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

Ganesh Khatri
kh6ganesh@gmail.com

Conversion of NFA to DFA

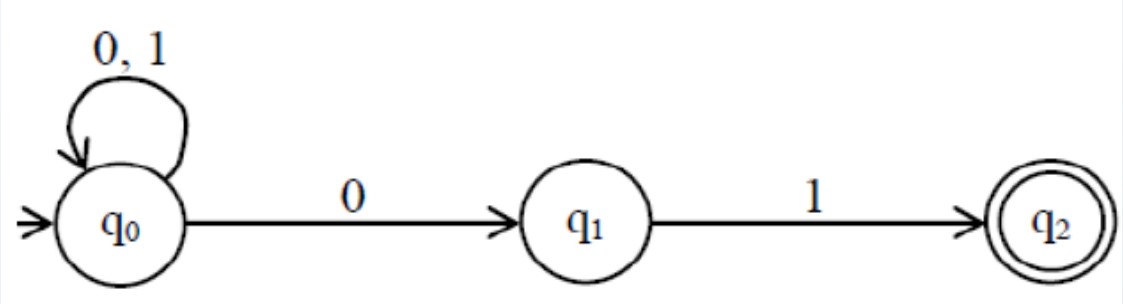
- Although there are many languages for which NFA is easier to construct than DFA, it can be proved that every language that can be described by some NFA can also be described by some DFA
- The DFA has more transition than NFA and in worst case, the smallest DFA can have 2^n state while the smallest 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 : Eample

- Convert given DFA to NFA.



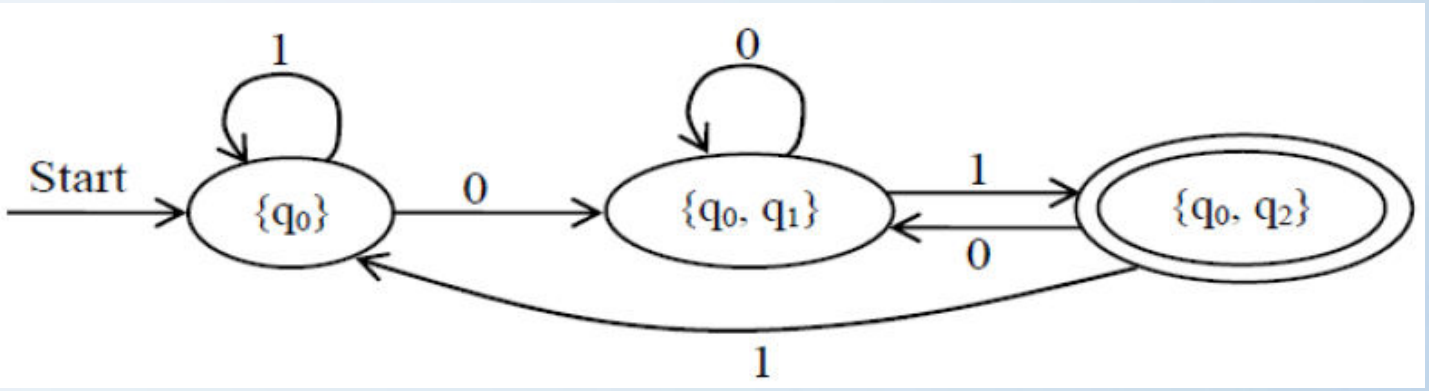
	δ:	0	1
A	ϕ	ϕ	ϕ
B	→ {q0}	{q0,q1}	{q0}
C	{q1}	ϕ	{q2}
D	*{q2}	ϕ	ϕ
E	{q0, q1}	{q0, q1}	{q0, q2}
F	*{q0, q2}	{q0, q1}	{q0}
G	*{q1, q2}	ϕ	{q2}
H	*{q0, q1, q2}	{q0, q1}	{q0, q2}

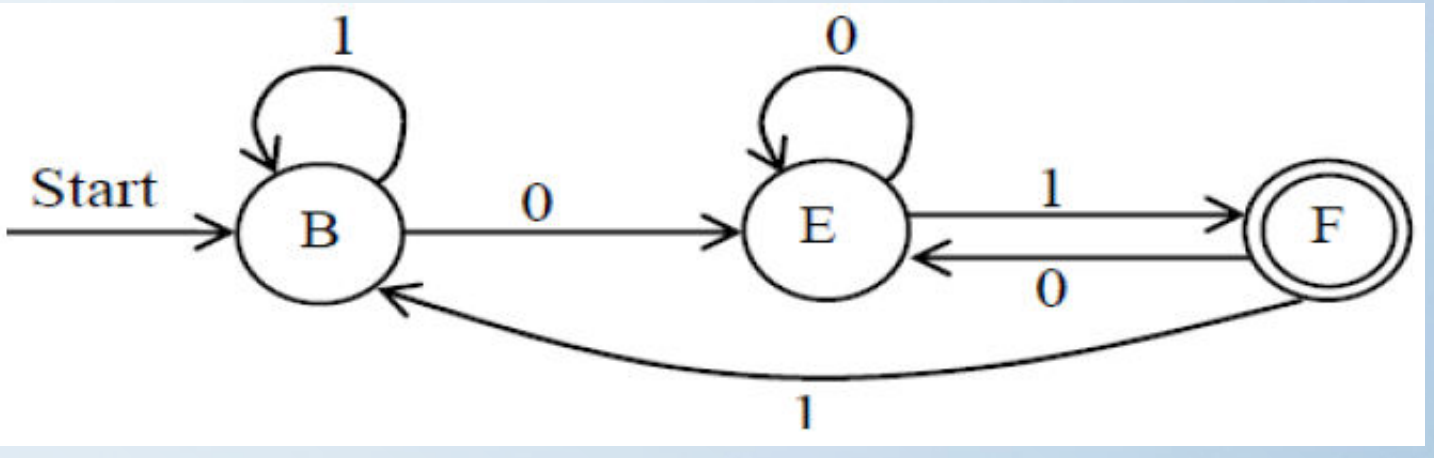
Conversion of NFA to DFA : Example

- 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

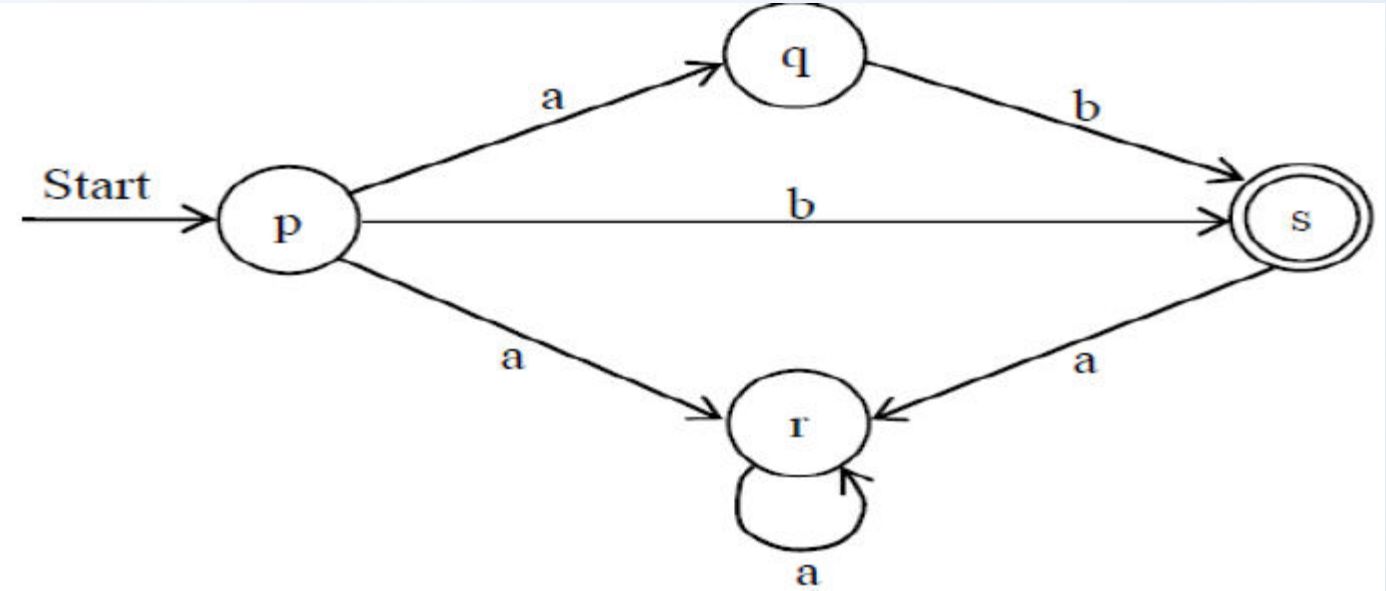


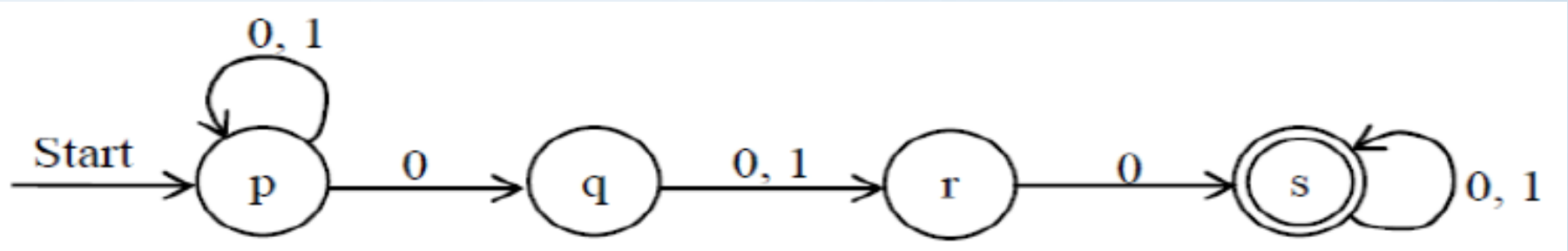


- The other state are removed because they are not reachable from start state

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