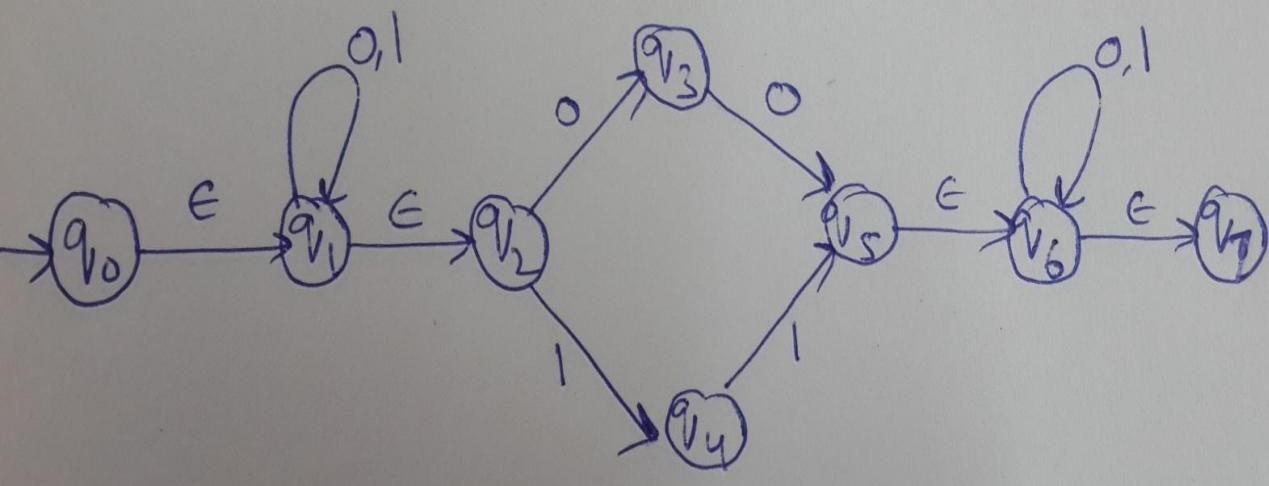
NULL-CLOSURE

- Null closure of a set Q is defined as a set of all the states, which are at zero distance from the state Q. A set of all the states, that can be reached from the state and along a null- transition.
- **ϵ -Closure(q_i)**- The set of all the states which are at zero distance from the state q_i is called ϵ -closure(q_i). Or the set of all the states that can be reached from the state q_i along ϵ labelled transition path, is known as ϵ -closure(q_i).



EQUIVALENCE BETWEEN NULL NFA TO NFA

- There will be no change in the initial state.
- No change in the total no. of states
- May be change in the number of final states.
- All the states will get the status of the final state in the resulting NFA, whose ϵ -closure contains at least one final state in the initial ϵ -NFA.

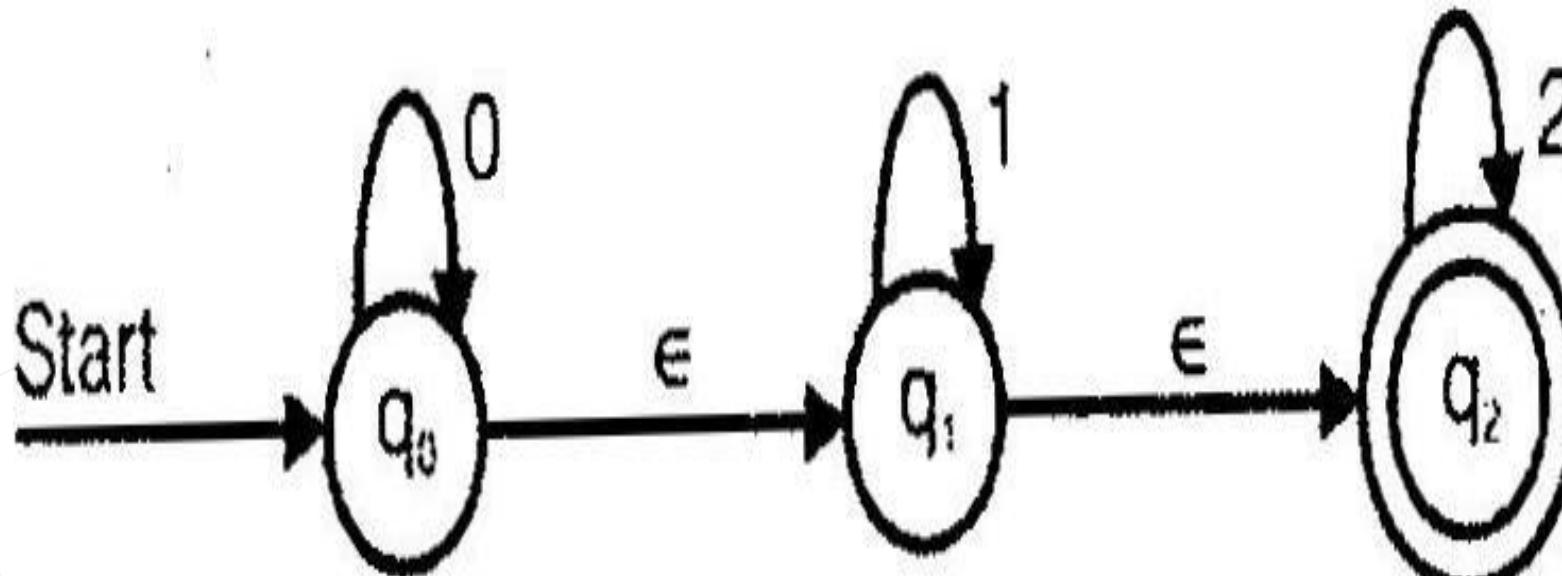


Q. If the given NFA is converted to NFA without \in moves , which of the following denotes the set of final states?

- a) $\{q_2\}$
- b) $\{q_1, q_2\}$
- c) $\{q_0, q_1, q_2\}$
- d) can't be determined

Q. Which of the following strings will not be accepted by the given NFA?

- a) 00 11 22
- b) 11 22
- c) 21
- d) 22



Q. Which of the following is False?

- a) The languages accepted by FAs are regular languages
- b) Every DFA is an NFA
- c) There are some NFAs for which no DFA can be constructed
- d) If L is accepted by an NFA with ϵ transitions then L is accepted by an NFA without ϵ transitions

Q. An FSM can be used to add two given integers. This remark is

- a) true
- b) false
- c) may be true
- d) none of the above

Q Let δ denote the transition function and δ' denoted the extended transition function of the ϵ -NFA whose transition table is given below: (GATE-2017) (2 Marks)

Then, $\delta'(q_2, aba)$ is

- a) \emptyset
- b) $\{q_1, q_2, q_3\}$
- c) $\{q_0, q_1, q_2\}$
- d) $\{q_0, q_2, q_3\}$

δ	ϵ	a	b
$\rightarrow q_0$	$\{q_2\}$	$\{q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_2\}$	$\{q_3\}$
q_2	$\{q_0\}$	\emptyset	\emptyset
q_3	\emptyset	\emptyset	$\{q_2\}$

Q Consider the following languages and find which of them are regular?

1. $L = \{a^m b^n \mid m, n \geq 0\}$

2. $L = \{a^m b^n c^p \mid m, n, p \geq 0\}$

3. $L = \{a^{x_1} b^{x_2} \dots z^{x_{26}} \mid x_i \geq 0, 1 \leq i \leq 26\}$

4. $L = \{a^m b^n \mid 1 \leq m \leq 100, 1 \leq n \leq 1200\}$

1. $L = \{a^m b^n \mid m * n = \text{finite}\}$

2. $L = \{a^n b^n \mid 1 \leq n \leq 2^{|GATE|}\}$

3. $L = \{a^n b^n \mid 1 \leq n \leq 2^{37\text{th prime}}\}$

4. $L = \{a^m b^n \mid m = n, 1 \leq n \leq 2^{2^{10}}\}$

1. $L = \{a^m b^n \mid m = n \mid m, n \geq 0\}$
2. $L = \{a^m b^n \mid m < n \mid m, n \geq 0\}$
3. $L = \{a^m b^n \mid m \neq n \mid m, n > 0\}$
4. $L = \{a^m b^n \mid m \text{ is divisible by } n\}$
5. $L = \{a^m b^n \mid m = n^p, p \geq 1\}$
6. $L = \{a^m b^n \mid \text{HCF}(m, n) = 1 \mid m, n \geq 1\}$

1. $L = \{a^m b^n \mid m + n = \text{even}\}$

2. $L = \{a^m b^n c^p \mid m = n = p\}$

3. $L = \{a^m b^n c^p \mid m + p = n\}$

4. $L = \{a^m b^n \mid m + n = \text{odd}\}$

1. $L = \{w c w^r \mid w \in \Sigma^*\}$
2. $L = \{w c w \mid w \in \Sigma^*\}$
3. $L = \{w c w^r \mid w \in \Sigma^+\}$
4. $L = \{w c w \mid w \in \Sigma^+\}$
5. $L = \{w w \mid w \in \Sigma^*\}$
6. $L = \{w w^r \mid w \in \Sigma^+\}$

1. $L = \{w c w^r \mid c, w \in \Sigma^*\}$

2. $L = \{c w w^r \mid c, w \in \Sigma^*\}$

3. $L = \{w w^r c \mid c, w \in \Sigma^*\}$

4. $L = \{w c w^r \mid c, w \in \Sigma^+$

5. $L = \{c w w^r \mid c, w \in \Sigma^+\}$

6. $L = \{w w^r c \mid c, w \in \Sigma^+\}$

1. $L = \{w c w \mid c, w \in \Sigma^*\}$
2. $L = \{c w w \mid c, w \in \Sigma^*\}$
3. $L = \{w w c \mid c, w \in \Sigma^*\}$
4. $L = \{w c w \mid c, w \in \Sigma^+\}$
5. $L = \{c w w \mid c, w \in \Sigma^+\}$
6. $L = \{w w c \mid c, w \in \Sigma^+\}$

Q If L is a regular language over $\Sigma = \{a, b\}$, which one of the following languages is NOT regular?
(GATE-2019) (2 Marks)

- a) $L \cdot L^R = \{xy \mid x \in L, y^R \in L\}$
- b) Suffix (L) = $\{y \in \Sigma^* \mid \exists x \in \Sigma^* \text{ such that } xy \in L\}$
- c) Prefix (L) = $\{x \in \Sigma^* \mid \exists y \in \Sigma^* \text{ such that } xy \in L\}$
- d) $\{w w^R \mid w \in L\}$

Q Which of the following languages is/are regular? (GATE-2015) (2 Marks)

$L_1: \{w x w^R \mid w, x \in \{a, b\}^* \text{ and } |w|, |x| > 0\}$ w^R is the reverse of string w

$L_2: \{a^n b^m \mid m \neq n \text{ and } m, n \geq 0\}$

$L_3: \{a^p b^q c^r \mid p, q, r \geq 0\}$

- (A) L_1 and L_3 only
- (B) L_2 only
- (C) L_2 and L_3 only
- (D) L_3 only

Q Let $L_1 = \{w \in \{0, 1\}^* \mid w \text{ has at least as many occurrences of } (110)^\text{'s as } (011)^\text{'s}\}$.

Let $L_2 = \{w \in \{0, 1\}^* \mid w \text{ has at least as many occurrences of } (000)^\text{'s as } (111)^\text{'s}\}$.

Which one of the following is TRUE? **(GATE-2014) (2 Marks)**

- (A) L_1 is regular but not L_2
- (B) L_2 is regular but not L_1
- (C) Both L_2 and L_1 are regular
- (D) Neither L_1 nor L_2 are regular

If $L_1 = \{a^n \mid n \geq 0\}$ and $L_2 = \{b^n \mid n \geq 0\}$, consider

- (I) $L_1 \cdot L_2$ is a regular language
- (II) $L_1 \cdot L_2 = \{a^n b^n \mid n \geq 0\}$

Which one of the following is CORRECT? **(GATE-2014) (2 Marks)**

- (A) Only (I)
- (B) Only (II)
- (C) Both (I) and (II)
- (D) Neither (I) nor (II)

Q Which one of the following is TRUE? (GATE-2014) (1 Marks)

- (A) The language $L = \{a^n b^n \mid n \geq 0\}$ is regular.
- (B) The language $L = \{a^n \mid n \text{ is prime}\}$ is regular.
- (C) The language $L = \{w \mid w \text{ has } 3k + 1 \text{ } b's \text{ for some } k \in \mathbb{N} \text{ with } \Sigma = \{a, b\}\}$ is regular.
- (D) The language $L = \{ww \mid w \in \Sigma^* \text{ with } \Sigma = \{0,1\}\}$ is regular.

Q Let P be a regular language and Q be context-free language such that $Q \subseteq P$. (For example, let P be the language represented by the regular expression p^*q^* and Q be $\{p^nq^n \mid n \in N\}$). Then which of the following is ALWAYS regular? **(GATE-2011) (1 Marks)**

- (A) $P \cap Q$
- (B) $P - Q$
- (C) $\Sigma^* - P$
- (D) $\Sigma^* - Q$

Q Which of the following are regular sets? (GATE-2008) (2 Marks)

- I. $\{a^n b^{2m} \mid n \geq 0, m \geq 0\}$
- II. $\{a^n b^m \mid n = 2m\}$
- III. $\{a^n b^m \mid n \neq m\}$
- IV. $\{xcy \mid x, y \in \{a, b\}^*\}$

- (A) I and IV only
- (B) I and III only
- (C) I only
- (D) IV only

Q Which of the following languages is (are) non-regular? (GATE-2008) (2 Marks)

$$L_1 = \{0^m 1^n \mid 0 \leq m \leq n \leq 10000\}$$

$$L_2 = \{w \mid w \text{ reads the same forward and backward}\}$$

$$L_3 = \{w \in \{0, 1\}^* \mid w \text{ contains an even number of 0's and an even number of 1's}\}$$

- (A) L_2 and L_3 only
- (B) L_1 and L_2 only
- (C) L_3 only
- (D) L_2 only

Q Which of the following languages is regular? (GATE-2007) (2 Marks)

a) $\{w w^R \mid w \in \{0, 1\}^+\}$

b) $\{w w^R x \mid x, w \in \{0, 1\}^+\}$

c) $\{w x w^R \mid x, w \in \{0, 1\}^+\}$

d) $\{x w w^R \mid x, w \in \{0, 1\}^+\}$

Q Consider the following two statements (GATE-2001) (2 Marks)

S1: $\{0^{2n} \mid n \geq 1\}$ is a regular language

S2: $\{0^m 1^n 0^{m+n} \mid m \geq 1 \text{ and } n \geq 1\}$ is a regular language

(A) Only S_1 is correct

(C) Both S_1 and S_2 are correct

(B) Only S_2 is correct

(D) None of S_1 and S_2 is correct

Q Consider the following languages (GATE-2001) (2 Marks)

$L_1 = \{ww \mid w \in \{a, b\}^*\}$

$L_2 = \{ww^R \mid w \in \{a, b\}^*, w^R \text{ is the reverse of } w\}$

$L_3 = \{0^{2i} \mid i \text{ is an integer}\}$

$L_4 = \{0^{i^2} \mid i \text{ is an integer}\}$

Which of the languages are regular?

- (A) Only L_1 and L_2
- (B) Only L_2 , L_3 and L_4
- (C) Only L_3 and L_4
- (D) Only L_3

Regular Expressions

- One way of describing regular language is via the notation of regular expression. An expression of strings which represents regular language is called regular expression.
- The regular expressions are useful for representing certain sets of strings(Language) in an algebraic fashion.
- We give a formal recursive definition of regular expressions over Σ as follows:
 - Any terminal symbol (i.e. an element of Σ), \in and Φ are regular expressions (**Primitive regular expressions**).
 - A regular expression is valid iff it can be derived from a primitive regular expression by a finite number of applications of operators.

Regular Language:- Any set(language) represented by a regular expression is called a **Regular language**. If for example, $a, b \in \Sigma$, then

- $R = a$ denotes the $L = \{a\}$
- $R = a.b$ denotes $L = \{ab\}$ concatenation
- $R = a + b$ denotes $L = \{a, b\}$ Union
- $R = a^*$ denotes the set $\{\epsilon, a, aa, aaa, \dots\}$ known as Kleene closure.
- $R = a^+$ Positive closure $\{a, aa, aaa\dots\}$
- $R = (a + b)^*$ denotes $\{a, b\}^*$

Operators

When we view a in Σ as a regular expression, we denote it by a .

- If R is a regular expression, then (R) is also a regular expression.
- The iteration (or closure) of a regular expression R written as R^* , is also a regular expression.
- The iteration (or closure) of a regular expression R written as R^+ , is also a regular expression.
- The concatenation of two regular expressions R_1 and R_2 , written as $R_1 R_2$, is also a regular expression.
- The union of two regular expressions R_1 and R_2 , written as $R_1 + R_2$, is also a regular expression.
- The precedence order to solve is ()Bracket, * (Kleene Closure), + Positive Closure, Concatenation, Union.

IDENTITIES FOR Regular Expression

- Two regular expressions P and Q are equivalent (we write $P = Q$)
 - if P and Q represent the same set of strings.
- Every regular expression can generate only one regular language but, a regular language can be generated by more than one regular expression i.e. means two different regular expression can generate same language.
- Two regular expression are said to be equal if they generate same language.
- $r_1 = a^*$
- $r_2 = a^* + (aa)^*$



हमारी language बताओ

- $R=\{a\}$
- $R=\{a + b\}$
- $R=\{a + b + c\}$
- $R=\{a.b\}$
- $R=\{a.b + a\}b$

Q Let S and T be language over $\Sigma = \{a, b\}$ represented by the regular expressions $(a + b^*)^*$ and $(a + b)^*$, respectively. Which of the following is true? (GATE-2000) (1 Marks)

(A) $S \subset T$

(B) $T \subset S$

(C) $S = T$

(D) $S \cap T = \emptyset$

FINITE AUTOMATA AND regular expression

Q Design a regular expression that represent a language ‘L’, where $L=\{a\}$ over the alphabet $\Sigma=\{a\}$.

Q design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$, where every accepted string 'w' starts with substring s

i) $s = b$

ii) $s = ab$

iii) $s = abb$

Q design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$. where every accepted string 'w' ends with substring 's'.

- i) s = ab
- ii) s = aa
- iii) s = bab

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$. where every accepted string 'w' contains sub string s.

i) abb

ii) aba

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with a.

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with same symbol.

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with different symbol.

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w , is like $w=SX$.

i) $s= aa/bb$

ii) $s=aaa/bbb$

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w, is like $w=XS$.

i) $s= aa/bb$

ii) $s=aaa/bbb$

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w, is like $w=XSX$.

i) $s= aa/bb$

ii) $s=aaa/bbb$

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' accepted must be like

- i) $|w| = 3$
- ii) $|w| \leq 3$
- iii) $|w| \geq 3$

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain exactly two a's.

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain at least two a's.

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain at most two a's.

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$ such that for every accepted string 2nd from left end is always b.

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$ such that for every accepted string 4th from right end is always a.

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' where $|W| = 0(\text{mod } 3)$?

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' where $|W| = 3(\text{mod } 4)$?

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' where $|W|_a \equiv 0 \pmod{3}$?

Q Design a regular expression that represent all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' where $|W|_a = 2 \pmod 3$?

$$Q L = \{a^m b^n \mid m, n \geq 0\}$$
$$Q L = \{a^m b^n \mid m \geq 1, n \geq 1\}$$
$$Q L = \{a^m b^n \mid m \geq 2, n \geq 3\}$$

$$Q L = \{a^m b^n c^p \mid m, n, p \geq 1\}$$
$$Q L = \{a^n b^n \mid n \geq 1\}$$

$Q L = \{a^m b^n \mid m + n = \text{even}\}$

Q Write a regular expression for the language, $L = \{w c w^r \mid w, c \in \{a, b\}^+\}$

Algebraic Properties of regular expression

Closure Property - regular expressions satisfy closure property with respect to Union, Concatenation and kleene closure. If R_1 and R_2 are regular expression then the following will also be regular expression.

$$r_1 + r_2$$

$$r_1 \cdot r_2$$

$$r_1^*$$

$$r_1^+$$

Associative Property- regular expression satisfy associative property with respect to union and intersection

$$(r_1 + r_2) + r_3 \boxed{\quad} r_1 + (r_2 + r_3)$$

$$(r_1 \cdot r_2) \cdot r_3 \boxed{\quad} r_1 \cdot (r_2 \cdot r_3)$$

Identity Property- The identity property is satisfied as follows-

$$r \cdot \boxed{\quad} = r$$

$$r + \boxed{\quad} = r$$

Inverse Property- The inverse property is not satisfied with respect to concatenation and union.

$$r \cdot \boxed{\quad} = \epsilon$$

$$r + \boxed{\quad} = \phi$$

Commutative Property-regular expressions are commutative with respect to union but not with respect to concatenation.

$$r_1 + r_2 \quad \boxed{} \quad r_2 + r_1$$

$$r_1 \cdot r_2 \quad \boxed{} \quad r_2 \cdot r_1$$

Distributive Property-regular expression satisfy this property as follows-

$$r_1(r_2 + r_3) \boxed{\quad} r_1r_2 + r_1r_3$$

$$(r_1 + r_2)r_3 \boxed{\quad} r_1r_3 + r_2r_3$$

$$r_1 + (r_2 \cdot r_3) \boxed{\quad} (r_1 + r_2)(r_1 + r_3)$$

$$(r_1 \cdot r_2) + r_3 \boxed{\quad} (r_1 + r_3) \cdot (r_2 + r_3)$$

Idempotent Property-regular expressions satisfies idempotent property with respect to union but not with respect to concatenation.

$$r_1 + r_1 \boxed{\quad} r_1$$

$$r_1 \cdot r_1 \boxed{\quad} r_1$$

Identities for regular expressions

1. $\phi + r =$

2. $\phi \cdot r = r \cdot \phi =$

3. $\epsilon \cdot r = r \cdot \epsilon =$

4. $\epsilon^* =$

5. $\phi^* =$

1. $r^*r^* =$

2. $r.r^* =$

3. $r^+.r^* =$

4. $r^+ r^+ =$

1. $r^+ \cup r^* =$

2. $r^+ \cap r^* =$

1. $(r^*)^* =$

2. $(r^+)^* =$

3. $((r^+)^*)^* =$

4. $((r^*)^*)r^+ =$

1) $(r_1 + r_2)^* \boxed{\quad} (r_1^* + r_2^*)^*$

2) $(r_1 + r_2)^* \boxed{\quad} (r_1 + r_2^*)^*$

3) $(r_1 + r_2)^* \boxed{\quad} (r_1^* + r_2^*)^*$

4) $(r_1 + r_2)^* \boxed{\quad} (r_1^* + r_2^*)$

$$1. \quad (r_1 + r_2)^* \boxed{\quad} (r_1^* \cdot r_2^*)^*$$

$$2. \quad (r_1 + r_2)^* \boxed{\quad} (r_1 \cdot r_2^*)^*$$

$$3. \quad (r_1 + r_2)^* \boxed{\quad} (r_1^* \cdot r_2)^*$$

$$4. \quad (r_1 + r_2)^* \boxed{\quad} (r_1 \cdot r_2)^*$$

$$5. \quad (r_1 + r_2)^* \boxed{\quad} (r_1^* \cdot r_2^*)$$

$$r_1(r_2 \cdot r_1)^* \square (r_1 \cdot r_2)^* r_1$$

Q Which of the following regular expressions represent(s) the set of all binary numbers that are divisible by three? Assume that the string ϵ is divisible by three.
(GATE 2021) (2 MARKS)

- a) $(0+1(01^*0)^*1)^*$
- b) $(0+11+10(1+00)^*01)^*$
- c) $(0^*(1(01^*0)^*1))^*$
- d) $(0+11+11(1+00)^*00)^*$

Q Which one of the following regular expressions represents the language: the set of all binary strings having two consecutive 0s and two consecutive 1s? **(GATE-2016) (2 Marks)**

- (A) $(0+1)^*0011(0+1)^* + (0+1)^*1100(0+1)^*$
- (B) $(0+1)^*(00(0+1)^*11 + 11(0+1)^*00)(0+1)^*$
- (C) $(0+1)^*00(0+1)^* + (0+1)^*11(0+1)^*$
- (D) $00(0+1)^*11 + 11(0+1)^*00$

Q Consider the languages $L_1 = \Phi$ and $L_2 = \{a\}$. Which one of the following represents $L_1 L_2^* \cup L_1^*$ (GATE-2013) (1 Marks)?

(A) $\{\epsilon\}$

(B) Φ

(C) a^*

(D) $\{\epsilon, a\}$

Q Which one of the following languages over the alphabet {0,1} is described by the regular expression: $(0+1)^*0(0+1)^*0(0+1)^*$? **(GATE-2009) (2 Marks)**

- (A)** The set of all strings containing the substring 00.
- (B)** The set of all strings containing at most two 0's.
- (C)** The set of all strings containing at least two 0's.
- (D)** The set of all strings that begin and end with either 0 or 1.

Q Which of the following regular expressions describes the language over {0, 1} consisting of strings that contain exactly two 1's? **(GATE-2008) (1 Marks)**

- (A)** $(0 + 1)^* 11(0 + 1)^*$ **(B)** $0^* 110^*$
- (C)** $0^* 10^* 10^*$ **(D)** $(0 + 1)^* 1(0 + 1)^* 1 (0 + 1)^*$

Q The regular expression $0^*(10^*)^*$ denotes the same set as **(GATE-2003)**

(1 Marks)

(A) $(1^*0)^*1^*$

(B) $0 + (0 + 10)^*$

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

(D) none of these

Q Which one of the following regular expressions is NOT equivalent to the regular expression $(a + b + c)^*$? **(GATE-2004) (2 Marks)**

(A) $(a^* + b^* + c^*)^*$

(B) $(a^*b^*c^*)^*$

(C) $((ab)^* + c^*)^*$

(D) $(a^*b^* + c^*)^*$

Q Which of the following is correct

a) $(xx)^*y = x(xy)^*$

b) $(xy)^*x = x(yx)^*$

c) $x(xy)^* = (xx)^*y$

d) $(xy)^* = (yx)^*$

Q find the no of strings of length ≤ 3 generated by the $(a + ab)^+$

Q which of the following statements are correct

- a) r^* , r^+ always represent finite language.
- b) r^* , r^+ always represent infinite language.
- c) $r^* = r^+$ if and only if $r = \epsilon$
- d) $r^* = r^+ = r$ if $r = \Phi$

EQUIVALENCE BETWEEN Regular Expression AND FINITE AUTOMATA

- **ARDEN'S THEOREM** is the mechanism for the construction of a regular expression from a finite automaton.

Q Consider a DFA and convert it into regular expression using Arden's theorem?

$$\delta(A, a) = A$$

$$\delta(A, b) = B$$

$$\delta(B, a) = B$$

$$\delta(B, b) = B$$

A is the initial state and B Is the final state

Q Consider a DFA and convert it into regular expression using Arden's theorem?

$$\delta(A, b) = A$$

$$\delta(A, b) = B$$

$$\delta(B, a) = C$$

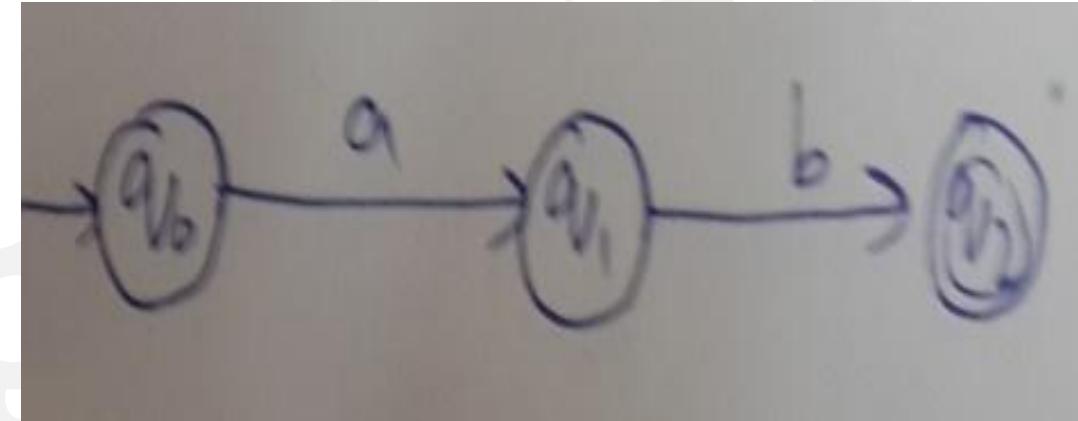
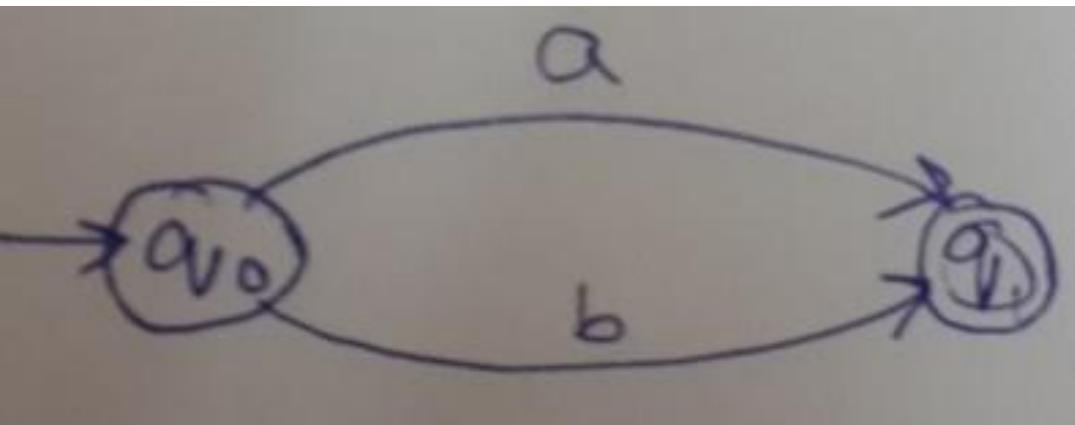
$$\delta(C, b) = C$$

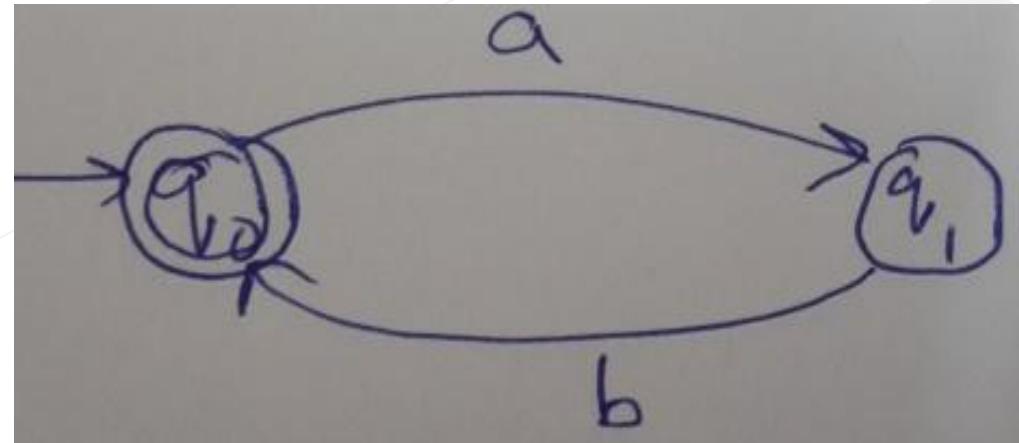
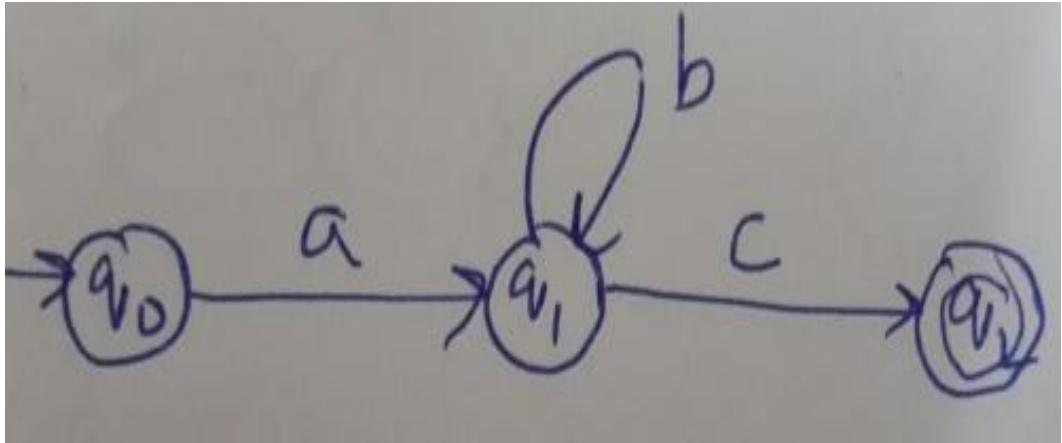
A is the initial state and B,C Is the final state

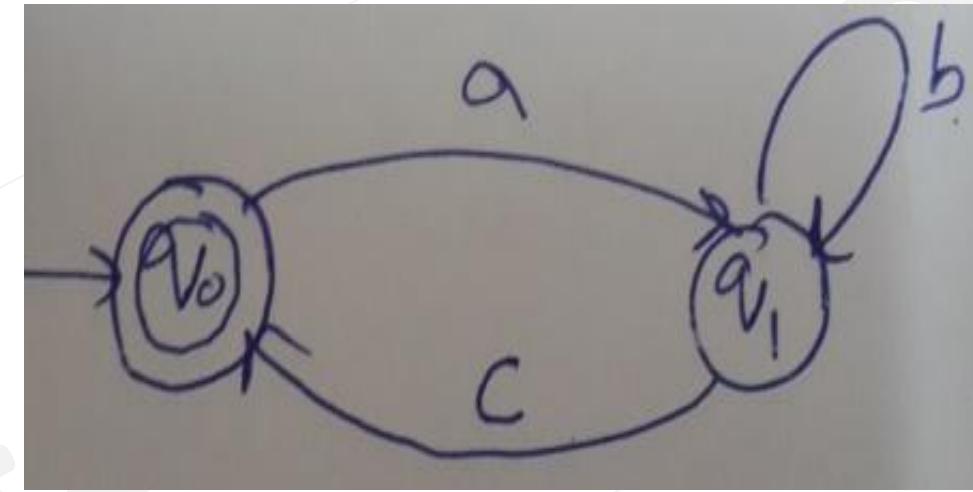
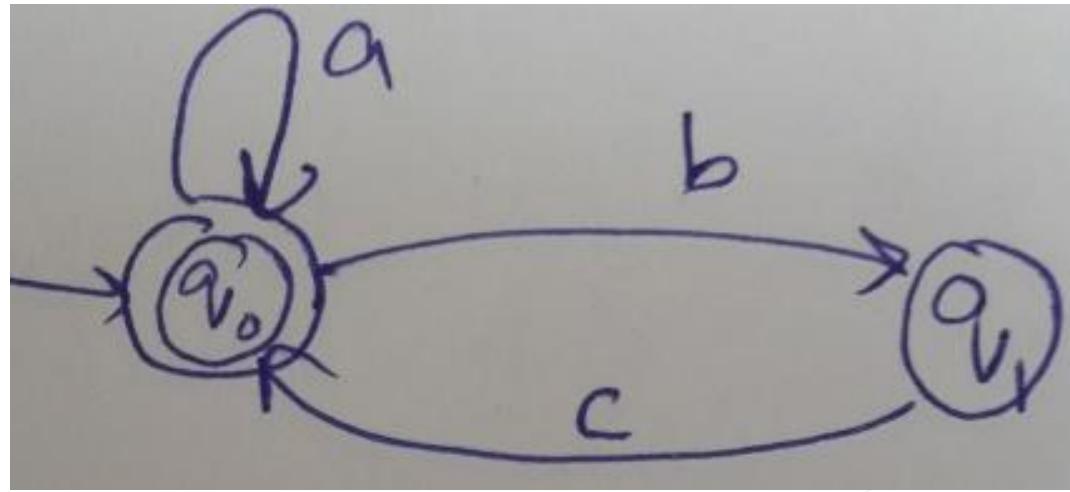
- Steps used-
 - For, every individual state of the DFA, write an expression for every incoming and outgoing input alphabet.
 - Apply Arden's theorem as follows-
 - If P is free from NULL, then equation $R=Q+RP$ has unique solution, $R=QP^*$
 - If P contains NULL, then equation $R=Q+RP$ has infinitely many solutions.

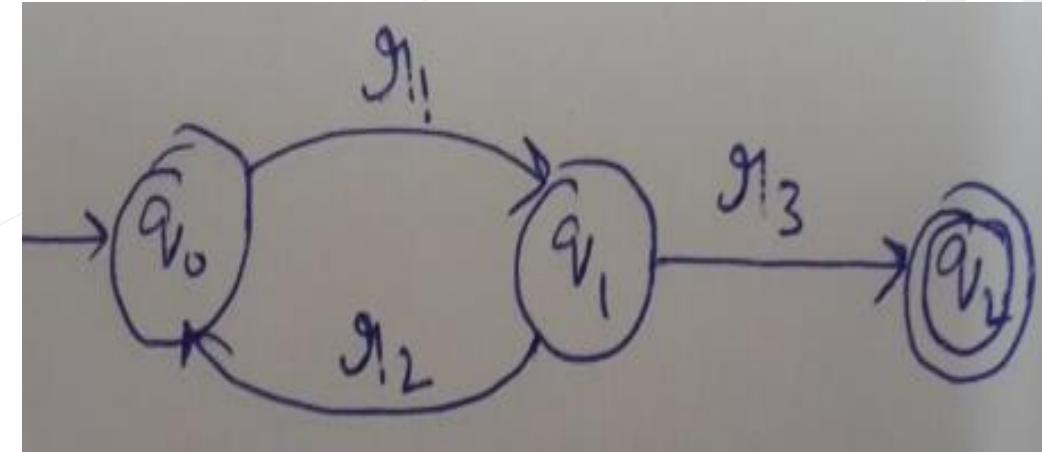
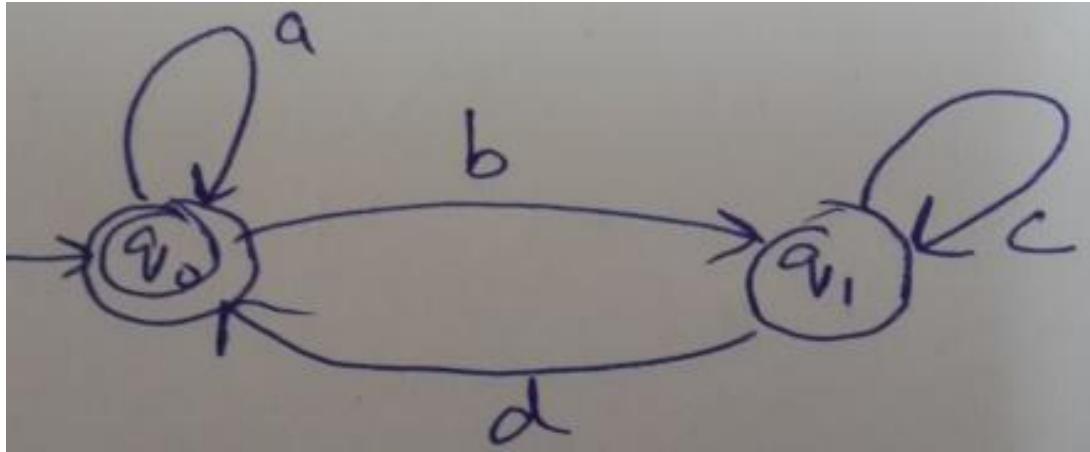
Conversion from Finite Automata to regular expression

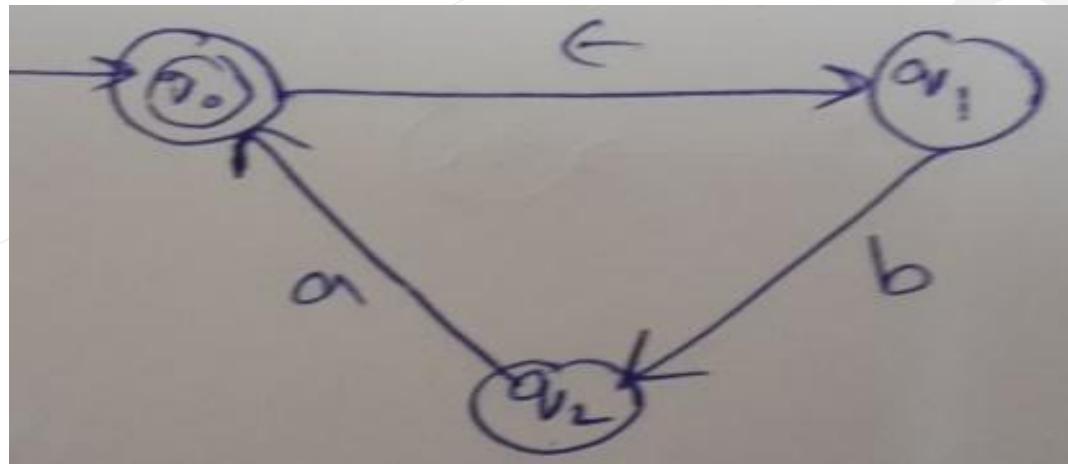
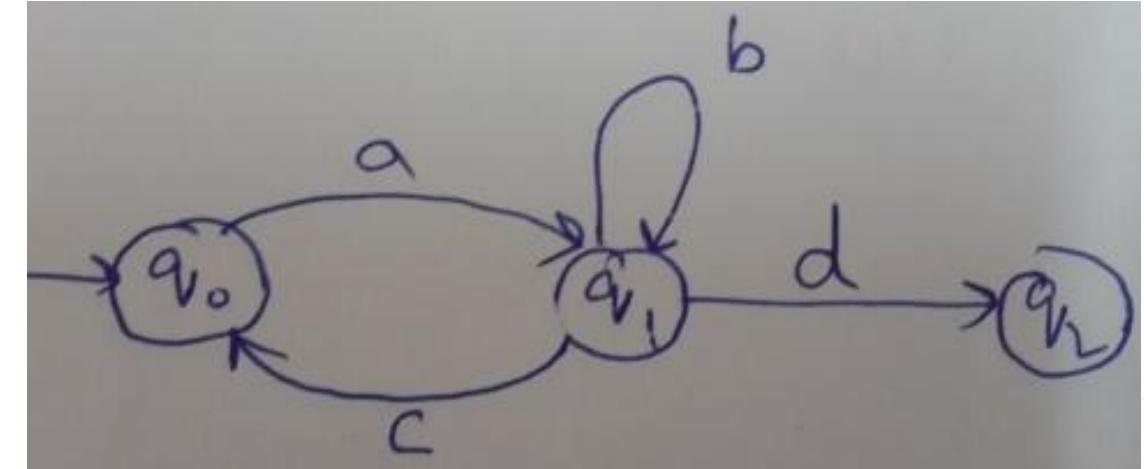
Q Write regular expressions for the following machines

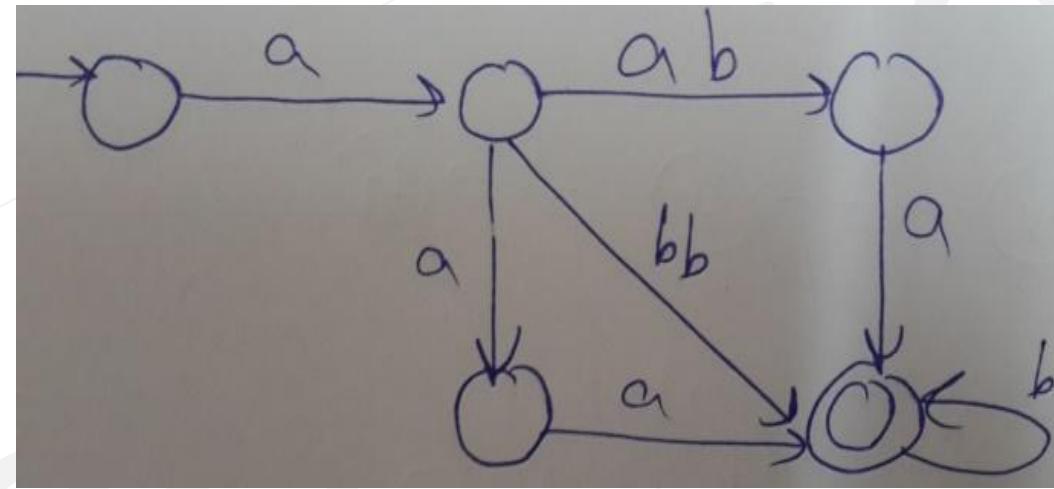
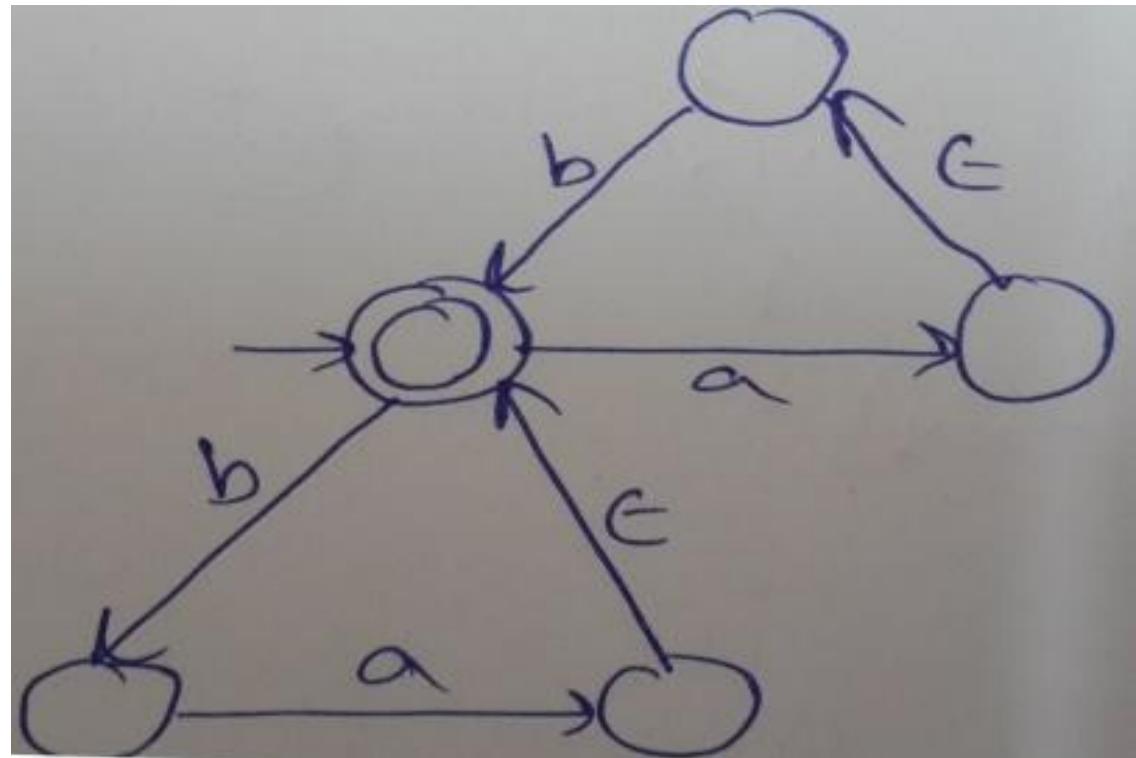


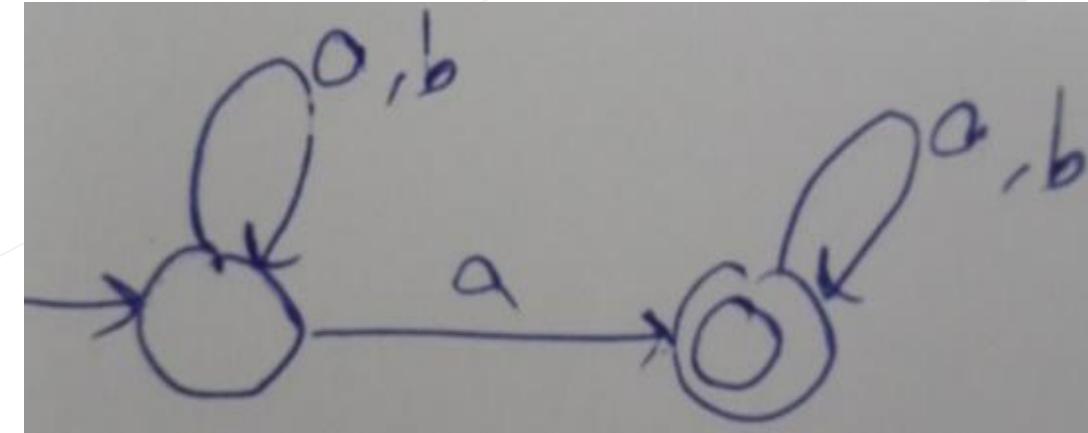
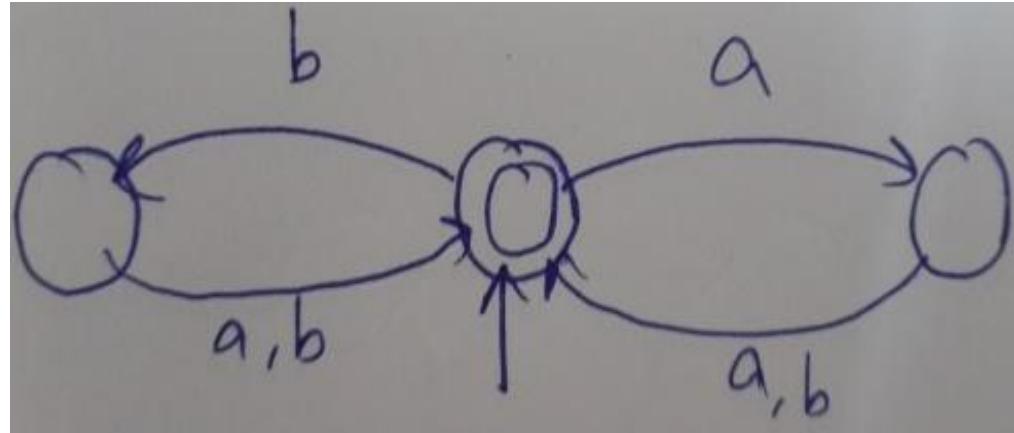


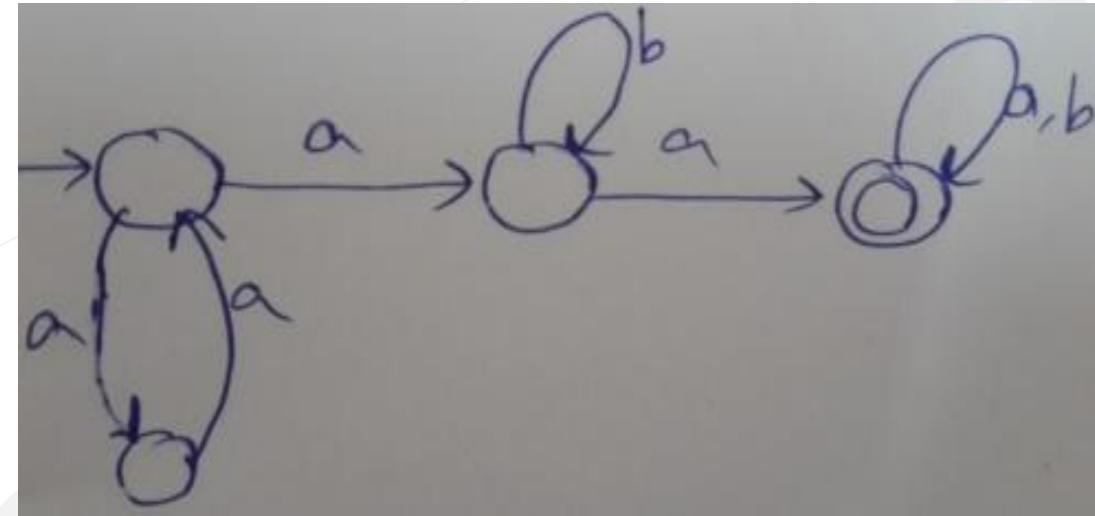
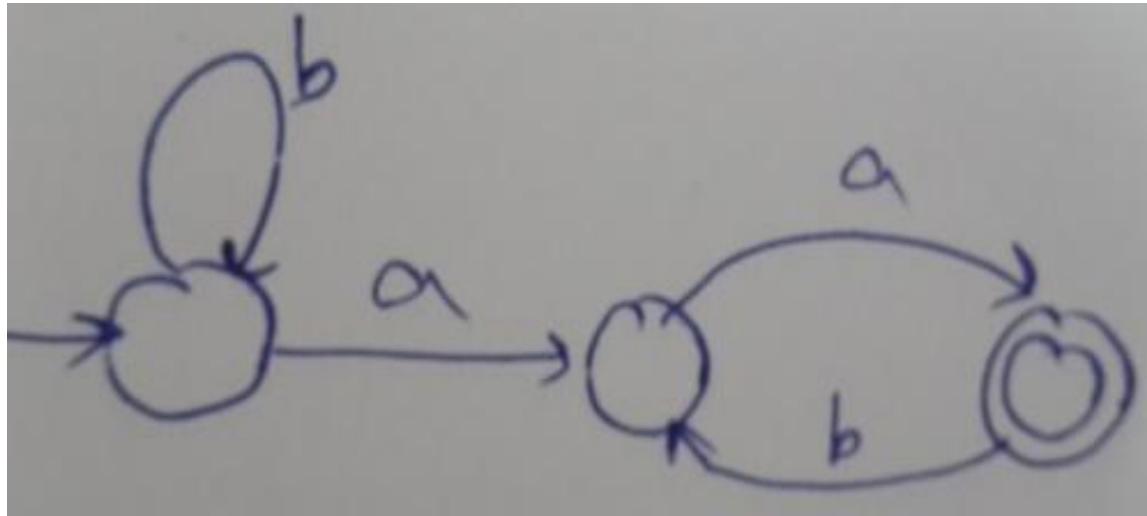


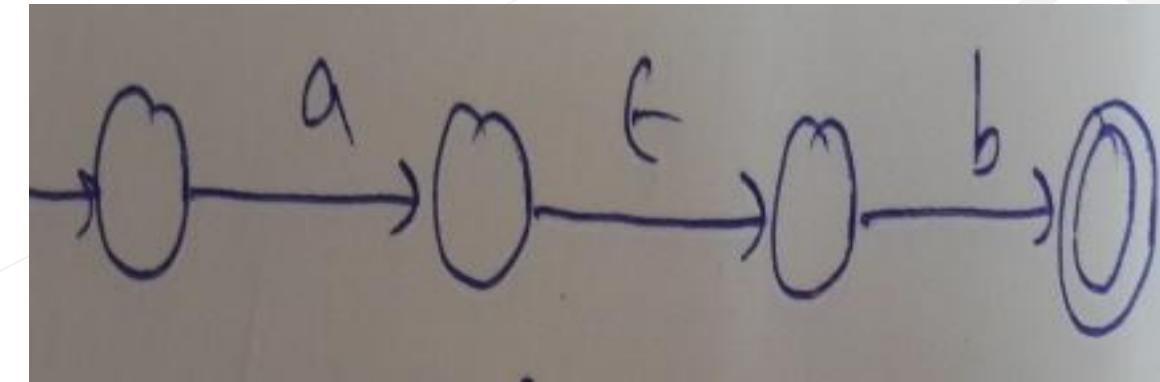
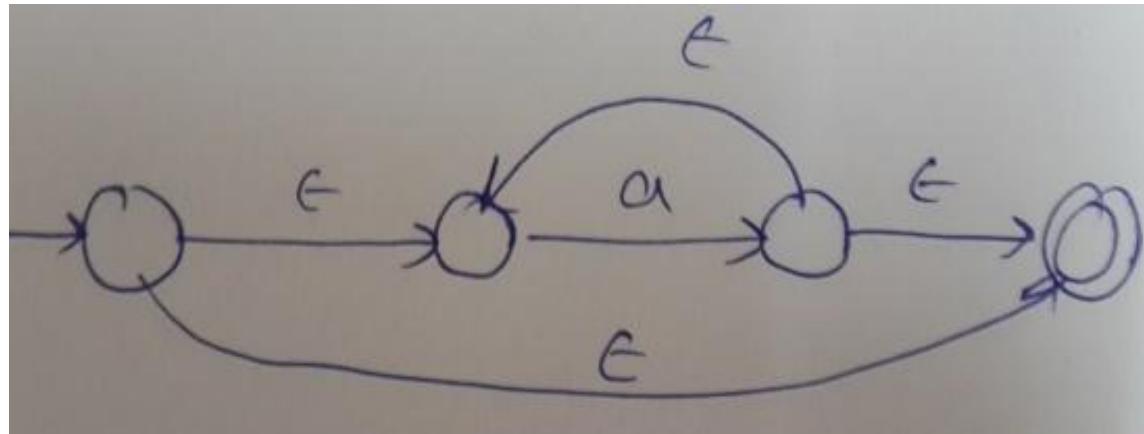










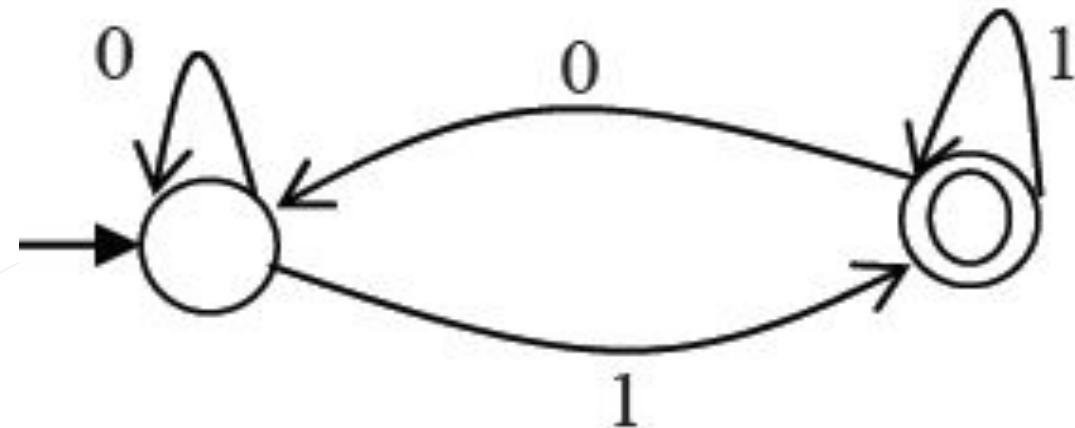


Q Which of the regular expressions given below represent the following DFA? (GATE-2014) (2 Marks)

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

II) $0^*1^*1+11^*0^*1$

III) $(0+1)^*1$



- (A) I and II only
- (B) I and III only
- (C) II and III only
- (D) I, II, and III

Q Let $L = \{w \in (0+1)^* \mid w \text{ has even number of } 1s\}$, i.e. L is the set of all bit strings with even number of 1s. Which one of the regular expression below represents L? **(GATE-2010)(2 Marks)**

a) $(0^* 1 0^* 1)^*$

b) $0^* (1 0^* 1 0^*)^*$

c) $0^* (1 0^* 1^*)^* 0^*$

d) $0^* 1 (1 0^* 1)^* 1 0^*$

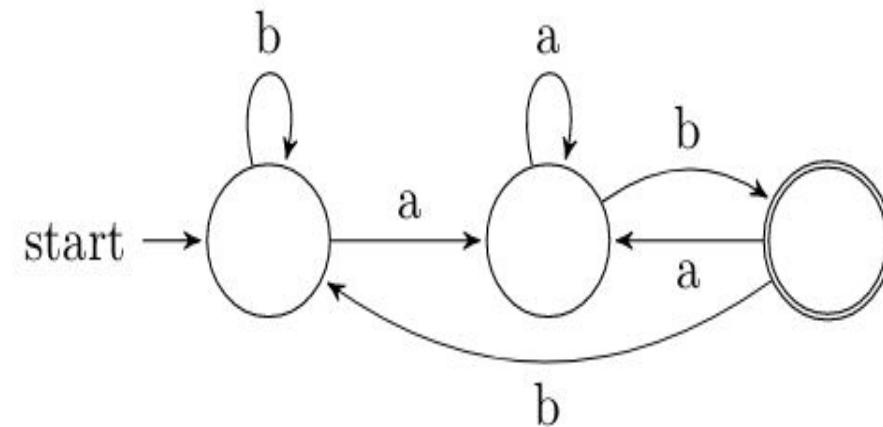
Q If the final states and non-final states in the DFA below are interchanged, then which of the following languages over the alphabet {a, b} will be accepted by the new DFA? **(GATE-2008) (1 Marks)**

(A) Set of all strings that do not end with ab

(B) Set of all strings that begin with either an a or a b

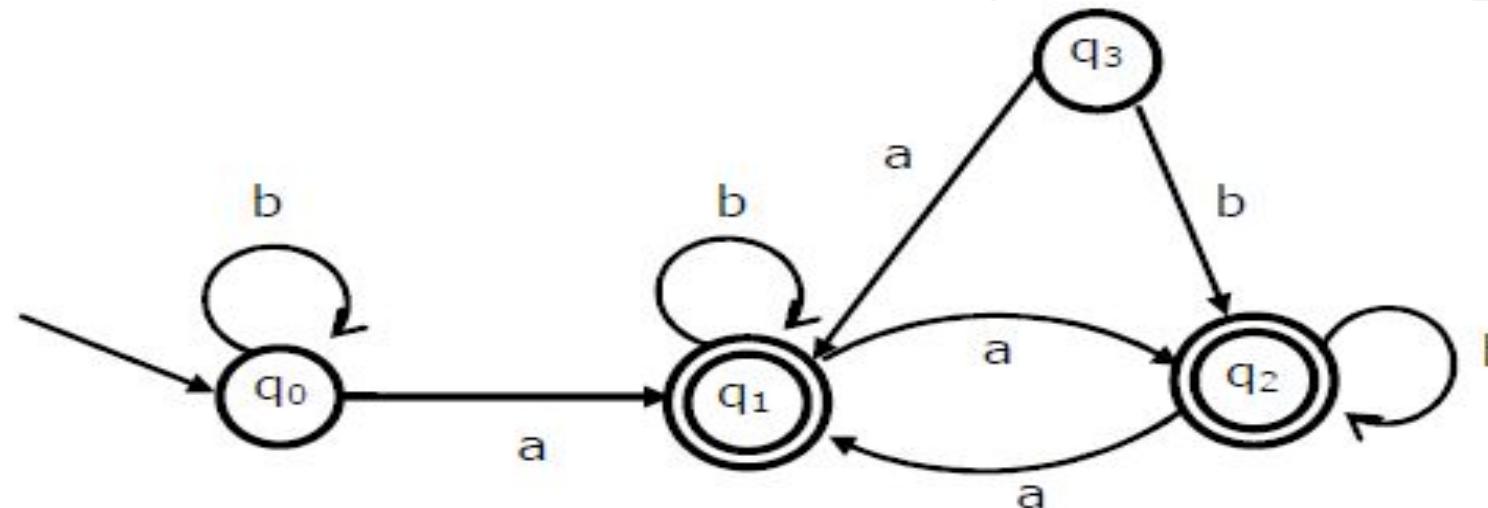
(C) Set of all strings that do not contain the substring ab

(D) The set described by the regular expression $b^*a a^*(ba)^*b^*$



Q Consider the following Finite State Automaton. The language accepted by this automaton is given by the regular expression **(GATE-2007) (2 Marks)**

- (A) $b^*ab^*ab^*ab^*$ (B) $(a+b)^*$ (C) $b^*a(a+b)^*$ (D) $b^*ab^*ab^*$



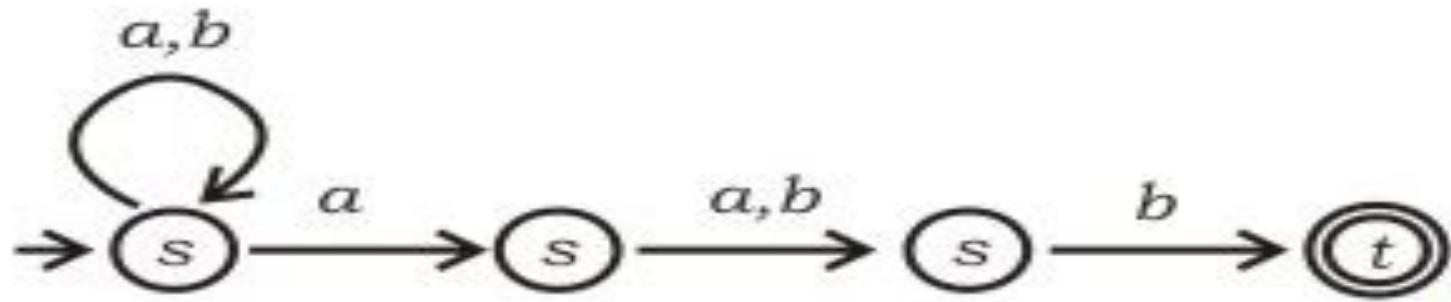
Q Which regular expression best describes the language accepted by the non-deterministic automaton below? **(GATE-2006) (1 Marks)**

(A) $(a + b)^* a(a + b)b$

(B) $(abb)^*$

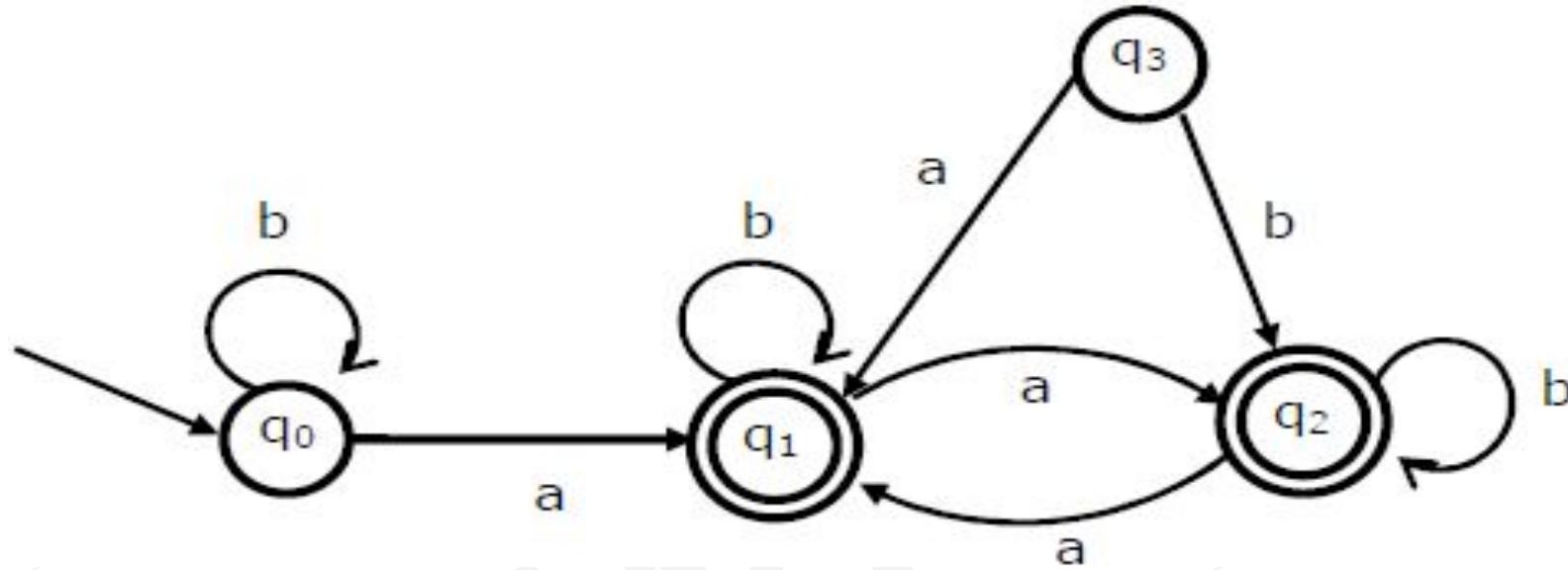
(C) $(a + b)^* a(a + b)^* b(a + b)^*$

(D) $(a + b)^*$



Q Consider the automata given in previous question. The minimum state automaton equivalent to the above FSA has the following number of states (**GATE-2007**) (1 Marks)

- (A) 1 (B) 2 (C) 3 (D) 4



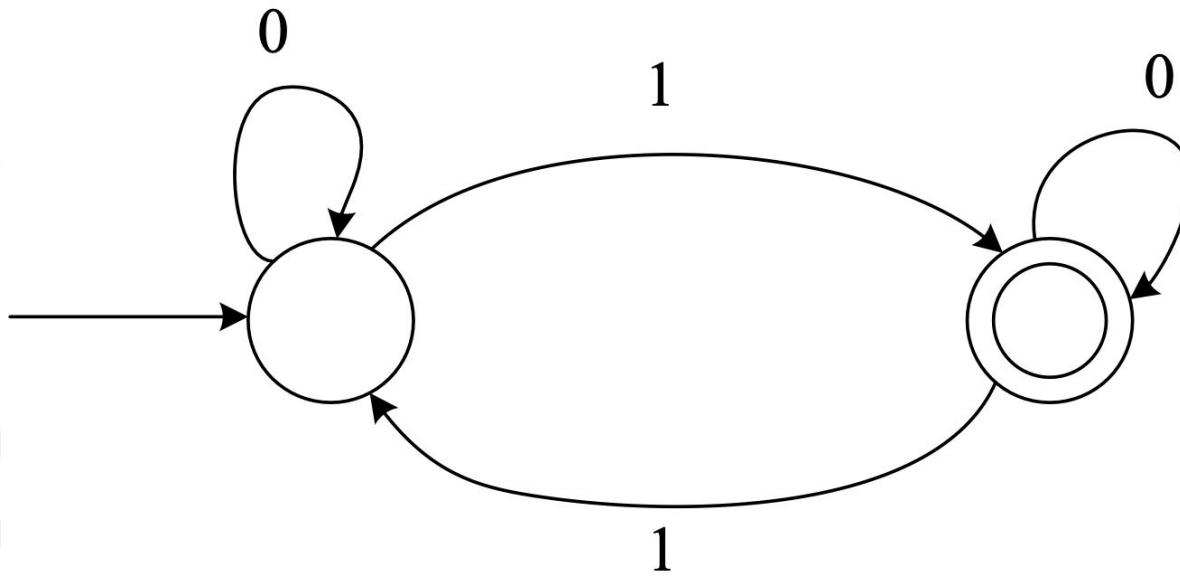
Q. Which one of the following regular expressions is equivalent to the language accepted by the DFA given below? (Gate 2024 CS)(1 Mark)(MCQ)

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

(b) $0^*(10^*11)^*0^*$

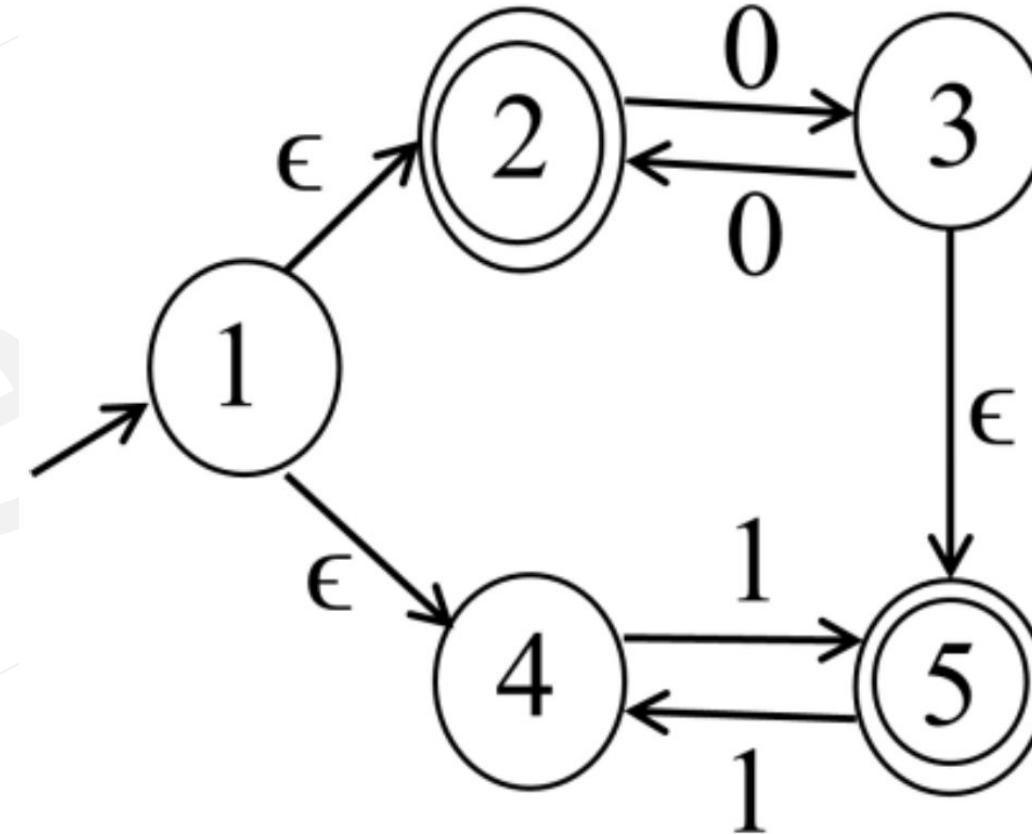
(c) $0^*1(0101^*1)^*0^*$

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



Q. Let M be the 5-state NFA with ϵ -transitions shown in the diagram below. Which one of the following regular expressions represents the language accepted by M ? (Gate 2024 CS) (2 Marks)(MCQ)

- (a) $(00)^* + 1(11)^*$
- (b) $0^* + (1 + 0(00)^*)(11)^*$
- (c) $(00)^* + (1 + (00)^*)(11)^*$
- (d) $0 + 1(11)^* + 0(11)^*$



Conversion from regular expression in Finite Automata

- i) R^*
- ii) $(R_1 \cdot R_2)^*$
- iii) $(R_1 + R_2)^*$

iv) $(R_1 * R_2 . R_3 *)$

v) $R = a^* b(ab)^*$

vi) $R = (a + ba)^* ab^*$

vii) $R = (aa + aaa)^*$

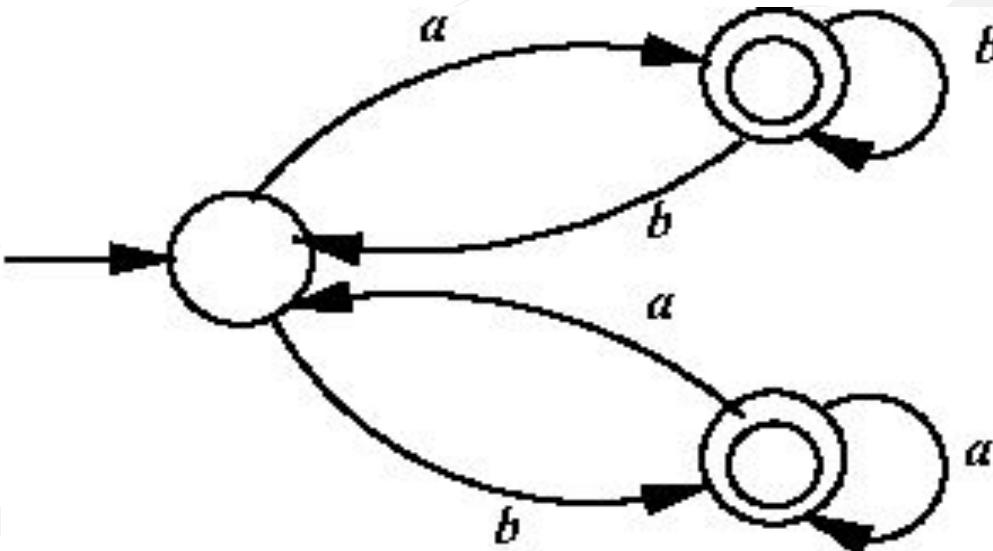
viii) $R = (a + aaaaa)^*$

1. $0^* 10^* 10^*$
2. $((ab)^* + c^*)^*$
3. $(\in + a + aa + aaa)b^* + (a + b)^* ba(a + b)^*$
4. $0^*(10^* 1^*)^*0^*$

1. $a^*b(b^* + aa^*b)^*$
2. $((11^*0+0)(0 + 1)^*0^*1^*)$
3. $(ab+ bc+ acc).(a+ bc)^*.(\in + bc^*a)^*$

Q Which one of the following regular expressions correctly represents the language of the finite automaton given below? (GATE 2022) (1 MARKS)

- (A) $ab^*bab^* + ba^*aba^*$
- (B) $(ab^*b)^*ab^* + (ba^*a)^*ba^*$
- (C) $(ab^*b + ba^*a)^* (a^* + b^*)$
- (D) $(ba^*a + ab^*b)^*(ab^* + ba^*)$



Q Consider the language L given by the regular expression $(a + b)^*b(a + b)$ over the alphabet $\{a, b\}$. The smallest number of states needed in a deterministic finite-state automaton (DFA) accepting L is _____. **(GATE-2017) (1 Marks)**

Q The number of states in the minimum sized DFA that accepts the language defined by the regular expression $(0+1)^*(0+1)(0+1)^*$ is _____ (GATE-2016) (2 Marks)

Q Let T be the language represented by the regular expression $\Sigma^*0011\Sigma^*$ where $\Sigma = \{0, 1\}$. What is the minimum number of states in a DFA that recognizes L' (complement of L)? **(GATE-2015) (2 Marks)**

- (A) 4 (B) 5 (C) 6 (D) 8

Q The number of states in the minimal deterministic finite automaton corresponding to the regular expression $(0 + 1)^*(10)$ is _____ (GATE-2015) (1 Marks)

- (A) 2 (B) 3 (C) 4 (D) 5

Q The length of the shortest string NOT in the language (over $\Sigma = \{a, b\}$) of the following regular expression is _____. (GATE-2014) (2 Marks)

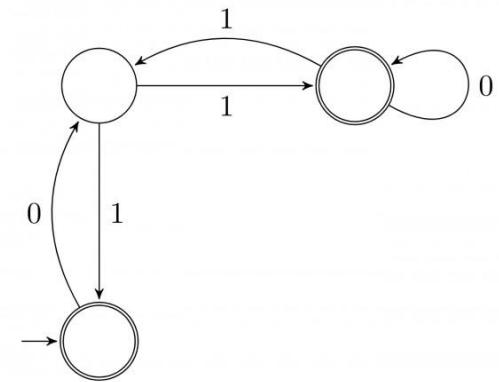
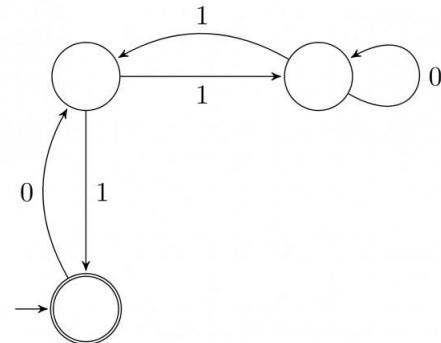
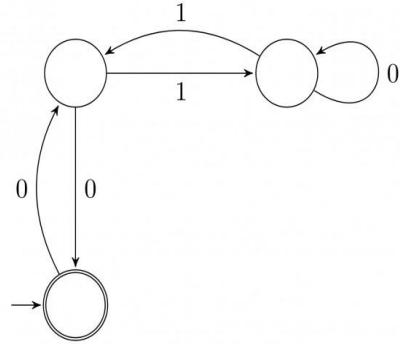
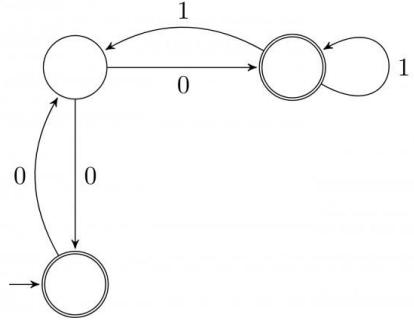
$a^*b^*(ba)^*a^*$

- (A) 2
- (B) 3
- (C) 4
- (D) 5

Q Consider the regular language $L = (111 + 11111)^*$. The minimum number of states in any DFA accepting this language is: **(GATE-2006) (1 Marks)**

- (A) 3
- (B) 5
- (C) 8
- (D) 9

Q Match the following NFAs with the regular expressions they correspond to (GATE-2008) (2 Marks)



1. $\epsilon + 0(01^*1 + 00)^*01^*$

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

3. $\epsilon + 0(10^*1 + 10)^*1$

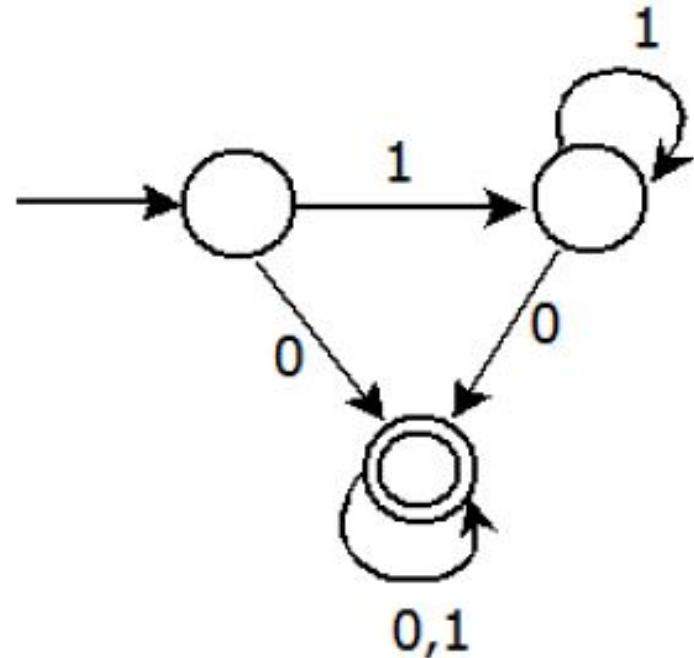
4. $\epsilon + 0(10^*1 + 10)^*10^*$

- (A) P - 2, Q - 1, R - 3, S - 4
- (B) P - 1, Q - 3, R - 2, S - 4
- (C) P - 1, Q - 2, R - 3, S - 4
- (D) P - 3, Q - 2, R - 1, S - 4

Q Consider the DFA given. (GATE-2013) (2 Marks)

Which of the following are FALSE?

1. Complement of $L(A)$ is context-free.
 2. $L(A) = L((11^*0+0)(0 + 1)^*0^*1^*)$
 3. For the language accepted by A , A is the minimal DFA.
 4. A accepts all strings over $\{0, 1\}$ of length at least 2.
- a) 1 and 3 only**
b) 2 and 4 only
c) 2 and 3 only
d) 3 and 4 only



Q the string 1101 does not belong to the set

- a)** $110^*(0+1)^*$
- b)** $1(0+1)^*101$
- c)** $(10)^*(10)^*(00+11)^*$
- d)** $(00 + (11)^*01)^*$

Q Which of the following statements is TRUE about the regular expression 01^*0 ?
(GATE-2005) (1 Marks)

- (A)** It represents a finite set of finite strings.
- (B)** It represents an infinite set of finite strings.
- (C)** It represents a finite set of infinite strings.
- (D)** It represents an infinite set of infinite strings

Q which of the following re, over the $\Sigma = \{0, 1\}$ denotes the set of all strings that not contain 100 as sub-string

- a) $0^*(0+1)^*$
- b) 0^*1010^*
- c) 0^*1^*01
- d) $0^*(10+1)^*$

Q. Consider the following two regular expressions over the alphabet {0,1}: (Gate 2024,CS) (2 Marks) (NAT)

$$r = 0^* + 1^*$$

$$s = 01^* + 10^*$$

The total number of strings of length less than or equal to 5, which are neither in r nor in s , is _____

Q. Let L_1 be the language represented by the regular expression $b^*ab^*(ab^*ab^*)^*$ and $L_2 = \{ w \in (a+b)^* \mid |w| \leq 4\}$, where $|w|$ denotes the length of string w . The number of strings in L_2 which are also in L_1 is _____ (Gate 2024 CS) (2 Marks) (NAT)

Q. Let $\Sigma = \{a, b, c\}$. For $x \in \Sigma^*$, and $\alpha \in \Sigma$, let $\#_\alpha(x)$ denote the number of occurrences of α in x .

Which one or more of the following option(s) define(s) regular language(s)? **(Gate 2025)**

- A) $\{a^m b^n \mid m, n \geq 0\}$
- B) $\{a, b\}^* \cap \{a^m b^n c^{m-n} \mid m \geq n \geq 0\}$
- C) $\{w \mid w \in \{a, b\}^*, \#_a(w) \equiv 2 \pmod{7}, \text{ and } \#_b(w) \equiv 3 \pmod{9}\}$
- D) $\{w \mid w \in \{a, b\}^*, \#_a(w) \equiv 2 \pmod{7}, \text{ and } \#_a(w) = \#_b(w)\}$

Formal Grammar

Language usually contains infinite number of strings. We cannot tabulate each and every string to represent the language, therefore like automata, grammar is also a mathematical model of representing a language, using which we can generate the entire language. Therefore, a grammar is usually thought of as a language generator. A phrase-structure grammar (or simply a grammar) is a 4-tuple (V_N, Σ, P, S) , where

- V_N is a finite nonempty set whose elements are called variables,
- Σ is a finite nonempty set whose elements are called terminals, $V_N \cap \Sigma = \Phi$.
- S is a special variable (i.e., an element of V_N ($S \in V_n$)) called the start symbol. Like every automaton has exactly one initial state, similarly every grammar has exactly one start symbol.
- P is a finite set whose elements are $\alpha \rightarrow \beta$. where α and β are strings on $V_N \cup \Sigma$. α has at least one symbol from V_N , the element of P are called productions or production rules or rewriting rules. $\{\Sigma \cup V_n\}^*$ some writer refers it as total alphabet.

For a formal valid production

$$\alpha \square \beta$$

$$\alpha \in (\Sigma \cup V_n)^* V_n (\Sigma \cup V_n)^*$$

$$\beta \in (\Sigma \cup V_n)^*$$

If $G = (\{S\}, \{0, 1\}, \{S \rightarrow 0S1, S \rightarrow \Lambda\}, S)$, find $L(G)$.

Some points to note about productions

- Reverse substitution is not permitted. For example, if $S \rightarrow AB$ is a production, then we can replace S by AB but we cannot replace AB by S .
- No inversion operation is permitted. For example, if $S \rightarrow AB$ is a production, it is not necessary that $AB \rightarrow S$ is a production.
- To generate a string in the language, one begins with a string consisting of only a single *start symbol*. The *production rules* are then applied in any order, until a string that contains neither the start symbol nor designated *nonterminal symbols* is produced. A sequence of rule applications is called a *derivation*.
- A production rule is applied to a string by replacing one occurrence of the production rule's left-hand side in the string by that production rule's right-hand side.
- $L(G)$ is the set of all terminal strings derived from the start symbol S . G_1 and G_2 are equivalent if $L(G_1) = L(G_2)$.

Defining a language by grammar

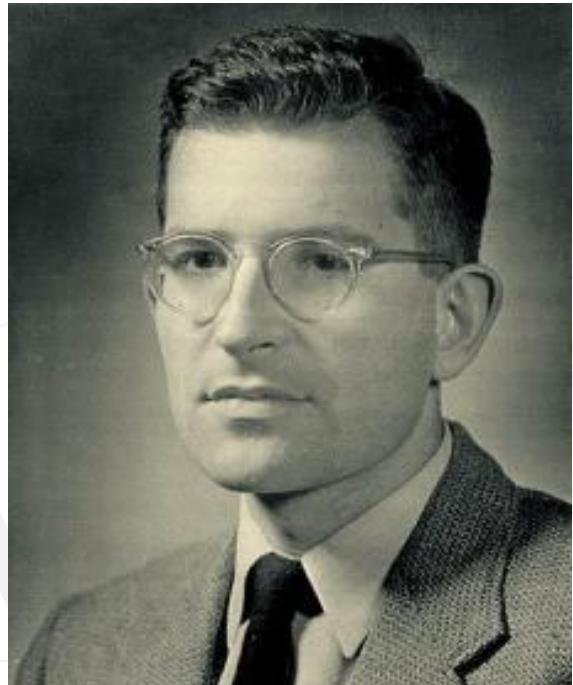
- The concept of defining a language using grammar is, starting from a start symbol using the production rules of the grammar any time, deriving the string. Here every time during derivation a production is used as its LHS is replaced by its RHS, all the intermediate stages(strings) are called sentential forms. The language formed by the grammar consists of all distinct strings that can be generated in this manner.

$$L(G) = \{w \mid w \in \Sigma^*, S \xrightarrow{*} w\}$$

- $\xrightarrow{*}$ (reflexive, transitive closure) means from s we can derive w in zero or more steps

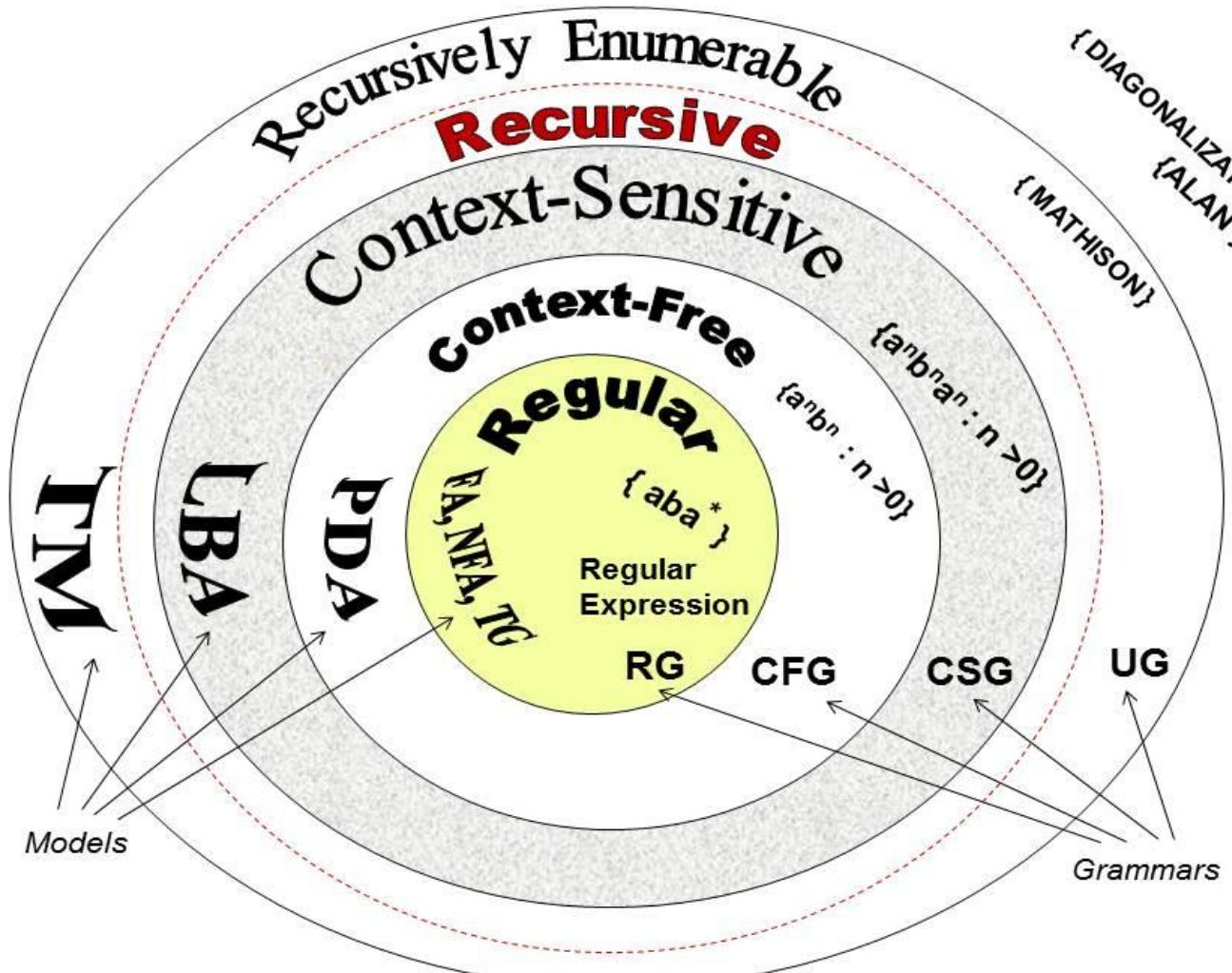
Chomsky Classification of Languages

- Chomsky classified the grammars into four types in terms of productions (types 0-3). This hierarchy of grammars was described by Noam Chomsky in 1956. From type 0 to type 3, we will be putting more and more restrictions. The more restrictive the grammar, the easier the language will be to process; the more liberal the grammar, the more complex the language will become.



LANGUAGES AND AUTOMATA

- Following are the machines that accepts the following grammars.



Type 0 Grammar

- Also known as Unrestricted Grammar, phrase structured grammar, recursively enumerable grammar used to generate recursive enumerable language which is accepted by a Turing machine.
- A type 0 grammar is without any restrictions.
- No restriction on the production rule, that is if there is a production from

$$\alpha \rightarrow \beta$$
$$\begin{aligned}\alpha &\in (\Sigma \cup V_n)^* \text{ and } (\Sigma \cup V_n)^* \\ \beta &\in (\Sigma \cup V_n)^*\end{aligned}$$

Type 1 Grammar

- Also known as case sensitive Grammar, length increasing grammar, non-contracting grammar, used to generate context sensitive language which is accepted by a linear bounded automaton.

$$\alpha A \beta \sqsupseteq \alpha \delta \beta$$
$$\alpha, \beta \in (\Sigma \cup V_n)^* \quad A \in V_n \quad \delta \in (\Sigma \cup V_n)^+$$

or

$$\alpha \sqsupseteq \beta$$
$$\alpha \in (\Sigma \cup V_n)^* \vee \beta \in (\Sigma \cup V_n)^*$$
$$\beta \in (\Sigma \cup V_n)^+$$
$$|\alpha| \leq |\beta|$$

- As from the rule we can understand that we cannot have null production, in order to solve that problem, Production $S \rightarrow \epsilon$, is allowed if S do not appear on the right-hand side of the production.
- A grammar is called type 1 or context-sensitive or context dependent if all its productions are type 1 productions.
- Very difficult to have a parse tree
- FORTRAN, PL1 with CGF we use STD

Type 2 Grammar

- Also known as Context Free Grammar, which will generate context free language that will be accepted by push down automata. (NPDA default case)
- if there is a production, from

$$\alpha \rightarrow \beta$$

$$\begin{array}{l} \alpha \in V_n \\ |\alpha| = 1 \\ \beta \in (\Sigma \cup V_n)^* \end{array}$$

- In other words, the L.H.S. has no left context or right context.
- A grammar is called a type 2 grammar if it contains only type 2 productions.
- Eg ALGOL 60, PASCAL

Type 3 Grammar

- Used to generate Regular Grammar which will generate regular language which will be accepted by finite machine.
- Regular grammar can be of two types either left linear or right linear.
- Left regular grammar, support two types of production
 $A \xrightarrow{} a / Ba$
 $A, B \in V_n \quad |A| = |B| = 1$
 $a \in \Sigma^*$
- Right regular grammar
 $A \xrightarrow{} a / aB$
 $A, B \in V_n \quad |A| = |B| = 1$
 $a \in \Sigma^*$
- however, if left-linear rules and right-linear rules are combined,
the language need no longer be regular.

Q Consider the grammar

$$S \rightarrow aaaS \mid a \mid aa$$

$$L(G) = ?$$

- a) $L(G) = \{w : |w| \bmod 3 = 0\}$
- b) $L(G) = \{w : |w| \bmod 3 = 1 \text{ or } 2\}$
- c) $L(G) = L(a^*)$
- d) $L(G) = L(a^*) - (\lambda)$

Q The basic limitation of FSM is that

- a)** It cannot remember arbitrary large amount of information
- b)** It sometimes fails to recognize grammars that are not regular
- c)** It sometimes fails to recognize grammars that are regular
- d)** All of these

Q Consider the following grammar and identify it's language?

$S \rightarrow aAb$

$A \rightarrow aB / b$

$B \rightarrow c$

Q Consider the following grammar and identify it's language?

S -> **AB / Bb**

A -> **b / c**

B -> **d**

Q Consider the following grammar and identify it's language?

$S \rightarrow aSb / \epsilon$

Q Consider the following grammar and identify it's language?

$S \rightarrow aA / abS$

$A \rightarrow bS / b$

Q Consider the following grammar and identify it's language?

$S \rightarrow aAB$

$A \rightarrow aA / \epsilon$

$B \rightarrow b$

Q Consider the following grammar and identify it's language?

$S \rightarrow AB$

$A \rightarrow aA / \epsilon$

$B \rightarrow bB / b$

Q Consider the following grammar and identify it's language?

$S \rightarrow aSa / bSb / \epsilon$

Q If G is a grammar with productions

$$S \rightarrow SaS \mid aSb \mid bSa \mid SS \mid \epsilon$$

where S is the start variable, then which one of the following strings is not generated by G? **(GATE-2017) (1 Marks)**

(A) abab

(B) aaab

(C) abba

(D) babba

Q Consider the following grammar (GATE – 2004) (2 Marks)

$$S \rightarrow bS \mid aA \mid b$$

$$A \rightarrow bA \mid aB \mid$$

$$B \rightarrow bB \mid aS \mid a$$

Let $N_a(w)$ and $N_b(w)$ denote the number of a's and b's in a string w respectively. The language $L(G) \cap \{a, b\}^*$ generated by G is

- a)** $\{w \mid N_a(w) > 3N_b(w)\}$
- b)** $\{w \mid N_b(w) > 3N_a(w)\}$
- c)** $\{w \mid N_b(w) = 3k, k \in \{0, 1, 2, \dots\}\}$
- d)** $\{w \mid N_a(w) = 3k, k \in \{0, 1, 2, \dots\}\}$

Q Consider the regular grammar: (GATE-2005) (1 Marks)

$S \rightarrow Xa \mid Ya$

$X \rightarrow Za$

$Z \rightarrow Sa \mid \epsilon$

$Y \rightarrow Wa$

$W \rightarrow Sa$

Where S is the starting symbol, the set of terminals is {a} and the set of non – terminals is {S, W, X, Y, Z}. We wish to construct a deterministic is (DFA) to recognize the same language. What is the minimum number of states required for the DFA?

- a) 2
- b) 3
- c) 4
- d) 5

Q What is the regular expression for the language generated by

$S \rightarrow aS \mid bA$

$A \rightarrow d \mid ccA$

- a) a^*bd
- b) $a^* (bd)(bcc)^* d$
- c) $a^* b(cc)^* d$
- d) None of these

Q Write the grammar for the regular expression a^*b^*

- a) $S \rightarrow AB, A \rightarrow aA \mid bB \mid \epsilon, B \rightarrow bB \mid aA \mid \epsilon$
- b) $S \rightarrow AB, A \rightarrow aA \mid \epsilon, B \rightarrow bB \mid \epsilon$
- c) $S \rightarrow ab \mid \epsilon, A \rightarrow aA \mid \epsilon, B \rightarrow bB \mid \epsilon$
- d) None of these

Q Language L_1 is defined by the grammar: $S_1 \square aS_1b \mid \epsilon$

Language L_2 is defined by the grammar: $S_2 \square abS_2 \mid \epsilon$

Consider the following statements **(GATE-2016) (2 Marks)**

P: L_1 is regular

Q: L_2 is regular

Which one of the following is TRUE?

(A) Both P and Q are true

(B) P is true and Q is false

(C) P is false and Q is true

(D) Both P and Q are false

Q Consider alphabet $\Sigma = \{0, 1\}$, the null/empty string \in and the sets of strings X_0 , X_1 and X_2 generated by the corresponding non-terminals of a regular grammar. X_0 , X_1 and X_2 are related as follows (GATE-2015) (2 Marks)

$$X_0 = 1 X_1 \quad X_1 = 0 X_1 + 1 X_2 \quad X_2 = 0 X_1 + \{\in\}$$

Which one of the following choices precisely represents the strings in X_0 ?

- (A) $10 (0^* + (10)^*)1$
- (B) $10 (0^* + (10)^*)^*1$
- (C) $1(0^* + 10)^*1$
- (D) $10 (0 + 10)^*1 + 110 (0 + 10)^*1$

Designing Grammar

Q Design a grammar that generates a language 'L', where $L=\{a\}$ over the alphabet $\Sigma=\{a\}$.

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$, where every accepted string ‘w’ starts with substring s

- i) $s = b$
- ii) $s = ab$
- iii) $s = abb$

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$. where every accepted string ‘w’ ends with substring ‘s’.

- i) $s = ab$
- ii) $s = aa$
- iii) $s = bab$

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$. where every accepted string 'w' contains sub string s.

i) abb

ii) aba

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with a.

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with same symbol.

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string start and end with different symbol.

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w , is like $w=SX$.

i) $s= aa/bb$

ii) $s=aaa/bbb$

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w , is like $w=XS$.

i) $s= aa/bb$

ii) $s=aaa/bbb$

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$ such that every accepted string w , is like $w=XSX$.

i) $s= aa/bb$

ii) $s=aaa/bbb$

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' accepted must be like

- i) $|w| = 3$
- ii) $|w| \leq 3$
- iii) $|w| \geq 3$

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain exactly two a's.

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain at least two a's.

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$, such that every string accepted must contain at most two a's.

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$ such that for every accepted string 2nd from left end is always b.

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' where $|W| = 0(\text{mod } 3)$?

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' where $|W| = 3(\text{mod } 4)$?

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' where $|W|_a = 0(\text{mod } 3)$?

Q Design a grammar that generates all strings over the alphabet $\Sigma = \{a, b\}$, such that every string 'w' where $|W|_a = 2 \pmod 3$?

- We actually did Regular Expression to Grammar
- Regular Grammar to Regular Expression

Q Consider the following Left recursive grammar and convert them into Regular Expression?

A -> Aα / β

A -> Aα / β₁ / β₂

A -> Aα / β₁ / β₂ ----- / β_n

$$A \rightarrow A\alpha_1 / A\alpha_2 / \beta$$
$$A \rightarrow A\alpha_1 / A\alpha_2 / \dots / A\alpha_n / \beta$$

A \rightarrow A α_1 / A α_2 /-----A α_n / β_1 / β_2 -----/ β_m

S -> 01S / 01

knowledgeGate

S -> 0011S / 01 / 10

knowledgeGate

S -> 01A / B11

A -> 011A / 01

B -> 101B / 11

S -> 011A / 101B

A -> 110A / 00

B -> 11B / S

S -> 01S / 1

knowledgeGate

S -> 011S / 01

knowledgeGate

$S \rightarrow 001S / 10A$

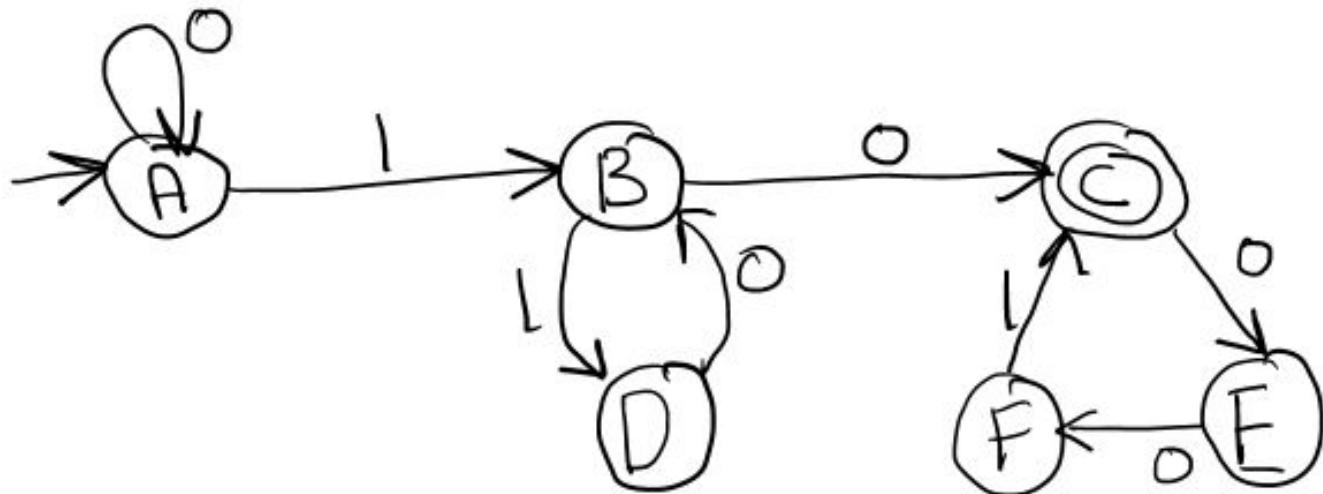
$A \rightarrow 101A / 0 / 1$

- We actually did Right Regular Grammar to (Finite Automata)
- Left Regular Grammar to (www.knowledgigate.in)

- Reverse the right hand side of every production.
- Construct the NFA.
- Interchange initial and final state.
- Change the direction of edges.

$S \rightarrow S10 / 01$

- Finite Automata to Regular Grammar



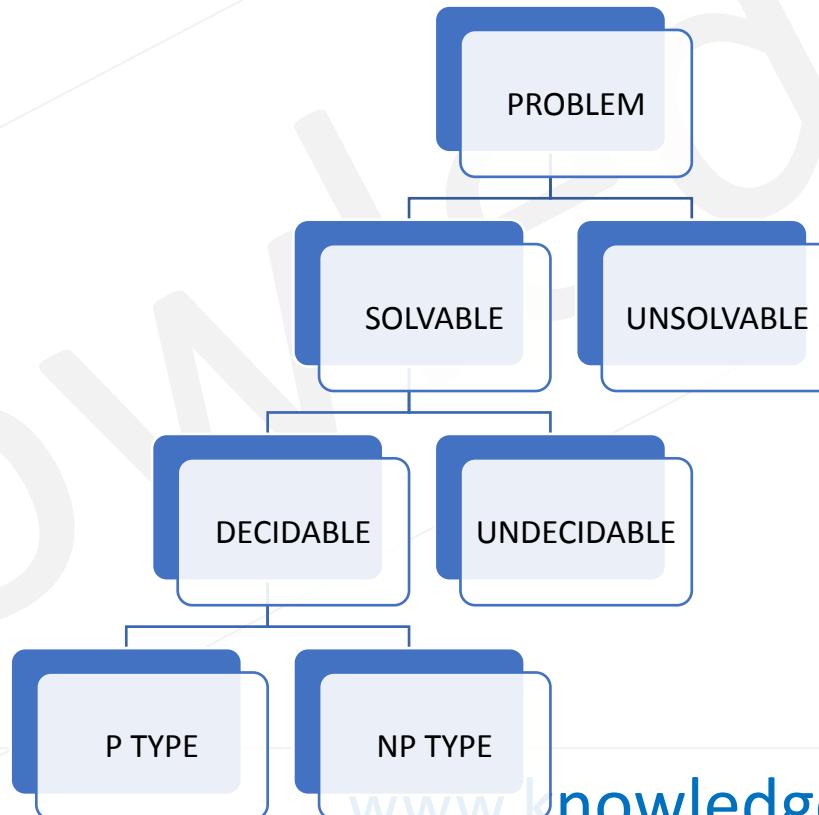
- We actually did (Finite Automata) To Right Regular Grammar
- Finite Automata to Left Regular Grammar

- Obtain the regular Expression
- Reverse the regular Expression
- Construct the finite automata
- Construct the right regular grammar
- Reverse right hand side of every production

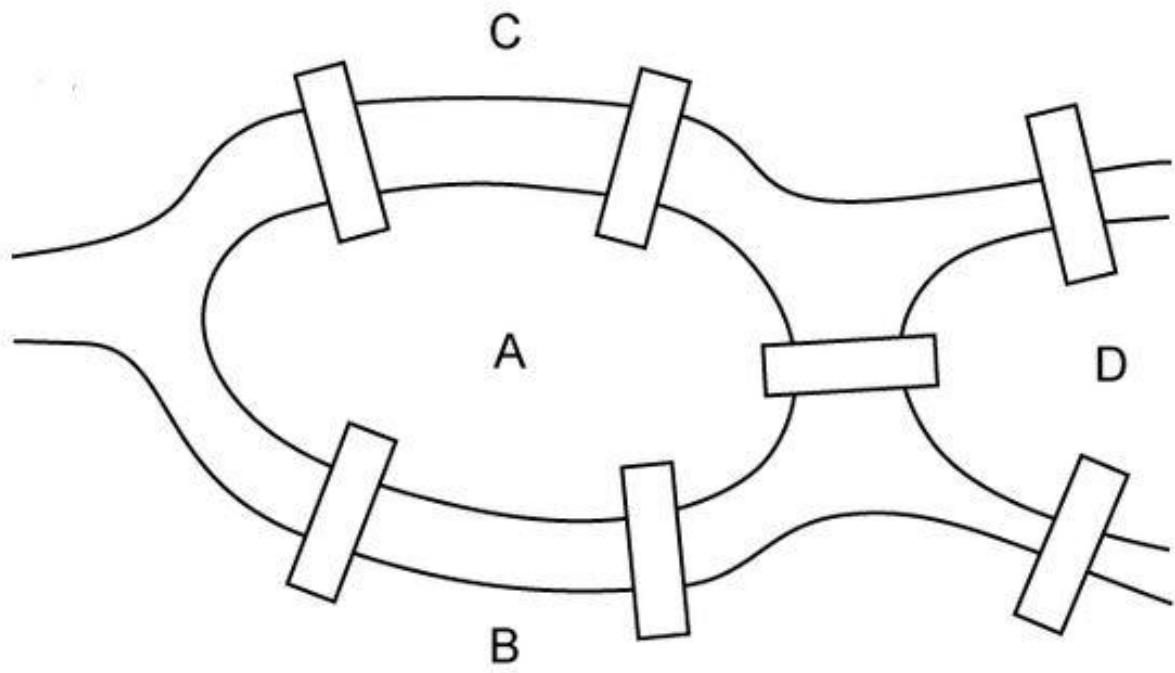
- Every Left regular grammar can be converted into right regular and every right regular grammar can be converted into left regular grammar.
- **Procedure to convert Right linear grammar to Left linear grammar**
 - Obtain the RE from the grammar
 - Reverse the RE
 - Construct the finite automata for the regular expression
 - Now construct the right linear grammar
 - Reverse the left hand side of every production to get a left linear grammar.

What computer science deals with?

- Don't
 - So we do not study how to design a computer
 - We do not study how to run a computer
- Do
 - We deal with problem solving, according to computer science a problem can be divided as follows



Konigsberg Bridge Problem



- **SOLVABLE** - A problem is said to be solvable if either we can solve it or if we can prove that the problem cannot be solved.
- **UNSOLVABLE** - A problem is said to be unsolvable if neither we can solve it, nor we can proof that the problem can not be solved.
- **P**- if there exist a polynomial time algorithm to solve a problem then problem is said to be decidable.
- **NP**- if there exist a non- polynomial time algo to solve a problem.

Decision properties

- Approximately all the properties are decidable in case a finite automaton. Here we will use machine model to proof decision properties.
 - i) Emptiness
 - ii) Non-emptiness
 - iii) Finiteness
 - iv) Infiniteness
 - v) Membership
 - vi) Equality

Emptiness & Non-emptiness

- Step 1: - select the state that cannot be reached from the initial states & delete them (remove unreachable states)
- Step 2: - if the resulting machine contains at least one final states, so then the finite automata accepts the non-empty language.
- Step 3: - if the resulting machine is free from final state, then finite automata accepts empty language.

Finiteness & Infiniteness

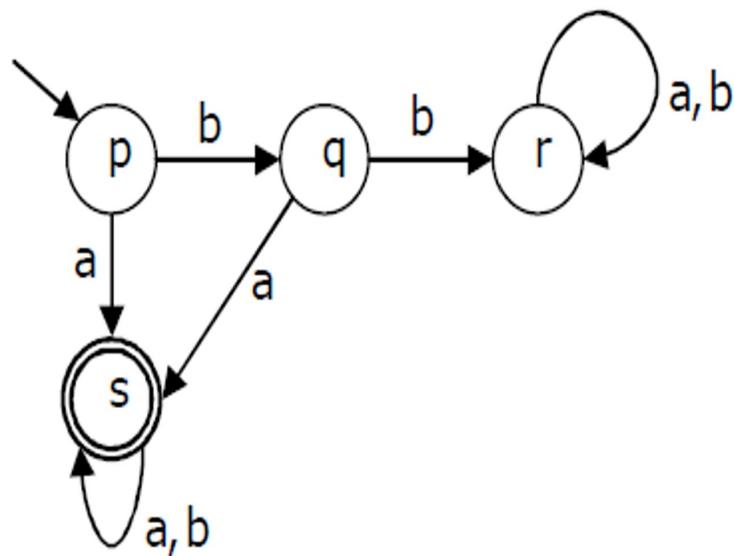
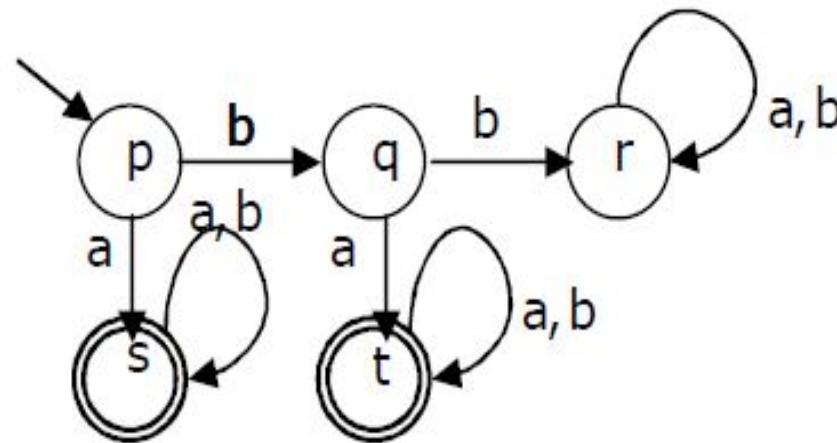
- Step 1: - Select the state that cannot be reached from the initial state & delete them (remove unreachable states)
- Step 2: - Select the state from which we cannot reach the final state & delete them (remove dead states)
- Step 3: - If the resulting machine contains loops or cycles then the finite automata accepts infinite language
- Step 4: - If the resulting machine do not contain loops or cycles then the finite automata accepts finite language.

Membership

- Membership is a property to verify an arbitrary string is accepted by a finite automaton or not i.e. it is a member of the language or not.
- Let M is a finite automata that accepts some strings over an alphabet, and let ' w ' be any string defined over the alphabet, if there exist a transition path in M , which starts at initial state & ends in anyone of the final state, then string ' w ' is a member of M , otherwise ' w ' is not a member of M .

Equality

- Two finite state automata M_1 & M_2 is said to be equal if and only if, they accept the same language.
- Minimise the finite state automata and and the minimal DFA will be unique.



	RL	DCFL	CFL	CSL	RS	RES
Emptiness	Y	Y	Y	X	N	N
Non-Emptiness	Y	Y	Y	X	N	N
Finiteness	Y	Y	Y	X	N	N
Infiniteness	Y	Y	Y	X	N	N
Membership	Y	Y	Y	X	Y	N
Equality	Y	N	N	X	N	N
Ambiguity	Y	N	N	X	N	N
Σ^*	Y	N	N	X	N	N
Halting	Y	Y	Y	X	Y	N

Closure Properties of Regular Languages

- Regular languages are closed under following operations
 - Kleen Closure
 - Positive closure
 - Complement
 - Reverse Operator
 - Prefix Operator
 - Suffix operator
 - Concatenation
 - Union
 - Intersection
 - Set Difference operator
 - Symmetric Difference

Q Consider the following two statements about regular languages:

- S_1 : Every infinite regular language contains an undecidable language as a subset.
- S_2 : Every finite language is regular.

Which one of the following choices is correct? **(GATE 2021) (2 MARKS)**

- (a) Only S_1 is true
- (b) Only S_2 is true
- (c) Both S_1 and S_2 are true
- (d) Neither S_1 nor S_2 is true

Q If L is a regular language over $\Sigma = \{a, b\}$, which one of the following languages is NOT regular? (GATE – 2019) (1 Marks)

- a) $L \cdot L^R \{xy \mid x \in L, y^R \in L\}$
- b) Suffix (L) = $\{y \in \Sigma^* \mid \exists x \in \Sigma^* \text{ such that } xy \in L\}$
- c) Prefix (L) = $\{x \in \Sigma^* \mid \exists y \in \Sigma^* \text{ such that } xy \in L\}$
- d) $\{ww^R \mid w \in L\}$

Q Consider the following two statements (GATE-2016) (1 Marks)

- I. If all states of an NFA are accepting states then the language accepted by the NFA is Σ^* .
- II. There exists a regular language A such that for all languages B, $A \cap B$ is regular.

Which one of the following is CORRECT?

- (A) Only I is true
- (B) Only II is true
- (C) Both I and II are true
- (D) Both I and II are false

Q Which of the following is TRUE? (GATE-2007) (2 Marks)

- a) Every subset of a regular set is regular.
- b) The union of two non-regular sets is not regular.
- c) Every finite subset of a non-regular set is regular.
- d) Infinite union of finite sets is regular

Q Which of the following is true? (GATE – 2007) (1 Marks)

- a)** Infinite union of regular set is regular
- b)** The union of two non-regular net is not regular
- c)** finite union of infinite set is regular
- d)** every R.L is also C.F.L

Q Which of the following statements about regular languages is NOT true?

(GATE-2006) (1-Marks)

- a)** Every language has a regular superset
- b)** Every language has a regular subset
- c)** Every subset of a regular language is regular
- d)** Every subset of a finite language is regular

Q Which of the following statement is false? (GATE – 1998) (1 Marks)

- a) Every finite subset of a non – regular set is regular
- b) Every subset of a regular set is regular
- c) Every finite subset of a regular set is regular
- d) The intersection of two regular sets in regular

Q. Let L_1, L_2 be two regular languages and L_3 a language which is not regular. Which of the following statements is/are always TRUE? (Gate 2024,CS) (1 Marks) (MSQ)

- (a) $L_1 = L_2$ if and only if $L_1 \cap L_2' = \emptyset$
- (b) $L_1 \cup L_3$ is not regular
- (c) $(L_3)'$ is not regular
- (d) $(L_1)' \cup (L_2)'$ is regular

Q. Consider the two lists List I and List II given below:

List I	List II
(i) Context free languages	(a) Closed under union
(ii) Recursive languages	(b) Not closed under complementation
(iii) Regular languages	(c) Closed under intersection

For matching of items in List I with those in List II, which of the following option(s) is/are CORRECT? **(Gate 2025)**

- A) (i) – (a), (ii) – (b), and (iii) – (c)
- B) (i) – (b), (ii) – (a), and (iii) – (c)
- C) (i) – (b), (ii) – (c), and (iii) – (a)
- D) (i) – (a), (ii) – (c), and (iii) – (b)

Q. Consider the following two languages over the alphabet $\{a, b\}$:

$$L_1 = \{ \alpha\beta\alpha \mid \alpha \in \{a, b\}^+ \text{ AND } \beta \in \{a, b\}^+ \}$$

$$L_2 = \{ \alpha\beta\alpha \mid \alpha \in \{a\}^+ \text{ AND } \beta \in \{a, b\}^+ \}$$

Which ONE of the following statements is CORRECT? (Gate 2025)

- A) Both L_1 and L_2 are regular languages.
- B) L_1 is a regular language but L_2 is not a regular language.
- C) L_1 is not a regular language but L_2 is a regular language.
- D) Neither L_1 nor L_2 is a regular language.

	RL	DCFL	CFL	CSL	RS	RES
Union	Y	N	Y	Y	Y	Y
Intersection	Y	N	N	Y	Y	Y
Complement	Y	Y	N	Y	Y	N
Set Difference	Y	N	N	Y	Y	N
Kleene Closure	Y	N	Y	Y	Y	Y
Positive Closure	Y	N	Y	Y	Y	Y
Concatenation	Y	N	Y	Y	Y	Y
Intersection with regular set	Y	Y	Y	Y	Y	Y
Reverse	Y	Y	Y	Y	Y	Y
Subset	N	N	N	N	N	N

	RL	DCFL	CFL	CSL	RS	RES
Homomorphism	Y	N	Y	N	N	Y
\in Free Homomorphism	Y	N	Y	Y	Y	Y
Inverse Homomorphism	Y	Y	Y	Y	Y	Y
Substitution	Y	N	Y	N	N	Y
\in Free Substitution	Y	N	Y	Y	Y	Y
Quotient with regular set	Y	Y	Y	N	Y	Y

Moore and Mealy Machine

- Both moore and mealy machine are special case of DFA
- Both acts like o/p producers rather than language acceptors
- In moore and mealy machine no need to define the final states
- No concepts of dead states and no concepts of final states
- Mealy and Moore Machines are equivalent in power.



George H Mealy



Edward F Moore

Moore Machine

A Moore machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where

- Q is a finite set of states:
- Σ is the input alphabet:
- Δ is the output alphabet.
- δ is the transition function $Q \times \Sigma$ into Q
- λ is the output function mapping Q into Δ and
- q_0 is the initial state.

Examples: The below table shows the transition table of a Moore Machine.

Present state	Next state δ		λ
	$a = 0$	$a = 1$	
$\rightarrow q_0$	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

- In moore machine for every state output is associated. If the length of i/p string is n, then length of o/p string will be n+1. Moore machine response for empty string \in .

Q construct a Moore machine take all the string of a's and b's as i/p and counts the no of a's in the i/p string in terms of 1, $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$?

Q construct a Moore machine take all the string of a's and b's as i/p and counts the no of occurrence of sub-string 'ab' in terms of 1, $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$?

Q construct a Moore machine take all the string of a's and b's as i/p and counts the no of occurrence of sub-string 'aa' in terms of 1, $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$?

Q construct a Moore machine where $\Sigma = \{0, 1\}$, $\Delta = \{a, b, c\}$, machine should give o/p a, if the i/p string ends with 10, b if i/p string ends with 11, c otherwise?

Mealy Machine

- Mealy machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where all the symbols except λ have the same meaning as in the Moore machine. λ is the output function mapping $Q \times \Sigma$ into Δ .
- In case of mealy machine, the output symbol depends on the transition.

Example: The below table shows the transition table of a Mealy Machine.

Present state	Next state			
	$a = 0$		$a = 1$	
	state	output	state	output
$\rightarrow q_1$	q_3	0	q_2	0
q_2	q_4	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

- If the length of i/p string is n, then length of o/p string will be n. Mealy machine do not response for empty string \in .

Q construct a Mealy machine take all the string of a's and b's as i/p and counts the no of a's in the i/p string in terms of 1, $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$?

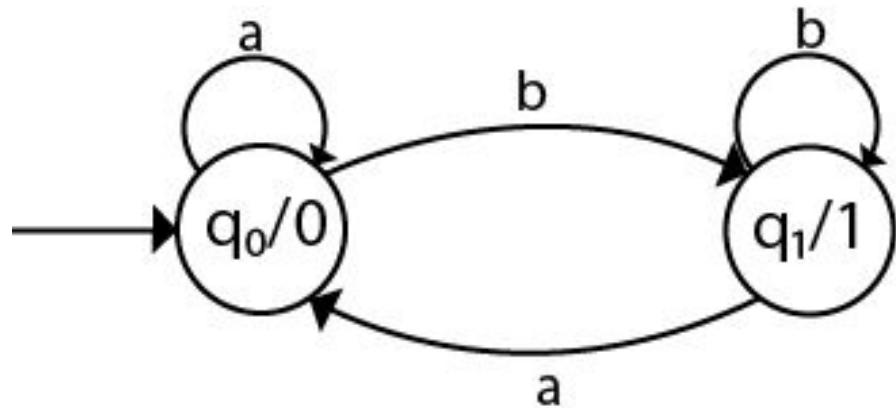
Q construct a Mealy machine take all the string of a's and b's as i/p and counts the no of occurrence of sub-string 'ab' in terms of 1, $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$?

Q construct a Mealy machine take all the string of a's and b's as i/p and counts the no of occurrence of sub-string 'aa' in terms of 1, $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$?

Q construct a Mealy machine where $\Sigma = \{0, 1\}$, $\Delta = \{a, b, c\}$, machine should give o/p a, if the i/p string ends with 10, b if i/p string ends with 11, c otherwise?

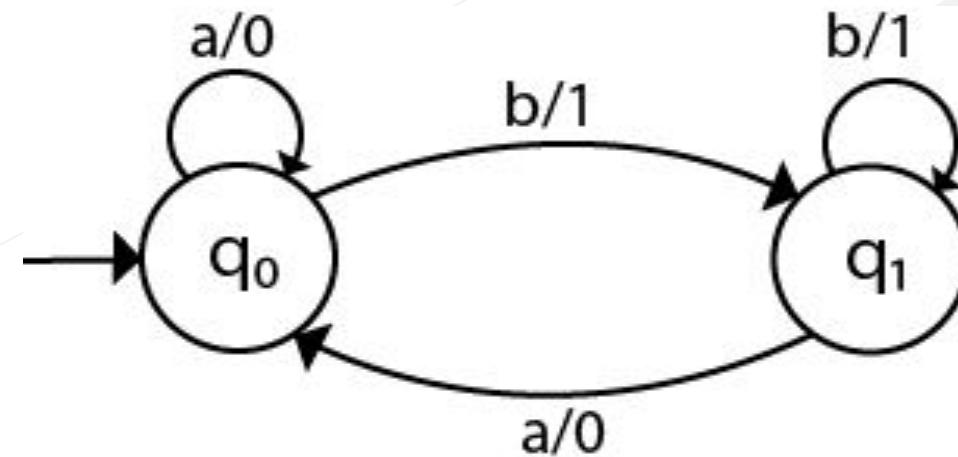
CONVERSION OF MOORE TO MEALY MACHINE

- Let us take an example to understand the conversion:
- Convert the following Moore machine into its equivalent Mealy machine.



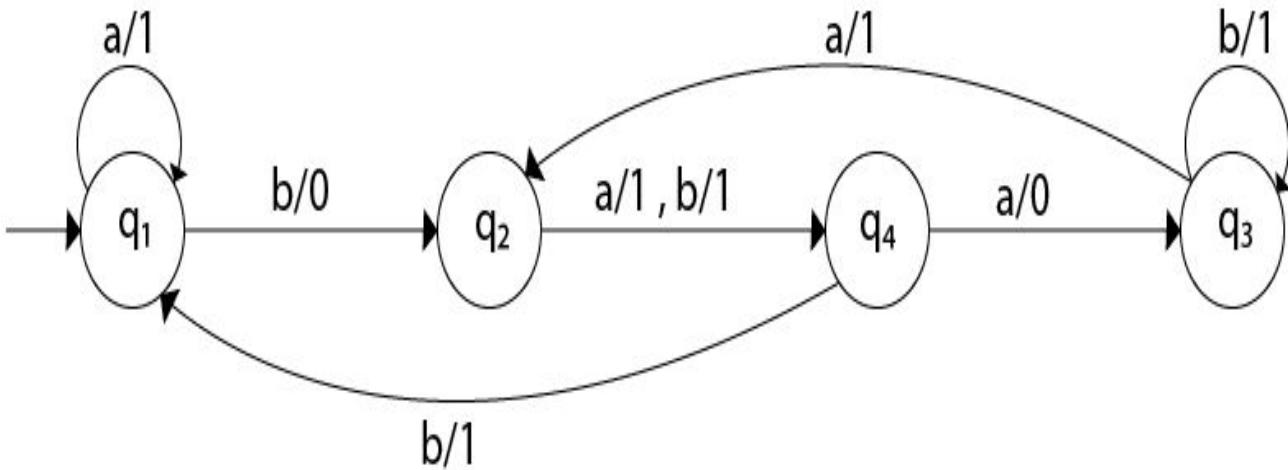
Q	a	b	Output(λ)
q_0	q_0	q_1	0
q_1	q_0	q_1	1

- To convert a mealy machine to moore machine all you need to do is just push out the outputs of states onto to the incoming transitions.
- While conversion from moore to mealy machine, the number of states will be same and there will be no extra states.
- The equivalent Moore Machine will be:



PROCEDURE FOR TRANSFORMING A MEALY MACHINE INTO A MOORE MACHINE

Consider the Mealy Machine:



Present State	Next State			
	a		b	
	State	O/P	State	O/P
q ₁	q ₁	1	q ₂	0
q ₂	q ₄	1	q ₄	1
q ₃	q ₂	1	q ₃	1
q ₄	q ₃	0	q ₁	1

Q Given the following state table of an FSM with two states A and B, one input and one output:
If the initial state is A=0, B=0, what is the minimum length of an input string which will take the
machine to the state A=0, B=1 with Output = 1? **(GATE-2009) (2 Marks)**

(A) 3

(B) 4

(C) 5

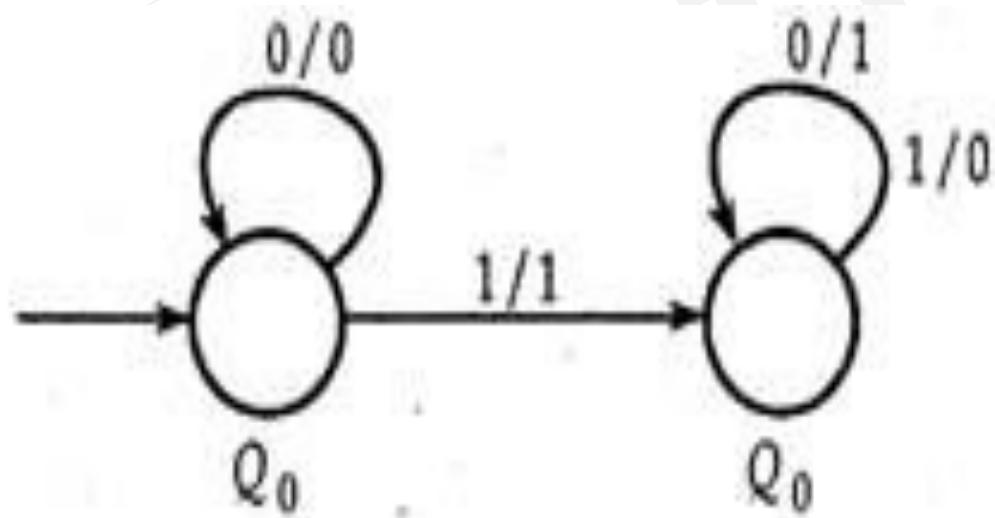
(D) 6

Present State A	Present State B	I/p	Next State A	Next State B	O/p
0	0	0	0	0	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	1	0	0
0	0	1	0	1	0
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	0	0	1

Q The following diagram represents a finite state machine which takes as input a binary number from the least significant bit. **(GATE-2005) (1 Marks)**

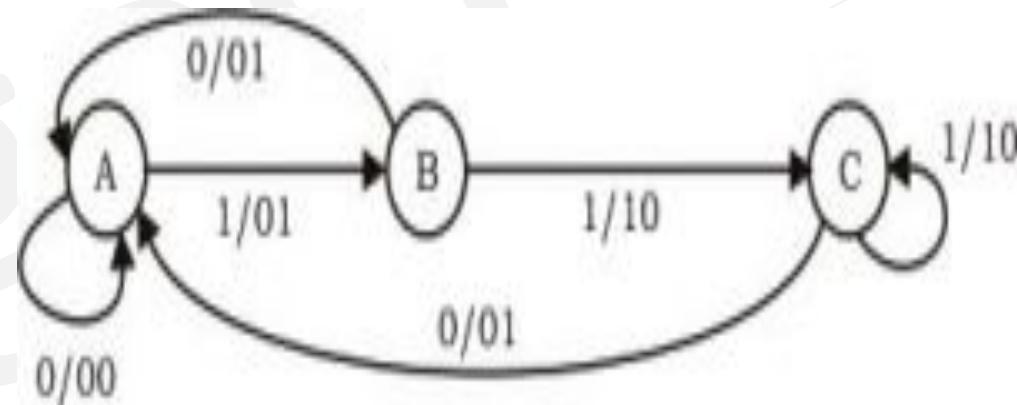
Which one of the following is TRUE?

- (A) It computes 1's complement of the input number
- (B) It computes 2's complement of the input number
- (C) It increments the input number
- (D) It decrements the input number



Q The Finite state machine described by the following state diagram with A as starting state, where an arc label is x / y and x stand for 1-bit input and y stands for 2- bit output (**GATE-2002**) (2 Marks)

- (A) Outputs the sum of the present and the previous bits of the input.
- (B) Outputs 01 whenever the input sequence contains 11.
- (C) Outputs 00 whenever the input sequence contains 10.
- (D) None of these



Q Suppose we want to design a synchronous circuit that processes a string of 0's and 1's. Given a string, it produces another string by replacing the first 1 in any subsequence of consecutive 1's by a 0. Consider the following example.

Input sequence : 00100011000011100

Output sequence : 00000001000001100

A Mealy Machine is a state machine where both the next state and the output are functions of the present state and the current input. The above mentioned circuit can be designed as a two-state Mealy machine. The states in the Mealy machine can be represented using Boolean values 0 and 1. We denote the current state, the next state, the next incoming bit, and the output bit of the Mealy machine by the variables s, t, b and y respectively. Assume the initial state of the Mealy machine is 0. What are the Boolean expressions corresponding to t and y in terms of s and b ? **(GATE 2021) (2 MARKS)**

(a) $t = s+b \quad y = sb$

(b) $t = b \quad y = sb$

(c) $t = b \quad y = sb'$

(d) $t = s+b \quad y = sb'$

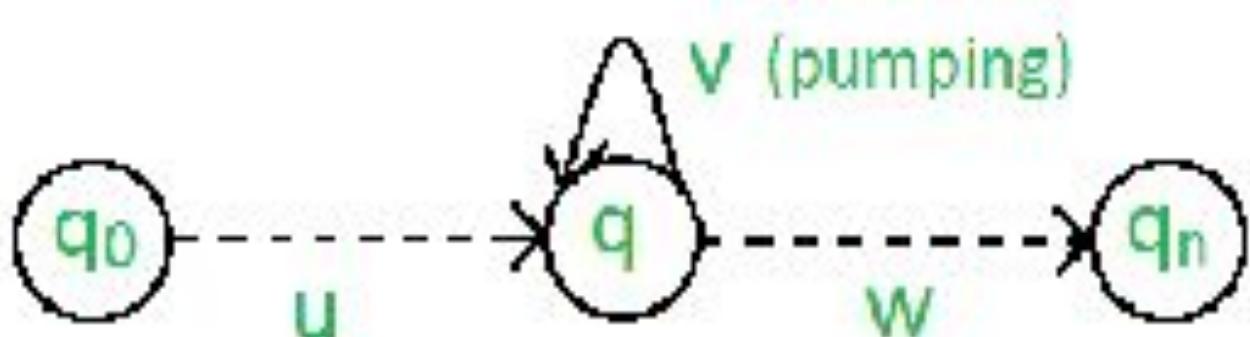
Pumping Lemma For Regular Languages

- Pumping Lemma is used as a proof for irregularity of a language. Thus, if a language is regular, it always satisfies pumping lemma. If there exists at least one string made from pumping which is not in L , then L is surely not regular.
- The opposite of this may not always be true. That is, if Pumping Lemma holds, it does not mean that the language is regular.
- Pumping Lemma is used to prove that some of the language is non-regular.
- For the pumping lemma, i/p is NRL & o/p is also NRL.

Pumping Lemma

Pumping Lemma For Regular Languages

- For any regular language L , there exists an integer n , such that for all $z \in L$ with $|z| \geq n$, there exists $u, v, w \in \Sigma^*$, such that $z = uvw$, and
 - $|uv| \leq n$
 - $|v| \geq 1$
 - for all $i \geq 0$: $uv^iw \in L$
- In simple terms, this means that if a string v is ‘pumped’, i.e., if v is inserted any number of times, the resultant string still remains in L .



Process of pumping Lemma

- Suppose that we need to prove that language L is a non-regular.
- Assume that L is a regular language, then L must satisfy pumping lemma property.
- Choose $z \in L$ such that $|z| \leq n$, split z into 3 parts.
 - $|uv| \leq n$
 - $|v| \geq 1$
- If there exist at least one variable for i such that $uv^iw \notin L$, then L does not satisfy pumping lemma property so it is a contradiction, therefore language L is non-regular.

Q Proof that language $L = \{a^m b^n \mid m=n\}$ is non-regular?

$L = \{ab, aabb, aaabbb, aaaabbbb, \dots\}$

$z \in L$

$z = a^k b^k$

For $i=1$ $uv^i w$ $a^k b^k$

For $i=2$ $uv^i w$ $a^{k+1} b^k \notin L$

Therefore L is not regular

Q Proof that language $L = \{a^m b^n \mid m < n\}$ is non-regular?

Q Proof that language $L = \{a^n b^n \mid n \text{ is a prime number}\}$?

Q Proof that language $L = \{a^{n^2} \mid n \geq 0\}$?

Q For $\Sigma = \{a, b\}$, let us consider the regular language

$$L = \{x \mid x = a^{2+3k} \text{ or } x = b^{10+12k}, k \geq 0\}$$

Which one of the following can be a pumping length (the constant guaranteed by the pumping lemma) for L? **(GATE- 2019)**

- (A) 3
- (B) 5
- (C) 9
- (D) 24

Q. Consider a finite state machine (FSM) with one input X and one output f , represented by the given state transition table. The minimum number of states required to realize this FSM is _____. (Answer in integer) (Gate 2025)

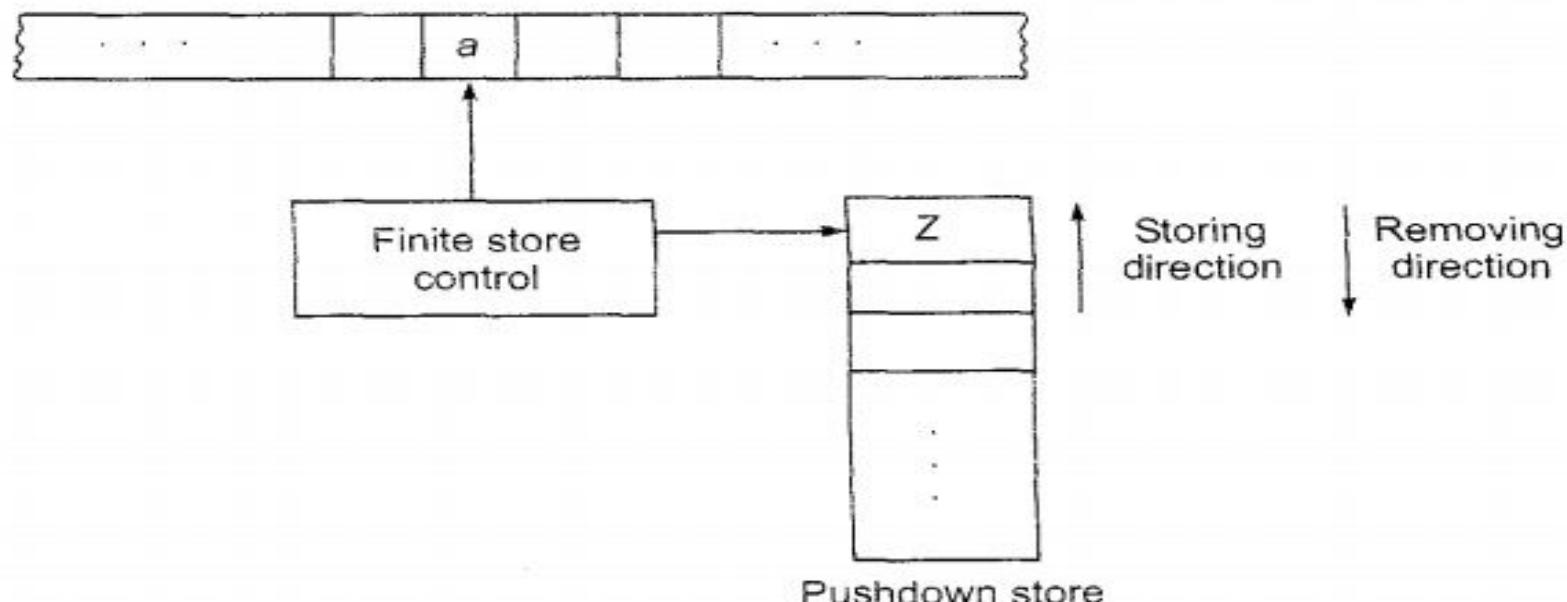
Present state	Next state		Output f	
	$X = 0$	$X = 1$	$X = 0$	$X = 1$
A	F	B	0	0
B	D	C	0	0
C	F	E	0	0
D	G	A	1	0
E	D	C	0	0
F	F	B	1	1
G	G	H	0	1
H	G	A	1	0

CONTEXT-FREE LANGUAGES AND PUSH DOWN AUTOMATA

- We Already understand the limitation of finite automata, that it cannot do the infinite comparison between the symbols.
- Let us consider $L = \{a^n b^n \mid n \geq 1\}$. This is not regular, as it has to remember the number of a's in a string and so it will require an infinite number of states, which is logically not possible.
- This difficulty can be avoided by adding an auxiliary memory in the form of a 'stack'. The reason we choose stack because it is the simplest memory possible.
- This type of arrangement where ***a finite automaton has a stack leads to the generation of a pushdown automaton.***

BLOCK DIAGRAM OF PDA

- Finite control unit is also called as memory unit it is static and limited. So to process the i/p string if the static memory is not sufficient then we can use the stack.
- i/p tape is divided into cells where is cell is capable of holding one symbol at a time. At stack of infinite size, which support three operations push, pop and skip.
- The accepting power of a pda is more than that of finite automata and less than that of linear bounder automata
- The power of non-deterministic pda is more than the power of deterministic pda.



Formal Definition of DPDA

A DPDA is a 7-tuple, namely $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where

- (i) Q – is a finite nonempty set of states,
- (ii) Σ – is a finite nonempty set of input symbols,
- (iii) Γ – is a finite nonempty set of pushdown symbols,
- (iv) q_0 – is a special state called the initial state,
- (v) Z_0 – is a special pushdown symbol called the initial symbol on the pushdown store.
- (vi) F – is a set of final states, a subset of Q and
- (vii) δ – is a transition function from $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$ to the set of finite subsets of $Q \times \Gamma^*$.

Representation of States

(1) **PUSH** – one symbol can be inserted into the stack at one time.

$$\delta(q_i, a, z_0) = (q_j, az_0)$$

Representation of States

(2) POP – one symbol can be deleted from the stack at one time.

$$\delta(q_i, a, z_0) = (q_j, \epsilon)$$

Representation of States

(3) SKIP – IT means no stack operation, status of the stack will remain same, before a after the operation

$$\delta(q_i, a, z_0) = (q_j, z_0)$$

note- if pda perform a push or a pop operation at least one's during processing of string than we say that pda is using the stack.

Instantaneous Description (ID)

- An **instantaneous description (ID)** is (q, x, α) , where $q \in Q$, $x \in \Sigma^*$ and $\alpha \in \Gamma^*$.
- An initial ID is (q_0, x, Z_0) , this means that initially the pda is in the initial state q_0 , the input string to be processed is x and the PDS has only one symbol, namely Z_0 .
- ACCEPTANCE BY PDA There is no change in the language acceptance capability of the pda either we accept by final state or empty stack.

Q Design a Deterministic Push Down Automata for $L = \{a, ab\}$?

Q Design a Deterministic Push Down Automata for $\{a^n b^n \mid n \geq 1\}$?

Q Design a Deterministic Push Down Automata for $\{a^n b^{2n} \mid n \geq 1\}$?

Q Design a PDA for $\{w c w^r \mid w \in (a, b)^*\}$?

Q Construct PDA that accepts $L = \{|w|_{a=b} \mid w \in (a, b)^*\}$?

Q consider the following mapping and find the correct language?

$$\delta(q_0, 1, z_0) = (q_0, xz_0)$$

$$\delta(q_0, 1, x) = (q_0, xx)$$

$$\delta(q_0, 0, x) = (q_1, \epsilon)$$

$$\delta(q_1, 0, x) = (q_1, \epsilon)$$

$$\delta(q_1, \epsilon, x) = (q_1, \epsilon)$$

$$\delta(q_1, \epsilon, z_0) = (q_F, z_0)$$

a) $L = \{a^m b^n \mid m = n\}$

b) $L = \{a^m b^n \mid m \neq n\}$

c) $L = \{a^m b^n \mid m \geq n\}$

d) $L = \{a^m b^n \mid m \leq n\}$

Q consider the following mapping and find the correct language?

$$\delta(q_0, a, z_0) = (q_1, z_0)$$

$$\delta(q_0, b, z_0) = (q_2, z_0)$$

$$\delta(q_1, a, z_0) = (q_1, z_0)$$

$$\delta(q_1, b, z_0) = (q_2, z_0)$$

$$\delta(q_2, a, z_0) = (q_2, z_0)$$

$$\delta(q_2, b, z_0) = (q_0, z_0)$$

$$\delta(q_1, \epsilon, z_0) = (q_f, z_0)$$

- a)** ending with a
- b)** ending with a, contain even number of a
- c)** ending with a, contain odd number of a
- d)** ending with a, contain even number of b

Non- Deterministic PDA

- Non- Deterministic PDA can also be defined using 7 tuples.
- $\delta: Q \times \{\sum U \in\} \times \Gamma \rightarrow 2^{(Q \times \Gamma^*)}$
- i.e. on a given input symbol and stack symbol a NPDA can move to more than one state.
- Rest all other tuples are same as DPDA.

Q Construct pda that accepts a language $L = \{w w^r \mid w \in (a, b)^*\}$?

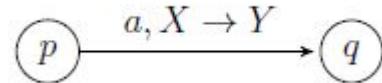
For a string w , we define w^R to be the reverse of w . For example, if $w = 01101$ then $w^R = 10110$.

(GATE-2021)

Which of the following languages is/are context-free?

- A. $\{wxw^Rx^R \mid w, x \in \{0, 1\}^*\}$
- B. $\{ww^Rxx^R \mid w, x \in \{0, 1\}^*\}$
- C. $\{wxw^R \mid w, x \in \{0, 1\}^*\}$
- D. $\{wx^Rw^R \mid w, x \in \{0, 1\}^*\}$

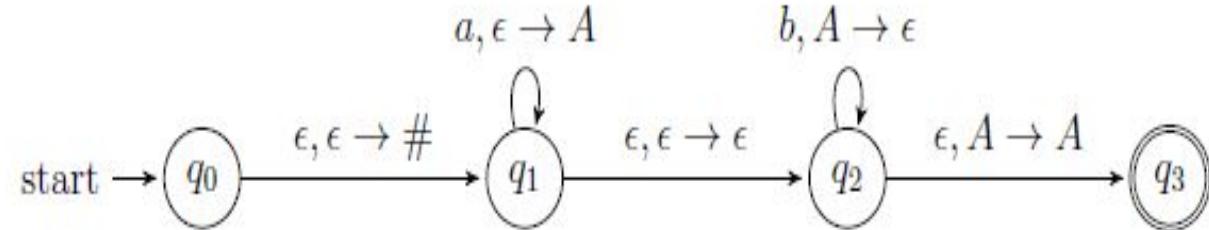
Q In a pushdown automaton $P=(Q,\Sigma,\Gamma,\delta,q_0,F)$, a transition of the form,



where $p,q \in Q$, $a \in \sigma \cup \{\epsilon\}$, $X,Y \in \Gamma \cup \{\epsilon\}$, represents

$$(q,Y) \in \delta(p,a,X)$$

Consider the following pushdown automaton over the input alphabet $\Sigma=\{a,b\}$ and stack alphabet $\Gamma=\{\#,A\}$.



The number of strings of length 100 accepted by the above pushdown automaton is _____. (GATE 2021) (2 MARKS)

Q Which one of the following languages over $\Sigma = \{a, b\}$ is NOT context-free?

(GATE-2019) (2 Marks)

- a) $\{a^n b^i \mid i \in \{n, 3n, 5n\}, n \geq 0\}$
- b) $\{w a^n w^R b^n \mid w \in \{a, b\}^*, n \geq 0\}$
- c) $\{w w^R \mid w \in \{a, b\}^*\}$
- d) $\{w a^n b^n w^R \mid w \in \{a, b\}^*, n \geq 0\}$

Q Consider the following languages:

- I. $\{a^m b^n c^p d^q \mid m + p = n + q, \text{ where } m, n, p, q \geq 0\}$
- II. $\{a^m b^n c^p d^q \mid m = n \text{ and } p = q, \text{ where } m, n, p, q \geq 0\}$
- III. $\{a^m b^n c^p d^q \mid m = n = p \text{ and } p \neq q, \text{ where } m, n, p, q \geq 0\}$
- IV. $\{a^m b^n c^p d^q \mid mn = p + q, \text{ where } m, n, p, q \geq 0\}$

Which of the above languages are context-free?

(GATE-2018) (2 Marks)

- a) I and IV only
- b) I and II only
- c) II and III only
- d) II and IV only

Linear Grammar in Computer Science

- A **linear grammar** is a type of context-free grammar where each production's right-hand side contains at most one nonterminal symbol.
- Example of a simple linear grammar:
- Let G be a grammar with nonterminal set $N=\{S\}$, terminal set $\Sigma=\{a,b\}$, and the start symbol S.
- Production rules:
 - $S \rightarrow aSb$
 - $S \rightarrow \epsilon$
 - This grammar generates the language $L=\{a^n b^n \mid n \geq 0\}$, which is a **linear language**.
- **Types of Linear Grammars:**
 - **Left-Linear Grammars:** Nonterminal symbols appear only at the left ends of the right-hand sides.
 - **Right-Linear Grammars:** Nonterminal symbols appear only at the right ends of the right-hand sides.
 - Both left-linear and right-linear grammars generate **regular languages**.

- **General Linear Grammars:**
 - Nonterminal symbols can appear at either end of the production, but not necessarily always at the same end.
 - Any linear grammar can be transformed into this form without changing the language generated.
- For example, the grammar:
 - $S \rightarrow aAS$
 - $A \rightarrow SbA$
 - $S \rightarrow \epsilon$
 - generates the same language as the example above.

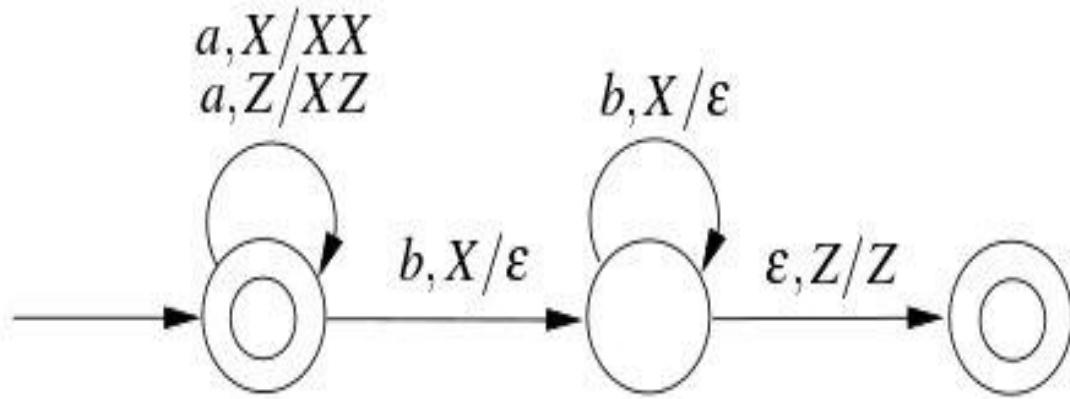
- **Relation to Other Languages:**

- All **regular languages** are linear, but not all linear languages are regular. For example, $\{anbn \mid n \geq 0\}$ is linear but non-regular.
- All linear languages are **context-free**, but not all context-free languages are linear. For example, the well-balanced bracket language is context-free but not linear.

- **Determinism and Complexity:**

- Regular linear languages are **deterministic**, but some linear languages are **nondeterministic**.
- For example, the language of even-length **palindromes** (generated by the grammar $S \rightarrow 0S0 \mid 1S1 \mid \epsilon$) is nondeterministic.
- **Nondeterministic** linear languages cannot always be recognized in **linear time**.
- Moreover, it is **undecidable** whether a given context-free language is linear.

Q Consider the transition diagram of a PDA given below with input alphabet $\Sigma = \{a, b\}$ and stack alphabet $\Gamma = \{X, Z\}$. Z is the initial stack symbol. Let L denote the language accepted by the PDA. **(GATE-2016) (2 Marks)**



Which one of the following is **TRUE**?

- (A) $L = \{a^n b^n \mid n \geq 0\}$ and is not accepted by any finite automata
- (B) $L = \{a^n \mid n \geq 0\} \cup \{a^n b^n \mid n \geq 0\}$ and is not accepted by any deterministic PDA
- (C) L is not accepted by any Turing machine that halts on every input
- (D) $L = \{a^n \mid n \geq 0\} \cup \{a^n b^n \mid n \geq 0\}$ and is deterministic context-free

Q Consider the following languages (GATE-2016) (1 Marks)

$$L_1 = \{a^n b^m c^n : m, n \geq 1\}$$

$$L_2 = \{a^n b^n c^{2n} : n \geq 1\}$$

Which one of the following is TRUE?

- (A) Both L_1 and L_2 are context-free
- (B) L_1 is context-free while L_2 is not context-free.
- (C) L_2 is context-free while L_1 is not context-free
- (D) Neither L_1 nor L_2 is context-free.

Q Which of the following languages are context-free? (GATE-2015) (2 Marks)

$$L_1 = \{a^m b^n a^n b^m \mid m, n \geq 1\}$$

$$L_2 = \{a^m b^n a^m b^n \mid m, n \geq 1\}$$

$$L_3 = \{a^m b^n \mid m = 2n + 1\}$$

(A) L_1 and L_2 only

(B) L_1 and L_3 only

(C) L_2 and L_3 only

(D) L_3 only

Consider the following languages over the alphabet $\Sigma = \{0,1,c\}$:

$$L_1 = \{0^n 1^n \mid n \geq 0\}$$

$$L_2 = \{wcw^r \mid w \in \{0,1\}^*\}$$

$$L_3 = \{ww^r \mid w \in \{0,1\}^*\}$$

Here, w^r is the reverse of the string w . Which of these languages are deterministic Context-free languages? (GATE-2014) (2 Marks)

- (A) None of the languages (B) Only L_1
- (C) Only L_1 and L_2 (D) All the three languages

Q Consider the language L_1 , L_2 , L_3 as given below. (GATE-2011) (2 Marks)

$$L_1 = \{a^p b^q \mid p, q \in N\}$$

$$L_2 = \{a^p b^q \mid p, q \in N \text{ and } p=q\}$$

$$L_3 = \{a^p b^q c^r \mid p, q, r \in N \text{ and } p = q = r\}$$

Which of the following statements is **NOT TRUE**?

- (A) Push Down Automata (PDA) can be used to recognize L_1 and L_2
- (B) L_1 is a regular language
- (C) All the three languages are context free
- (D) Turing machine can be used to recognize all the three languages

Q Consider the languages (GATE-2010) (2 Marks)

$$L_1 = \{0^i 1^j \mid i \neq j\}$$

$$L_2 = \{0^i 1^j \mid i = j\}$$

$$L_3 = \{0^i 1^j \mid i = 2j+1\}$$

$$L_4 = \{0^i 1^j \mid i \neq 2j\}$$

- (A) Only L_2 is context free
- (B) Only L_2 and L_3 are context free
- (C) Only L_1 and L_2 are context free
- (D) All are context free

Q Which one of the following is FALSE? (GATE-2009) (2 Marks)

- (A) There is unique minimal DFA for every regular language
- (B) Every NFA can be converted to an equivalent PDA.
- (C) Complement of every context-free language is recursive.
- (D) Every nondeterministic PDA can be converted to an equivalent deterministic PDA.

Q Consider the following languages. (GATE-2008) (2 Marks)

$$L_1 = \{a^i b^j c^k \mid i = j, k \geq 1\}$$

$$L_2 = \{a^i b^j \mid j = 2i, i \geq 0\}$$

Which of the following is true?

- (A) L_1 is not a CFL but L_2 is
- (B) $L_1 \cap L_2 = \emptyset$ and L_1 is non-regular
- (C) $L_1 \cap L_2$ is not a CFL but L_2 is
- (D) There is a 4-state PDA that accepts L_1 ,
but there is no DPDA that accepts L_2

Q The language $L = \{0^i 2 1^i \mid i \geq 0\}$ over the alphabet $\{0, 1, 2\}$ is: **(GATE-2007) (2 Marks)**

- a) not recursive.
- b) is recursive and is a deterministic CFL.
- c) is a regular language.
- d) is not a deterministic CFL but a CFL

Q (GATE-2006) (1 Marks)

Let $L_1 = \{0^{n+m}1^n 0^m \mid n, m \geq 0\}$, $L_2 = \{0^{n+m}1^{n+m} 0^m \mid n, m \geq 0\}$, and

$L_3 = \{0^{n+m} 1^{n+m}0^{n+m} \mid n, m \geq 0\}$. Which of these languages are NOT context free?

- (A) L_1 only
- (C) L_1 and L_2

- (B) L_3 Only
- (D) L_2 and L_3

Q Consider the languages: (GATE-2005) (2 Marks)

$$L_1 = \{w w^R \mid w \in \{0, 1\}^*\}$$

$$L_2 = \{w \# w^R \mid w \in \{0, 1\}^*\}, \text{ where } \# \text{ is a special symbol}$$

$$L_3 = \{w w \mid w \in \{0, 1\}^*\}$$

Which one of the following is TRUE?

- (A) L_1 is a deterministic CFL
- (B) L_2 is a deterministic CFL
- (C) L_3 is a CFL, but not a deterministic CFL
- (D) L_3 is a deterministic CFL

Q Let N_f and N_p denote the classes of languages accepted by non-deterministic finite automata and non-deterministic push-down automata, respectively. Let D_f and D_p denote the classes of languages accepted by deterministic finite automata and deterministic push-down automata, respectively. Which one of the following is TRUE? (GATE-2005) (1 Marks)

- (A) $D_f \subset N_f$ and $D_p \subset N_p$ (B) $D_f \subset N_f$ and $D_p = N_p$
- (C) $D_f = N_f$ and $D_p = N_p$ (D) $D_f = N_f$ and $D_p \subset N_p$

Q. The language $\{0^n 1 2^n \mid 1 \leq n \leq 10^6\}$ is : (GATE 2005, 1 Marks)

- a) Regular
- b) Context free but not regular
- c) Context free but its complement is not context free
- d) Not Context free

Q The language $\{a^m b^n C^{m+n} \mid m, n \geq 1\}$ is **(GATE-2004) (1 Marks)**

- (A) regular
- (B) context-free but not regular
- (C) context sensitive but not context free
- (D) type-0 but not context sensitive

Q The language accepted by a Pushdown Automation in which the stack is limited to 10 items is best described as **(GATE-2003) (1 Marks)**

(A) Context Free

(B) Regular

(C) Deterministic Context Free

(D) Recursive

Q. Which of the following language over $\{a,b,c\}$ is accepted by a deterministic pushdown automaton? (GATE - 1997, 1 Marks)

- a) $\{w \in w^R \mid w \in (a,b)^*\}$
- b) $\{ww^R \mid w \in (a,b,c)^*\}$
- c) $\{a^n b^n c^n \mid n \geq 0\}$
- d) {w | w is a palindrome over $\{a,b,c\}$ }

Q Identity the language generated by following grammar where S is the start variable. **(GATE-2017) (2 Marks)**

$S \rightarrow XY$

$X \rightarrow aX \mid a$

$Y \rightarrow aYb \mid \epsilon$

- a) $\{a^m b^n \mid m \geq n, n > 0\}$
- b) $\{a^m b^n \mid m \geq n, n \geq 0\}$
- c) $\{a^m b^n \mid m > n, n \geq 0\}$
- d) $\{a^m b^n \mid m > n, n > 0\}$

Q Consider the context-free grammars over the alphabet {a, b, c} given below. S and T are non-terminals. (GATE-2017) (2 Marks)

$G_1: S \rightarrow aSb \mid T, T \rightarrow cT \mid \epsilon$

$G_2: S \rightarrow bSa \mid T, T \rightarrow cT \mid \epsilon$

The language $L(G_1) \cap L(G_2)$ is

- (A) Finite
- (B) Not finite but regular
- (C) Context-Free but not regular
- (D) Recursive but not context-free

Q Consider the following context-free grammar over the alphabet $\Sigma = \{a, b, c\}$ with S as the start symbol: **(GATE-2017) (2 Marks)**

$S \rightarrow abScT \mid abcT$

$T \rightarrow bT \mid b$

Which of the following represents the language generated by the above grammar?

a) $\{(ab)^n(cb)^n \mid n \geq 1\}$

b) $\{((ab)^n c b^{m_1} c b^{m_2} \dots c b^{m_n}) \mid n, m_1, m_2, \dots, m_n \geq 1\}$

c) $\{(ab)^n (cb^m)^n \mid m, n \geq 1\}$

d) $\{(ab)^n (cb^n)^m \mid m, n \geq 1\}$

Q Which of the following languages is generated by the given grammar?

(GATE-2016) (2 Marks)

$$S \rightarrow aS \mid bS \mid \epsilon$$

(A) $\{a^n b^m \mid n, m \geq 0\}$

(B) $\{w \in \{a, b\}^* \mid w \text{ has equal number of } a's \text{ and } b's\}$

(C) $\{a^n \mid n \geq 0\} \cup \{b^n \mid n \geq 0\} \cup \{a^n b^n \mid n \geq 0\}$

(D) $\{a, b\}^*$

Q Consider the following context-free grammars:

$$G_1: S \rightarrow aS|B, B \rightarrow b|bB$$

$$G_2: S \rightarrow aA|bB, A \rightarrow aA|B|\epsilon, B \rightarrow bB|\epsilon$$

Which one of the following pairs of languages is generated by G_1 and G_2 , respectively **(GATE-2016) (2 Marks)**

- (A) $\{a^m b^n | m > 0 \text{ or } n > 0\}$ and $\{a^m b^n | m > 0 \text{ and } n > 0\}$
- (B) $\{a^m b^n | m > 0 \text{ and } n > 0\}$ and $\{a^m b^n | m > 0 \text{ or } n \geq 0\}$
- (C) $\{a^m b^n | m \geq 0 \text{ or } n > 0\}$ and $\{a^m b^n | m > 0 \text{ and } n > 0\}$
- (D) $\{a^m b^n | m \geq 0 \text{ and } n > 0\}$ and $\{a^m b^n | m > 0 \text{ or } n > 0\}$

Q S -> aSa | bSb | a | b (GATE-2009) (2 Marks)

The language generated by the above grammar over the alphabet {a, b} is the set of

- (A) All palindromes
- (B) All odd length palindromes.
- (C) Strings that begin and end with the same symbol
- (D) All even length palindromes

Q Consider a CFG with the following productions. (GATE-2008) (2 Marks)

$$S \rightarrow AA \mid B$$

$$A \rightarrow 0A \mid A0 \mid 1$$

$$B \rightarrow 0B00 \mid 1$$

S is the start symbol, A and B are non-terminals and 0 and 1 are the terminals. The language generated by this grammar is

(A) $\{0^n 10^{2n} \mid n \geq 1\}$

(B) $\{0^i 10^j 10^k \mid i, j, k \geq 0\} \cup \{0^n 10^{2n} \mid n \geq 0\}$

(C) $\{0^i 10^j \mid i, j \geq 0\} \cup \{0^n 10^{2n} \mid n \geq 0\}$

(D) The set of all strings over {0, 1} containing at least two 0's

Q Language L_1 is defined by the grammar: $S_1 \rightarrow aS_1b \mid \epsilon$

Language L_2 is defined by the grammar: $S_2 \rightarrow abS_2 \mid \epsilon$

Consider the following statements:

P: L_1 is regular

Q: L_2 is regular

Which one of the following is TRUE? (GATE-2007) (2 Marks)

- a)** Both P and Q are true
- b)** P is true and Q is false
- c)** P is false and Q is true
- d)** Both P and Q are false

Q The two-grammar given below generate a language over alphabet {x, y, z} (**GATE – 2007**) (**2 Marks**)

$$G_1: S \rightarrow x \mid z \mid xS \mid zS \mid yB$$

$$B \rightarrow y \mid z \mid yB \mid zB$$

$$G_2: S \rightarrow y \mid z \mid yS \mid zS \mid xB$$

$$B \rightarrow y \mid yS$$

Which one of the following choices describes the properties satisfied by the strings in these languages?

a) G_1 : No y appears before any x, G_2 : Every x is followed by at least one y

b) G_1 : No y appears before any x, G_2 : No x appears before any y

c) G_1 : No y appears after any x, G_2 : Every x is followed by at least one y

d) G_1 : No y appears after any x, G_2 : Every y is followed by at least one x

Q Consider the grammar given below (GATE – 2007) (2 Marks)

S → xB | yA

A → x | x S | y AA

B → y | y S | x BB

Consider the following strings.

- i) xxyyx ii) xxyyxy iii) xyxy**
- iv) yxxx v) yxx vi) xyx**

Which of the above strings are generated by the grammar?

- a) i), ii) and iii)**
- b) ii), v) and vi)**
- c) ii), iii) and iv)**
- d) i), iii) and iv)**

Q Which one of the following grammars generates the language $L = \{a^i b^j \mid i \neq j\}$ (GATE-2006) (1 Marks)

(A)

(B) $S \rightarrow aS \mid Sb \mid a \mid b$

$S \rightarrow AC \mid CB$

$C \rightarrow aC \mid b \mid a \mid b$

$A \rightarrow aA \mid \epsilon$

$B \rightarrow Bb \mid \epsilon$

(C)

(D)

$S \rightarrow AC \mid CB$

$S \rightarrow AC \mid CB$

$C \rightarrow aC \mid b \mid \epsilon$

$C \rightarrow aC \mid b \mid \epsilon$

$A \rightarrow aA \mid \epsilon$

$A \rightarrow aA \mid a$

$B \rightarrow Bb \mid \epsilon$

$B \rightarrow Bb \mid b$

Q In the context-free grammar below, S is the start symbol, a and b are terminals, and ϵ denotes the empty string **(GATE-2006) (1 Marks)**

$$S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$$

Which of the following strings is NOT generated by the grammar?

- (A) aaaa (B) baba
- (C) abba (D) babaaabab

Q. Let L denotes the language generated by the grammar $S \rightarrow 0S0/00$. (GATE 2000, 1 Marks)

Which of the following is True?

- a) $L = 0^+$
- b) L is regular but not 0^+
- c) L is context free but not regular
- d) L is not Context free

Q. Which of the following definitions below generates the same language as L, where $L = \{x^n y^n \text{ such that } n \geq 1\}$ (GATE -1995, 2 Marks)

1. $E \rightarrow xEy \mid xy$
 2. $xy \mid (x^+ syy^+)$
 3. x^+y^+
-
- a) 1 only
 - b) 1 and 2
 - c) 2 and 3
 - d) 2 only

Q. Consider a context-free grammar G with the following 3 rules.

$$S \rightarrow aS$$

$$S \rightarrow aSbS$$

$$S \rightarrow c$$

Let $w \in L(G)$. Let $n_a(w)$, $n_b(w)$, $n_c(w)$ denote the number of times a , b , c occur in w , respectively. Which of the following statements is/are TRUE? **(Gate 2024 CS) (2 Marks)**

(MSQ)

(a) $n_a(w) > n_b(w)$

(b) $n_a(w) > n_c(w) - 2$

(c) $n_c(w) = n_b(w) + 1$

(d) $n_c(w) = n_b(w) * 2$

**Q. Which ONE of the following languages is accepted by a deterministic pushdown automaton?
(Gate 2025)**

- A) Any regular language.**
- B) Any context-free language.**
- C) Any language accepted by a non-deterministic pushdown automaton.**
- D) Any decidable language.**

Decision properties

Following properties are decidable in case a CFL. Here we will use Grammar model to proof decision properties.

- i) Emptiness
- ii) Non-emptiness
- iii) Finiteness
- iv) Infiniteness
- v) Membership

Q consider the following CFG and identify which of the following CFG generate Empty language?

$S \rightarrow aAB / Aa$

$A \rightarrow a$

$S \rightarrow aAB$

$A \rightarrow a / b$

$S \rightarrow aAB / aB$

$A \rightarrow aBb$

$B \rightarrow aA$

Q consider the following CFG and identify which of the following CFG generate Finite language?

S -> **SS / AB**

A -> **BC / a**

B -> **CC / b**

S -> **AB**

A -> **B / a**

S -> **AB**

A -> **BC / a**

B -> **CC / b**

C -> **AB**

Q consider the following CFG and check out the membership properties?

$S \rightarrow AB / BB$

$A \rightarrow BA / AS / b$

$B \rightarrow AA / SB / a$

$w_1 = aba$

$w_2 = abaab$

$w_3 = abababba$

CYK algorithm

- In computer science, the **Cocke–Younger–Kasami algorithm** (alternatively called **CYK**, or **CKY**) is a parsing algorithm for context-free grammars, named after its inventors, John Cocke, Daniel Younger and Tadao Kasami. It employs bottom-up parsing and dynamic programming.
- The standard version of CYK operates only on context-free grammars given in Chomsky normal form (CNF). However any context-free grammar may be transformed to a CNF grammar expressing the same language (Sipser 1997).
- The importance of the CYK algorithm stems from its high efficiency in certain situations. Using Big O notation, the worst case running time of CYK is $O(n^3 \cdot |G|)$, Where n is the length of the parsed string and $|G|$ is the size of the CNF grammar G.
- This makes it one of the most efficient parsing algorithms in terms of worst-case **asymptotic complexity**, although other algorithms exist with better average running time in many practical scenarios.

Following properties are Undecidable in case a CFL

- i) Equality
- ii) Ambiguity

Q Which of the following problems is undecidable? (GATE-2014) (1 Marks)

- (A) Deciding if a given context-free grammar is ambiguous.
- (B) Deciding if a given string is generated by a given context-free grammar.
- (C) Deciding if the language generated by a given context-free grammar is empty.
- (D) Deciding if the language generated by a given context-free grammar is finite.

Q Which of the following are decidable? (GATE-2008) (2 Marks)

- I. Whether the intersection of two regular languages is infinite
 - II. Whether a given context-free language is regular
 - III. Whether two push-down automata accept the same language
 - IV. Whether a given grammar is context-free
-
- a). I and II
 - b). I and IV
 - c). II and III
 - d). II and IV

Q Which of the following problems is undecidable? (GATE-2007) (1 Marks)

(A) Membership problem for CFGs (B) Ambiguity problem for CFGs.

(C) Finiteness problem for FSAs.

(D) Equivalence problem for FSAs.

Q Which one of the following statements is FALSE? (GATE-2004) (1 Marks)

- a) There exist context free languages such that all the context free grammars generating them are ambiguous.
- b) An unambiguous context free grammar always has a unique parse tree for each string of the language generated by it.
- c) Both deterministic and non – deterministic pushdown automata always accept the same set of languages
- d) A finite set of strings from one alphabet is always a regular language.

Q Consider the following decision problems? (GATE-2000) (2 Marks)

(P₁) Does a given finite state machine accept a given string

(P₂) Does a given context free grammar generate an infinite number of strings

Which of the following statements is true?

- (A)** Both (P₁) and (P₂) are decidable
- (B)** Neither (P₁) nor (P₂) are decidable
- (C)** Only (P₁) is decidable
- (D)** Only (P₂) is decidable

	RL	DCFL	CFL	CSL	RS	RES
Emptiness	Y	Y	Y	X	N	N
Non-Emptiness	Y	Y	Y	X	N	N
Finiteness	Y	Y	Y	X	N	N
Infiniteness	Y	Y	Y	X	N	N
Membership	Y	Y	Y	X	Y	N
Equality	Y	N	N	X	N	N
Ambiguity	Y	N	N	X	N	N
Σ^*	Y	N	N	X	N	N
Halting	Y	Y	Y	X	Y	N

Closure Properties of Deterministic Context Free Languages

- Deterministic Context Free Languages are closed under following operations
 - Complement
 - Intersection with regular set
 - Inverse Homeomorphism
- Deterministic Context Free Languages are not closed under following operations
 - Union
 - Concatenation
 - Kleen closure
 - homomorphism
 - Substitution
 - Reverse operator
 - Intersection

Closure Properties of Context Free Languages

- Context Free Languages are closed under following operations
 - Union
 - Concatenation
 - Kleen Closure
 - Substitution
 - Homomorphism
 - Inverse Homomorphism
 - Reverse Operator
 - Intersection with regular set
- Context Free Languages are not closed under following operations
 - Intersection
 - Complement
 - Symmetric Difference

	RL	DCFL	CFL	CSL	RS	RES
Union	Y	N	Y	Y	Y	Y
Intersection	Y	N	N	Y	Y	Y
Complement	Y	Y	N	Y	Y	N
Set Difference	Y	N	N	Y	Y	N
Kleene Closure	Y	N	Y	Y	Y	Y
Positive Closure	Y	N	Y	Y	Y	Y
Concatenation	Y	N	Y	Y	Y	Y
Intersection with regular set	Y	Y	Y	Y	Y	Y
Reverse	Y	Y	Y	Y	Y	Y
Subset	N	N	N	N	N	N

	RL	DCFL	CFL	CSL	RS	RES
Homomorphism	Y	N	Y	N	N	Y
\in Free Homomorphism	Y	N	Y	Y	Y	Y
Inverse Homomorphism	Y	Y	Y	Y	Y	Y
Substitution	Y	N	Y	N	N	Y
\in Free Substitution	Y	N	Y	Y	Y	Y
Quotient with regular set	Y	Y	Y	N	Y	Y

Q For a string w , we define wR to be the reverse of w . For example, if $w = 01101$ then $wR = 10110$. Which of the following languages is/are context-free? **(GATE 2021) (2 MARKS)**

- (A) $\{wxwRxR \mid w, x \in \{0,1\}^*\}$
- (B) $\{wwRxxR \mid w, x \in \{0,1\}^*\}$
- (C) $\{wxwR \mid w, x \in \{0,1\}^*\}$
- (D) $\{wxxRwR \mid w, x \in \{0,1\}^*\}$

Q Consider the following languages:

$$L_1 = \{ww \mid w \in \{a,b\}^*\}$$

$$L_2 = \{a^n b^n c^m \mid m, n \geq 0\}$$

$$L_3 = \{a^m b^n c^n \mid m, n \geq 0\}$$

Which of the following statements is/are FALSE? (GATE 2022) (2 MARKS)

(A) L_1 is not context-free but L_2 and L_3 are deterministic context-free.

(B) Neither L_1 nor L_2 is context-free.

(C) L_2 , L_3 and $L_2 \cap L_3$ all are context-free.

(D) Neither L_1 nor its complement is context-free.

Q Consider the following languages:

$$L_1 = \{a^n w a^n \mid w \in \{a,b\}^*\}$$

$$L_2 = \{wxw^R \mid w, x \in \{a,b\}^*, |w|, |x| > 0\}$$

Note that w^R is the reversal of the string w . Which of the following is/are TRUE? **(GATE 2022) (2 MARKS)**

- (A) L_1 and L_2 are regular.
- (B) L_1 and L_2 are context-free.
- (C) L_1 is regular and L_2 is context-free.
- (D) L_1 and L_2 are context-free but not regular.

Q Suppose that L_1 is a regular language and L_2 is a context-free language.
Which one of the following languages is NOT necessarily context-free?
(GATE 2021)

- (a) $L_1 \cap L_2$
- (b) $L_1 \cdot L_2$
- (c) $L_1 - L_2$
- (d) $L_1 \cup L_2$

Q Let L1 be a regular language and L2 be a context-free language. Which of the following languages is/are context-free? (GATE 2021) (1 MARKS)

- (A) $L_1 \cap L_2'$
- (B) $(L_1' \cup L_2')'$
- (C) $L_1 \cup (L_2 \cup L_2')$
- (D) $(L_1 \cap L_2) \cup (L_1 \cap L_2')$

Q Let L_1 and L_2 be any context-free language and R be any regular language. Then, which of the following is correct? (GATE-2017) (2 Marks)

I. $L_1 \cup L_2$ is context-free.

II. L_1' is context-free.

III. $L_1 - R$ is context-free.

IV. $L_1 \cap L_2$ is context-free

a) I, II and IV only

b) I and III only

c) II and IV only

d) I only

Q Consider the following languages over the alphabet $\Sigma = \{a, b, c\}$.

Let $L_1 = \{a^n b^n c^m \mid m, n \geq 0\}$ and $L_2 = \{a^m b^n c^n \mid m, n \geq 0\}$.

Which of the following are context-free languages? **(GATE-2017) (2 Marks)**

I. $L_1 \cup L_2$

II. $L_1 \cap L_2$

- a) I only
- c) I and II

- b) II only
- d) Neither I nor II

Q Which one of the following statements is FALSE? (GATE-2013) (1 Marks)

$$L_1 = \{0^p 1^q 0^r \mid p, q, r \geq 0\}$$

$$L_2 = \{0^p 1^q 0^r \mid p, q, r \geq 0, p \neq r\}$$

- (A) L_2 is context-free.
- (B) L_1 intersection L_2 is context-free.
- (C) Complement of L_2 is recursive.
- (D) Complement of L_1 is context-free but not regular.

Q. Let $L = L_1 \cap L_2$ are languages as defined below: (GATE 2009, 2 Marks)

$$L_1 = \{a^m b^m c a^n b^n \mid m, n \geq 0\}$$

$$L_2 = \{a^i b^j c^k \mid i, j, k \geq 0\}$$

- a) Non recursive
- b) Regular
- c) Context Free but not regular
- d) Recursively enumerable but not context free

**Q Let L be a context - free language and M a regular language. Then the language $L \cap M$ is
(GATE-2006) (1 Marks)**

a) Always regular

b) Never regular

c) Always a deterministic context free language

d) Always a context – free language

Q Consider the languages: (GATE-2005) (2 Marks)

$$L_1 = \{a^n b^n c^m \mid n, m > 0\}$$

$$L_2 = \{a^n b^m c^m \mid n, m > 0\}$$

Which one of the following statements is FALSE?

- (A) $L_1 \cap L_2$ is a context-free language
- (B) $L_1 \cup L_2$ is a context-free language
- (C) L_1 and L_2 are context-free language
- (D) $L_1 \cap L_2$ is a context sensitive language

Q Let L be a regular language and M be a context-free language, both over the alphabet Σ . Let L^c and M^c denote the complements of L and M respectively. Which of the following statements about the language $L^c \cup M^c$ is TRUE? **(GATE-2005) (2 Marks)**

- (A) It is necessarily regular but not necessarily context-free
- (B) It is necessarily context-free.
- (C) It is necessarily non-regular.
- (D) None of the above

Q Which of the following statements is true? (GATE-2001) (1 Marks)

- (A)** If a language is context free it can always be accepted by a deterministic push-down automaton
- (B)** The union of two context free languages is context free
- (C)** The intersection of two context free languages is context free
- (D)** The complement of a context free language is context free

Q Consider the following decision problems (GATE-2000) (2 Marks)

(P₁) Does a given finite state machine accept a given string

(P₂) Does a given context free grammar generate an infinite number of strings

Which of the following statements is true?

- (A)** Both (P₁) and (P₂) are decidable
- (B)** Neither (P₁) nor (P₂) are decidable
- (C)** Only (P₁) is decidable
- (D)** Only (P₂) is decidable

Q Let L_1 is context free language and L_2 is a regular language which of the following is/are false (GATE – 1999) (1 Marks)

- a) $L_1 - L_2$ is not context free
- b) $L_1 \cap L_2$ is context free
- c) $\sim L_1$ is context free
- d) $\sim L_2$ is regular

Q Context – free language is closed under: (GATE – 1999) (1 Marks)

a) Union, intersection

b) Union, Kleene closure

c) Intersection, complement

d) Complement, Kleene closure

Q. Let L_1 and L_2 are context free language and R a regular set, one of the languages below is not necessarily a context free language. Which one? (GATE 1996, 1 Marks)

- a) L_1, L_2
- b) $L_1 \cap L_2$
- c) $L_1 \cap R$
- d) $L_1 \cup L_2$

Q Context – free languages are (GATE – 1992) (1 Marks)

a) Closed under union

b) Closed under complementation

c) Closed under intersection

d) Closed under Kleene closure

Q. Let G_1, G_2 be Context Free Grammars (CFGs) and R be a regular expression. For a grammar G , let $L(G)$ denote the language generated by G .

Which ONE among the following questions is decidable? **(Gate 2025)**

- A) Is $L(G_1) = L(G_2)$?
- B) Is $L(G_1) \cap L(G_2) = \emptyset$?
- C) Is $L(G_1) = L(R)$?
- D) Is $L(G_1) = \emptyset$?

Q. Consider the following two languages over the alphabet $\{a, b, c\}$, where m and n are natural numbers.

$$L_1 = \{a^m b^m c^{m+n} \mid m, n \geq 1\}$$

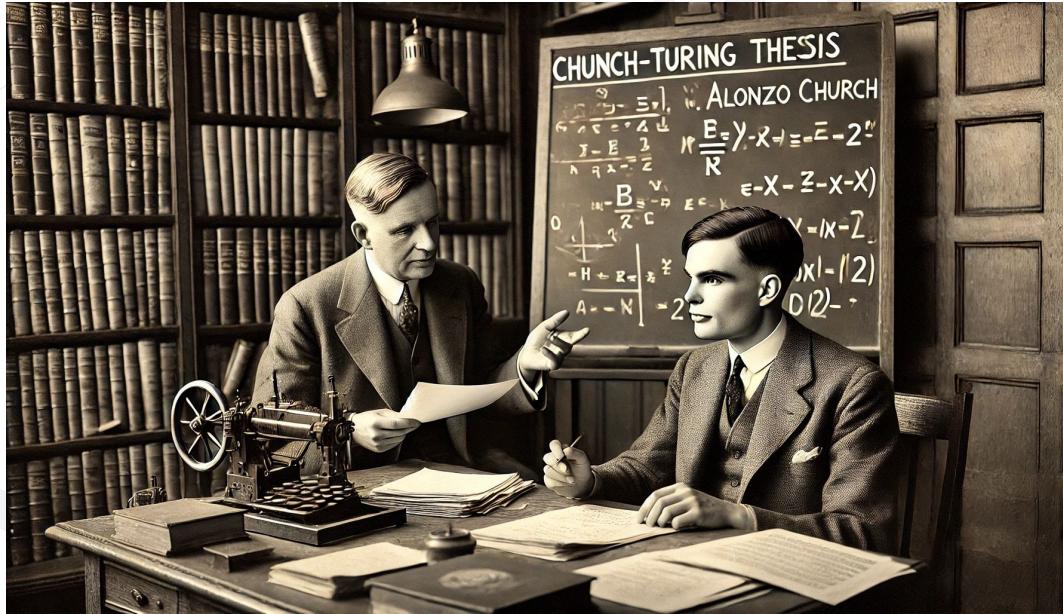
$$L_2 = \{a^m b^n c^{m+n} \mid m, n \geq 1\}$$

Which ONE of the following statements is CORRECT? (Gate 2025)

- A) Both L_1 and L_2 are context-free languages.
- B) L_1 is a context-free language but L_2 is not a context-free language.
- C) L_1 is not a context-free language but L_2 is a context-free language.
- D) Neither L_1 nor L_2 are context-free languages.

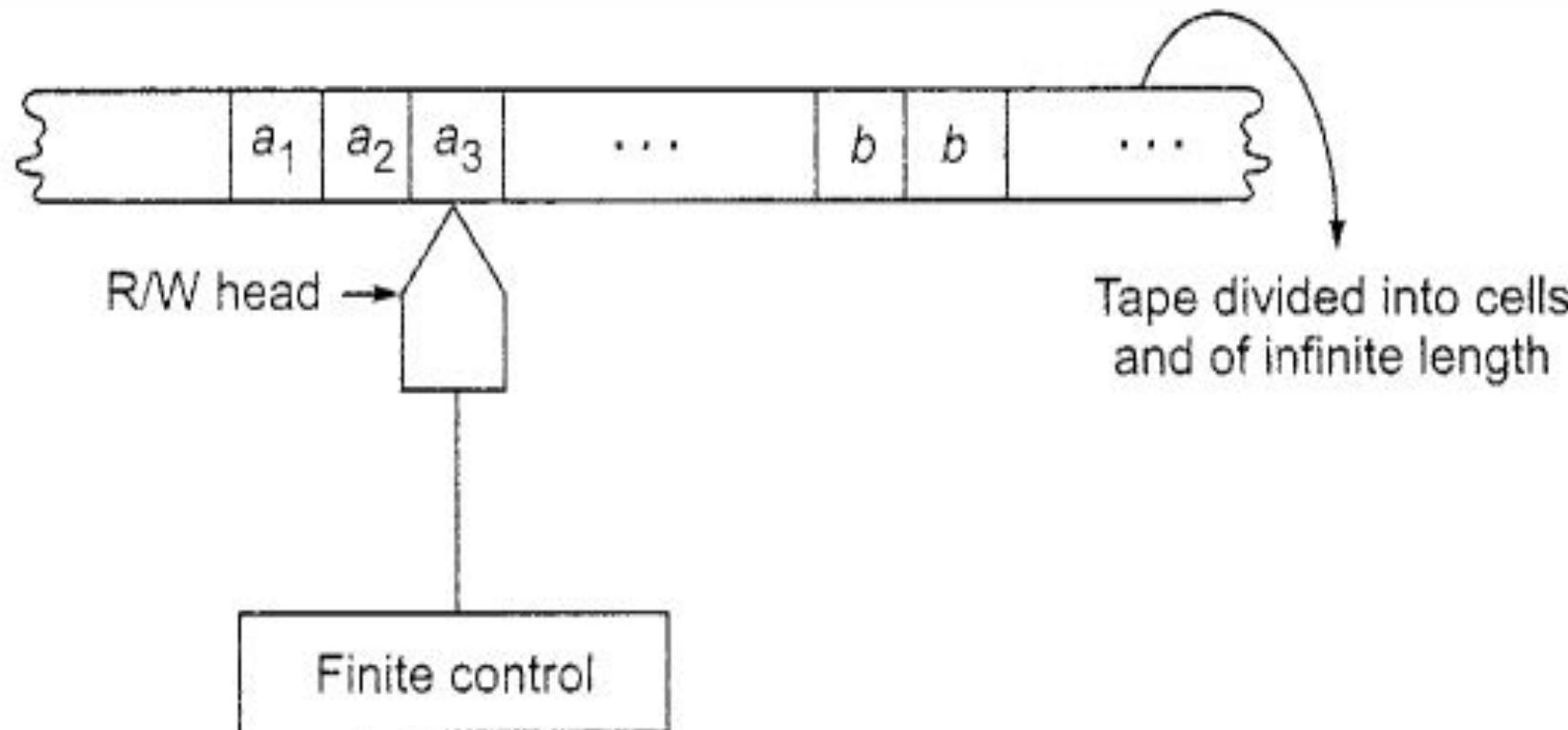
Turing Machine

- The **Church-Turing thesis** posits that any computational task performable by a human or machine can be executed by a Turing machine. It is widely regarded as a fundamental theoretical model of computation in computer science.
- Turing machines serve several key purposes:
 - **General Automaton:** Turing machines are the most general form of automaton, capable of simulating any algorithmic process.
 - **Type-0 Language Acceptance:** They can recognize type-0 languages, which are the most complex in the Chomsky hierarchy.
 - **Function Computation:** Turing machines can compute any computable function.
 - **Undecidability & Complexity:** They help determine the undecidability of certain languages and are used to measure both time and space complexity for computational problems.



- Components of a Turing Machine:

- **Infinite Tape**: The tape acts as both input storage and memory. It is infinitely long and divided into cells, each capable of holding a single symbol. The tape can be accessed randomly in either direction, referred to as a two-way infinite tape.
- **Read-Write Head**: The head reads symbols from the tape and can also write over them. After reading or writing, it moves one cell either to the left or right.
- **Finite Control Unit**: The control unit governs transitions based on the current state and symbol under the read-write head. It determines the next action, ultimately leading to the machine's output.



FORMAL DEFINITION

A Turing machine M is a 7-tuple, namely $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$, where

- Q is a finite nonempty set of states.
- Γ is a finite nonempty set of tape symbols,
- $b \in \Gamma$ is the blank.
- Σ is a nonempty set of input symbols and is a subset of Γ and $b \notin \Sigma$.
- δ is the transition function mapping $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L/R\}$. It says that, on providing a tape symbol, from a particular state there will be a transition to another state, with a different or same tape symbol with defining whether next the machine needs to move in left/ right.
- $q_0 \in Q$ is the initial state, and
- $F \subseteq Q$ is the set of final states.

Q Design a Turing machine for $L = \{ab\}^*$?

Q Design a Turing machine for $L = \{a, ab\}$?

Q Consider the following machines and corresponding transition function

Machine	Transition function
1. DFA	A. $Q \times \Sigma \cup \{\epsilon\} \times \Gamma \rightarrow Q \times \Gamma^*$
2. NFA	B. $Q \times \Sigma \rightarrow 2^Q$
3. PDA	C. $Q \times \Sigma \rightarrow Q$
4. TM	D. $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
	E. $Q \times \Sigma \rightarrow \Delta$
	F. $Q \rightarrow \Delta$

REPRESENTATION OF TURING MACHINES

Three representations:

- Instantaneous descriptions using move-relations.
- Transition table, and
- Transition diagram (transition graph).

LANGUAGE ACCEPTABILITY BY TURING MACHINES

- A string w in Σ^* is said to be accepted by a $\text{TM}(M)$ if after parsing the string w Turing machine must halts on final state.
- $q_0 w \vdash^* \alpha_1 p \alpha_2$ for some $p \in F$ and $\alpha_1, \alpha_2 \in \Gamma^*$.
- M does not accept w if the machine M either halts in a non-accepting state or does not halt and goes into a loop.

Q Design a Turing machine for $L = \{a^n b^n \mid n \geq 1\}$?

Q Design a Turing machine for $L = \{a^n b^n c^n \mid n \geq 0\}$?

Q Design a Turing machine for $L = \{w c w \mid w \in \{0, 1\}^*\}$?

Q Design a Turing machine for addition of two number in unary?

Q Design a Turing machine for converting unary number into binary number?

Q A single tape Turing Machine M has two states q_0 and q_1 , of which q_0 is the starting state. The tape alphabet of M is $\{0, 1, B\}$ and its input alphabet is $\{0, 1\}$. The symbol B is the blank symbol used to indicate end of an input string. The transition function of M is described in the following table. Which of the following statements is true about M ?

- (A) M does not halt on any string in $(0 + 1)^+$
- (B) M does not halt on any string in $(00 + 1)^*$
- (C) M halts on all string ending in a 0
- (D) M halts on all string ending in a 1

	0	1	B
q_0	$q_1, 1, R$	$q_1, 1, R$	Halt
q_1	$q_1, 1, R$	$q_0, 1, L$	q_0, B, L

Q Consider the transition table of a TM given below. Here “b” represents the blank symbol. Which of the following strings will not be accepted?

- a) 010101
- b) 101010
- c) 110011
- d) Both a and b

δ	0	1	b
q_0	$q_0, 0R$	$q_1, 0R$	$q_2, 0R$
q_1	$q_1, 0R$	$q_0, 0R$	-
q_2	-	-	-

Q The given Turing machine accepts

- a) set of all even palindromes over $\{0, 1\}$

- b) strings over $\{0, 1\}$ containing even number of 1's

- c) strings over $\{0, 1\}$ containing even number of 1's
and odd no. of 0's

- d) string over $\{0, 1\}$ starting with zero

δ	0	1	b
q_0	$q_0 \text{ OR}$	$q_1 \text{ OR}$	$q_2 \text{ OR}$
q_1	$q_1 \text{ OR}$	$q_0 \text{ OR}$	-
q_2	-	-	-

Q Consider the following languages.

$$L_1 = \{a^p \mid p \text{ is a prime number}\}$$

$$L_2 = \{a^n b^m c^{2m} \mid n \geq 0, m \geq 0\}$$

$$L_3 = \{a^n b^n c^{2n} \mid n \geq 0\}$$

$$L_4 = \{a^n b^n \mid n \geq 1\}$$

Which of the following are CORRECT? (GATE-2017) (2 Marks)

- I. L_1 is context free but not regular.
- II. L_2 is not context free.
- III. L_3 is not context free but recursive
- IV. L_4 is deterministic context free

- a) I, II and IV only
- b) II and III only
- c) I and IV only
- d) III and IV only

Q Which of the following is true for the language

$$\{a^p \mid p \text{ is a prime}\}?$$

(GATE-2008) (2 Marks)

(A) It is not accepted by a Turing Machine

(B) It is regular but not context-free

(C) It is context-free but not regular

(D) It is neither regular nor context-free, but accepted by a Turing machine

Q For $S \in (0 + 1)^*$ let $d(s)$ denote the decimal value of s (e.g. $d(101) = 5$).

Let $L = \{s \in (0 + 1)^* \mid d(s) \bmod 5 = 2 \text{ and } d(s) \bmod 7 \neq 4\}$. **(GATE-2006) (2 Marks)**

Which one of the following statements is true?

(A) L is recursively enumerable, but not recursive

(B) L is recursive, but not context-free

(C) L is context-free, but not regular

(D) L is regular

Q L_1 is a recursively enumerable language over Σ . An algorithm A effectively enumerates its words as w_1, w_2, w_3, \dots . Define another language L_2 over Σ Union {#} as $\{w_i \# w_j : w_i, w_j \in L_1, i < j\}$. Here # is a new symbol. Consider the following assertions. (GATE-2004) (2 Marks)

S_1 : L_1 is recursive implies L_2 is recursive

S_2 : L_2 is recursive implies L_1 is recursive

Which of the following statements is true ?

- (A) Both S_1 and S_2 are true
- (B) S_1 is true but S_2 is not necessarily true
- (C) S_2 is true but S_1 is not necessarily true
- (D) Neither is necessarily true

- **Deterministic and Non-Deterministic Turing Machines:**
 - **Deterministic:** Each state and symbol pair leads to a single possible move.
 - **Non-Deterministic:** Multiple moves may be possible for a given state and symbol. However, non-deterministic Turing machines do not add computational power and can always be converted into a deterministic Turing machine.

Versions of Turing Machines:

- **Multi-Tape Turing Machine**: Uses multiple tapes with corresponding read/write heads. However, every multi-tape machine can be simulated by a single-tape machine.
- **Multi-Head Turing Machine**: A machine with multiple read/write heads.
- **Multi-Dimensional Turing Machine**: Operates on tapes of multiple dimensions (e.g., 2D grid).
- **Turing Machine with Stay Option**: The head can remain in place instead of moving left or right.
- **One-Way Infinite Tape (Semi-Infinite Tape)**: The tape is infinite in only one direction.
- **Offline Turing Machine**: The input tape is read-only and cannot be modified.
- **Jumping Turing Machine**: Can make multiple moves in one transition.
- **Non-Erasing Turing Machine**: Cannot erase or overwrite the input symbols with blanks.
- **Always Writing Turing Machine**: Must replace every symbol it reads with another symbol.
- **Finite Automaton with Queue**: Combines a finite automaton with an unbounded queue.
- **Turing Machine with 3 States**: A simplified Turing machine with only 3 states.
- **Multi-Tape TM with Stay Option and 2 States**: A variation with multiple tapes and stay options but limited to two states.
- **Non-Deterministic TM**: Similar to the non-deterministic Turing machine but with a stay option.
- **NPDA with Two Stacks**: A non-deterministic pushdown automaton with two independent stacks.

Q Which of the following pairs have DIFFERENT expressive power? (GATE-2011) (1 Marks)

- a) Deterministic finite automata (DFA) and non – deterministic finite automata (NFA)**

- b) Deterministic push down automata (DPDA) and Non – deterministic push down automata (NPDA)**

- c) Deterministic single – tape Turing machine and Non – deterministic single tape Turing machine**

- d) Single – tape Turing machine and multi – tape Turing machine**

Halt in a Turing Machine

- When a Turing machine reaches a state where no further transitions are defined or needed, it is said to **halt**. There are two types of halts:
 - **Final Halt:** The machine halts in a final (accepting) state, indicating that the input string is accepted.
 - **Non-Final Halt:** The machine halts in a non-final state, meaning the input string is rejected.
- After processing an input string, a Turing machine can experience one of three outcomes:
 - **Final Halt:** The machine accepts the string.
 - **Non-Final Halt:** The machine rejects the string.
 - **Infinite Loop:** The machine enters an infinite loop, in which case it is undecidable whether the string is accepted or rejected.

Recursively Enumerable Languages

- **Recursive Set:**

- A language L is a recursive set if it is accepted by a Turing machine in such a way that:
 - For all $w \in L$, the machine halts in a final state (accepts).
 - For all $w \notin L$, the machine halts in a non-final state (rejects).
- The membership property is clearly defined here—every string is either accepted or rejected.

- **Recursively Enumerable Set (REL):**

- A language L is recursively enumerable if it is accepted by a Turing machine, where:
 - For all $w \in L$, the machine halts in a final state (accepts).
 - For all $w \notin L$, the machine may either halt in a non-final state (reject) or enter an infinite loop.
- The membership property is not defined because the machine might never halt for certain strings.

- **Decidability:**

- A set is **decidable** if both the set and its complement are recognizable by a Turing machine.
- Some sets are **not recognizable**, meaning that even the members of the set cannot be identified, as there are more languages than programs available to recognize them.

Q Let L be a language and L' be its complement. Which one of the following is NOT a viable possibility? **(GATE-2014) (1 Marks)**

- (A) Neither L nor L' is recursively enumerable (r.e.).
- (B) One of L and L' is r.e. but not recursive; the other is not r.e.
- (C) Both L and L' are r.e. but not recursive.
- (D) Both L and L' are recursive

Q Which of the following statements is false? (GATE-2008) (1 Marks)

- (A) Every NFA can be converted to an equivalent DFA
- (B) Every non-deterministic Turing machine can be converted to an equivalent deterministic Turing machine
- (C) Every regular language is also a context-free language
- (D) Every subset of a recursively enumerable set is recursive

Q If L and L' are recursively enumerable, then L is (GATE-2008) (1 Marks)

(A) regular

(B) context-free

(C) context-sensitive

(D) recursive

Q Let L_1 be a recursive language, and let L_2 be a recursively enumerable but not a recursive language. Which one of the following is TRUE? **(GATE-2005) (1 Marks)**

L_1' Complement of L_1

L_2' Complement of L_2

(A) L_1' is recursive and L_2' is recursively enumerable

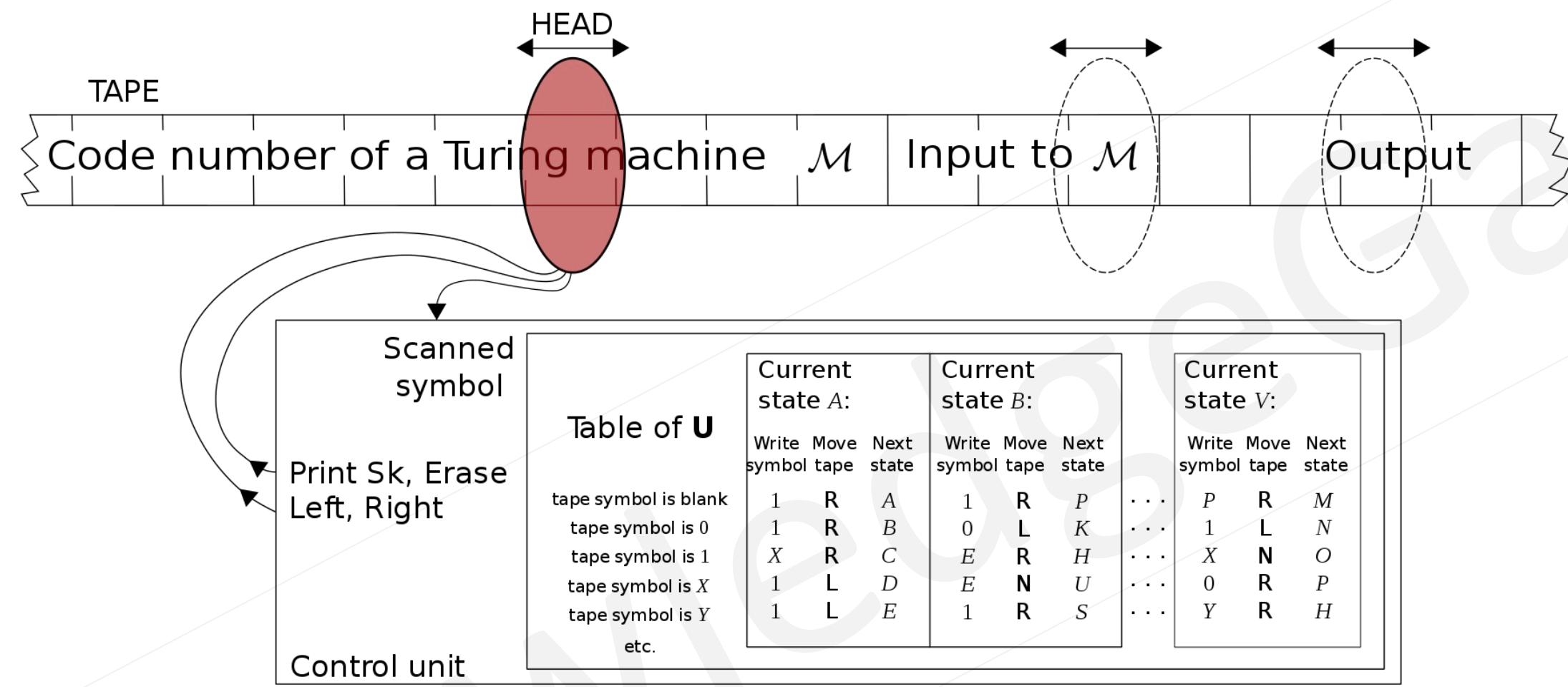
(B) L_1' is recursive and L_2' is not recursively enumerable

(C) L_1' and L_2' are recursively enumerable

(D) L_1' is recursively enumerable and L_2' is recursive

Universal Turing Machine (UTM)

- In computer science, a **Universal Turing Machine** (UTM) is a Turing machine capable of simulating any other Turing machine on any input. It does this by reading both the description of the machine to be simulated and the input for that machine from its own tape.
 - Every Turing machine computes a fixed **partial computable function** from input strings over its alphabet, behaving like a computer with a specific program.
 - The action table of any Turing machine can be encoded into a string. A UTM can then read this string, which describes the action table, followed by a string that describes the input tape, and simulate the encoded Turing machine's behavior.
- In his 1936 paper, Turing described this concept in detail, stating:
 - *"It is possible to invent a single machine which can be used to compute any computable sequence. If this machine UUU is supplied with a tape containing the 'standard description' (S.D.) of some machine MMM, UUU will compute the same sequence as MMM."*



Q For a Turing machine M, $\langle M \rangle$ denotes an encoding of M. Consider the following two languages.

- **L1** = { $\langle M \rangle$ | M takes more than 2021 steps on all inputs }
- **L2** = { $\langle M \rangle$ | M takes more than 2021 steps on some input }

Which one of the following options is correct? **(GATE 2021) (2 MARKS)**

- (a) Both L1 and L2 are decidable
- (b) L1 is decidable and L2 is undecidable
- (c) L1 is undecidable and L2 is decidable
- (d) Both L1 and L2 are undecidable

Q Let $\langle M \rangle$ denote an encoding of an automaton M . Suppose that $\Sigma = \{0,1\}$. Which of the following languages is/are NOT recursive? **(GATE 2021)**

- (a) $L = \{\langle M \rangle \mid M \text{ is a DFA such that } L(M) = \emptyset\}$
- (b) $L = \{\langle M \rangle \mid M \text{ is a DFA such that } L(M) = \Sigma^*\}$
- (c) $L = \{\langle M \rangle \mid M \text{ is a PDA such that } L(M) = \emptyset\}$
- (d) $L = \{\langle M \rangle \mid M \text{ is a PDA such that } L(M) = \Sigma^*\}$

Q $L_1 = \{ \langle M \rangle \mid M \text{ takes at least } 2016 \text{ steps on some input}\},$

$L_2 = \{ \langle M \rangle \mid M \text{ takes at least } 2016 \text{ steps on all inputs}\}$ and

$L_3 = \{ \langle M \rangle \mid M \text{ accepts } \epsilon\},$

where for each Turing machine M , $\langle M \rangle$ denotes a specific encoding of M .

Which one of the following is TRUE? **(GATE-2016) (2 Marks)**

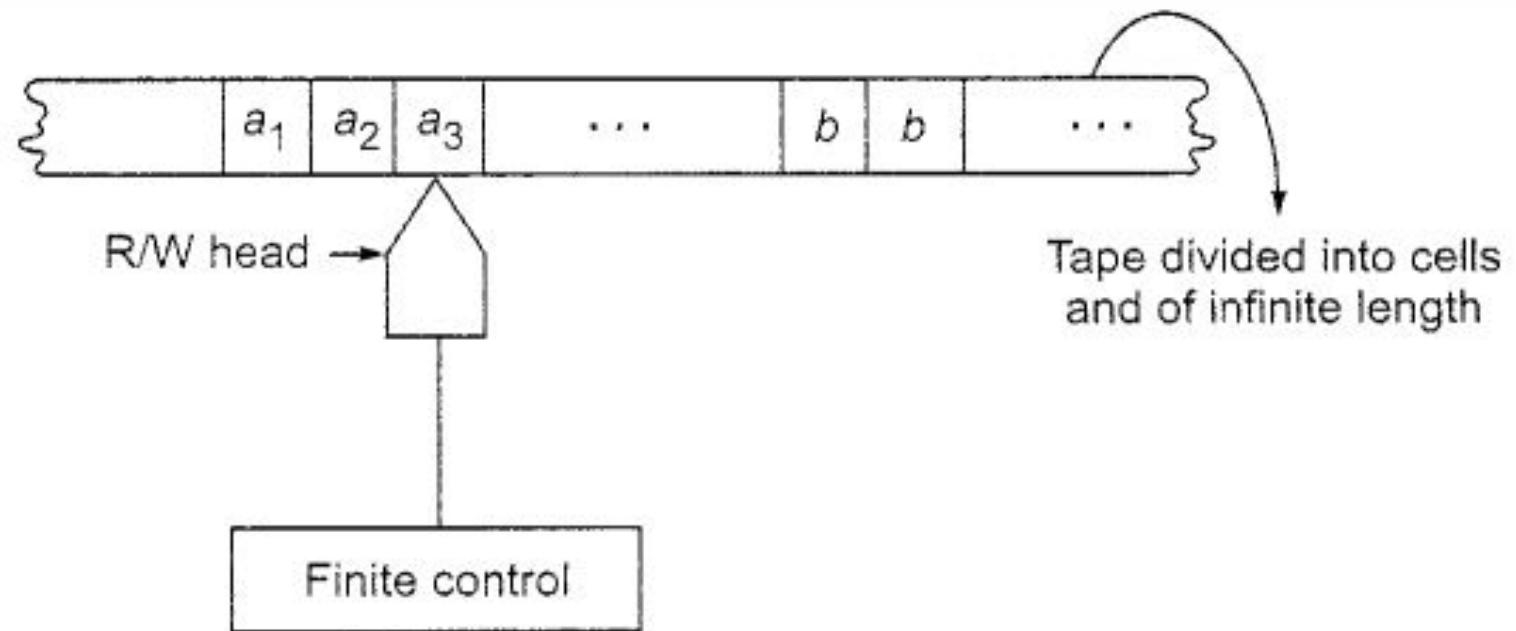
- (A) L_1 is recursive and L_2, L_3 are not recursive
- (B) L_2 is recursive and L_1, L_3 are not recursive
- (C) L_1, L_2 are recursive and L_3 is not recursive
- (D) L_1, L_2, L_3 are recursive

Q Let $\langle M \rangle$ be the encoding of a Turing machine as a string over $\{0, 1\}$.
Let $L = \{\langle M \rangle \mid M \text{ is a Turing machine that accepts a string of length 2014}\}$.
Then, L is **(GATE-2014) (2 Marks)**

- (A)** decidable and recursively enumerable
- (B)** undecidable but recursively enumerable
- (C)** undecidable and not recursively enumerable
- (D)** decidable but not recursively enumerable

Linear Bounded Automaton (LBA)

- An **LBA** is a variant of a Turing machine with the following property: The tape length is finite and bounded, unlike a traditional Turing machine that has an infinite tape.



Decision properties

- Following properties are decidable in case a RS.
 - Membership
- All properties are undecidable in case of a REL.

	RL	DCFL	CFL	CSL	RS	RES
Emptiness	Y	Y	Y	X	N	N
Non-Emptiness	Y	Y	Y	X	N	N
Finiteness	Y	Y	Y	X	N	N
Infiniteness	Y	Y	Y	X	N	N
Membership	Y	Y	Y	X	Y	N
Equality	Y	N	N	X	N	N
Ambiguity	Y	N	N	X	N	N
Σ^*	Y	N	N	X	N	N
Halting	Y	Y	Y	X	Y	N

Q Which of the following is/are undecidable? (GATE 2022) (2 MARKS)

- (A) Given two Turing machines M_1 and M_2 , decide if $L(M_1) = L(M_2)$.
- (B) Given a Turing machine M , decide if $L(M)$ is regular.
- (C) Given a Turing machine M , decide if M accepts all strings.
- (D) Given a Turing machine M , decide if M takes more than 1073 steps on every string.

Q Consider the following problems. $L(G)$ denotes the language generated by a grammar G. $L(M)$ denotes the language accepted by a machine M.

- (I)** For an unrestricted grammar G and a string w, whether $w \in L(G)$
- (II)** Given a Turing machine M, whether $L(M)$ is regular
- (III)** Given two grammar G_1 and G_2 , whether $L(G_1) = L(G_2)$
- (IV)** Given an NFA N, whether there is a deterministic PDA P such that N and P accept the same language

Which one of the following statement is correct? **(GATE-2018) (2 Marks)**

- a)** Only I and II are undecidable
- b)** Only II is undecidable
- c)** Only II and IV are undecidable
- d)** Only I, II and III are undecidable

Q Let $L(R)$ be the language represented by regular expression R . Let $L(G)$ be the language generated by a context free grammar G . Let $L(M)$ be the language accepted by a Turing machine M . Which of the following decision problems are undecidable? **(GATE-2017) (2 Marks)**

I. Given a regular expression R and a string w , is $w \in L(R)$?

II. Given a context-free grammar G , is $L(G)=\emptyset$

III. Given a context-free grammar G , is $L(G)=\Sigma^*$ for some alphabet Σ ?

IV. Given a Turing machine M and a string w , is $w \in L(M)$?

a) I and IV only

b) II and III only

c) II, III and IV only

d) III and IV only

Q Which of the following decision problems are undecidable (GATE-2016) (2)

- I. Given NFAs N_1 and N_2 , is $L(N_1) \cap L(N_2) = \Phi$?
- II. Given a CFG $G = (N, \Sigma, P, S)$ and a string $x \in \Sigma^*$, does $x \in L(G)$?
- III. Given CFGs G_1 and G_2 , is $L(G_1) = L(G_2)$?
- IV. Given a TM M , is $L(M) = \Phi$?

(A) I and IV only

(B) II and III only

(C) III and IV only

(D) II and IV only

Q Which of the following is/are undecidable? (GATE-2013) (1 Marks)

- 1) G is a CFG. Is $L(G)=\emptyset$?
 - 2) G is a CFG. Is $L(G)=\Sigma^*$?
 - 3) M is a Turing machine. Is $L(M)$ regular?
 - 4) A is a DFA and N is an NFA. Is $L(A)=L(N)$?
- (A) 3 only (B) 3 and 4 only
(C) 1, 2 and 3 only (D) 2 and 3 only

Q Which of the following problems are decidable? (GATE-2012) (2 Marks)

- a) Does a given program ever produce an output?**
 - b) If L is a context-free language, then, is L' also context-free?**
 - c) If L is a regular language, then, is L' also regular?**
 - d) If L is a recursive language, then, is L' also recursive?**
-
- a) 1, 2, 3, 4**
 - b) 1, 2**
 - c) 2, 3, 4**
 - d) 3, 4**

Q Which of the following is not decidable? (GATE – 1997) (1 Marks)

- a) Given a Turing machine M, a strings s and an integer k, M accepts s within k steps
- b) Equivalence of two given Turing machines
- c) Language accepted by a given finite state machine is not empty
- d) Languages generated by a context free grammar is non empty

Closure Properties of Recursive Set

- Recursive languages are closed under following operations
 - Union
 - Concatenation
 - Intersection
 - Reverse
 - Complement
 - Inverse homomorphism
 - Intersection with regular set
 - Set Difference
 - Kleen closure
- Recursive languages are not closed under following operations
 - Homomorphism
 - Substitution

Closure Properties of Recursive Enumerable Set

- Recursive Enumerable are closed under following operations
 - Union
 - Concatenation
 - Kleen Closure
 - Intersection
 - Substitution
 - Homomorphism
 - Inverse Homomorphism
 - Intersection with regular set
 - Reverse operation
- Recursive Enumerable are not closed under following operations
 - Compliment
 - Set Difference

	RL	DCFL	CFL	CSL	RS	RES
Union	Y	N	Y	Y	Y	Y
Intersection	Y	N	N	Y	Y	Y
Complement	Y	Y	N	Y	Y	N
Set Difference	Y	N	N	Y	Y	N
Kleene Closure	Y	N	Y	Y	Y	Y
Positive Closure	Y	N	Y	Y	Y	Y
Concatenation	Y	N	Y	Y	Y	Y
Intersection with regular set	Y	Y	Y	Y	Y	Y
Reverse	Y	Y	Y	Y	Y	Y
Subset	N	N	N	N	N	N

	RL	DCFL	CFL	CSL	RS	RES
Homomorphism	Y	N	Y	N	N	Y
\in Free Homomorphism	Y	N	Y	Y	Y	Y
Inverse Homomorphism	Y	Y	Y	Y	Y	Y
Substitution	Y	N	Y	N	N	Y
\in Free Substitution	Y	N	Y	Y	Y	Y
Quotient with regular set	Y	Y	Y	N	Y	Y

Q Which of the following statements is/are TRUE? (GATE 2022) (1 MARKS)

- (A) Every subset of a recursively enumerable language is recursive.
- (B) If a language L and its complement L' are both recursively enumerable, then L must be recursive.
- (C) Complement of a context-free language must be recursive.
- (D) If L_1 and L_2 are regular, then $L_1 \cap L_2$ must be deterministic context-free.

Q The set of all recursively enumerable languages is **(GATE-2018) (1 Marks)**

a) closed under complementation.

b) closed under intersection.

c) a subset of the set of all recursive languages.

d) an uncountable set.

Q Consider the following types of languages (GATE-2016) (2 Marks)

L_1 : Regular, L_2 : Context-free L_3 : Recursive, L_4 : Recursively enumerable.

Which of the following is/are TRUE?

I. $L_3' \cup L_4$ is recursively enumerable

II. $L_2 \cup L_3$ is recursive

III. $L_1^* \cup L_2$ is context-free

IV. $L_1 \cup L_2'$ is context-free

(A) I only (B) I and III only

(C) I and IV only (D) I, II and III only

Q For any two languages L_1 and L_2 such that L_1 is context free and L_2 is recursively enumerable but not recursive, which of the following is/are necessarily true? **(GATE-2015) (2 Marks)**

1. L_1' is recursive
 2. L_2' is recursive
 3. L_1' is context-free
 4. $L_1' \cup L_2$ is recursively enumerable
- (A) 1 only (B) 3 only
(C) 3 and 4 only (D) 1 and 4 only

Q Let L be a language and L' be its complement. Which one of the following is NOT a viable possibility? **(GATE-2014) (2 Marks)**

- a) Neither L nor L' is recursively enumerable (r.e.).
- b) One of L and L' is r.e. but not recursive; the other is not r.e.
- c) Both L and L' are r.e. but not recursive.
- d) Both L and L' are recursive

Q Which of the following statements is/are FALSE? (GATE-2013) (2 Marks)

1. For every non-deterministic Turing machine, there exists an equivalent deterministic Turing machine.
 2. Turing recognizable languages are closed under union and complementation.
 3. Turing decidable languages are closed under intersection and complementation.
 4. Turing recognizable languages are closed under union and intersection.
- (A) 1 and 4 only (B) 1 and 3 only
(C) 2 only (D) 3 only

Q Let L_1 be a recursive language. Let L_2 and L_3 be languages that are recursively enumerable but not recursive. Which of the following statements is not necessarily true? **(GATE-2010) (1 Marks)**

(A) $L_2 - L_1$ is recursively enumerable.

(B) $L_1 - L_3$ is recursively enumerable

(C) $L_2 \cap L_1$ is recursively enumerable

(D) $L_2 \cup L_1$ is recursively enumerable

Q Which one of the following is FALSE? (GATE-2009) (1 Marks)

- A) There is unique minimal DFA for every regular language**

- B) Every NFA can be converted to an equivalent PDA**

- C) Complement of every context-free language is recursive.**

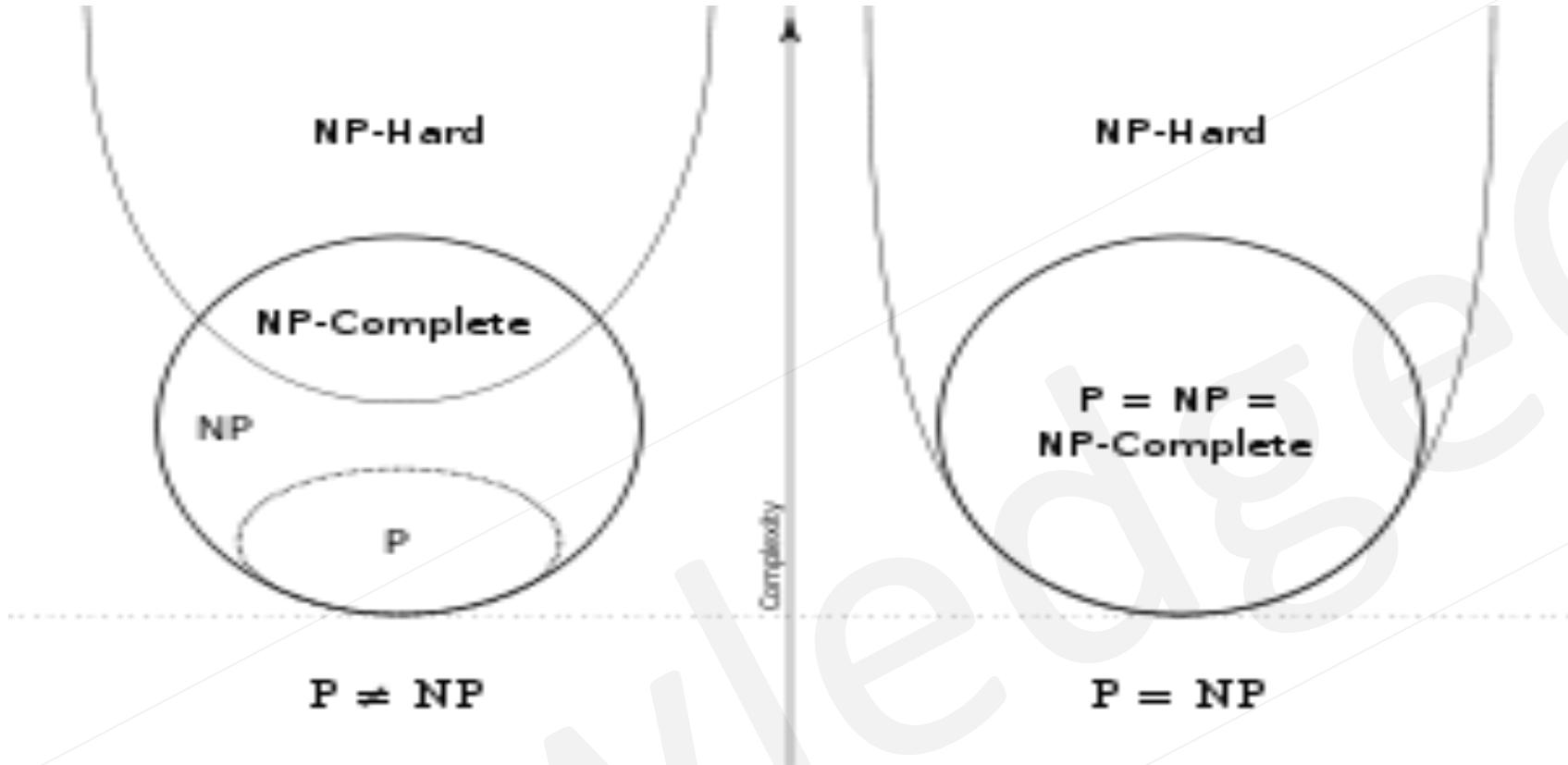
- D) Every non-deterministic PDA can be converted to an equivalent deterministic PDA**

Q If the strings of a language L can be effectively enumerated in lexicographic (i.e., alphabetic) order, which of the following statements is true? **(GATE-2003) (1 Marks)**

- (A)** L is necessarily finite
- (B)** L is regular but not necessarily finite
- (C)** L is context free but not necessarily regular
- (D)** L is recursive but not necessarily context free

Q Which of the following is true? (GATE-2002) (1 Marks)

- (A) The complement of a recursive language is recursive.
- (B) The complement of a recursively enumerable language is recursively enumerable.
- (C) The complement of a recursive language is either recursive or recursively enumerable.
- (D) The complement of a context-free language is context-free.



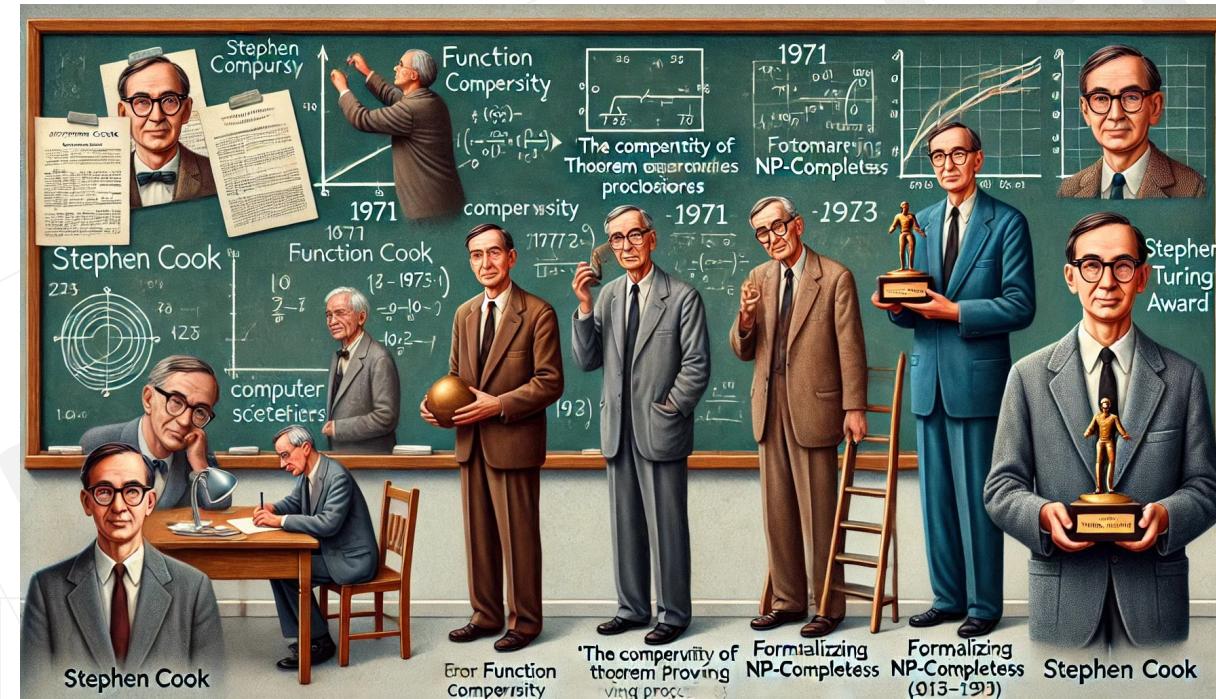
Cook-Levin Theorem (Cook's Theorem)

- In computational complexity theory, the **Cook-Levin theorem** states that the **Boolean satisfiability problem (SAT)** is **NP-complete**. This means:
 - SAT is in NP, and any problem in NP can be **reduced** in polynomial time by a deterministic Turing machine to SAT.
 - An important implication of this theorem is that if a deterministic polynomial-time algorithm exists for SAT, then every NP problem can be solved in polynomial time. This leads to the famous **P vs NP problem**, one of the most important open questions in theoretical computer science.



Stephen Cook's Contributions:

- During his PhD, Cook worked on function complexity, focusing on multiplication. In his 1971 landmark paper, "**The Complexity of Theorem Proving Procedures**", Cook formalized the concepts of **polynomial-time reduction** (also known as **Cook reduction**) and **NP-completeness**. He also proved the existence of NP-complete problems, showing that SAT is NP-complete.
- This theorem explores whether optimization problems, whose solutions can be efficiently verified, can also be solved efficiently.
- Cook proposed that some optimization problems (with easily verifiable solutions) **cannot** be solved by efficient algorithms, i.e., **P \neq NP**. This conjecture has spurred extensive research in computational complexity, enhancing our understanding of the difficulty of computational problems. However, the conjecture remains unresolved.
- In 1982, Cook received the **Turing Award** for his groundbreaking contributions to complexity theory.



Q Let X be a recursive language and Y be a recursively enumerable but not recursive language. Let W and Z be two languages such that Y reduces to W, and Z reduces to X (reduction means the standard many-one reduction). Which one of the following statements is TRUE (GATE-2016) (2 Marks)

- (A) W can be recursively enumerable and Z is recursive.
- (B) W can be recursive and Z is recursively enumerable.
- (C) W is not recursively enumerable and Z is recursive.
- (D) W is not recursively enumerable and Z is not recursive

Q Consider the following statements (GATE-2015) (2 Marks)

1. The complement of every Turing decidable language is Turing decidable
2. There exists some language which is in NP but is not Turing decidable
3. If L is a language in NP, L is Turing decidable

Which of the above statements is/are True?

- (A) Only 2 (B) Only 3
- (C) Only 1 and 2 (D) Only 1 and 3

Q Language L_1 is polynomial time reducible to language L_2 . Language L_3 is polynomial time reducible to L_2 , which in turn is polynomial time reducible to language L_4 . Which of the following is/are True? (GATE-2015) (2 Marks)

Q Consider two decision problems Q_1 , Q_2 such that Q_1 reduces in polynomial time to 3-SAT and 3-SAT reduces in polynomial time to Q_2 . Then which one of the following is consistent with the above statement? **(CS-2015) (2 Marks)**

- (A) Q_1 is in NP, Q_2 is NP hard
- (B) Q_2 is in NP, Q_1 is NP hard
- (C) Both Q_1 and Q_2 are in NP
- (D) Both Q_1 and Q_2 are in NP hard

Q consider the following two problems of graph. (GATE-2015) (2 Marks)

- 1)** Given a graph, find if the graph has a cycle that visits every vertex exactly once except the first visited vertex which must be visited again to complete the cycle.
- 2)** Given a graph, find if the graph has a cycle that visits every edge exactly once.

Which of the following is true about above two problems.

- (A)** Problem 1 belongs NP Complete set and 2 belongs to P
- (B)** Problem 1 belongs to P set and 2 belongs to NP Complete set
- (C)** Both problems belong to P set
- (D)** Both problems belong to NP complete set

Q Let $A \leq_m B$ denotes that language A is mapping reducible (also known as many-to-one reducible) to language B. Which one of the following is FALSE? **(GATE-2014) (2 Marks)**

- (A) If $A \leq_m B$ and B is recursive then A is recursive.
- (B) If $A \leq_m B$ and A is undecidable then B is undecidable.
- (C) If $A \leq_m B$ and B is recursively enumerable then A is recursively enumerable.
- (D) If $A \leq_m B$ and B is not recursively enumerable then A is not recursively enumerable.

Q (GATE-2014) (2 Marks)

Consider the decision problem 2CNFSAT defined as follows:

{ ϕ | ϕ is a satisfiable propositional formula in CNF with at most two literals per clause }

For example, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_4)$ is a Boolean formula and it is in 2CNFSAT.

The decision problem 2CNFSAT is

- (A)** solvable in polynomial time by reduction to directed graph reachability
- (B)** solvable in constant time since any input instance is satisfiable
- (C)** solvable in constant time since any input instance is satisfiable.
- (D)** NP-hard, but not NP-complete

Q Assuming $P \neq NP$, which of the following is TRUE? (GATE-2012) (1 Marks)

a) $NP\text{-complete} = NP$

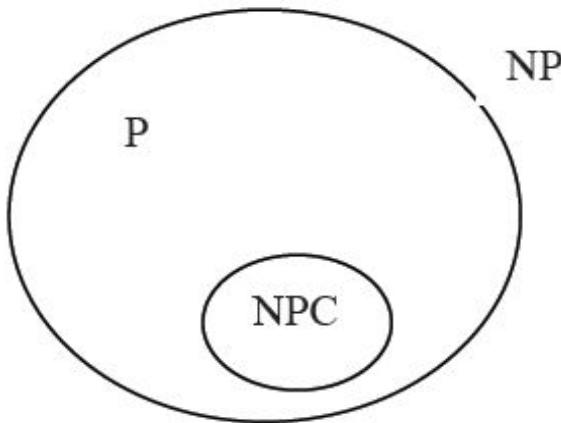
b) $NP\text{-complete} \cap P = \emptyset$

c) $NP\text{-hard} = NP$

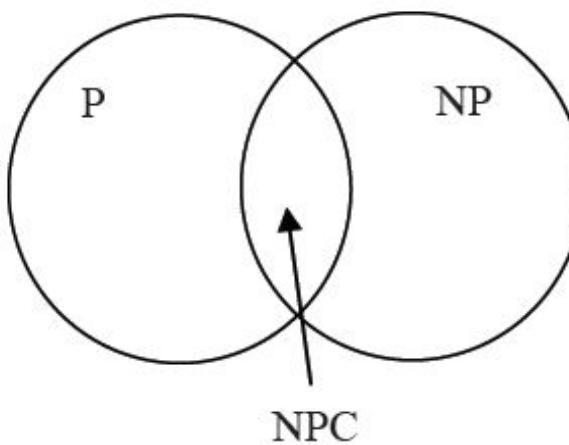
d) $P = NP\text{-complete}$

Q Suppose a polynomial time algorithm is discovered that correctly computes the largest clique in a given graph. In this scenario, which one of the following represents the correct Venn diagram of the complexity classes P, NP and NP Complete (NPC)? (GATE-2012) (1 Marks)

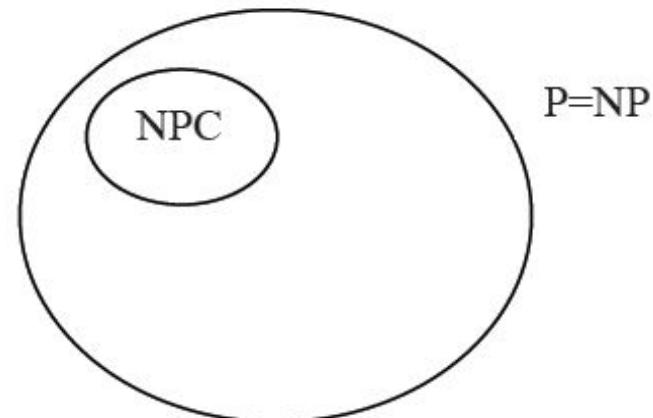
(A)



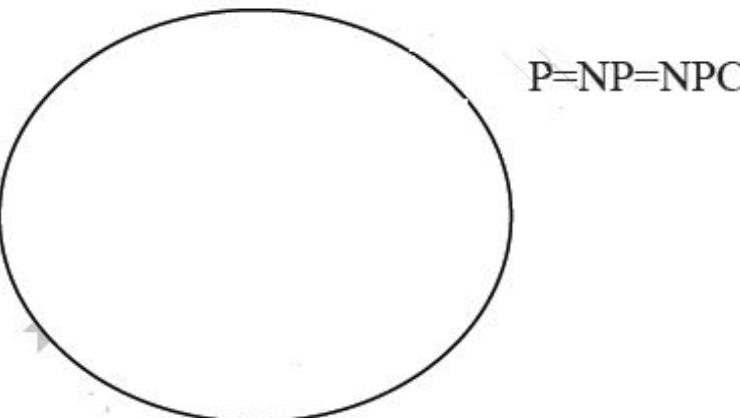
(B)



(C)



(D)



Q Let π_A be a problem that belongs to the class NP. Then which one of the following is TRUE? (GATE-2009) (1 Marks)

- a) There is no polynomial time algorithm for π_A
- b) If π_A can be solved deterministically in polynomial time, then P = NP
- c) If π_A is NP-hard, then it is NP-complete
- d) π_A may be undecidable

Q For problem X and Y, Y is NP – complete and X reduces to Y in polynomial time.

Which of the following is TRUE? (GATE – 2008) (2 Marks)

- a) IF X can be solved in polynomial, time then so can Y
- b) X is NP – complete
- c) X is NP – hard
- d) X is in NP – but not necessarily NP – complete

Q The subset-sum problem is defined as follows: Given a set S of n positive integers and a positive integer W , determine whether there is a subset of S whose elements sum to W . An algorithm Q solves this problem in $O(nW)$ time. Which of the following statements is false?
(GATE-2008) (2 Marks)

- a) Q solves the subset-sum problem in polynomial time when the input is encoded in unary
- b) Q solves the subset-sum problem in polynomial time when the input is encoded in binary
- c) The subset sum problem belongs to the class NP
- d) The subset sum problem is NP-hard

Q A problem in NP is NP – complete if [GATE - 2006] (2 Marks)

- a) It can be reduced to the 3 – SAT problem polynomial time**

- b) The 3-SAT problems can be reduced to it in polynomial time**

- c) It can reduce to any other problem in NP in polynomial time**

- d) Some problem in NP can be reduced to it in polynomial time**

Q. Let SHAM_3 be the problem of finding a Hamiltonian cycle in graph $G = (V, E)$ with $|V|$ divisible by 3 and DHAM_3 be the problem of determining if a Hamiltonian cycle exists in such graphs. Which one of the following is true? (GATE 2006, 1 Marks)

- a) Both DHAM_3 and SHAM_3 are NP - Hard
- b) SHAM_3 is NP-Hard but DHAM_3 is not
- c) DHAM_3 is NP-Hard ut SHAM_3 is not
- d) Neither DHAM_3 nor SHAM_3 is NP - Hard

Q Let S be an NP – complete problem Q and R be two other problems not known to be in NP. Q is polynomial – time reducible to S and S is polynomial – time reducible to R. Which one of the following statements is true? **(GATE-2006) (2 Marks)**

a) R is NP – complete

b) R is NP – hard

c) Q is NP – complete

d) Q is NP- hard

Q Consider three decision problems P_1 , P_2 and P_3 . It is known that P_1 is decidable and P_2 is undecidable. Which one of the following is TRUE? (GATE-2005) (2 Marks)

- (A) P_3 is decidable if P_1 is reducible to P_3
- (B) P_3 is undecidable if P_3 is reducible to P_2
- (C) P_3 is undecidable if P_2 is reducible to P_3
- (D) P_3 is decidable if P_3 is reducible to P_2 's complement

Q. Consider the following two problems on undirected graphs: (GATE 2005, 1 Marks)

- α : Given $G(V,E)$ does G have an independent set of size $|V| - 4$?
 β : Given $G(V,E)$ does G have an independent set of size 5?

Which one of the following is true?

- a) α is in P and β NP - Complete
- b) α is NP Complete and β is in P
- c) Both α and β are NP - Complete
- d) Both α and β are in P

Q Ram and Shyam have been asked to show that a certain problem Π is NP-complete. Ram shows a polynomial time reduction from the 3-SAT problem to Π , and Shyam shows a polynomial time reduction from Π to 3-SAT. Which of the following can be inferred from these reductions? **(GATE-2003) (2 Marks)**

(A) Π is NP-hard but not NP-complete

(B) Π is in NP, but is not NP-complete

(C) Π is NP-complete

(D) Π is neither NP-hard, nor in NP