CS & IT ENGINEERING

THEORY OF COMPUTATION

Regular expressions

Lecture No.- 03

By- Venkat sir

Recap of Previous Lecture

ϵ-NFA = NFA = DFA

Topic ?????

NFA $\rightarrow$ DFA

ϵ-NFA $\rightarrow$ NFA

ϵ-NFA

$\downarrow$

NFA

$\downarrow$

DFA ✓

Topics to be Covered

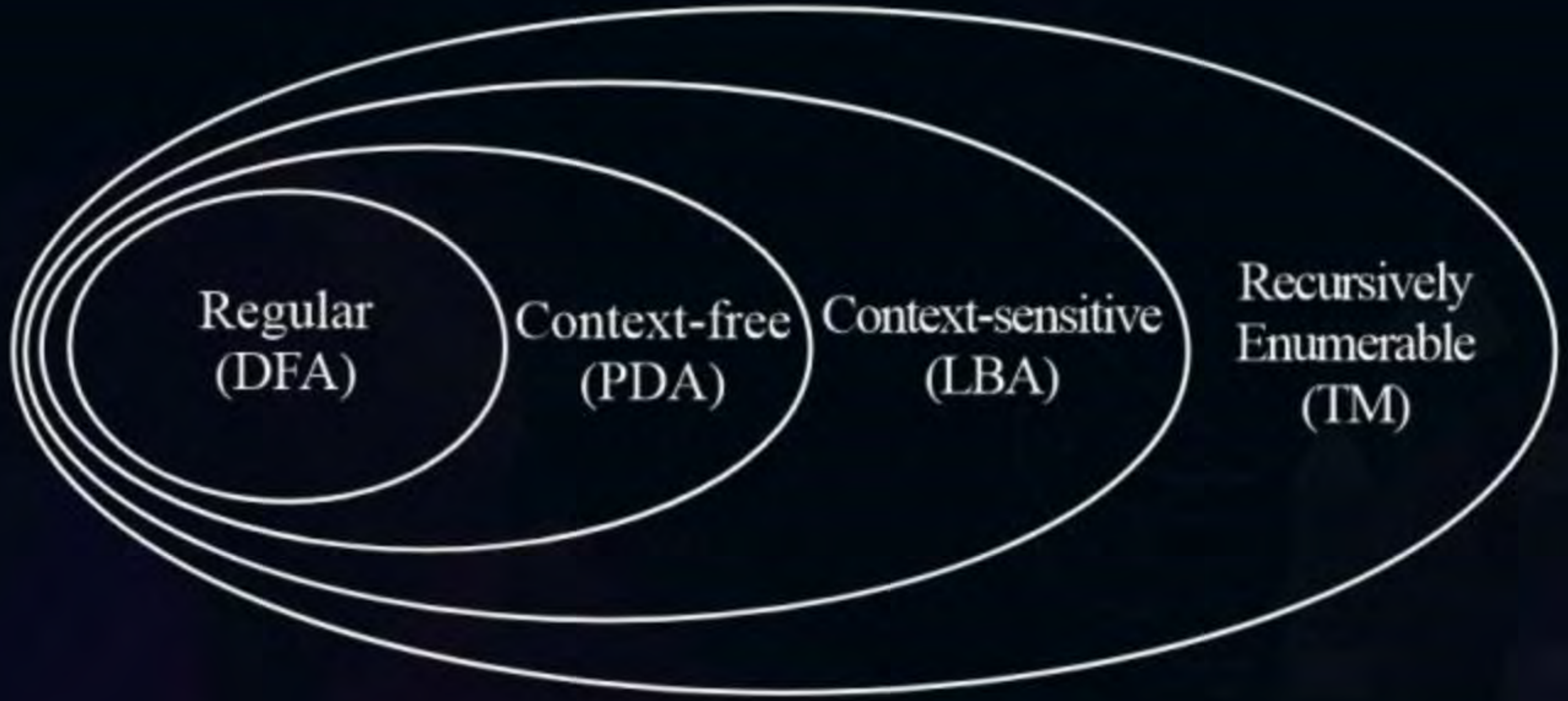
Topic Conversion from $\in$-NFA to NFA

Topic ??

Topic ??

Topic ??

Topic : Theory of Computation



Topic : ϵ -NFA

NOTE: Construction of $\in$ - NFA is easy than NFA

$$\boxed{\{Q, \Sigma, q_0, F, \delta\}}$$

Q	-	Finite number of states (set of state)
Σ	-	Input alphabet
q_0	-	initial state
F	-	Set of final states
δ	-	transition function

$$\delta: Q \times \Sigma \cup \{\in\} \rightarrow 2^Q$$

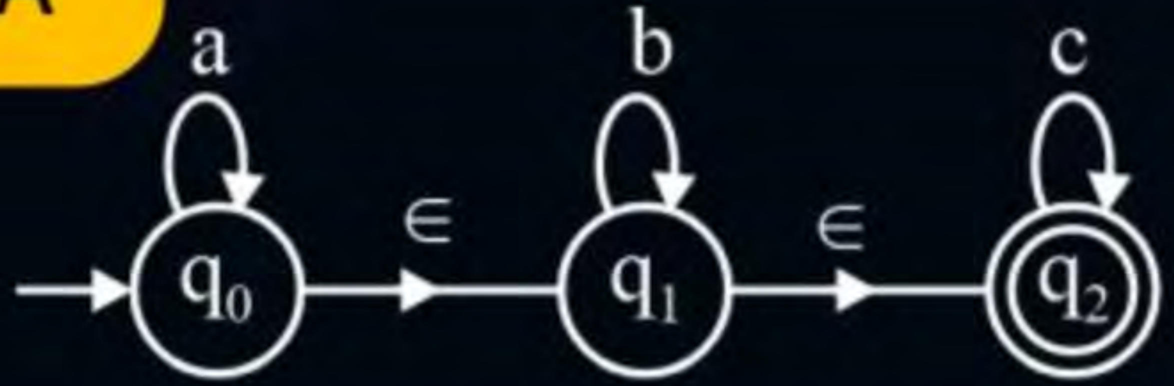
Topic : ϵ -NFA

$L = \{a^n b^m c^k / n, m, k \geq 0\}$ construct $\in$-NFA for L

$\in$NFA

Topic : ϵ -NFA

ϵ-NFA



Topic : ϵ -NFA

While converting $\in$-NFA into NFA (without $\in$) the following are the possibilities

→ No. of states are same ✓

→ Initial state is same ✓

→ Final state may changes ✓

→ Transitions may changes ✓

Topic : Conversion from $\in$-NFA to NFA

1. Number of states in $\in$-NFA is same of NFA

2. Initial state of $\in$- NFA is same as NFA

3. In NFA make states as final where $\in$-closure of that state contains a final state of $\in$-NFA.

Topic : Conversion from $\in$-NFA to NFA

Transitions of NFA is

$$\boxed{\delta^{1}(q_1, a) = \in\text{-closure}\,(\delta(\in\text{-closure}\,(q), a)}$$

Topic : Conversion from $\in$-NFA to NFA

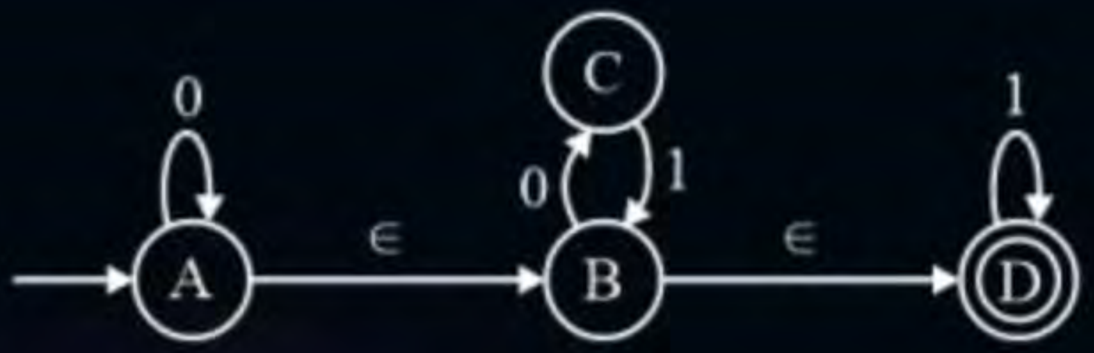
$\in$-closure (q) = set of all states which are reachable from state q by reading only $\in$.

#Q. Construct an equivalent NFA for the following Ɛ-NFA

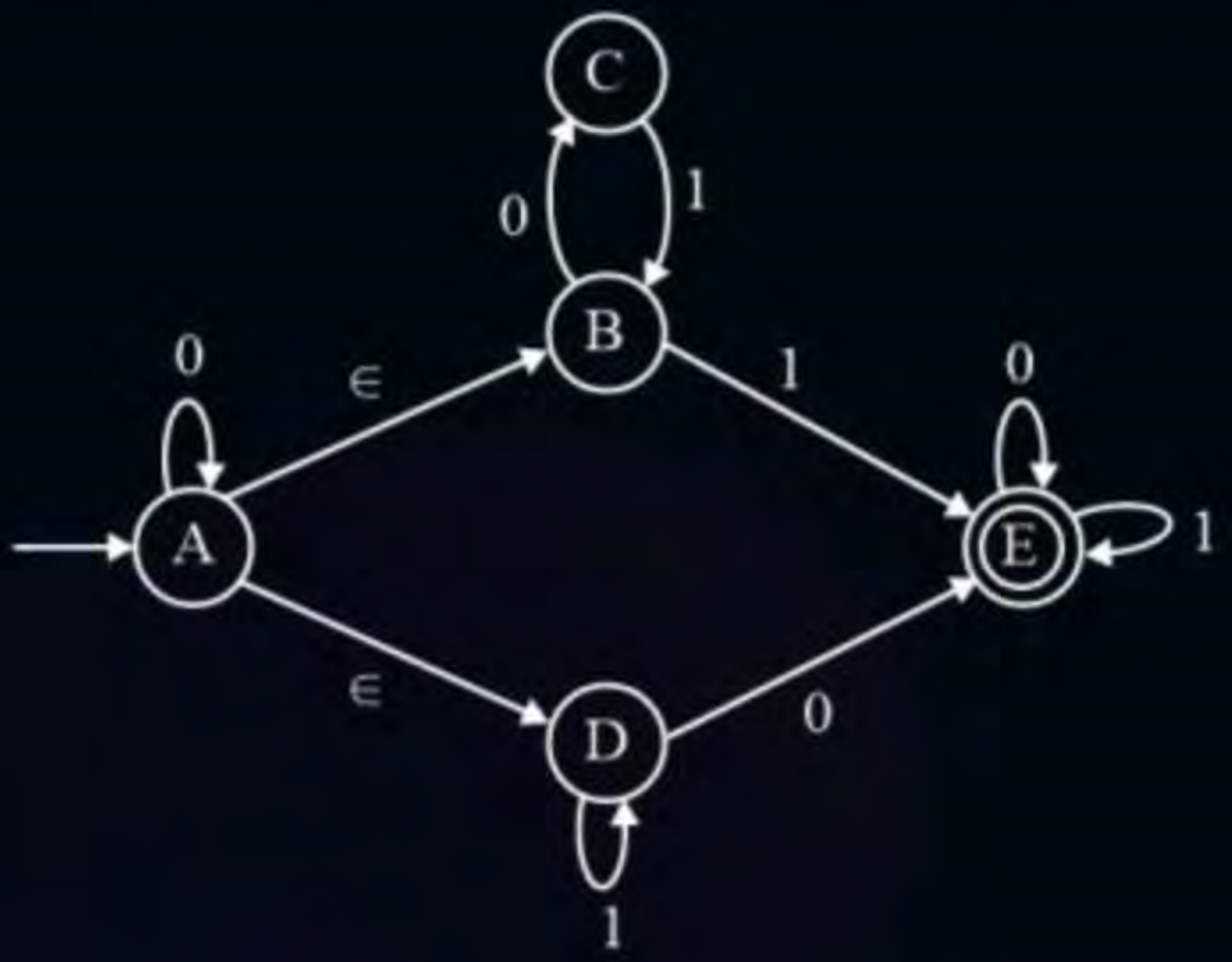
#Q. Construct an equivalent NFA for the following Ɛ-NFA

#Q. Construct an equivalent NFA for the following Ɛ-NFA

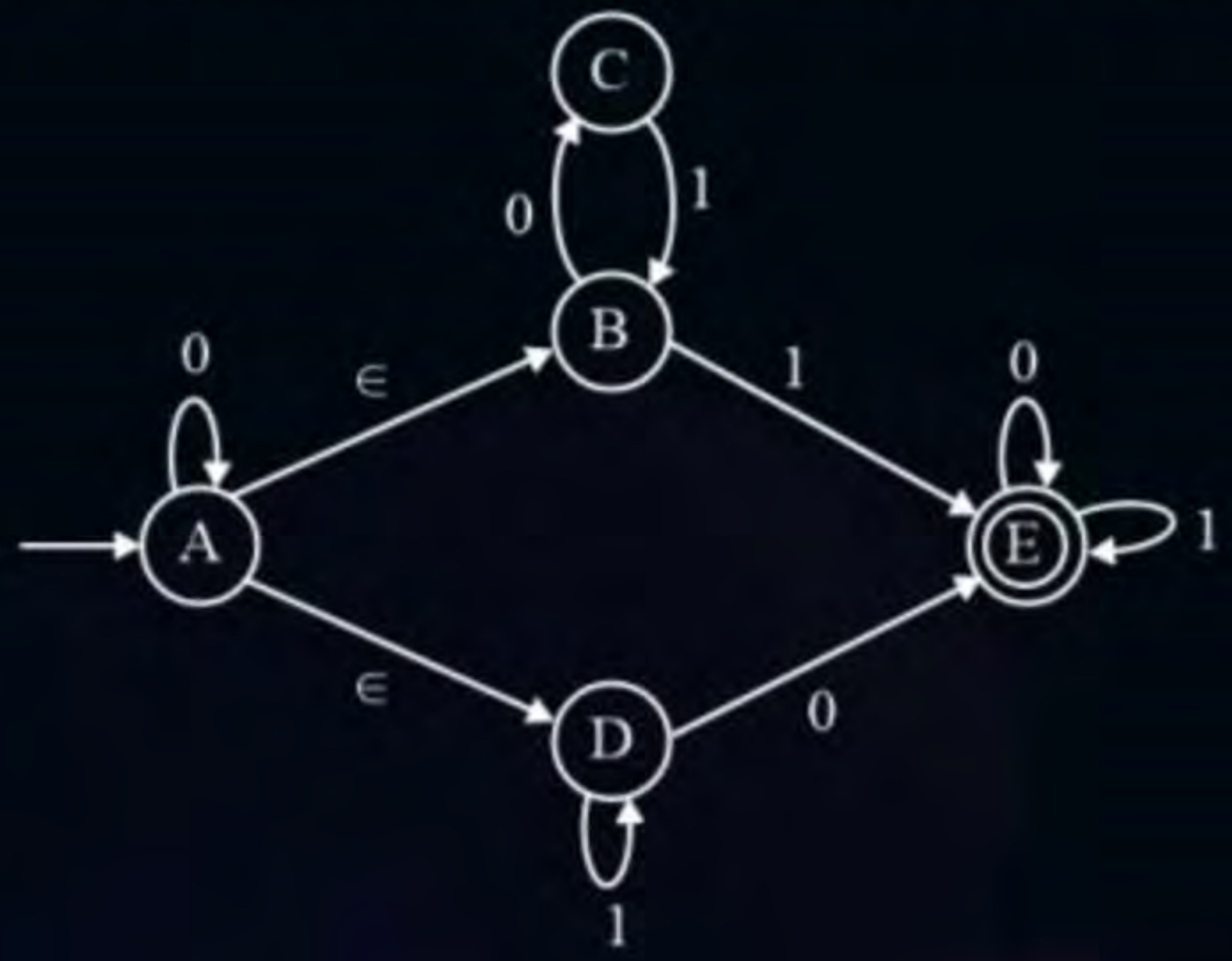
#Q. Construct an equivalent NFA for the following Ɛ-NFA



#Q. Construct an equivalent NFA for the following Ɛ-NFA



#Q. Construct an equivalent NFA for the following Ɛ-NFA



#Q. Construct an equivalent NFA for the following Ɛ-NFA

Topic : Conversion from $\in$-NFA to NFA

$\in\text{-closure}[\delta(q_0, a) \cup (q_2, a)] = \in\text{-closure}(q_1)$

$\in\text{-closure}[\delta(q_0, q_2), a] = \{q_0, q_1, q_2\}$

$\delta(q_2, a) = \in\text{-closure}(\delta(\in\text{-closure}(q_2), a)$

Transitions of NFA is

$$\boxed{\delta^1(q_1, a) = \in\text{-closure}(\delta(\in\text{-closure}(q), a)}$$

$\delta(q_1, a) = \in\text{-closure}(\delta(\in\text{-closure}(q_1)), a))$

$\in\text{-closure}[\delta(q_0, q_1, q_2), a)]$

Direct transitions

" $\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)$

$\in\text{-closure}(q_1) = \{q_0, q_1, q_2\}$

$\in\text{-closure}[(q_1 \cup \phi \cup \phi)] \rightarrow$

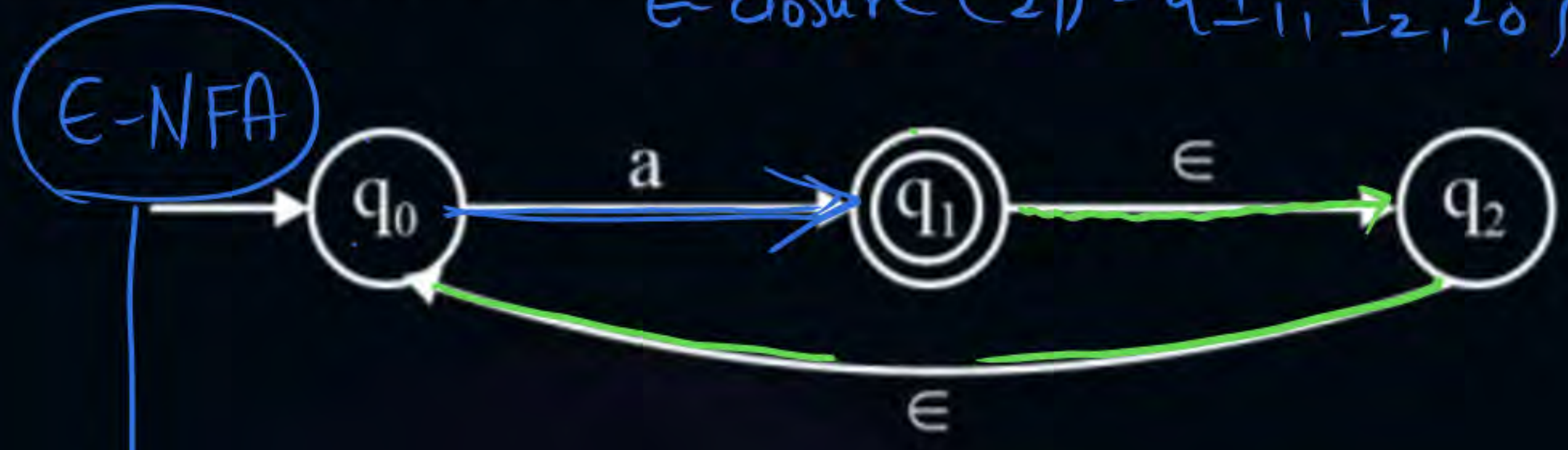
$$\boxed{\delta^1 (q_1, a) = \epsilon\text{-closure}\,(\delta(\epsilon\text{-closure}\,(q),a)}$$

$\delta(q_0, a) = \epsilon\text{-closure}(\delta(q_0, a)$

$\Sigma = \{a\}$

#Q. Construct an equivalent NFA for the following ε-NFA

$\epsilon\text{-closure}(q_1) = \{q_1, q_2, q_0\}$



$\epsilon\text{-closure}(q_0) = \{q_0\}$

$\epsilon\text{-closure}(q_1) = \{q_1, q_2, q_0\}$

$\epsilon\text{-closure}(q_2) = \{q_2, q_0\}$

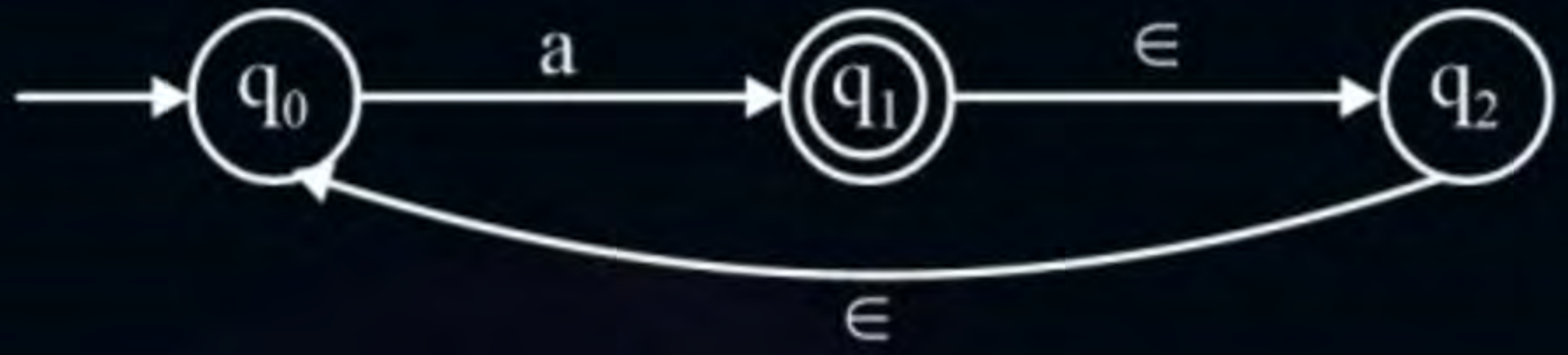
$\delta(q_0, a) = \{q_0, q_1, q_2\}$

$\delta(q_1, a) = \{q_0, q_1, q_2\}$

$\delta(q_2, a) = \{q_0, q_1, q_2\}$

$\delta^1(q_1, a) = \in\text{-closure}\,(\delta(\in\text{-closure}\,(q), a)$

#Q. Construct an equivalent NFA for the following Ɛ-NFA

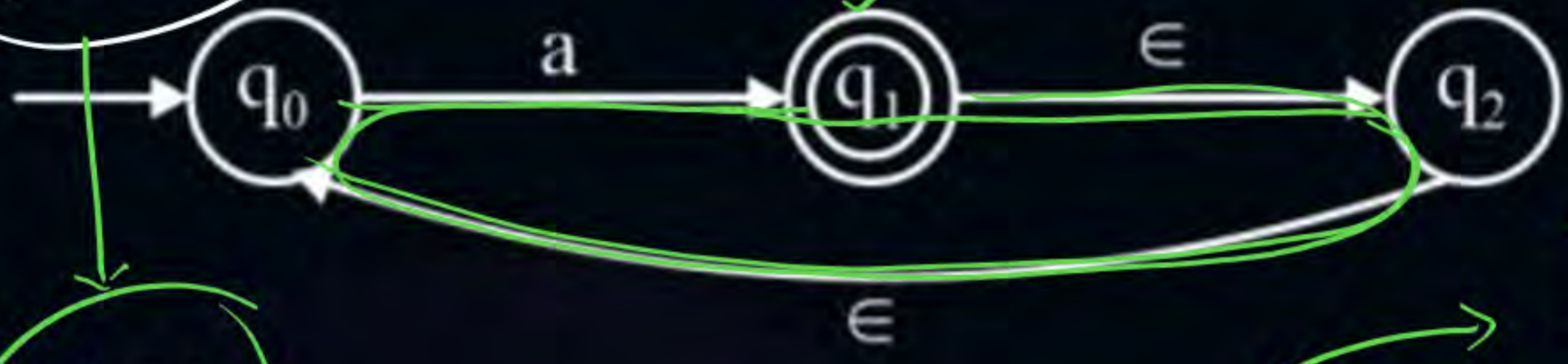


DFA

$\to q_0 \xrightarrow{a} q_0 q_1 q_2 \circlearrowleft a$

#Q. Construct an equivalent [redacted] for the following Ɛ-NFA

ϵ-NFA



NFA

	a
$\to q_0$	$\{q_0, q_1, q_2\}$
q_1	$\{q_0, q_1, q_2\}$
q_2	$\{q_0, q_1, q_2\}$

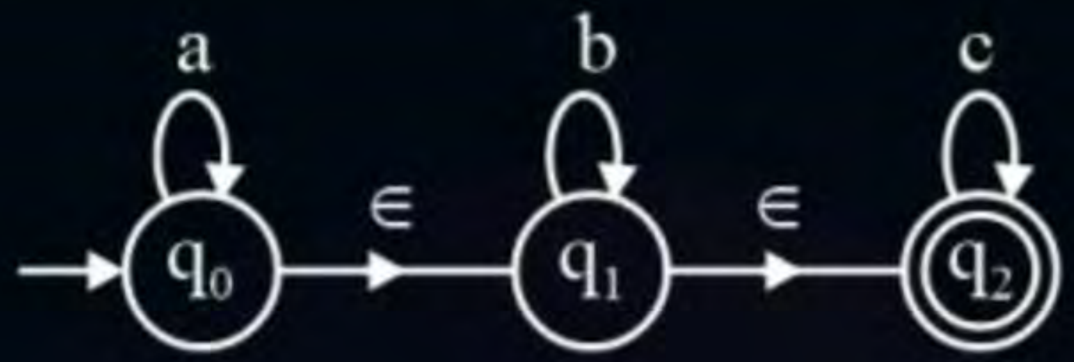
min DFA

	a
$\to q_0$	$[q_0 q_1 q_2]$
$[q_0 q_1 q_2]$	$[q_0 q_1 q_2]$

ϵ-NFA $\to$ NFA $\to$ DFA

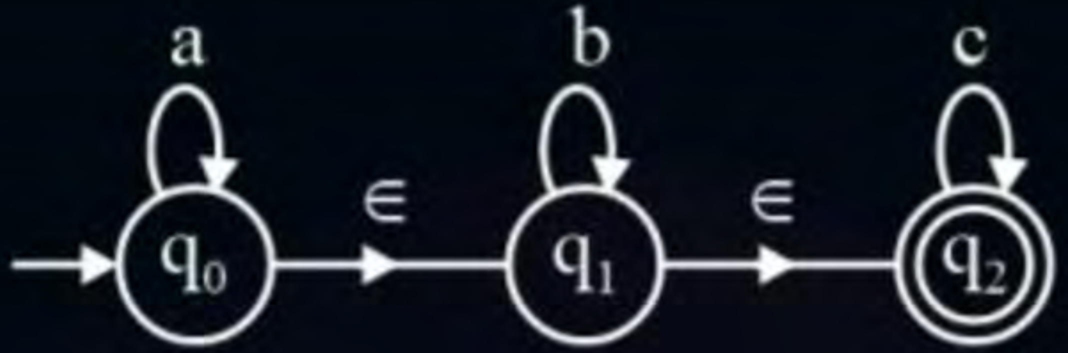
Topic : ϵ -NFA

Construct a DFA for the following $\in$-NFA



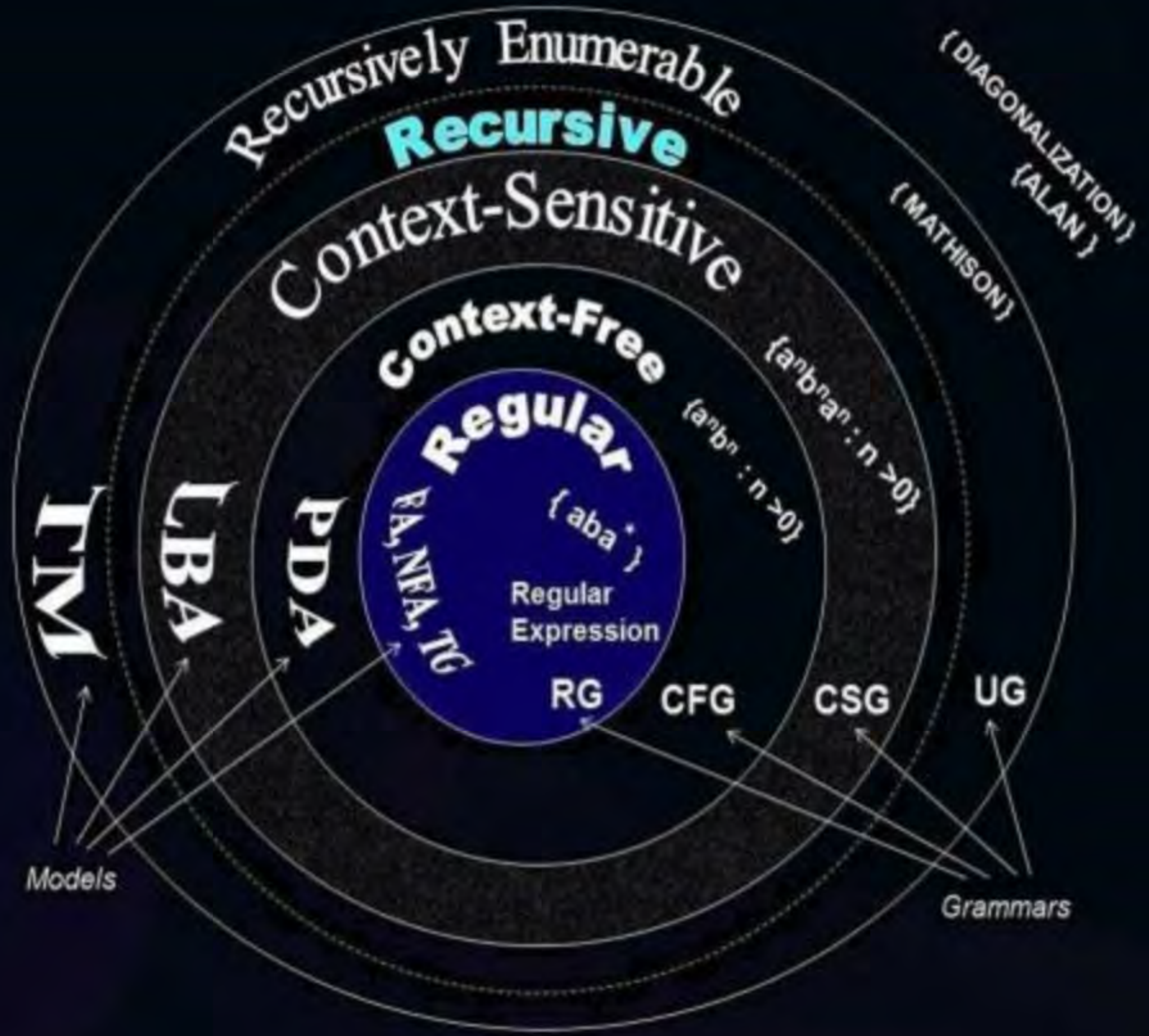
Topic : ϵ -NFA

Construct a DFA for the following $\in$-NFA



#Q. Construct an equivalent DFA for the following Ɛ-NFA

Topic : Theory of Computation



Topic : Expressive Power

Number of languages accepted by particular automata is knowns as expressive power.

(TM > LBA > PDA > FA)

1. Expressive power of NFA and DFA same. Hence every NFA is converted into DFA.
2. Expressive power of NPDA is more than DPDA. Hence conversion not possible
3. Expressive power of DTM and NTM is same.

MCQ

#Q. Let D_f, D_p are number of languages accepted by DFA and DPDA respectively. Let N_f, N_P are number of languages accepted NFA and NPDA respectively. Which of the following is true.

A $N_f = D_f$
$N_p = D_p$

B $N_f \supset D_f$
$N_p \supset D_p$

C $N_f = D_f$
$N_p \subset D_p$

D None

MCQ

#Q. In which of the cases stated below the following statement is false?
"Every nondeterministic machine M_1 there exists an equivalent deterministic machine M_2 recognizing the same language"

A M_1 is non deterministic FA

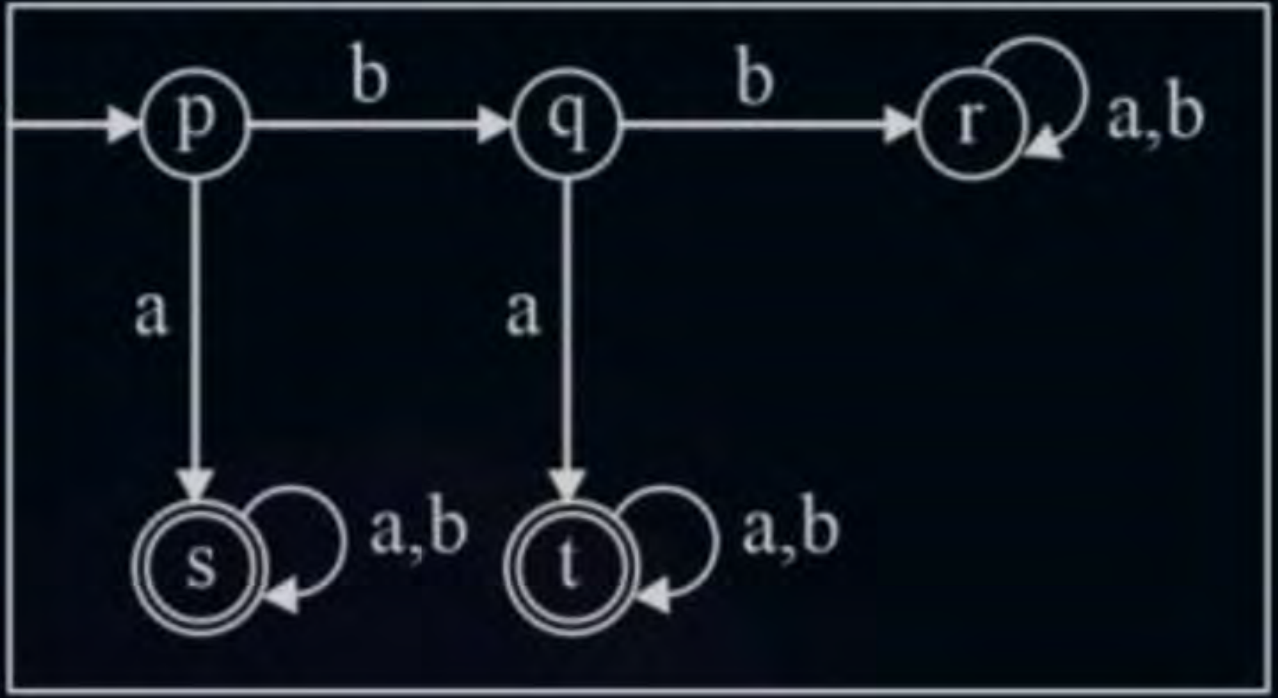
B M_1 is non deterministic turing machine

C M_1 Is non deterministic PDA

D None

Q

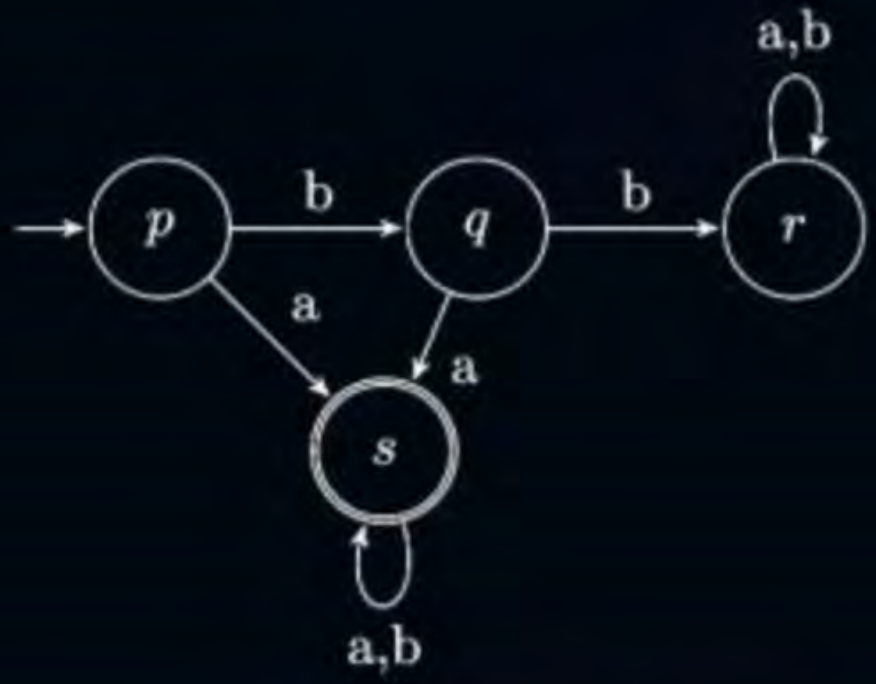
A deterministic finite automaton (DFA) D with alphabet $\Sigma = \{a, b\}$ is given below:



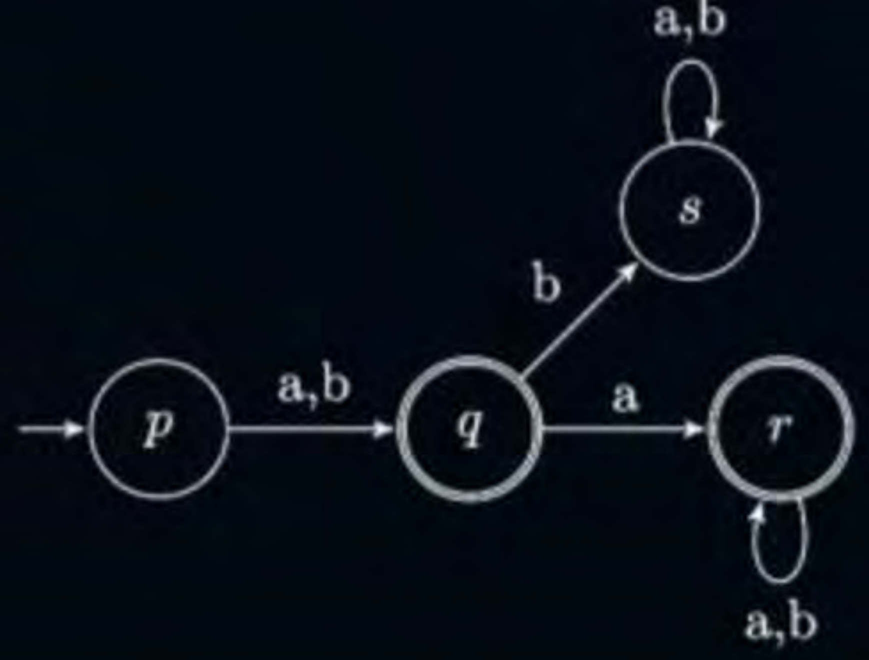
Which of the following finite state machines is a valid minimal DFA which accepts the same language as D?

[2011:2 Mark]

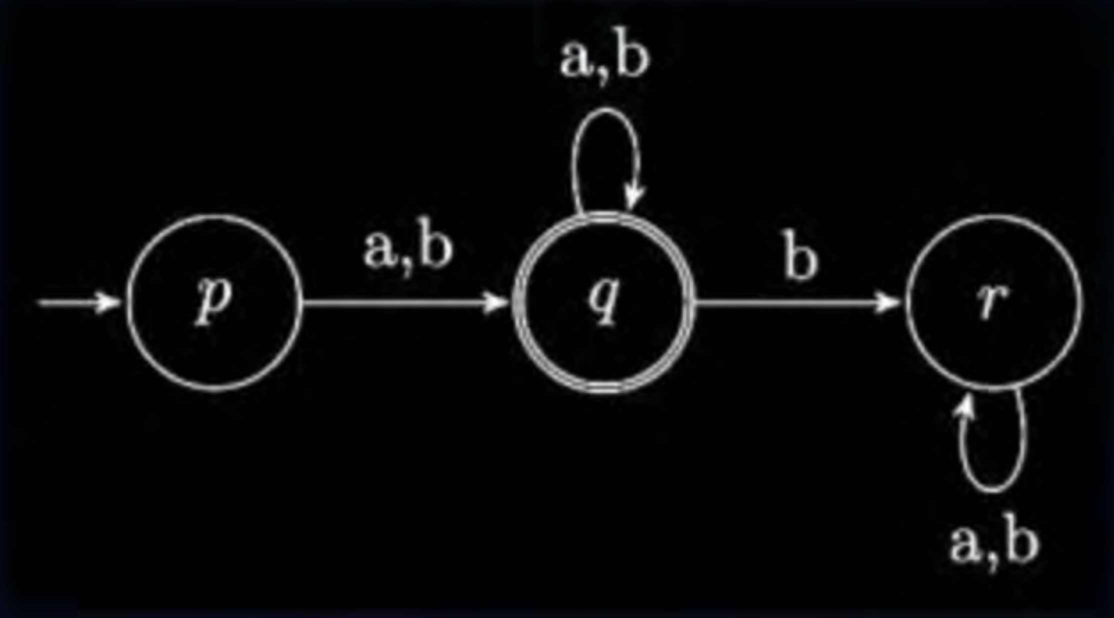
A



B



C

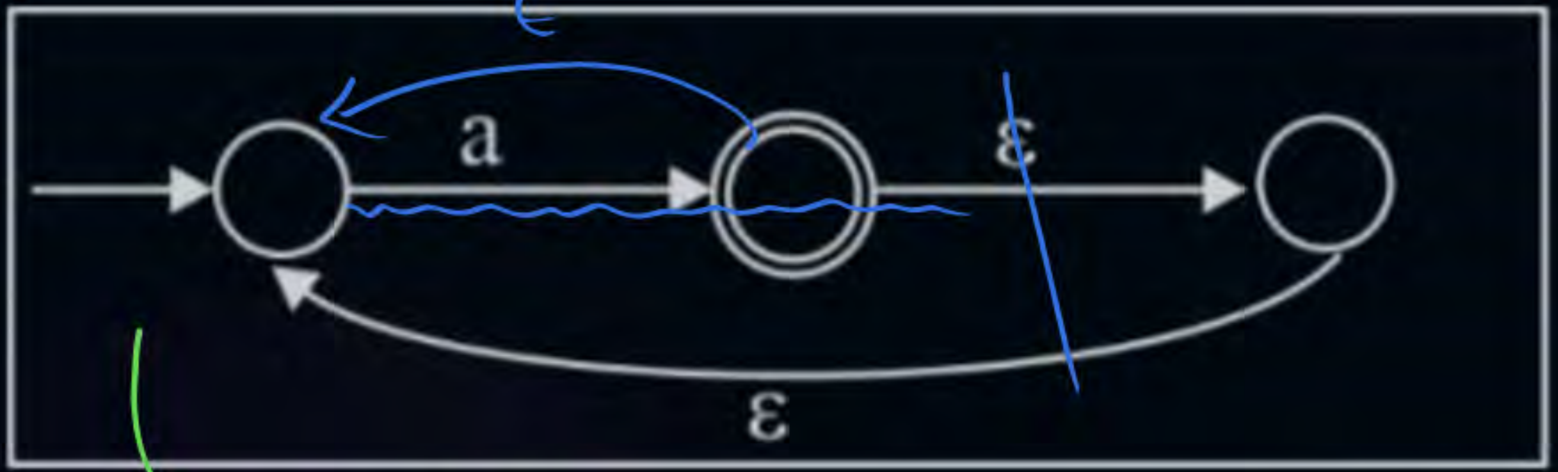


D

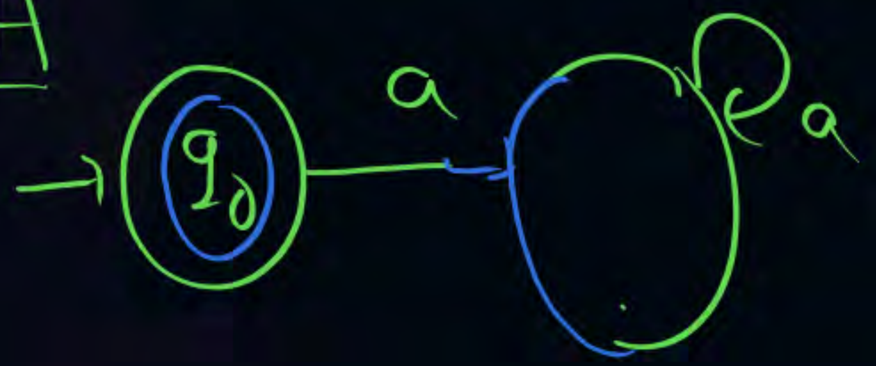
Q

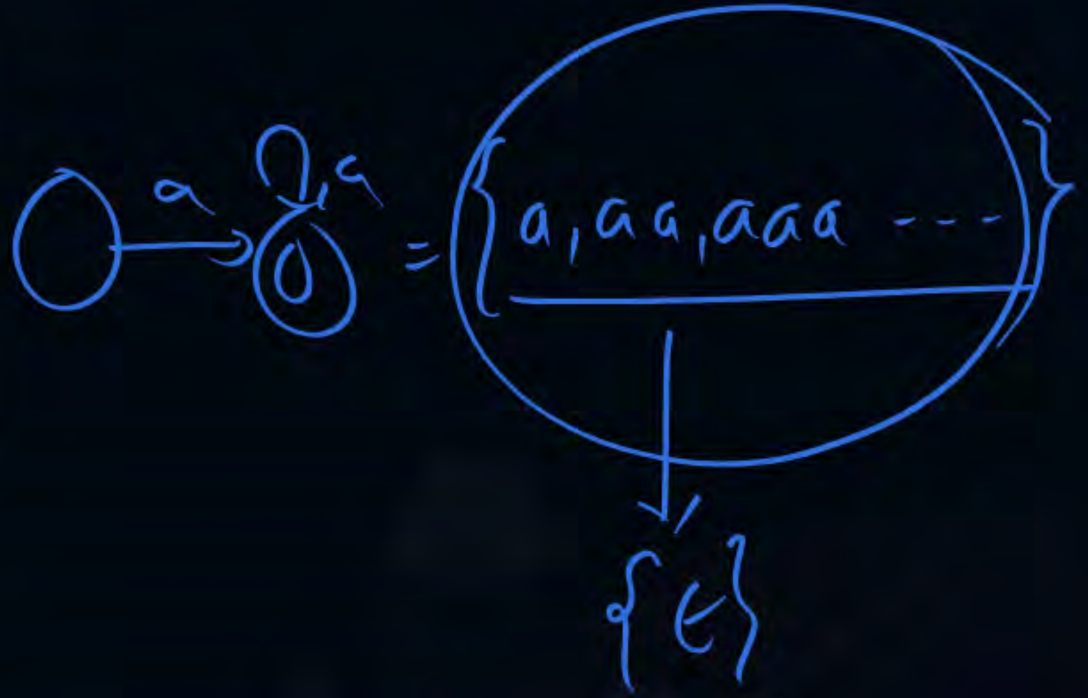
What is the complement of the language accepted by the NFA shown below? Assume $\Sigma = \{a\}$ and ε is the empty string.

[2012: 1 Mark]



NFA → DFA





A $\varnothing$

B $\{\varepsilon\}$

C a^*

D $\{a, \varepsilon\}$

DFA → Language Detection

DFA → minimization

DFA → Construction

$\uparrow$

NFA ✓

$\uparrow$

ϵ-NFA ✓

Topic : Regular Expression

- 1 The simplest way of representing a regular language is known as Regular expression.
- 2 For every regular language regular expression can be constructed.
- To construct regular expression following 3 operators are used.
- + is known as union operator ✓
- . is known as concatenation operator ✓
- * is known as Kleene closure operator ✓

$$\overset{(OR)}{\gamma_1 + \gamma_2} = \{\gamma_1, \gamma_2\}$$

$$\overset{(and)}{\underline{\gamma_1 \cdot \gamma_2}} = \underline{\gamma_1 \cdot \gamma_2}$$

<u>Repeat</u> $\gamma^* = \{\epsilon, \gamma, \gamma^2, \gamma^3, \gamma^4 ---\}$

positive closure $\gamma^+ = \{\gamma, \gamma^2, \gamma^3, \gamma^4 ---\}$

$$L = \{a^n b^n \mid n \geq 0\}$$

$\downarrow$

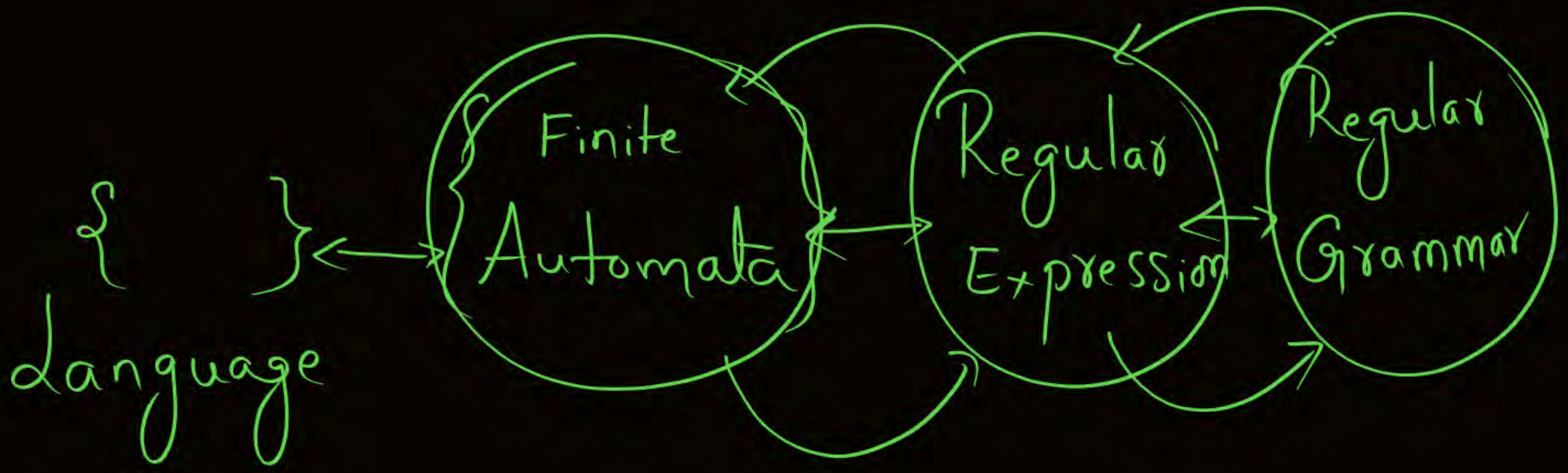
not possible

Complete Language

$$L = \{\epsilon, a, b, aa, ab, ba, bb \ldots\}$$

$$\Downarrow$$

$$\Sigma^*$$


{ }
Language
Finite
Automata
Regular
Expression
Regular
Grammar

Topic : NOTE

- For one regular language many number of regular expressions can be possible.
- One regular expression can generate only one regular language.

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

$L_1 = \{ \ \} \longrightarrow \phi$

$L_2 = \{ \epsilon \} \longrightarrow \epsilon$

$L_3 = \{ a \} \longrightarrow a$

$L_4 = \{ a, b \} \longrightarrow a + b$

$L_5 = \{ \epsilon, a, aa, aaa \dots \} \longrightarrow a^*$

$L_6 = \{ a, aa, aaa \dots \} \longrightarrow a^+$ positive closure.

$$L_1 = \{ a^{2n} \mid n \geq 0 \} = \{ \epsilon, a^2, a^4, a^6 --- \} = (aa)^* \qquad (a \cdot a)^*$$

$$L_2 = \{ a^n b^m \mid n, m \geq 0 \} = a^* b^*$$

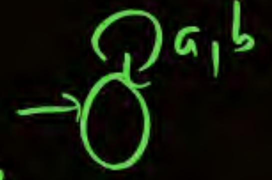
$$L_3 = \left\{ a^n b^m \,\middle|\, \begin{matrix} n \geq 0 \\ m \geq 1 \end{matrix} \right\} = a^* b^+$$

$$L_4 = \left\{ a^n b^m \,\middle|\, \begin{matrix} n \geq 2 \\ M \geq 3 \end{matrix} \right\} = aa\,a^*\,bbbb^* = aa^+ bbb^+$$

$$L_5 = \{ a^n b^n \mid n \geq 1 \} = \text{not Regular} = \left\{ \begin{matrix} \text{Regular Expression} \\ \text{not possible.} \end{matrix} \right\}$$

× $L_1 = \{a^n b^m \mid n > m\} \xrightarrow{\times}$ Not Regular → R.E not possible.

× $L_2 = \{a^n b^m \mid n > m \text{ (or) } n < m\} \Rightarrow \{a^n b^m \mid n \neq m\}$ = not possible



✓ $L_3 = \{a^n b^m \mid n > m \text{ and } n < m\}$ → possible $\Rightarrow \{\ \} = \phi$

× $L_4 = \{a^n b^m \mid n \geq m \text{ and } n \leq m\} \Rightarrow \{\epsilon, ab, a^2b^2, a^3b^3 \ldots\} = \{a^n b^n\}$

not possible

$L_5 = \{a^n b^m c^k \mid n, m, k \geq 0\} = a^* b^* c^*$

$a^* \; b^* \; c^*$

a^{10} 0

$a^* b^*$
$a \; bb$

$L_1 = \{ a^n b^m \mid (n+m) \text{ is even} \} = \text{possible}$

even + even

odd + odd

even

$$\boxed{(aa)^* (bb)^* + a(aa)^* b(bb)^*}$$

even even odd odd

$L_2 = \{ a^n b^m \mid (n+m) \text{ is odd} \}$

$$\boxed{(aa)^* b(bb)^* + a(aa)^* (bb)^*}$$

even odd + odd even

#Q. Construct regular expression that generates set of all even length palindrome strings over {a}.

madam

nitin

naman

$$\{\epsilon, a^2, a^4, a^6, a^8 \ldots\} = (aa)^*$$

#Q. Construct regular expression that generates set of all odd length palindrome strings over {a}.

$$\{a, a^3, a^5, a^7 \text{---}\} = a(aa)^*$$

#Q. Construct regular expression that generates set of all even length palindrome strings over {a, b}.

$L = \{\epsilon, aa, bb, abba, baab \ldots\}$

(a) $(aa)^* + (bb)^*$

(b) $a(aa)^* b(bb)^*$

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

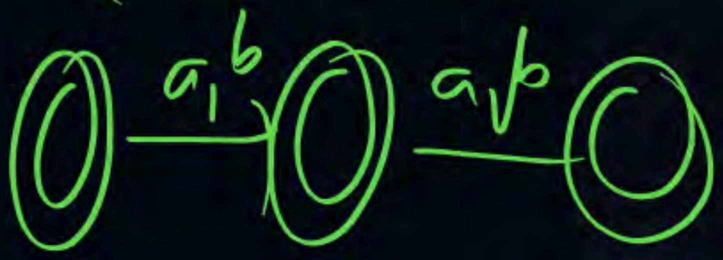
(d) none

not possible

#Q. Construct regular expression that generates set of all odd length palindrome strings over {a, b}.

not possible

$(a+b+\epsilon)(a+b+\epsilon)$



#Q. Construct regular expression that generates set of all odd length palindrome strings of English language.

$$\left\{ \begin{array}{l} \text{mom} \\ \text{dad} \\ \text{madam} \\ \vdots \end{array} \right\}$$

(a) possible

~~(b)~~ not possible

Topic : NOTE

- Palindrome languages over more than one symbol are not regular .Hence regular expression not possible.
- Palindrome languages over one symbol are regular.

2 mins Summary

Topic One

Topic Two

Topic Three

Topic Four

Topic Five

Conversion from ϵ-NFA to NFA

THANK - YOU