CSC-257
Theory Of Computation
(BSc CSIT, TU)

Ganesh Khatri
kh6ganesh@gmail.com

Conversion of NFA to DFA

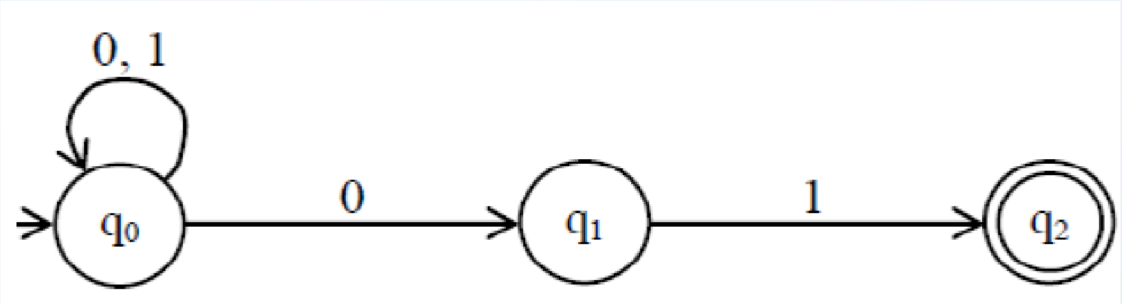
- Although there are many languages for which NFA is easier to construct than DFA, it can be proved that every language that can be described by some NFA can also be described by some DFA
- The DFA has more transition than NFA and in worst case, the smallest DFA can have 2^n state while the smallest NFA for the same language has only n states
- DFAs & NFAs accept exactly the same set of languages. That is non-determinism does not make a finite automaton more powerful
- We can convert an NFA to a DFA using "subset construction algorithm".
- The key idea behind the algorithm is that; the equivalent DFA simulates the NFA by keeping track of the possible states it could be in.
- Each state of DFA corresponds to a subset of the set of states of the NFA, hence the name of the algorithm. If NFA has n-states, the DFA can have 2^n states (at most), although it usually has many less.

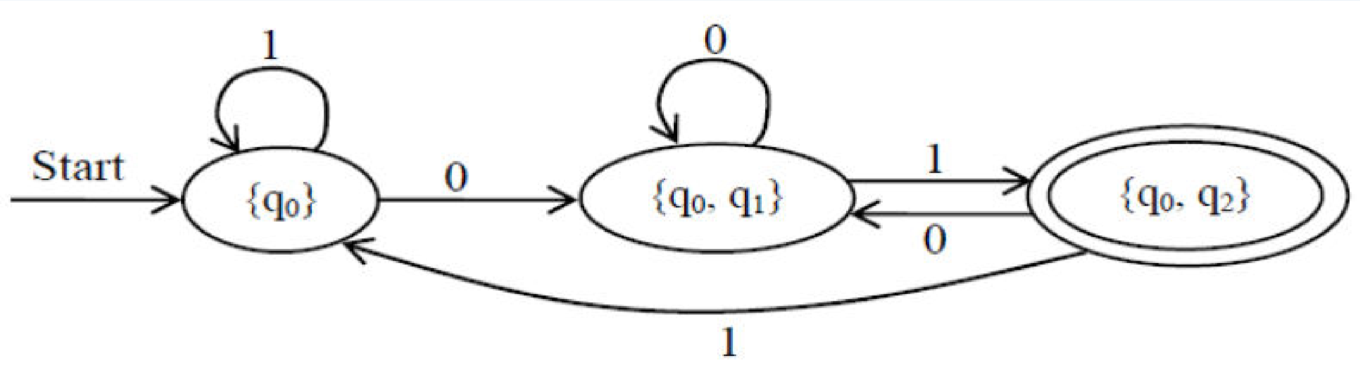
Conversion of NFA to DFA

- To convert a NFA, $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ into an equivalent DFA $D = (Q_D, \Sigma, \delta_D, q_0, F_D)$, we have following steps :
- 1.The start state of D is the set of start states of N i.e. if q0 is start state of N then D has start state as {q0}
- 2.Q_D is set of subsets of Q_N i.e. $Q_D = 2^{Q_N}$. So, Q_D is power set of Q_N. So if Q_N has n states then Q_D will have 2^n states. However, all of these states may not be accessible from start state of Q_D so they can be eliminated. So Q_D will have less than 2^n states.
- 3.F_D is set of subsets S of Q_N such that $S \cap F_N \neq \varphi$ i.e. F_D is all sets of N's states that include at least one final state of N
- For each set $S \subseteq Q_N$ & each input $a \in \Sigma$, $\delta_D(S, a) = \bigcup_{p \text{ in } s} \delta N(p,a)$
- i.e. for any state {q0, q1, q2, … qk} of the DFA & any input a, the next state of the DFA is the set of all states of the NFA that can result as next states if the NFA is in any of the state's q0, q1, q2, … qk when it reads a.

Conversion of NFA to DFA : Example

- Convert given DFA to NFA.



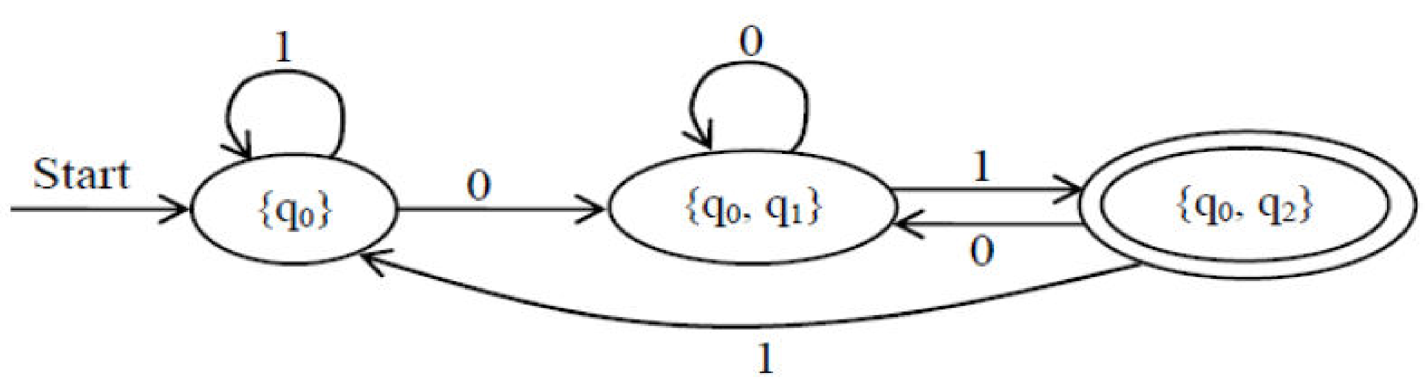


	δ:	0	1
A	ϕ	ϕ	ϕ
B	$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
C	$\{q_1\}$	ϕ	$\{q_2\}$
D	$*\{q_2\}$	ϕ	ϕ
E	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
F	$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
G	$*\{q_1, q_2\}$	ϕ	$\{q_2\}$
H	$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

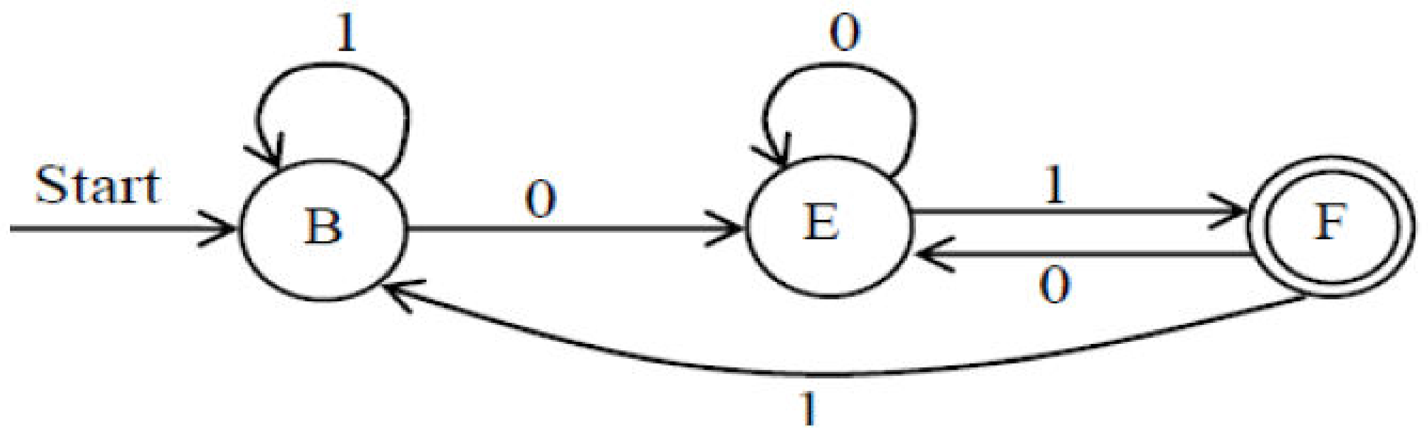
Conversion of NFA to DFA : Example

- The same table can be represented with renaming the state on table entry as
- So, the equivalent DFA is

δ:	0	1
A	A	A
→B	E	B
C	A	D
*D	A	A
E	E	F
*F	E	B
*G	A	D
*H	E	F



or



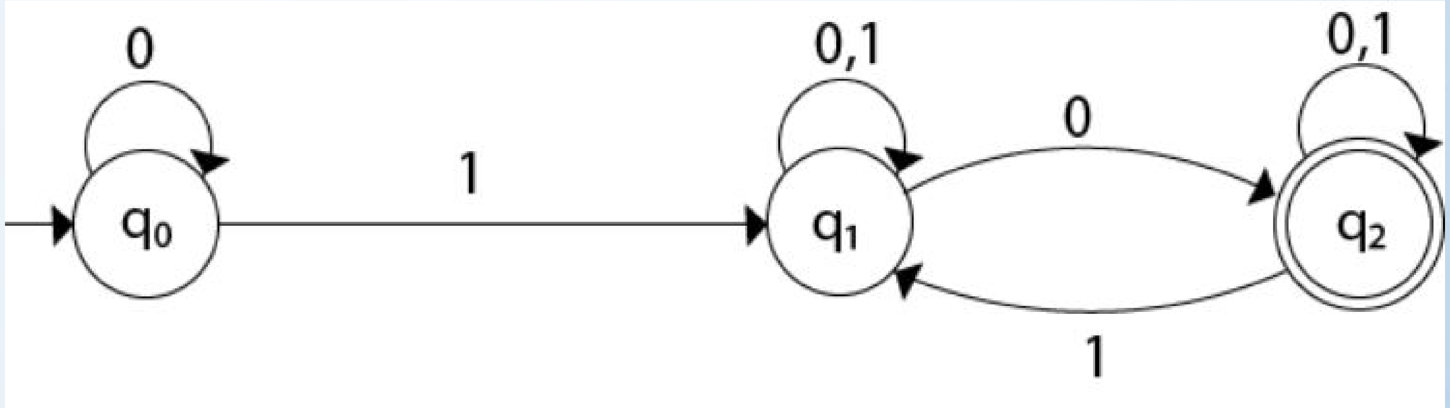
- The other state are removed because they are not reachable from start state

Steps for converting NFA to DFA

- Let, M = (Q, ∑, δ, q0, F) is an NFA which accepts the language L(M).
- There should be equivalent DFA denoted by M' = (Q', ∑, q0, δ', F') such that L(M) = L(M')
- **Step 1:** Initially Q' = ϕ
- **Step 2:** Add q0 of NFA to Q'. Then find the transitions from this start state.
- **Step 3:** In Q', find the possible set of states for each input symbol. If this set of states is not in Q', then add it to Q'
- **Step 4:** In DFA, the final state will be all the states which contain F(final states of NFA)

Steps for converting NFA to DFA

- Convert the given NFA to DFA



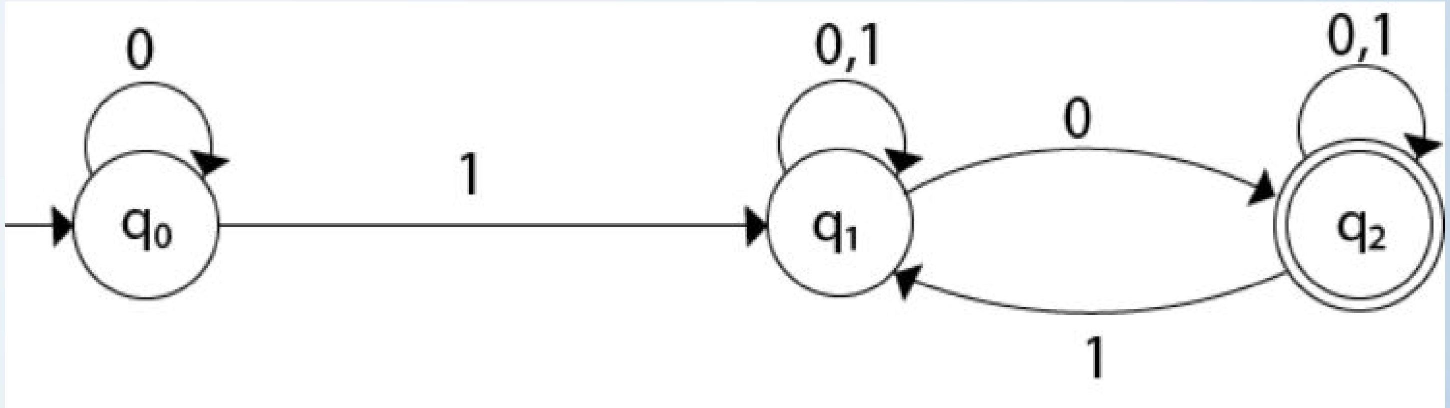
- For the given transition diagram we will first construct the transition table

State	0	1
→[q0]	[q0]	[q1]
[q1]	[q1, q2]	[q1]
*[q2]	[q2]	[q1, q2]
*[q1, q2]	[q1, q2]	[q1, q2]

- Now we will obtain δ' transition for state q0
  - δ'([q0], 0) = [q0]
  - δ'([q0], 1) = [q1]

Steps for converting NFA to DFA

- The δ' transition for state q1 is obtained as
  - δ'([q1], 0) = [q1, q2]
  - δ'([q1], 1) = [q1]



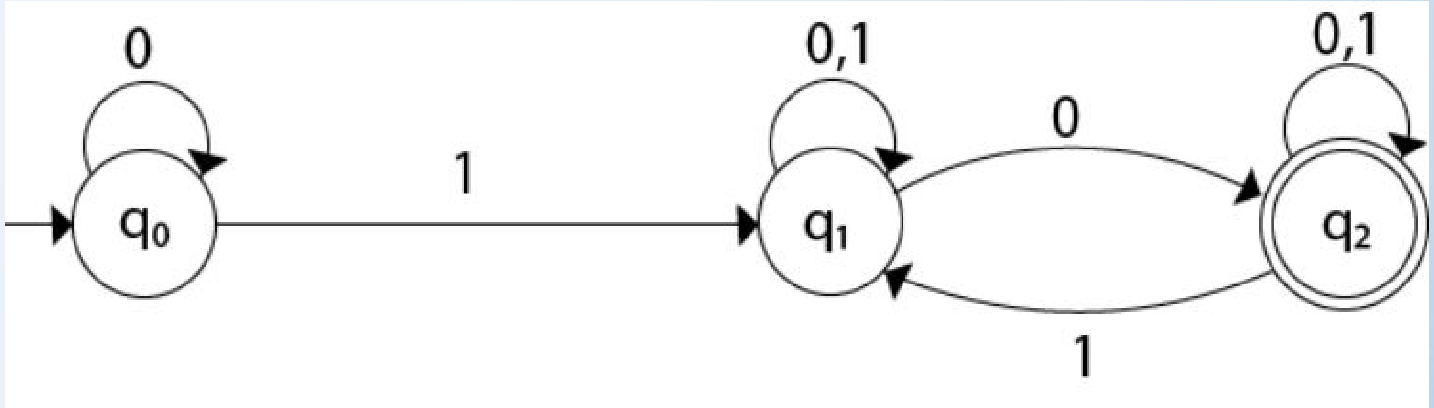
- The δ' transition for state q2 is obtained as:
  - δ'([q2], 0) = [q2]
  - δ'([q2], 1) = [q1, q2]

State	0	1
→[q0]	[q0]	[q1]
[q1]	[q1, q2]	[q1]
*[q2]	[q2]	[q1, q2]
*[q1, q2]	[q1, q2]	[q1, q2]

Steps for converting NFA to DFA

- ow we will obtain δ' transition on [q1, q2]
  - $\delta'([q1, q2], 0) = \delta(q1, 0) \cup \delta(q2, 0)$
    $= \{q1, q2\} \cup \{q2\}$
    $= [q1, q2]$
  - $\delta'([q1, q2], 1) = \delta(q1, 1) \cup \delta(q2, 1)$
    $= \{q1\} \cup \{q1, q2\}$
    $= \{q1, q2\}$
    $= [q1, q2]$

- The state [q1, q2] is the final state as well because it contains a final state q2

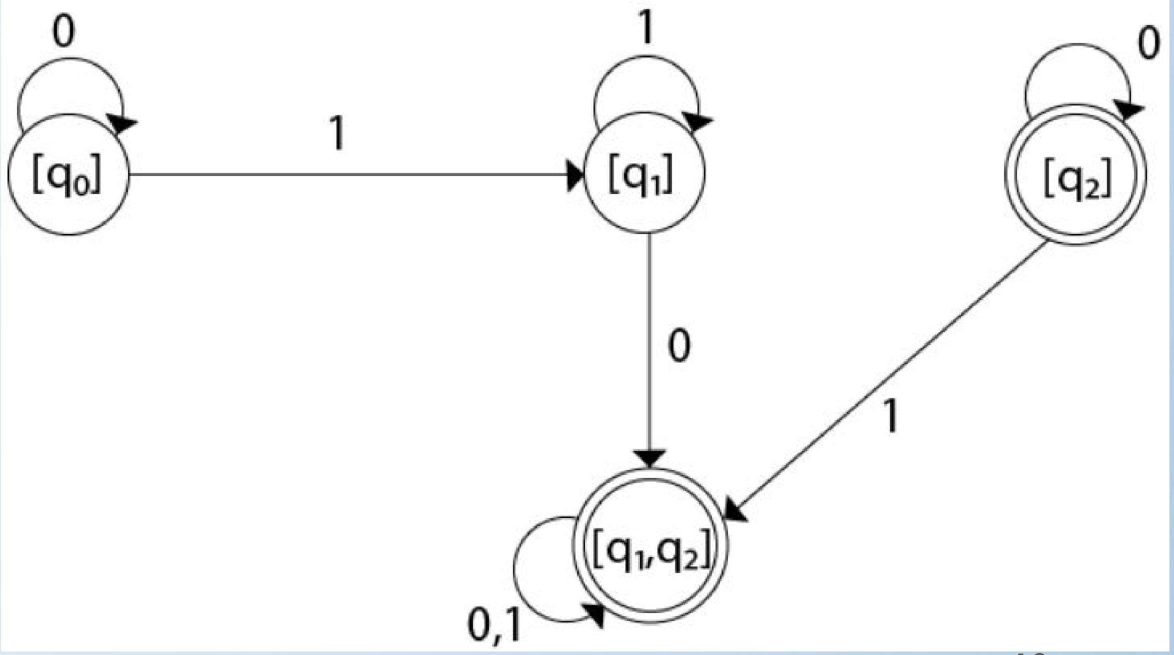


State	0	1
→[q0]	[q0]	[q1]
[q1]	[q1, q2]	[q1]
*[q2]	[q2]	[q1, q2]
*[q1, q2]	[q1, q2]	[q1, q2]

Steps for converting NFA to DFA

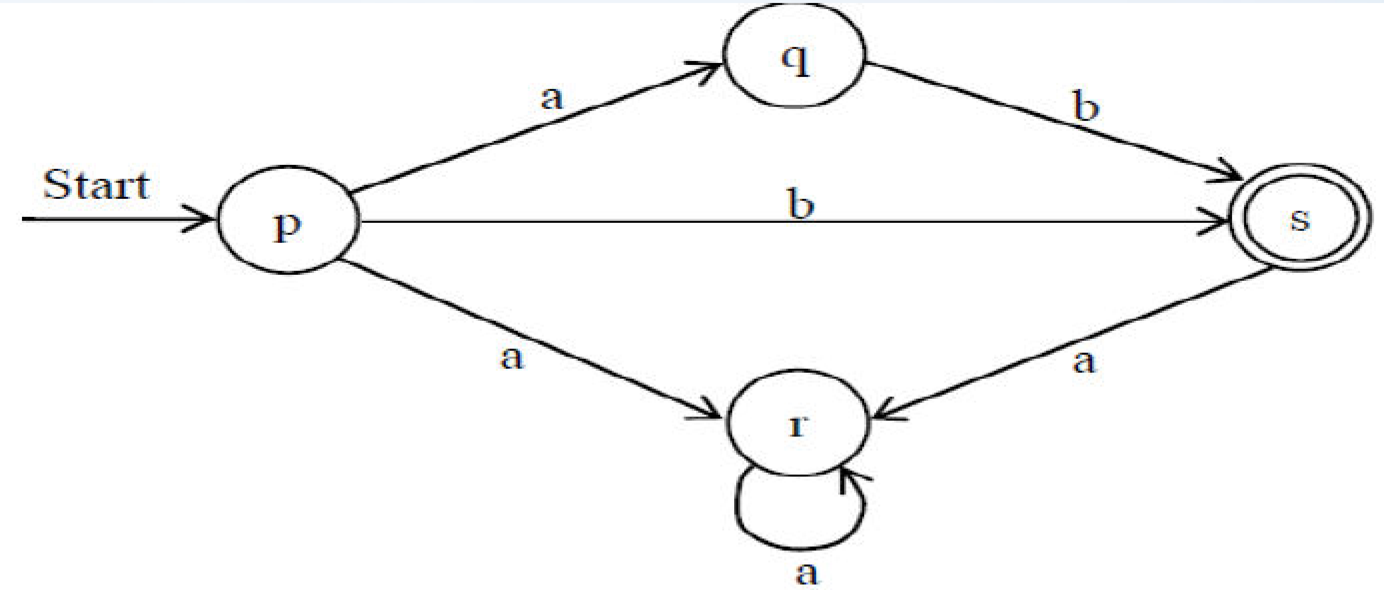
- The equivalent DFA will be :
- **Note :** The state q2 can be eliminated because q2 is an unreachable state
- **Reference :** https://www.javatpoint.com/automata-conversion-from-nfa-to-dfa

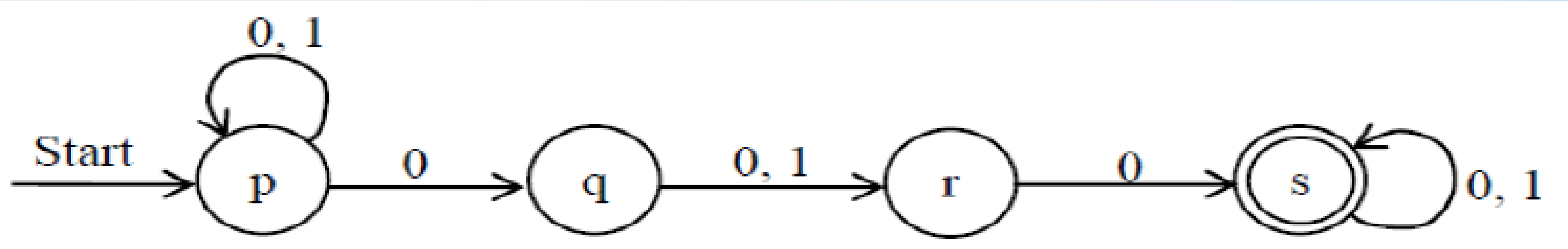
State	0	1
→[q0]	[q0]	[q1]
[q1]	[q1, q2]	[q1]
*[q2]	[q2]	[q1, q2]
*[q1, q2]	[q1, q2]	[q1, q2]



Conversion of NFA to DFA : Examples

- Convert the following NFAs to DFAs



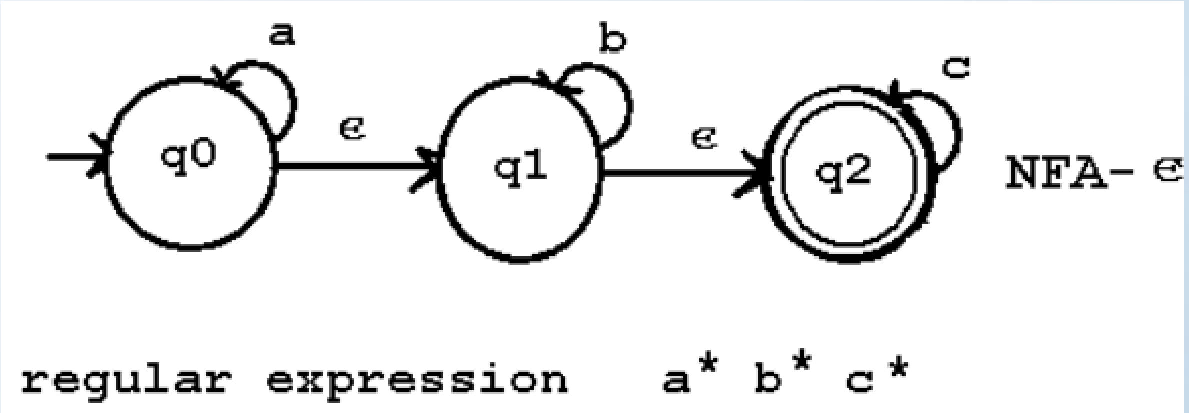


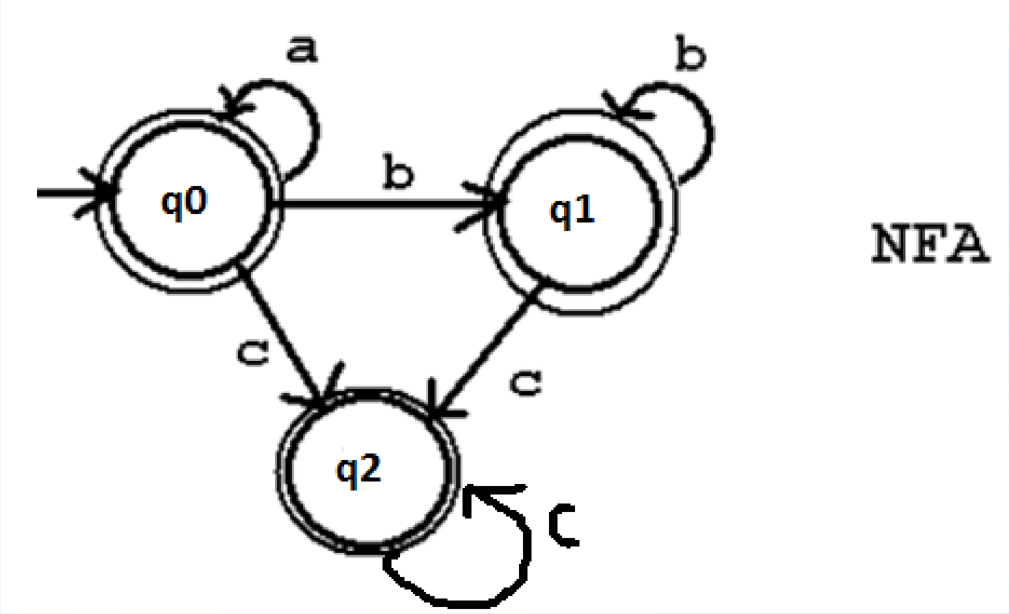
NFA with epsilon moves(ε-NFA)

- The ε-NFA (also sometimes called NFA-λ or NFA with epsilon moves) replaces the transition function with one that allows the empty string ε as a possible input, so that one has instead $\Delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$
- It can be shown that ordinary NFA and NFA-ε are equivalent, in that, given either one, one can construct the other, which recognizes the same language.
- We can extend NFA by introducing a "feature" that allows us to make a transition on , the empty string.
- Just as non-determinism made NFA's more convenient to represent some problems than DFA's but were not more powerful; the same applies to ε-NFA’s.
- anything we can represent with an ε-NFA, can be represented with a DFA that has no ε transitions.
- The ε (epsilon) transition refers to a transition from one state to another without the reading of an input symbol (i.e. without the tape containing the input string moving).
- Epsilon transitions can be inserted between any states.
- There is also a conversion algorithm from a NFA with epsilon transitions to a NFA without epsilon transitions

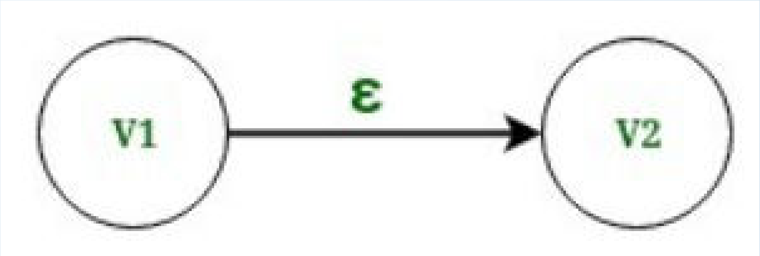
NFA with epsilon moves(ε-NFA)

- Consider the NFA-epsilon move machine M = { Q, Σ, δ, q0, F}
- where
  - Q = { q0, q1, q2 }
  - Σ= { a, b, c } and ε moves
  - q0 = q0
  - F = { q2 }
- The language accepted by the above NFA with epsilon moves is the set of strings over {a,b,c} including the null string and all strings with any number of a's followed by any number of b's followed by any number of c's





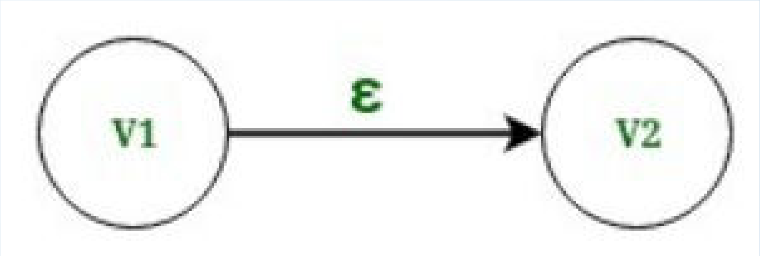
Conversion of Epsilon-NFA(ε-NFA) to NFA



- **Step1 :** Consider the two vertexes having the epsilon move. Here in Fig.1 we have vertex v1 and vertex v2 having epsilon move from v1 to v
- **Step 2:** Now find all the moves to any other vertex that start from vertex v2 (other than the epsilon move that is considering)
- After finding the moves, duplicate all the moves that start from vertex v2, with the same input to start from vertex v1 and remove the epsilon move from vertex v1 to vertex v2
- **Step 3 :** See that if the vertex v1 is a start state or not.
- If vertex v1 is a start state, then we will also make vertex v2 as a start state. If vertex v1 is not a start state, then there will not be any change

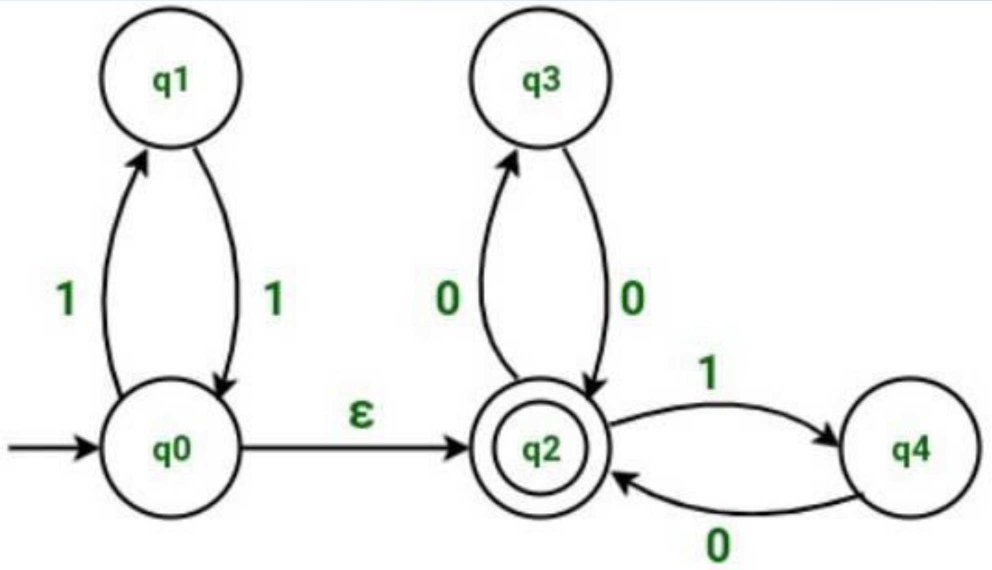
Conversion of Epsilon-NFA(ε-NFA) to NFA

- **Step 4 :** See that if the vertex v2 is a final state or not.
- If vertex v2 is a final state, then we will also make vertex v1 as a final state.
- If vertex v2 is not a final state, then there will not be any change.
- Repeat the steps(from step 1 to step 4) until all the epsilon moves are removed from the NFA.



Conversion of Epsilon-NFA(ε-NFA) to NFA

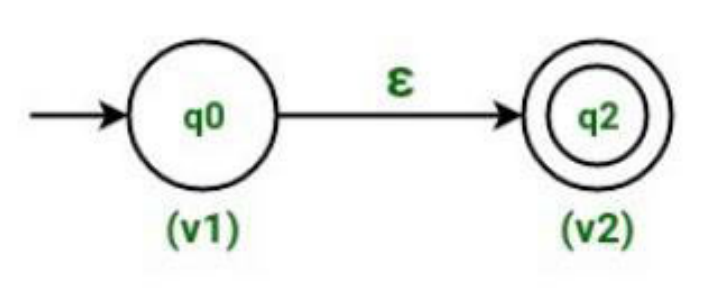
- Consider the example having states q0, q1, q2, q3, and q4
- Transition table for the above NFA is:
- We can see that we have an epsilon move from state q0 to state q2, which is to be removed.
- To remove epsilon move from state q0 to state q1, we will follow the steps as :



STATES/INPUT	INPUT 0	INPUT 1	INPUT EPSILON
q0	–	q1	q2
q1	–	q0	–
q2	q3	q4	–
q3	q2	–	–
q4	q2	–	–

Conversion of Epsilon-NFA(ε-NFA) to NFA

- **Step 1 :** Considering the epsilon move from state q0 to state q2. Consider the state q0 as vertex v1 and state q2 as vertex v2



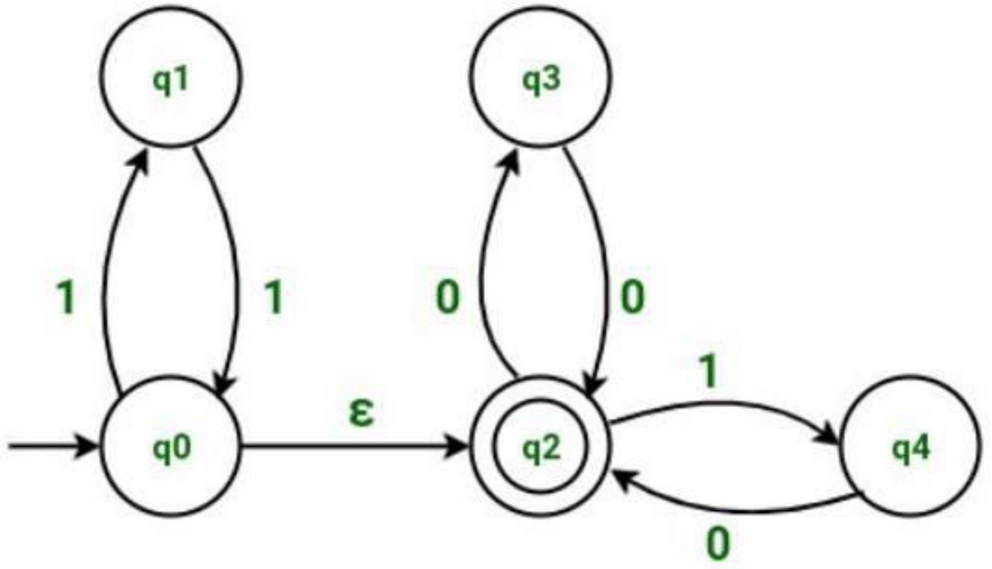
- Now find all the moves that starts from vertex v2 *(i.e. state q2)*.

- After finding the moves, duplicate all the moves that start from vertex v2 *(i.e state q2)* with the same input to start from vertex v1 *(i.e. state q0)* and remove the epsilon move from vertex v1 *(i.e. state q0)* to vertex v2 *(i.e. state q2)*.

- Since state q2 on getting input 0 goes to state q3. Hence on duplicating the move, we will have state q0 on getting input 0 also to go to state q3.

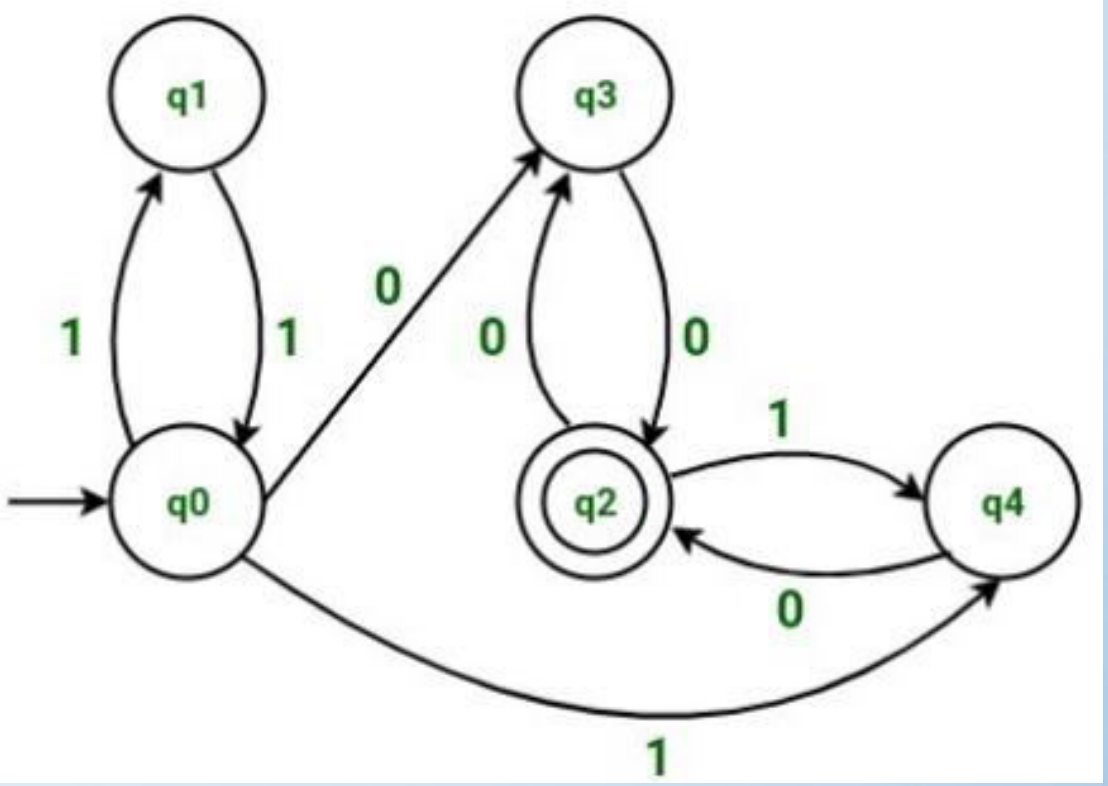
- Similarly state q2 on getting input 1 goes to state q4.

- Hence on duplicating the move, we will have state q0 on getting input 1 also to go to state q4



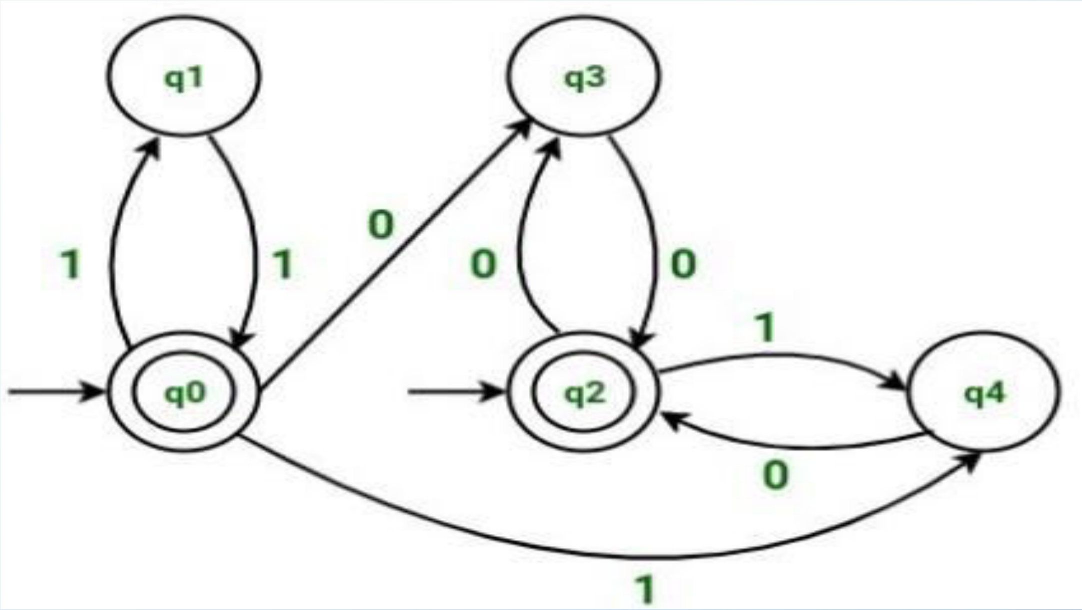
Conversion of Epsilon-NFA(ε-NFA) to NFA

- So, NFA after duplicating the moves is
- **Step 3 :** Since vertex v1 (i.e. state q0) is a start state.
- Hence we will also make vertex v2 (i.e. state q2) as a start state.
- Note that state q2 will also remain as a final state as we had initially.
- NFA after making state q2 also as a start state is



Conversion of Epsilon-NFA(ε-NFA) to NFA

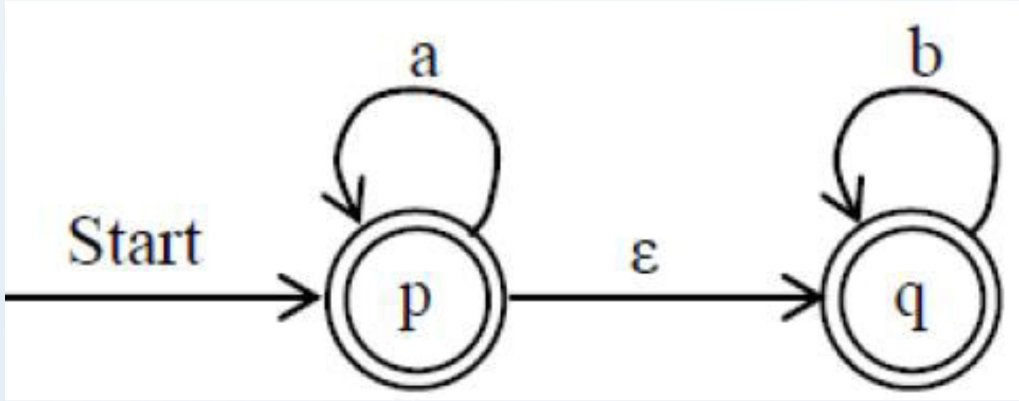
- Step 4 : Since vertex v2 (i.e. state q2) is a final state.
- Hence we will also make vertex v1 (i.e. state q0) as a final state.
- Note that state q0 will also remain as a start state as we had initially. After making state q0 also as a final state, the resulting NFA is
- The transition table for the above resulting NFA is:
- **Reference :** https://www.geeksforgeeks.org/conversion-of-epsilon-nfa-to-nfa/



STATES/INPUT	INPUT 0	INPUT 1
q0	q3	q1,q4
q1	–	q0
q2	q3	q4
q3	q2	–
q4	q2	–

NFA with epsilon moves(ε-NFA)

- A NFA with ε-transition is defined by five tuples (Q, Σ, δ, q0, F),
  - where; Q = set of finite states
  - Σ = set of finite input symbols
  - q0 = Initial state, $q0 \in Q$
  - F = set of final states; $F \subseteq Q$
  - δ = a transition function that maps; $Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$

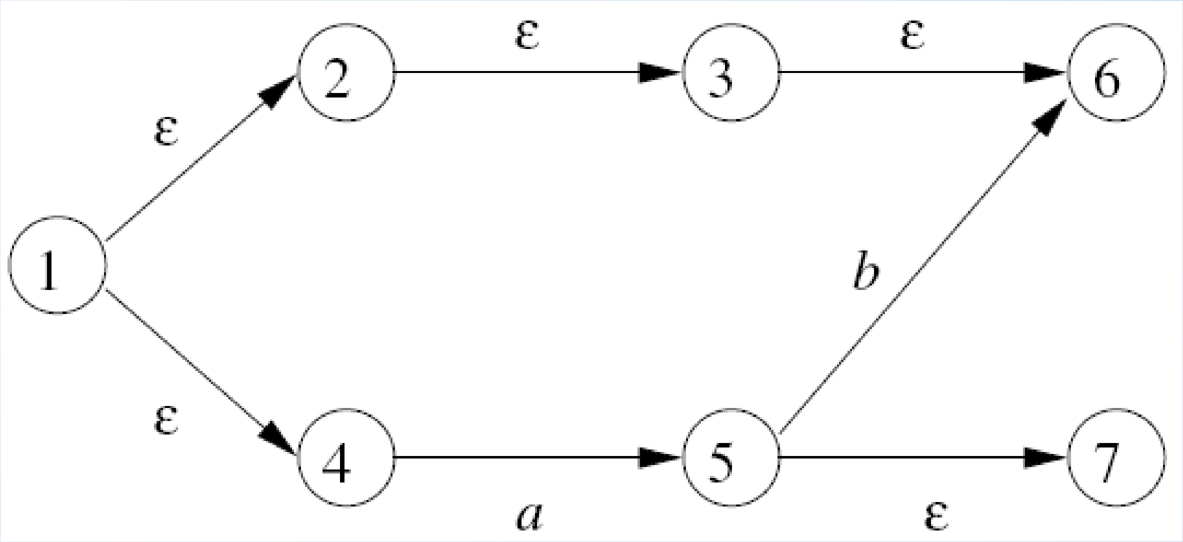


ε-closure of a state

- ε-closure of a state 'q' can be obtained by following all the transitions out of q that are labeled ε.
- After we get to another state by following ε, we follow the ε-transitions out of those states & so on, eventually finding every state that can be reached from q along any path whose arcs are all labeled with ε.
- Formally, we can define ε-closure of the state q as;
- **Basis :** state q is in ε-closure(q)
- **Induction :** If state p is reached with ε-transition from state q, p is in ε-closure(q). And if there is an arc from p to r labeled ε, then r is in ε-closure(q) and so on…

ε-closure of a state

- In this ε-NFA,
- ε-closure(1) = { 1, 2, 3, 4, 6 }
- ε-closure(2) = { 2, 3, 6 } etc.



Conversion of ε-NFA to NFA(multiple ε-transitions)

- Let $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ is an ε-NFA.
- The NFA equivalent to given ε-NFA is constructed as : $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$
- Start state of NFA = Start state of ε-NFA i.e. $q_N = q_0$
- Q_N= set of subsets of Q_E i.e $Q_N \subseteq Q_E$
- For any state $q \in Q_N$ and $a \in \Sigma$:
  - $\delta_N(q, a) = \varepsilon\text{-closure}(\delta_E(\varepsilon\text{- closure}(q), a))$
- $F_N = \{ q \in Q_N : \varepsilon\text{- closure}(q) \cap F_E \neq \phi\}$

Conversion of ε-NFA to NFA(ε-transitions)

- **Convert given ε-NFA to NFA**
- Start state = p
- $\delta_N(p,a)=\varepsilon\text{-closure}(\delta_E(\varepsilon\text{-closure}(p),a))$

  $= \varepsilon\text{-closure}(\delta_E(\{p,q,s\}, a))$

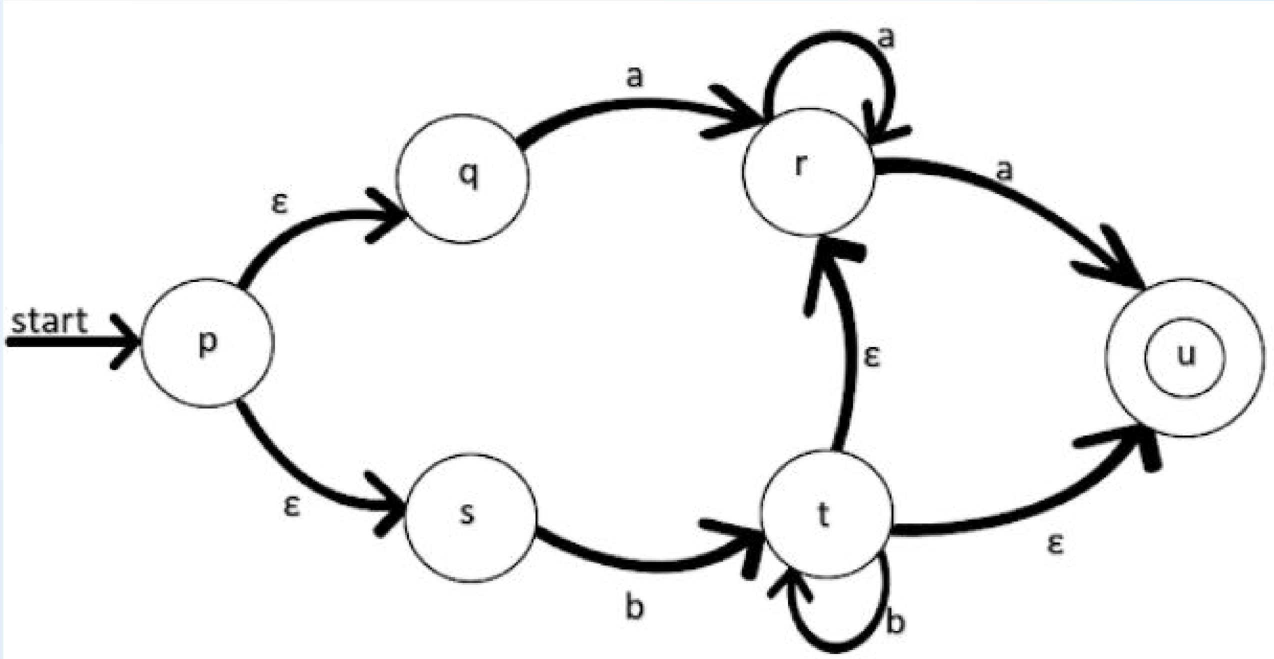
  $= \varepsilon\text{-closure}(\{r\})$

  $= \{r\}$
- $\delta_N(p,b)=\varepsilon\text{-closure}(\delta_E(\varepsilon\text{-closure}(p),b))$

  $= \varepsilon\text{-closure}(\delta_E(\{p,q,s\}, b))$

  $= \varepsilon\text{-closure}(\{t\})$

  $= \{t,u,r\}$



Conversion of ε-NFA to NFA(ε-transitions)

- **Convert given ε-NFA to NFA**
- δ_N(r,a)=ε-closure(δ_E(ε-closure(r),a))

  = ε-closure(δ_E({r}, a))

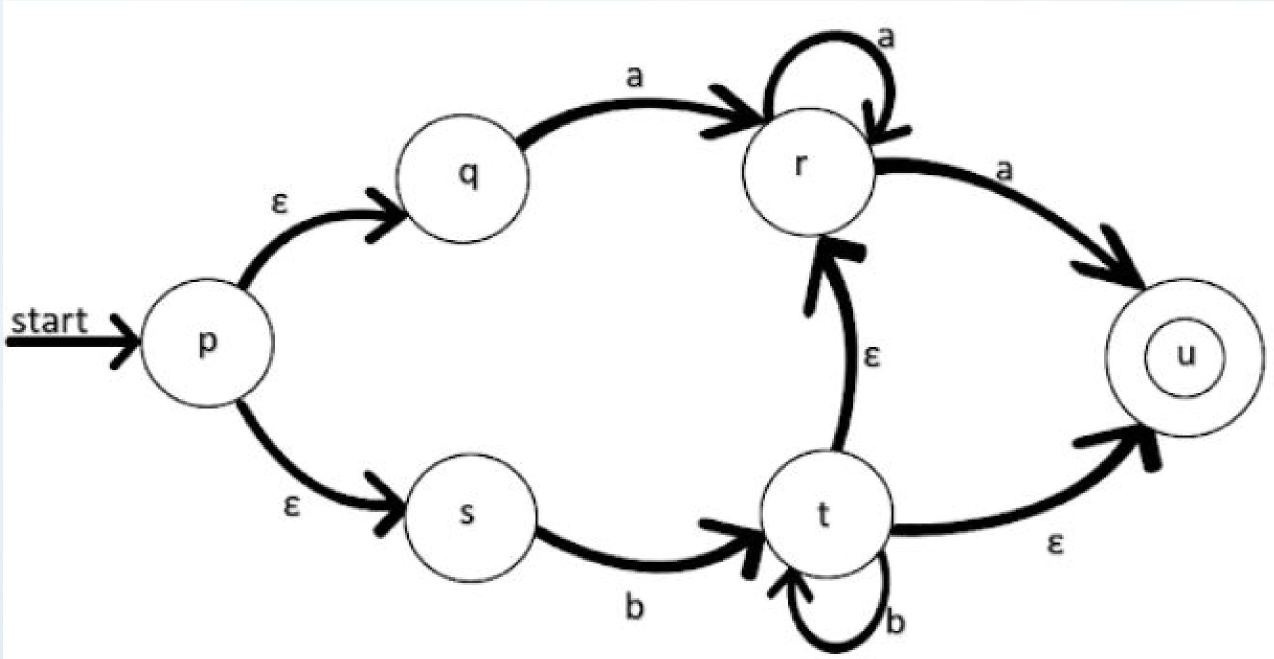
  = ε-closure({r,u})

  = {r,u}
- δ_N(r,b)=ε-closure(δ_E(ε-closure(r),b))

  = ε-closure(δ_E({r}, b))

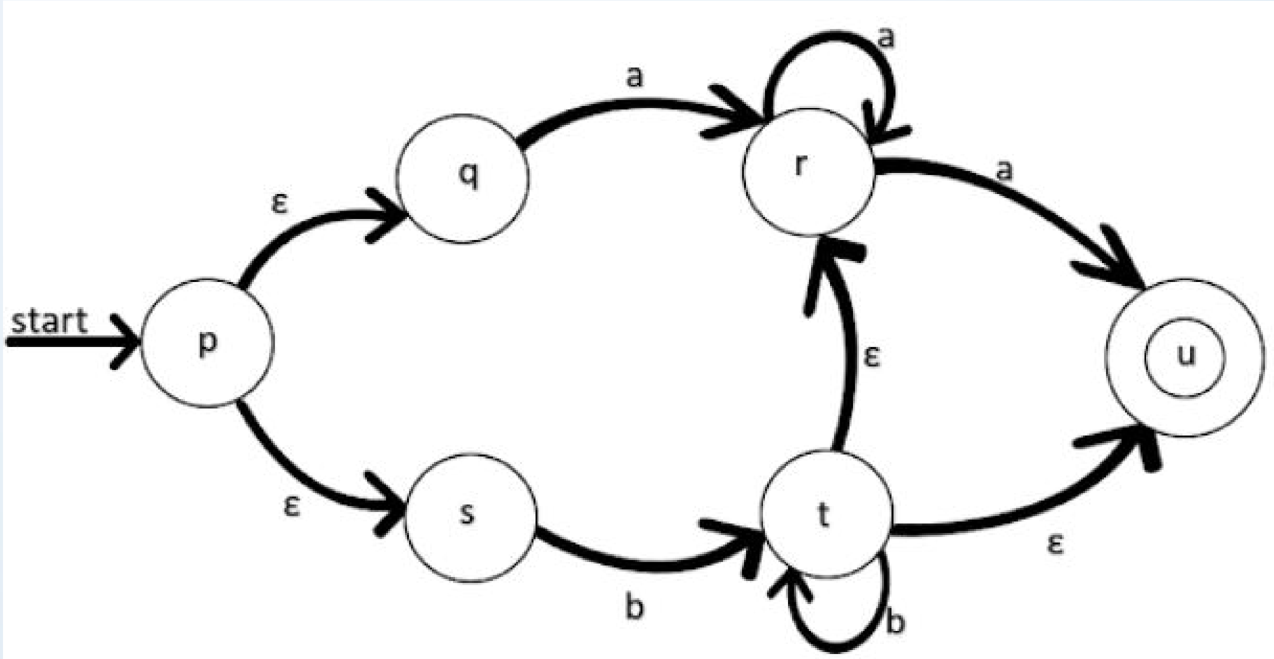
  = ε-closure({ϕ})

  = {ϕ}



Conversion of ε-NFA to NFA(ε-transitions)

- **Convert given ε-NFA to NFA**
- Similarly,
- $\delta_N(t, a) = \{r, u\}$
- $\delta_N(t, b) = \{t, r, u\}$
- $\delta_N(u, a) = \{\phi\}$
- $\delta_N(u, a) = \{\phi\}$



- Transition Table for NFA is

State	a	B
→{p}	{r}	{t, u, r}
{r}	{r, u}	{Ø}
{t}	{r, u}	{t, r, u}
*{u}	{Ø}	{Ø}
{Ø}	Ø	Ø

Conversion of ε-NFA to NFA(ε-transitions)

- Equivalent NFA is

State	a	B
→{p}	{r}	{t, u, r}
{r}	{r, u}	{Ø}
{t}	{r, u}	{t, r, u}
*{u}	{Ø}	{Ø}
{Ø}	Ø	Ø