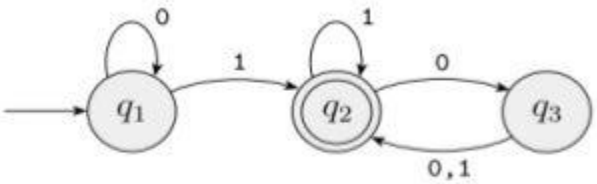
NFA

CSE 2233

DFA vs NFA

DFA:

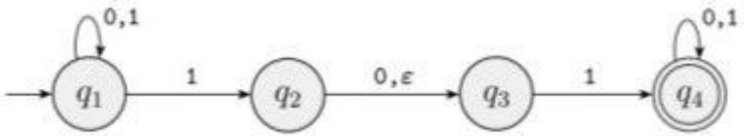
Every state of a DFA always has exactly one exiting transition arrow for each symbol in the alphabet.



In a DFA, labels on the transition arrows are symbols from the alphabet set Σ

NFA:

In an NFA, a state may have zero, one or more exiting arrows for each alphabet symbol.

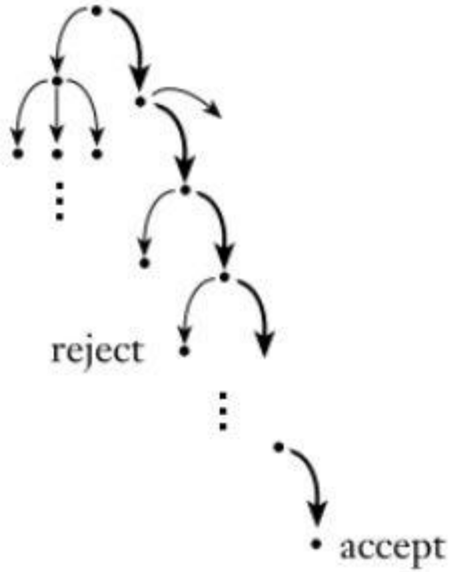


In NFA, labels on the transition arrows are symbols from the alphabet or ε. Zero, one or many arrows may exit from each state with the label ε.

DFA vs NFA

Deterministic computation

Nondeterministic computation



Why NFA ??

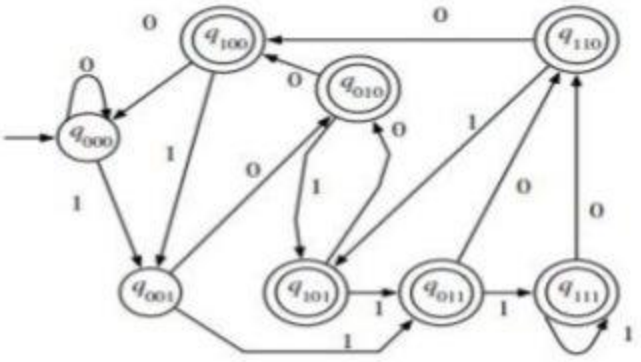
DFA:

1. **Faster** : It follows only one path.
2. **Complex** : More no of states and transitions.

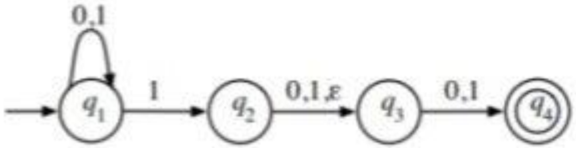
NFA:

1. **Slower** : It chooses between many paths.
2. **Simple** : Easy to express and join multiple machines.

Ex: Language that accepts all strings over {0,1} that contain a 1 either at the third position from the end or at the second position from the end



DFA



NFA

DFA vs NFA

DFA:

A finite automaton is a 5 tuple (Q, Σ, δ, q_0, F) where,

- Q is a finite set called the states
- Σ is a finite set called the alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state and
- $F \subseteq Q$ is the set of accept states

NFA:

A nondeterministic finite automaton is a 5-tuple (Q, Σ, δ, q_0, F), where

- Q is a finite set of states
- Σ is a finite alphabet set
- $\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)$ is the transition function
- $q_0 \in Q$ is the start state and
- $F \subseteq Q$ is the set of accept states

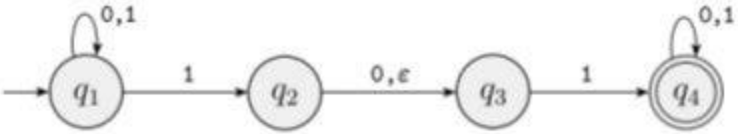
- Every Non Deterministic Finite automaton has an equivalent deterministic finite automaton
- Every DFA is by default NFA

NFA Machine, N_1 – Formal Definition

NFA :

A nondeterministic finite automaton is a 5-tuple (Q, Σ , δ, q_0, F), where

- Q is a finite set of states
- Σ is a finite alphabet set
- $\delta : Q \times \Sigma_\varepsilon \rightarrow P(Q)$ is the transition function
- $q_0 \in Q$ is the start state and
- $F \subseteq Q$ is the set of accept states



The formal description of N_1 is $(Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3, q_4\}$,

2. $\Sigma = \{0,1\}$,

3. δ is given as

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$,

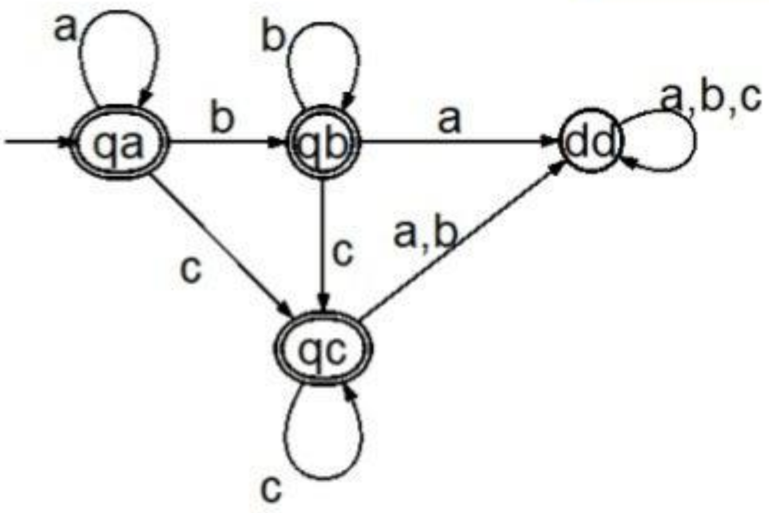
4. q_1 is the start state, and

5. $F = \{q_4\}$.

How does a DFA compute?

Steps:

- DFA will start from the start state, q_a
- It will take input one symbols from the input string from left to right consecutively and will traverse to the next states accordingly.
- After reaching the last state when no other input symbols left, if the last state is an accept state then this machine will accept that string otherwise it can't accept/reject that string.

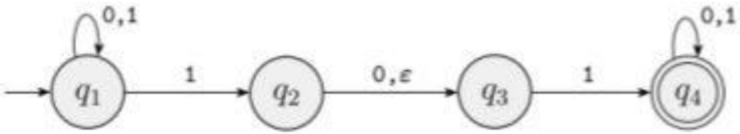


For input string "**aabcc**", the sequence of states the machine will visit are:

$$\rightarrow q_a \xrightarrow{a} q_a \xrightarrow{a} q_a \xrightarrow{b} q_b \xrightarrow{c} q_c \xrightarrow{c} q_c$$

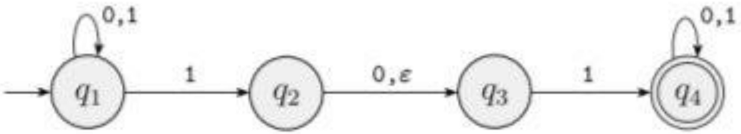
As the final state q_c is an accept state, so this string will be accepted.

How does an NFA compute?



- Suppose we are in state q_1 in NFA and that the next input symbol is a 1. After reading that symbol, the machine splits into multiple copies of itself and follows all the possibilities in parallel.
- Each copy of the machine takes one of the possible ways to proceed and continues as before. If there are subsequent choices, the machine splits again.
- If the next input symbol doesn't appear on any of the arrows exiting the state occupied by a copy of the machine, that copy of the machine dies, along with the branch of computation associated with it.
- Finally, if any one of these copies of the machine is in an accept state at the end of the input, the NFA accepts the input string.

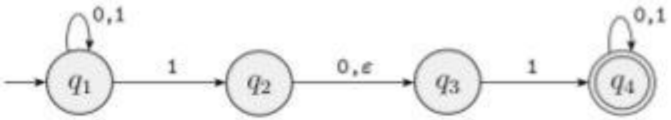
How does an NFA compute?

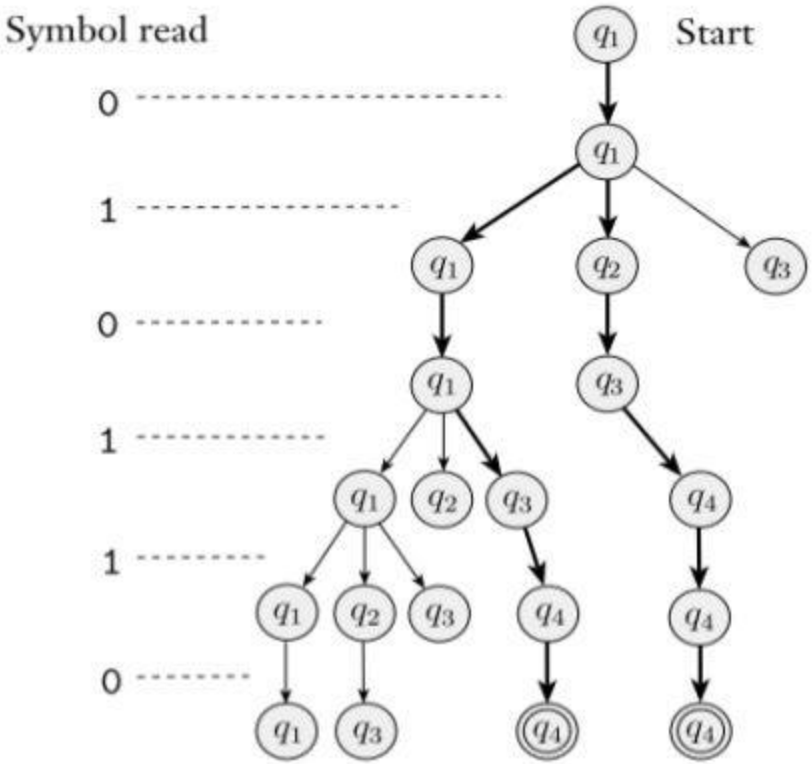


- If a state with an ε symbol on an exiting arrow is encountered, something similar happens. Without reading any input, the machine splits into multiple copies, one following each of the exiting ε –labeled arrows and one staying at the current state.
- Then the machine proceeds nondeterministically as before.

How does an NFA compute?

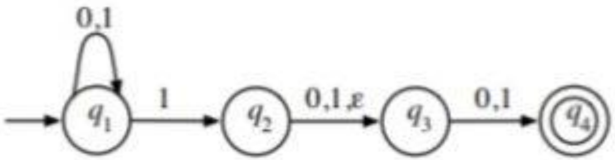
Let's compute the following NFA machine for input string ***010110***





How does an NFA compute?

- Let's compute the following NFA machine for input string ***010110***



NFA – Design

Draw the state diagram of NFA machines that can recognize the following languages:

- L(M) = { w | w begins with *101* } over ∑={0,1}
- L(M) = { w | w begins with *abb* } over ∑={a,b}

- L(M) = { w | w ends with *101* } over ∑={0,1}
- L(M) = { w | w ends with *aa* } over ∑={a,b}

- L(M) = { w | w contains *110* as substring } over ∑={0,1}
- L(M) = { w | w contains *abb* as substring } over ∑={a,b}

- L(M) = { w | w is exactly 101 }

- L(M) = { w | w contains a 1 in the 3rd position from the end } over ∑={0,1}

Regular Operations on NFA - Union

UNION:

The class of regular languages is closed under the union operation

That means,
if A_1 and A_2 are regular languages, then $A_1 \cup A_2$ is also a regular language.

Here, $A_1 \cup A_2 = \{ x \mid x \in A_1 \text{ or } x \in A_2 \}$

Example:

$A_1 = \{ \text{good, bad} \}$ and $A_2 = \{ \text{boy, girl} \}$

Then,

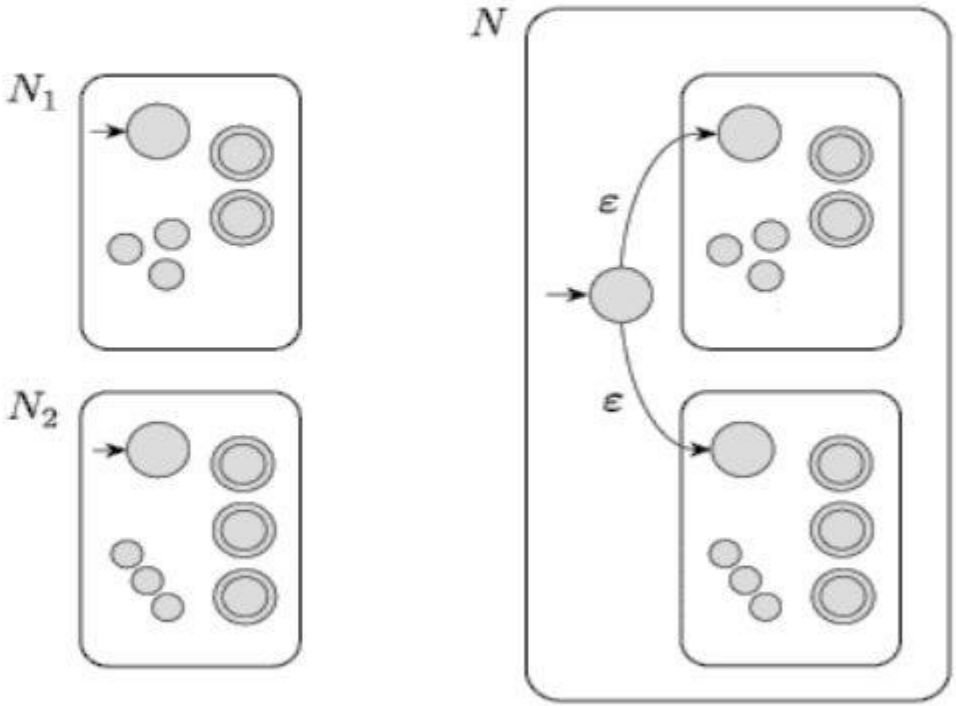
$A_1 \cup A_2 = \{ \text{good, bad, boy, girl} \}$

In this case, if N_1 and N_2 represent the NFAs to recognize A_1 and A_2 then we need to build a machine N from N_1 and N_2 so that N can also recognize $A_1 \cup A_2$

Regular Operations on NFA - Union

UNION of two NFAs

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$



Regular Operations on NFA - Concatenation

Concatenation:

The class of regular languages is closed under the concatenation operation

That means,
if A_1 and A_2 are regular languages, then $A_1 \circ A_2$ is also a regular language.

Here, $A_1 \circ A_2 = \{ xy \mid x \in A_1 \text{ and } y \in A_2 \}$

Example:

$A_1 = \{ \text{good, bad} \}$ and $A_2 = \{ \text{boy, girl} \}$

Then,

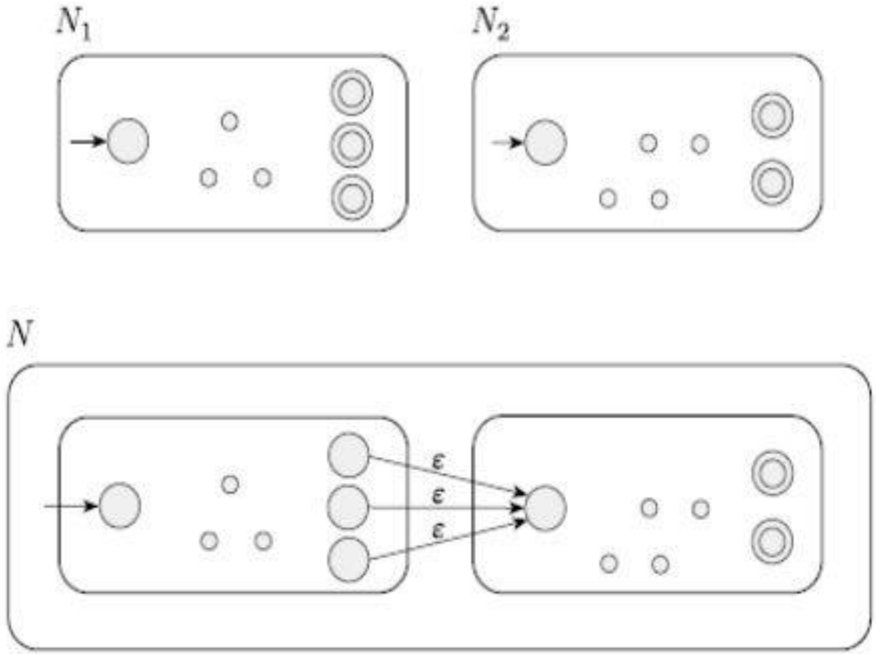
$A_1 \circ A_2 = \{ \text{goodboy, goodgirl, badboy, badgirl} \}$

In this case, if N_1 and N_2 represent the NFAs to recognize A_1 and A_2 then we need to build a machine N from N_1 and N_2 so that N can also recognize $A_1 \circ A_2$

Regular Operations on NFA - Concatenation

Concatenation of two NFAs

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2. \end{cases}$$



Regular Operations on NFA - Star

Star:

The class of regular languages is closed under the star operation

That means,
if A_1 is a regular language, then A_1^* is also a regular language.

Here, $A_1^* = \{ x_1x_2x_3 \dots \dots x_k \mid k>=0 \text{ and each } x_i \in A_1 \}$

Example:

$A_1 = \{ \text{good, bad} \}$

Then,

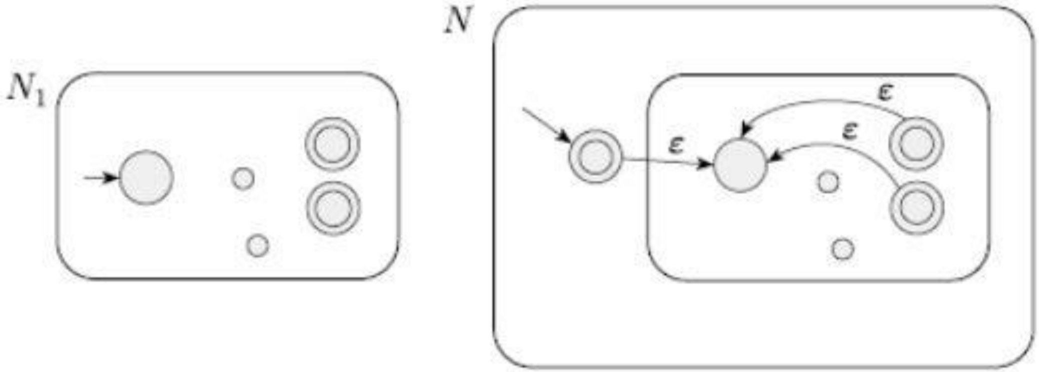
$A_1^* = \{ \varepsilon$, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad, … … }

In this case, if N_1 represents the NFAs to recognize A_1 then we need to build a machine N from N_1 so that N can also recognize A_1^*

Regular Operations on NFA - Star

Star on NFA

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$



NFA – Design

Draw the state diagram of NFA machines that can recognize the following languages:

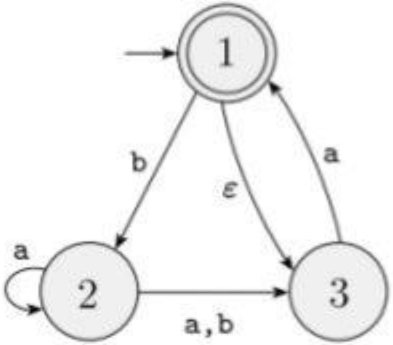
- Union: A be the language consisting of all strings of the form 0^k over {0} where k is a multiple of 2 or 3
- Union: All strings beginning with 101 or with 110
- Concatenation: All strings beginning with 101 and ending with 101
- Star: All strings consisting of 0 or more repetitions of 101
- Plus: All strings consisting of 1 or more repetitions of 101
- Complement: All strings that doesn't contain substring 101
- Concat + Complement: All strings with exactly 1 occurrence of 101

Equivalence of NFAs and DFAs

- Two machines are equivalent if they recognize the same language.
- Deterministic and Nondeterministic finite automata recognize the same class of languages.
- Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

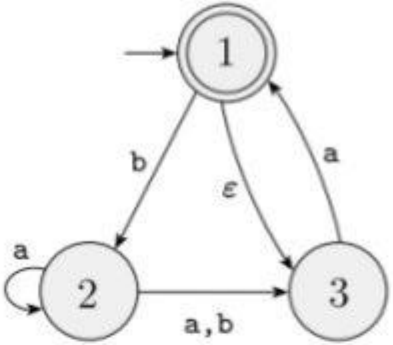
Converting NFA into equivalent DFA

- Let N = (Q, Σ, δ, q_0, F) be the NFA recognizing some language A
- We need to construct a DFA machine M = (Q', Σ, δ', q_0', F') recognizing A



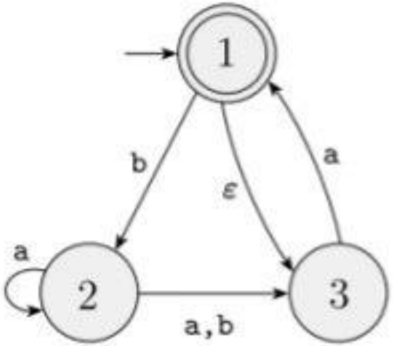
Step 1 : $M = (Q', \Sigma, \delta', q_0', F')$

- $Q' = P(Q)$ = set of all subsets of $Q = \{ \emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\} \}$



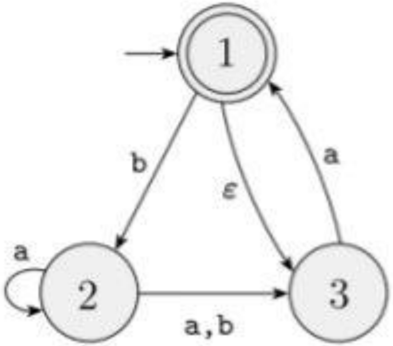
Step 2 : M = (Q', Σ , δ', q_0', F')

- For R ⊆ Q,
  ε - closure of R = E(R) = { q | q can be reached from members of R by traveling along 0 or more ε arrows }
- For this case, $q_0' = E(q_0) = E(\{1\}) = \varepsilon$ - closure of {1} = { 1, 3 } is our start state.



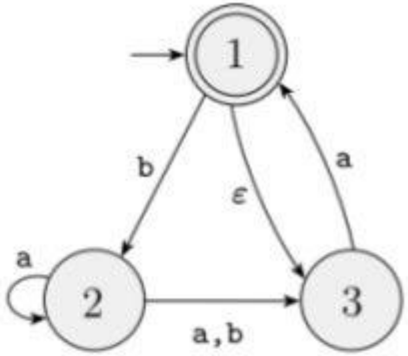
Step 3 : M = (Q', Σ , δ', q_0', F')

- F' = { R ∈ Q' | R contains an accept state of N }
  i.e. the machine M accepts if one of the possible states that N could be in at this point is an accept state.
- For this case, F' = { {1}, {1,2}, {1,3}, {1,2,3} }



Step 4 : M = (Q', Σ , δ', q_0', F')

- For R ⊆ Q,
  ε - closure of R = E(R) = { q | q can be reached from members of R by traveling along 0 or more ε arrows }



- For input symbol a, the transition function can be defined as,
  δ'(R, a) = { q ∈ Q | q ∈ E(δ(R, a)) for some r ∈ R } = $\bigcup_{r \in R} E(\delta(r, a))$
- In our case for example,

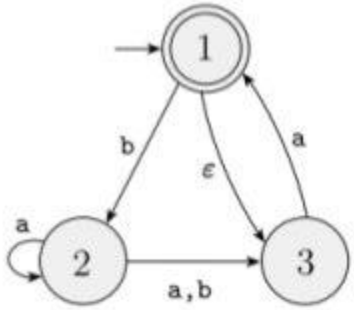
δ'(R, a) = δ'({ 2, 3 } , a) = $\mathrm{E}(\delta(\{2\}, a))$ U $\mathrm{E}(\delta(\{3\}, a))$ = E({2, 3}) U E({1}) = {2, 3} U { 1, 3 } = { 1, 2, 3 }

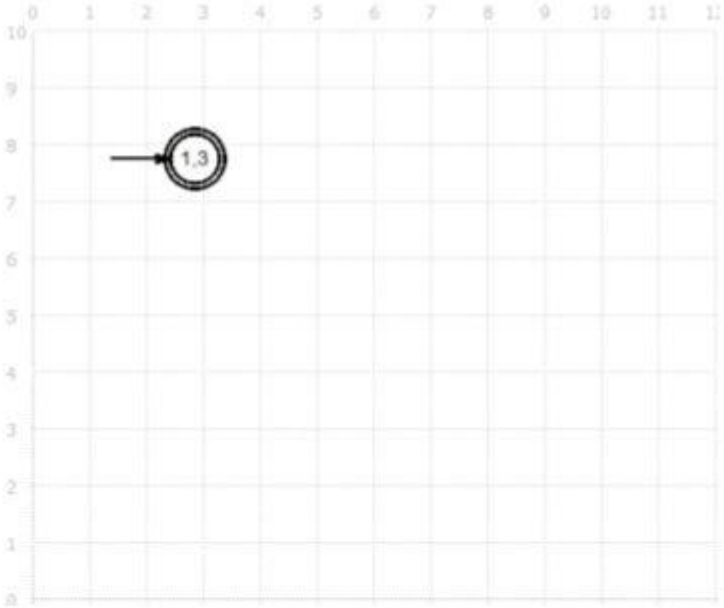
δ'(R, a) = δ'({ 1, 2 } , a) = $\mathrm{E}(\delta(\{1\}, a))$ U $\mathrm{E}(\delta(\{2\}, a))$ = E({ }) U E({2, 3}) = { } U { 2, 3 } = { 2, 3 }

δ'(R, a) = δ'({ 3 } , a) = $\mathrm{E}(\delta(\{3\}, a))$ = E({ 1 }) = {1, 3}

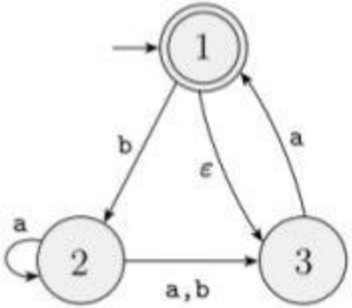
δ'(R, a) = δ'({ 3 } , b) = $\mathrm{E}(\delta(\{3\}, b))$ = E({ }) = { }

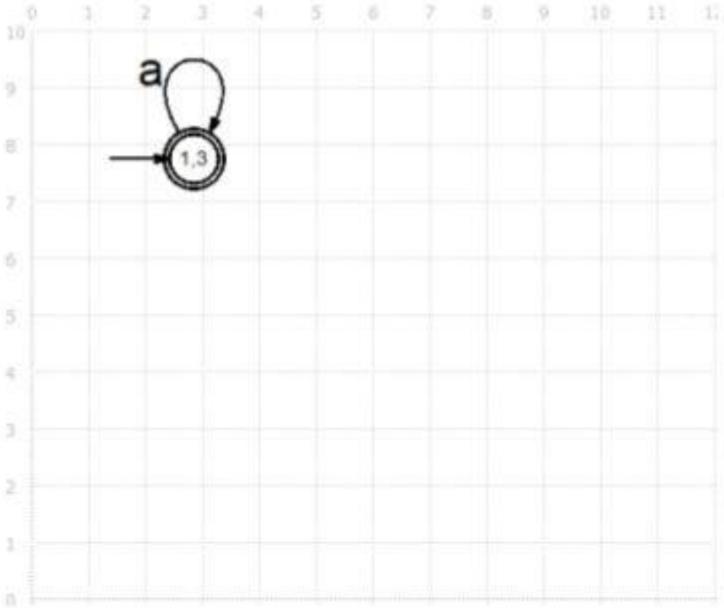
Step 4 : M = (Q', Σ , δ', q_0', F')





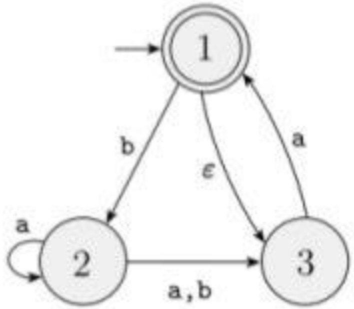
Step 4 : M = (Q', Σ , δ', q_0', F')

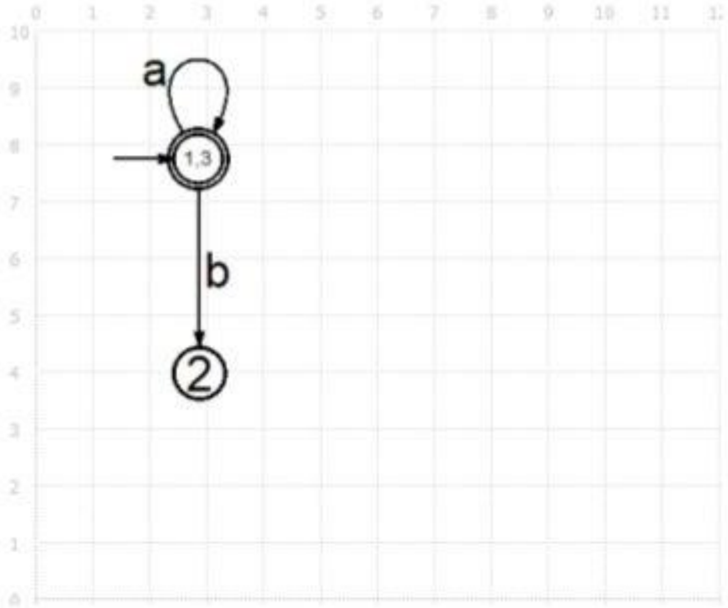




State	1, 3
Next States without ε	1
Final States with ε-closure	1, 3

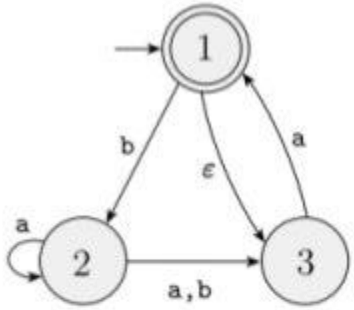
Step 4 : M = (Q', Σ , δ', q_0', F')

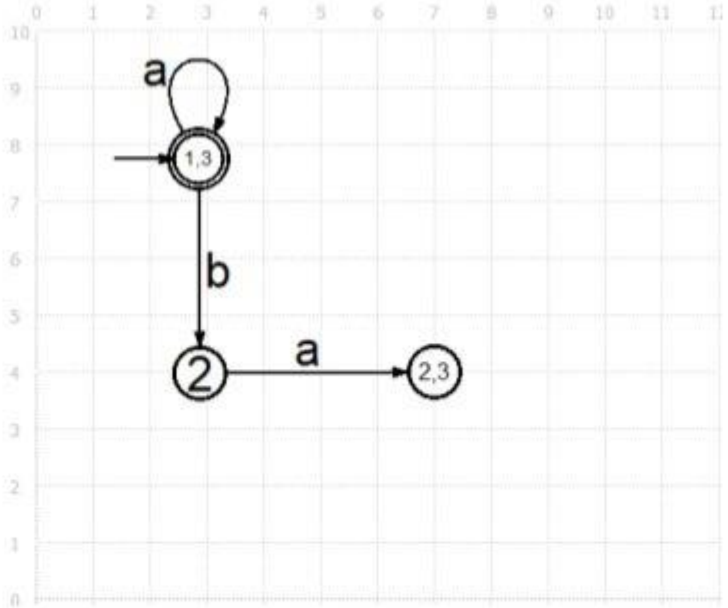




State	1, 3
Next States without ε	2
Final States with ε-closure	2

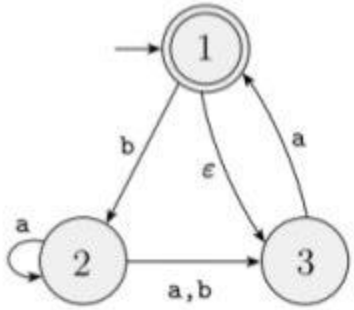
Step 4 : M = (Q', Σ , δ', q0', F')

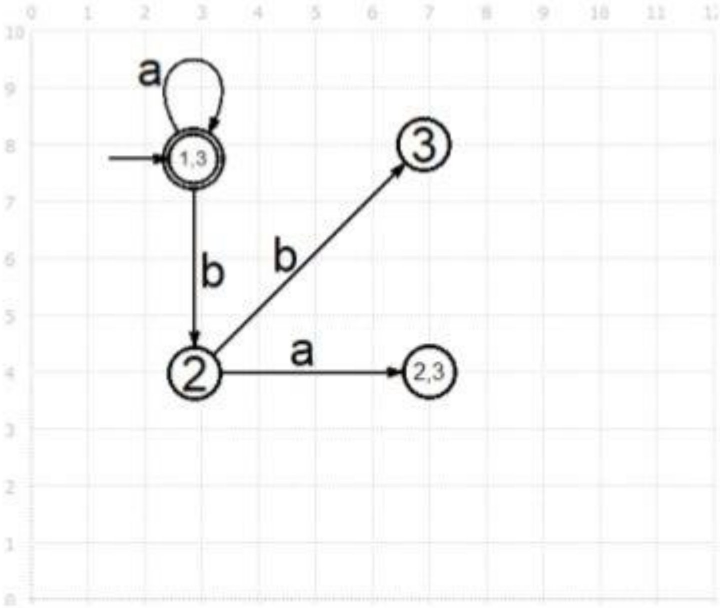




State	2
Next States without ε	2, 3
Final States with ε-closure	2, 3

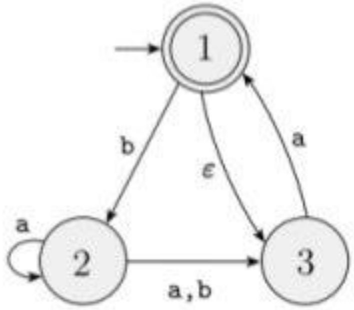
Step 4 : M = (Q', Σ , δ', q_0', F')

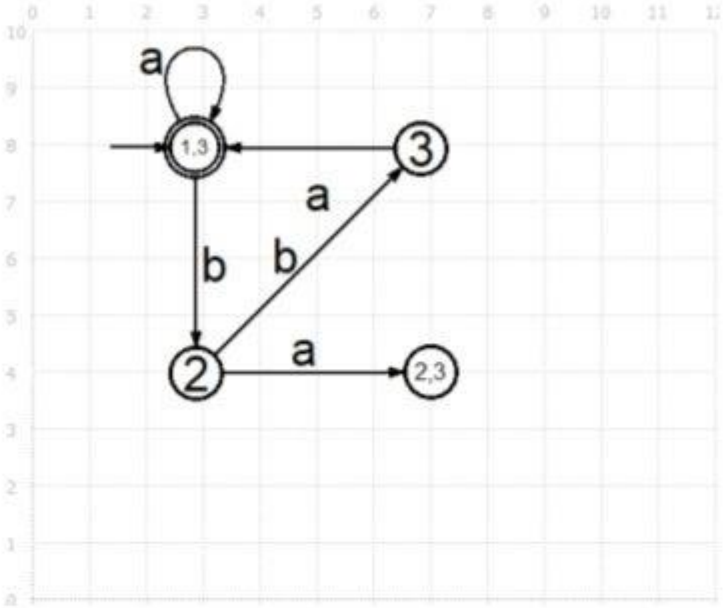




State	2
Next States without ε	3
Final States with ε-closure	3

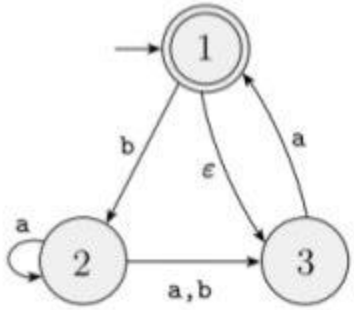
Step 4 : M = (Q', Σ , δ', q_0', F')

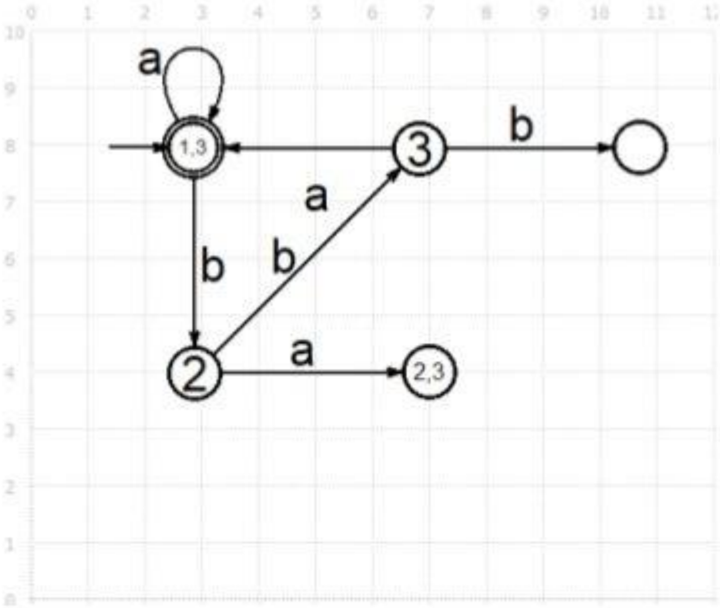




State	3
Next States without ε	1
Final States with ε-closure	1, 3

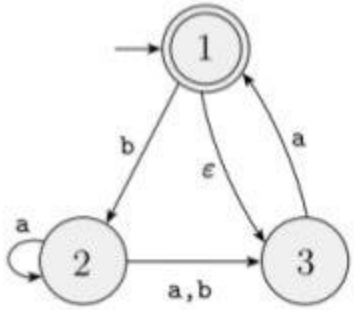
Step 4 : M = (Q', Σ , δ', q_0', F')

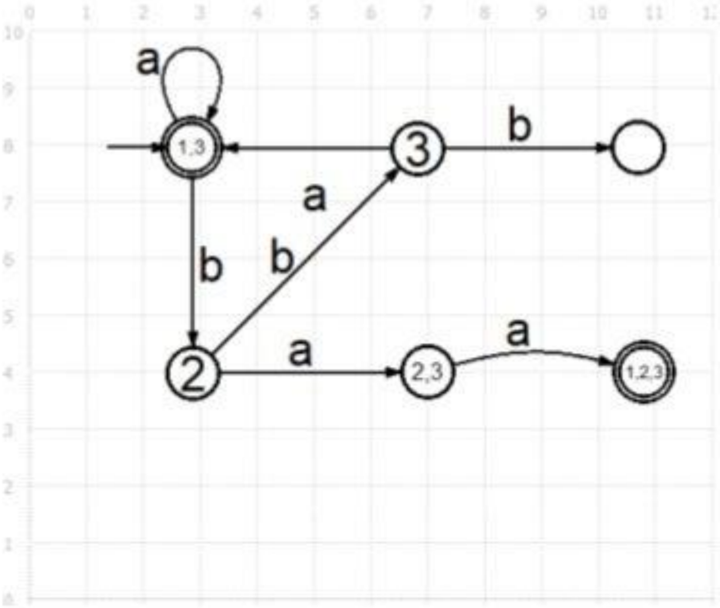




State	3
Next States without ε	{ }
Final States with ε-closure	{ }

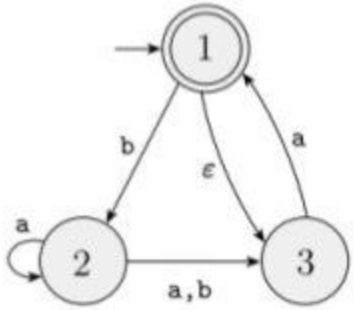
Step 4 : M = (Q', Σ , δ', q_0', F')

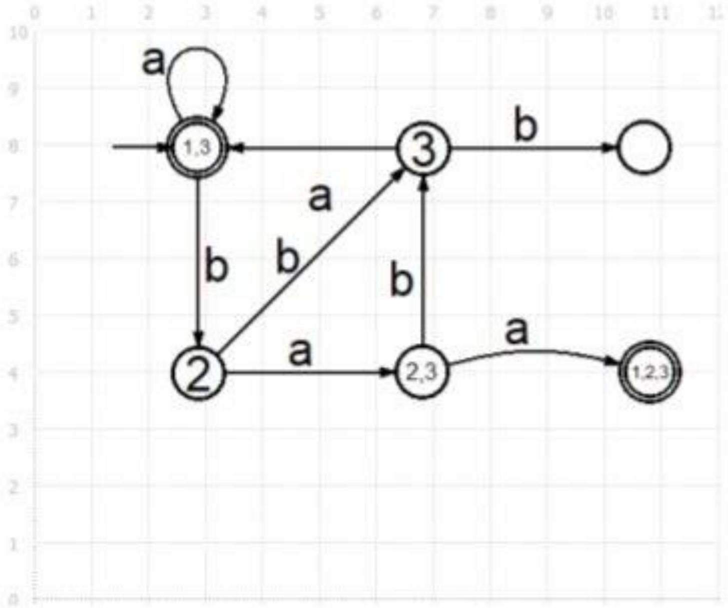




State	2, 3
Next States without ε	2, 3, 1
Final States with ε-closure	2, 3, 1

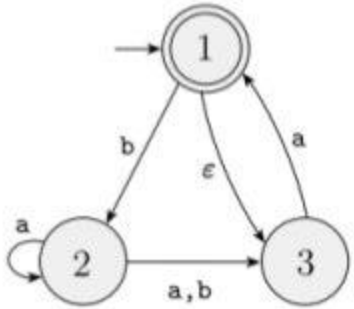
Step 4 : M = (Q', Σ , δ', q_0', F')

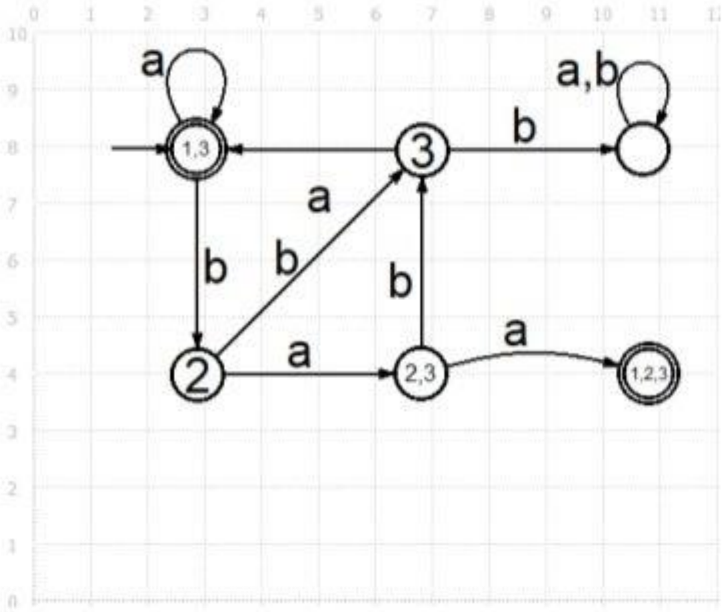




State	2, 3
Next States without ε	3
Final States with ε-closure	3

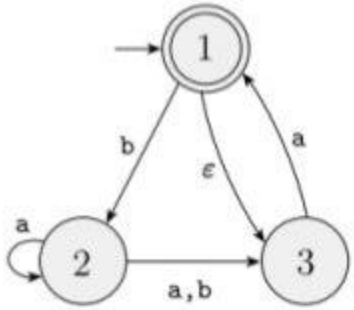
Step 4 : M = (Q', Σ , δ', q₀', F')

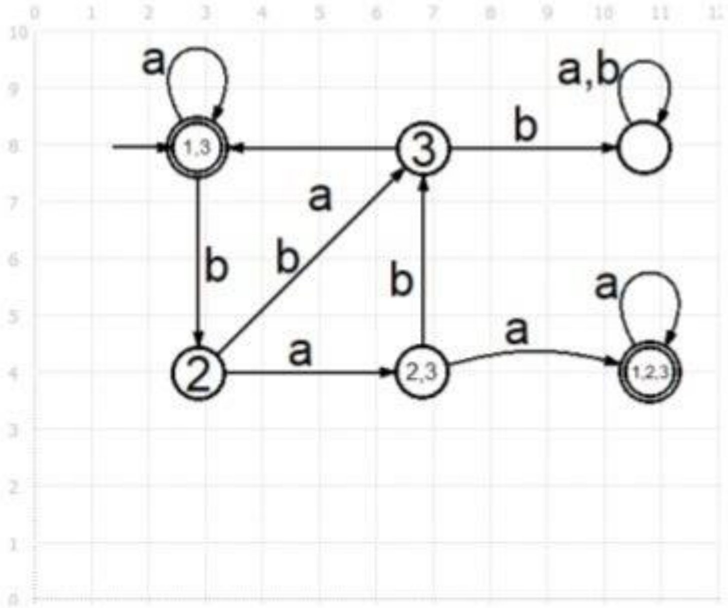




State	{ }
Next States without ε	{ }
Final States with ε-closure	{ }

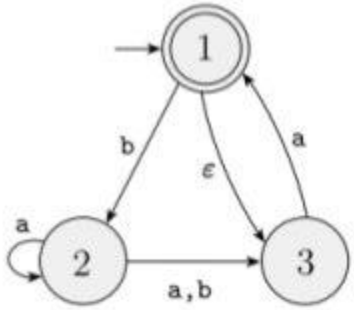
Step 4 : M = (Q', Σ , δ', q_0', F')

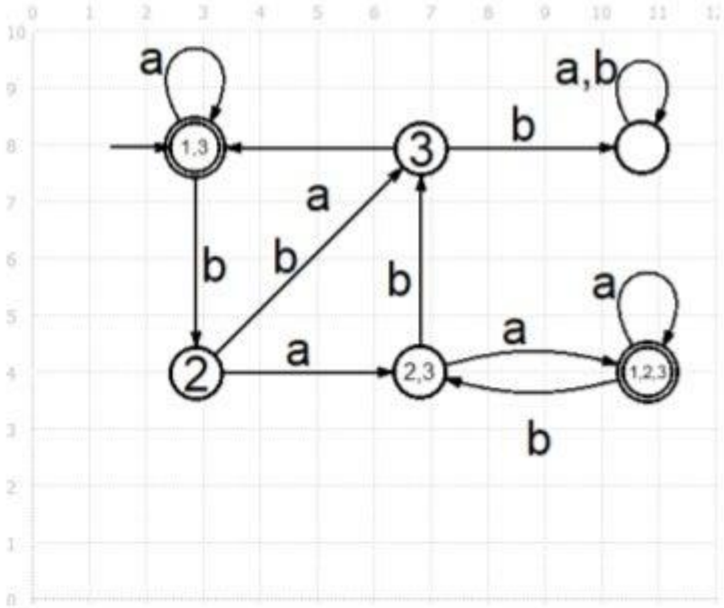




State	1, 2, 3
Next States without ε	2, 3, 1
Final States with ε-closure	2, 3, 1

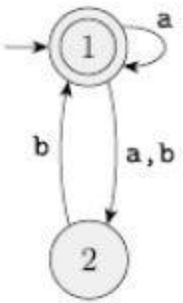
Step 4 : $M = (Q', \Sigma, \delta', q_0', F')$

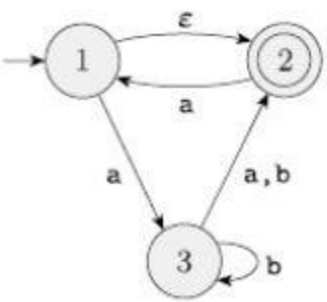


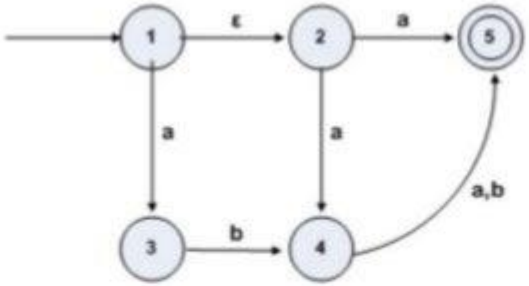


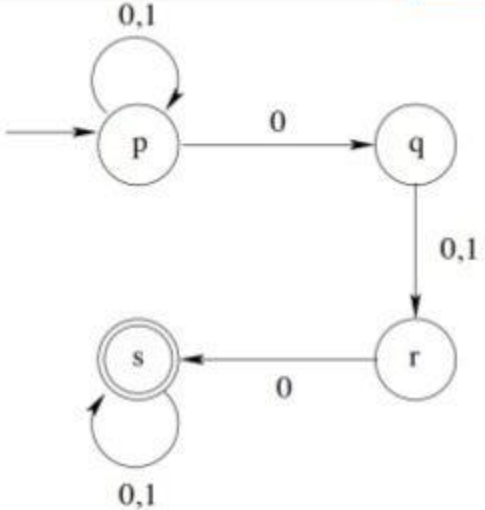
State	1, 2, 3
Next States without ε	2, 3
Final States with ε-closure	2, 3

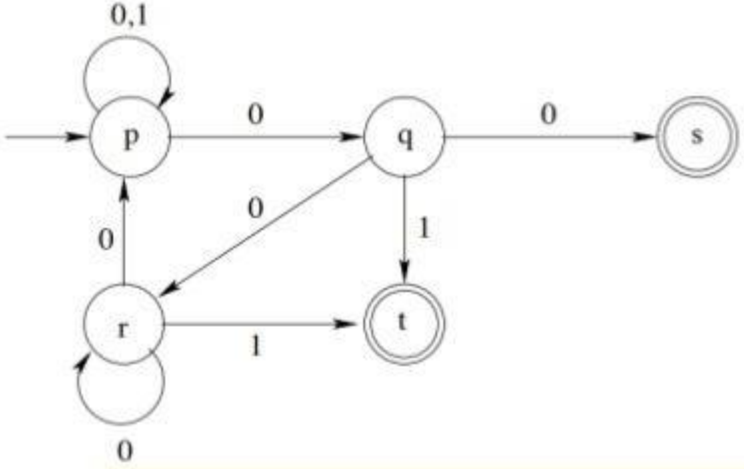
NFA → DFA : Practices

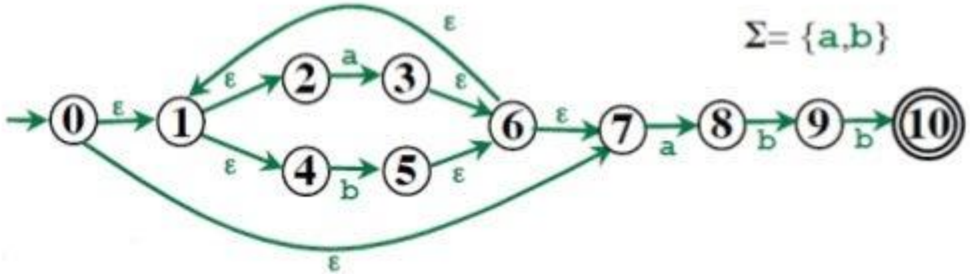










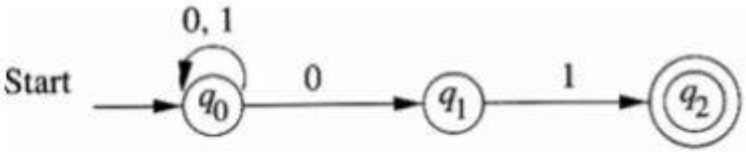


Question Archive

1. Design the state diagram for the following NFAs
   a) Draw the state diagram of an NFA / ε-NFA which accepts strings those do not contain substring "main". Here, $\Sigma = \{ a,b,c,d,...,z\}$
   b) Draw the state diagram of an NFA / ε-NFA which accepts binary strings which has even values. Here, $\Sigma = \{ 0, 1\}$ (Accepted: 01010, Not accepted : 10101)
   c) Draw the state diagram of an NFA / ε-NFA which recognizes FIFA World Cup years in 4 digits. Assume World Cup occurs every 4 years starting from 2002. Here, $\Sigma = \{ 0,1,2,3,4,5,6,7,8,9\}$.
2. Design the state diagram for the following DFA's
   a) Draw the state diagram of an NFA/ ε-NFA which accepts strings of length at most 5. The set of accepted symbols is {0, 1, 2}
   b) Draw the state diagram of an NFA/ ε-NFA for alphabet set {a, b, c} which starts with 'abc' or ends with 'bb'.
   c) Draw the state diagram of an NFA/ ε-NFA which accepts those binary strings that has odd decimal values. The set of accepted symbols is {0, 1}

Question Archive

3. Design the state diagram for the following NFAs
   a) Draw the state diagram of an NFA/ϵ-NFA for alphabet set {a, b} which starts and ends with "ab".
   b) Draw the state diagram of an NFA/ϵ-NFA for alphabet set {a, b} which contains a 'b' in its third position from the last. [Sample accepted strings: "abba", "baa", "ababab"].
   c) Consider the following NFA, and show with the help of NFA-tree whether the string "001010" is accepted or not.



4. Design the state diagram for the following DFA's
   a) Draw the state diagram of a NFA/ Є-NFA over alphabet set {a, b, c} that starts with 'ab' or 'ac' and does not end with 'a'. [Sample Accepted Strings: abcc, acbab, abaacb]
   b) Draw the state diagram of a NFA/ Є-NFA over alphabet set {0, 1} that contains '1011' and '000' as a substring.

Question Archive

5. Design the state diagram for the following NFAs
   a) Draw the state diagram of an NFA / ε-NFA which accepts strings having both 'web' and 'security' as substrings . Here, $\Sigma = \{ a, b, c, d, ..., z \}$
   b) Draw the state diagram of an NFA / ε-NFA which accepts strings having *1* at the 3rd position from the last. Here, $\Sigma = \{0, 1\}$
   c) Draw the state diagram of an NFA / ε-NFA which recognizes leap years. Assume that, leap year occurs every 4 years starting from 0. Here, $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$.

6. Design the state diagram for the following DFA's
   a) Draw the state diagram of an NFA or ε-NFA that accepts all binary strings which start with 1 or end with 001
   b) Draw the state diagram for alphabet set $\{0, 1, 2, \ldots, 9\}$ of an NFA or ε-NFA that accepts strings that ends with the digit *5* and also 5 is the summation of the first two digits.
   c) Draw the state diagram of an NFA or ε-NFA for the language $\{ w \in \Sigma^* \mid$ w contains at least two 0's or exactly two 1's