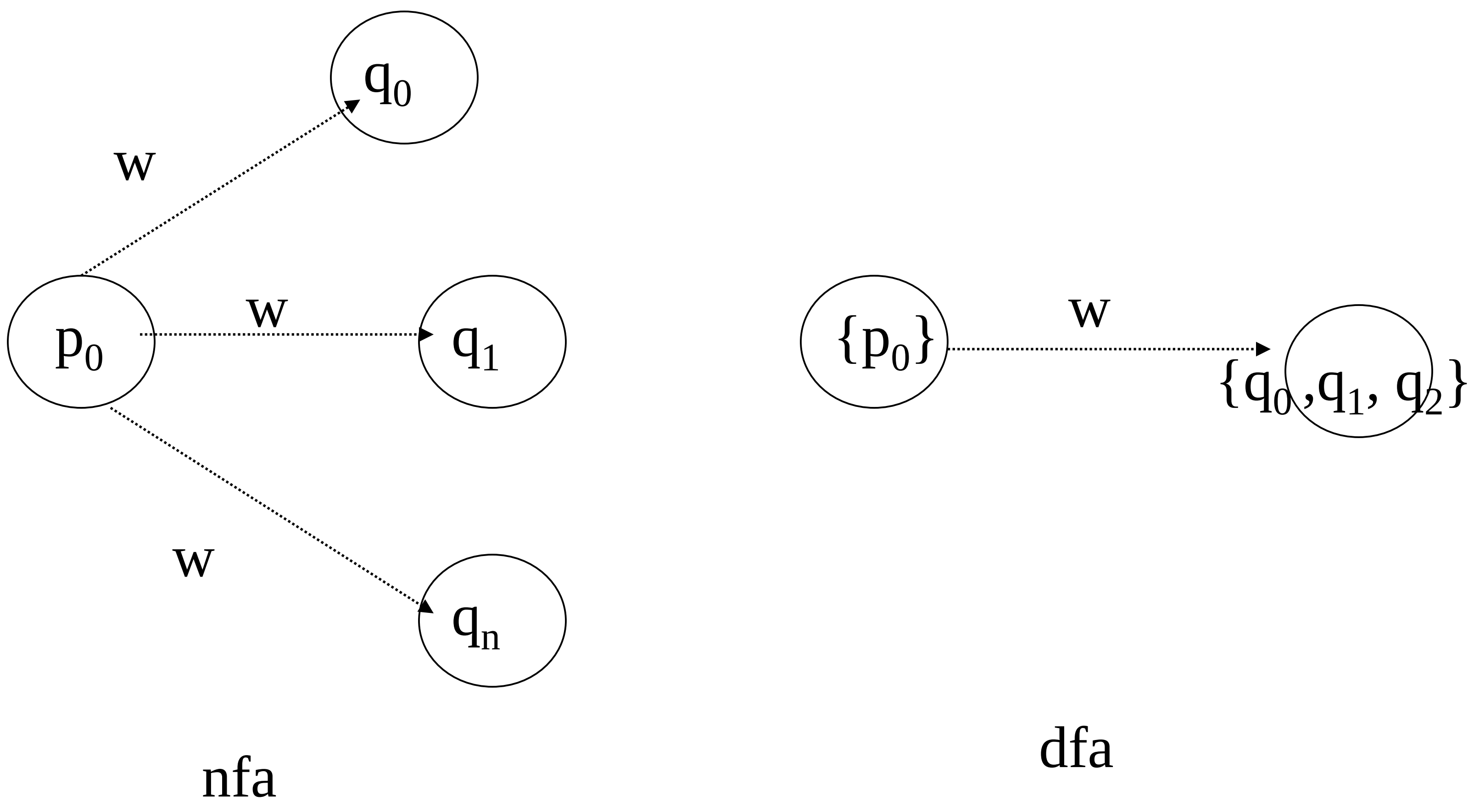
Automata Theory(SCS2112)

Dr Damitha D Karunaratna

- For every language accepted by some NFA there is a DFA that accepts the same language
  - How a NFA can be converted into an equivalent DFA ?
    - After an NFA has read a string w, it is in one state of a set of possible states.
    - After reading the same string by an equivalent DFA, it must be in some definite state.

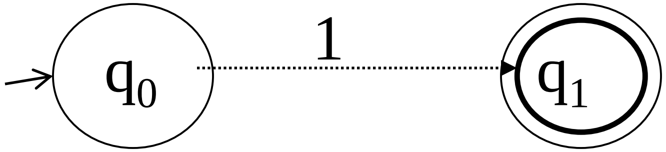
    How this conflict can be solved ?

Case1

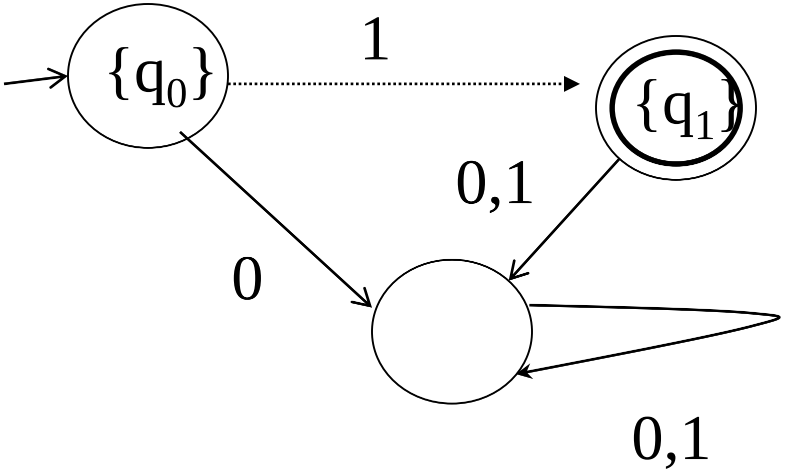


Case 2

Let $\sum = \{0,1\}$



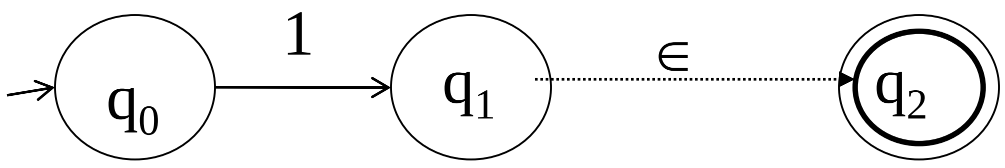
nfa



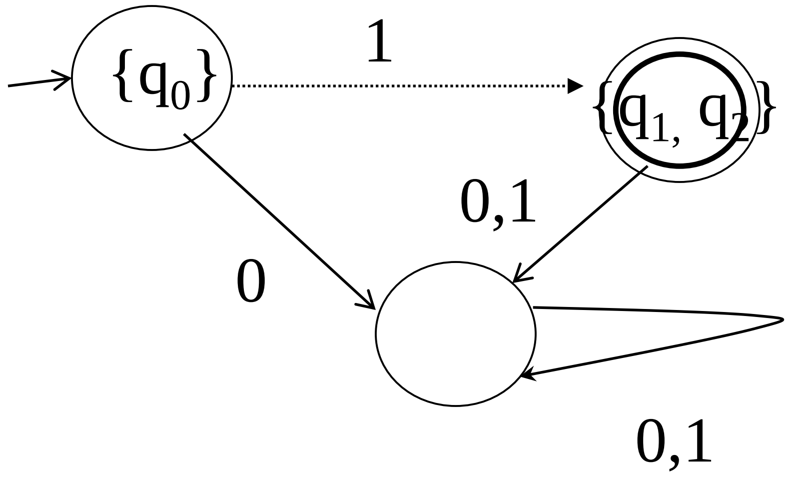
dfa

Case 3

Let $\sum = \{0,1\}$



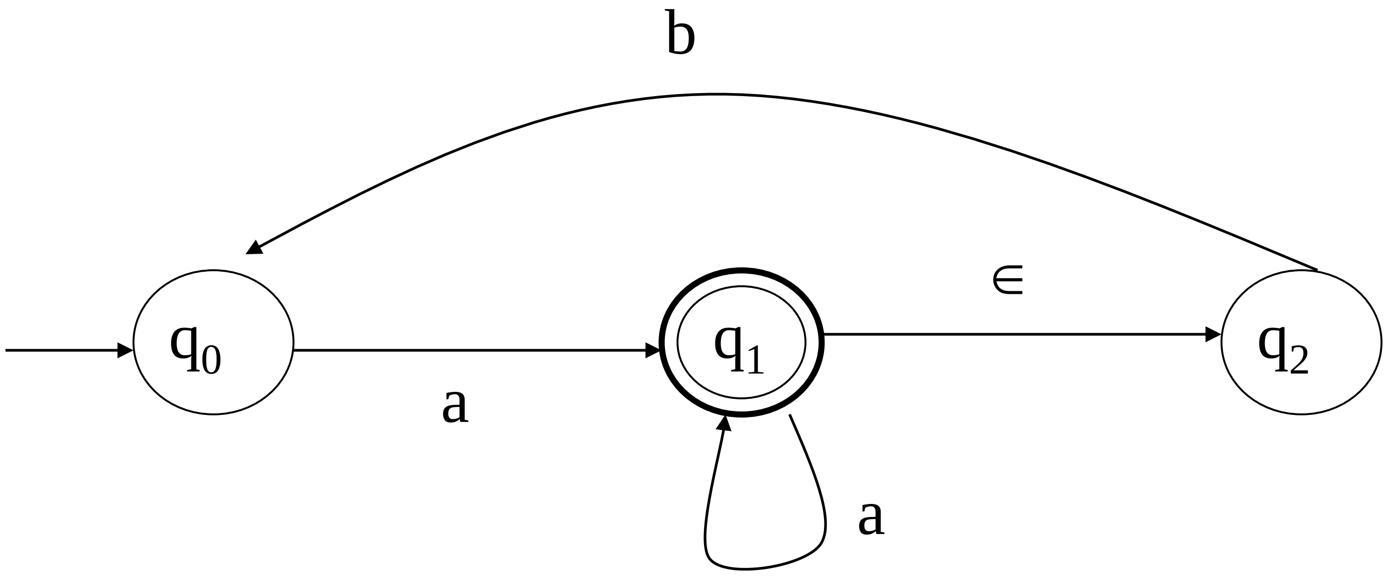
nfa

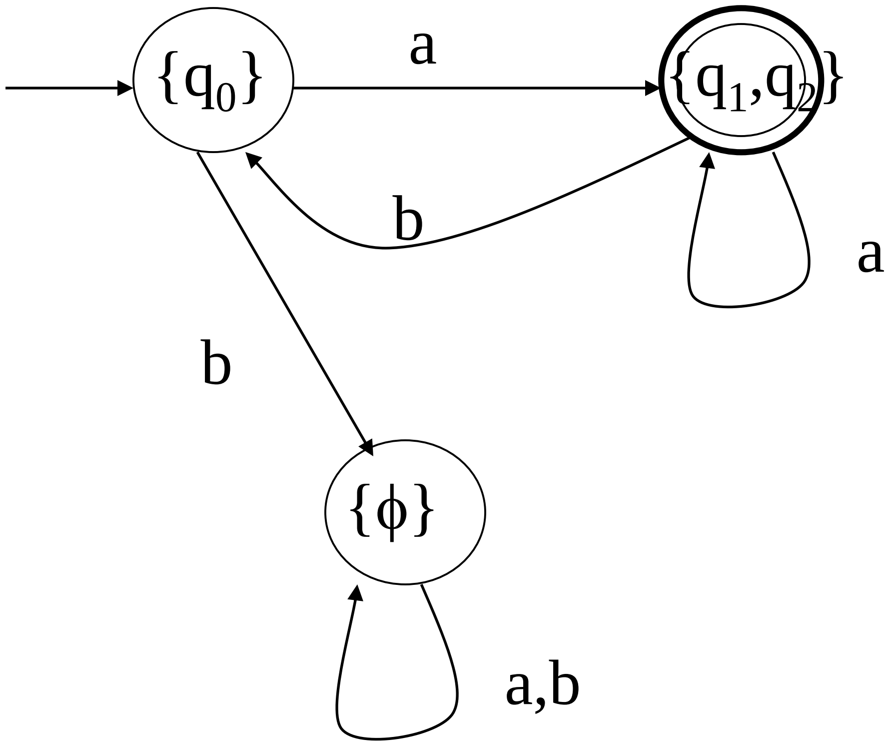


dfa

Label the states of the dfa with an appropriate set of states.

Example





- the state $\{q_1,q_2\}$ corresponds to two states of the nfa
- The state labeled $\{\phi\}$ represents an impossible move for the dfa and therefore means non acceptance of the string

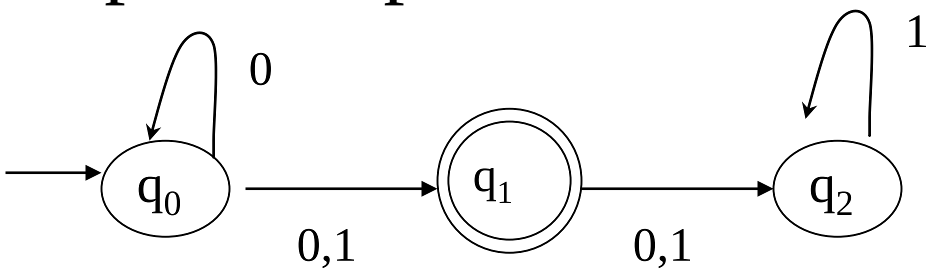
Conversion an NFA into DFA

- The basic idea behind the NFA-to-DFA construction is that each DFA state corresponds to a set of NFA states.
  - If A, B, C are states in the NFA then {A},{B},{C},{A,B},{A,C},{B,C},{A,B,C} are the possible states in the corresponding DFA
- If the number of states in the NFA is n, then the maximum number of states the DFA can have is 2^n.

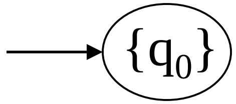
Theorem :

Let $L(M_N)$ be the language accepted by a nondeterministic finite accepter $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$. Then there exists a deterministic finite accepter $M_D = (Q_D, \Sigma, \delta_D, Q_0, F_D)$ such that $L(M_N) = L(M_D)$.

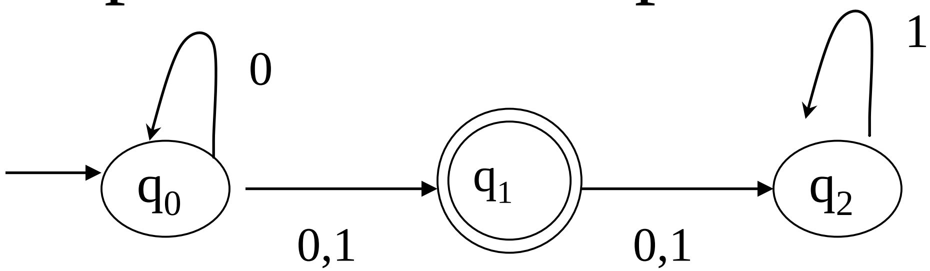
Example : Step 1

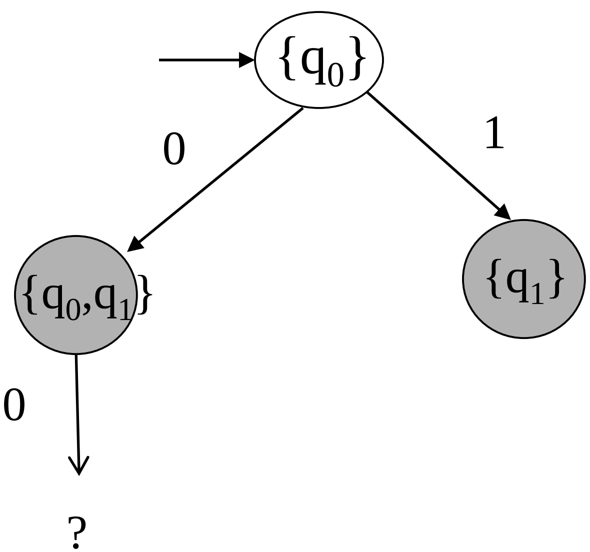


$\in$-closure(q_0) = {q_0}



Example : After n steps





1) $\delta(q_0,0) = \{q_{0,}\ q_1\}$

$\in$-closure$(\delta(q_0,0)\ )$

$= \in$-closure$\{q_{0,}\ q_1\}$

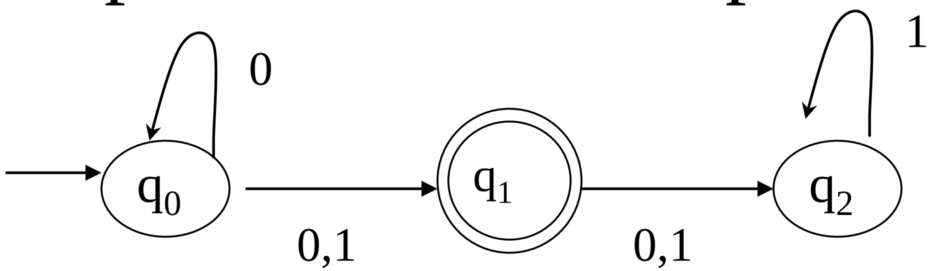
$= \{q_{0,}\ q_1\}$

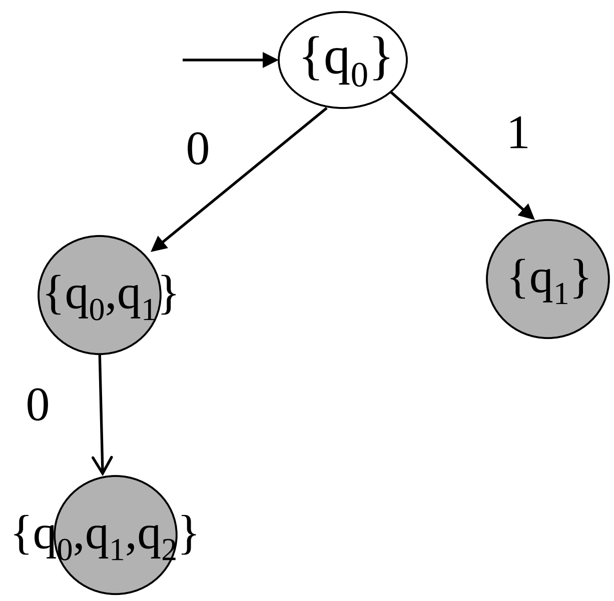
2)$\in$-closure$(\delta(q_1,0)\ )$

$= \{q_2\}$

1) $\cup$ 2) $= \{q_{0,}\ q_{1,}\ q_2\}$

Example : After n+1 step





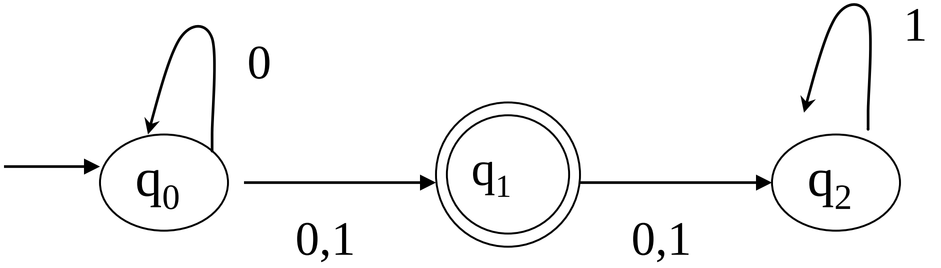
NFA to DFA algorithm

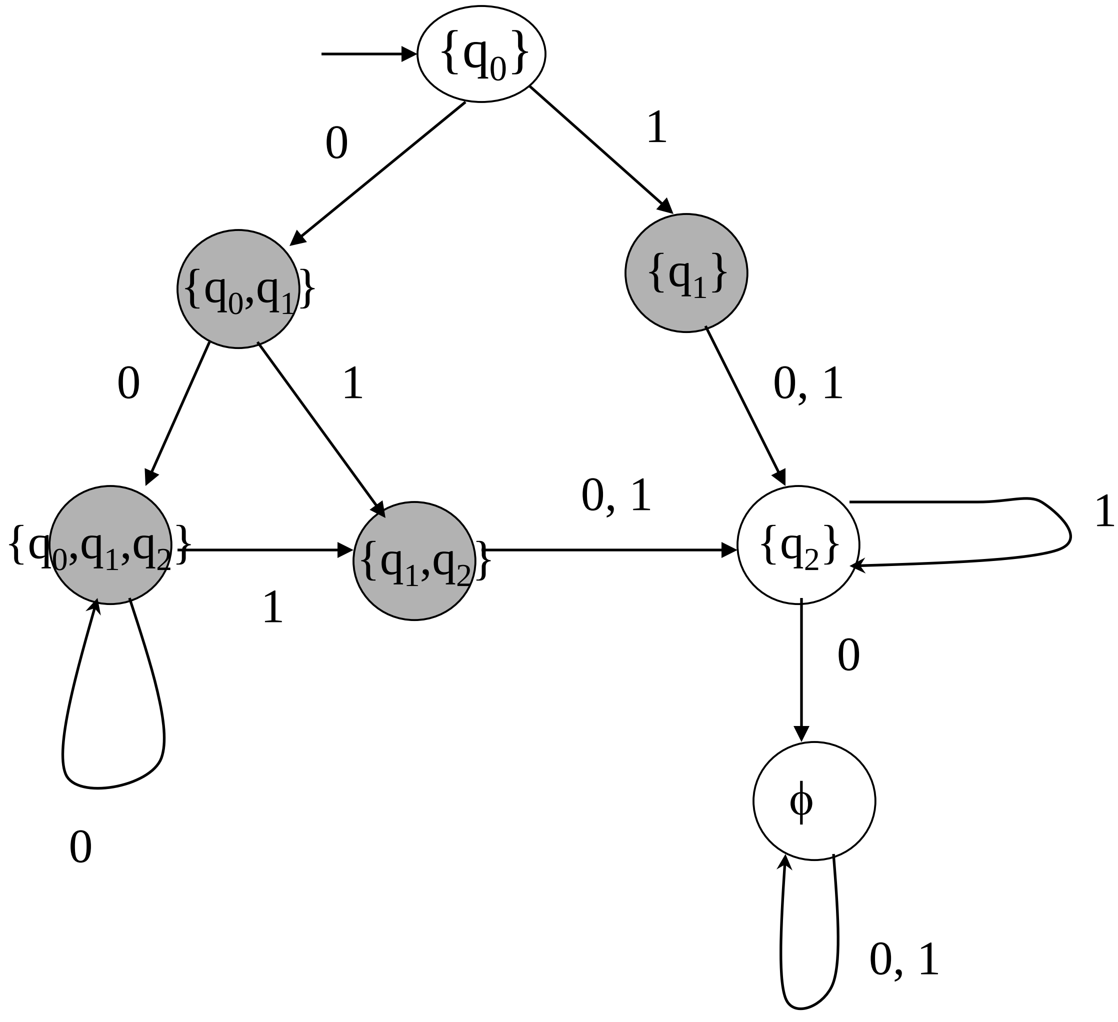
1. Create a graph G_D with vertex $\in$-closure(q_0) . Mark this vertex as the initial vertex.
2. Repeat the following steps for all edges.
   1. Take any vertex $\{q_i, q_j, \ldots, q_k\}$ of G_D that has no outgoing edge for some $a \in \Sigma$.
   2. Compute $\in$-closure($\delta(q_i,a)$) $\cup$ $\in$-closure($\delta(q_j,a)$) $\cup$ …. $\in$-closure($\delta(q_k,a)$)

      Let the result be the set $\{q_l, q_m, \ldots, q_t\}$
   3. Create a vertex for G_D labeled $\{q_l, q_m, \ldots, q_t\}$ if it does not already exist.
   4. Add to G_D an edge from $\{q_i, q_j, \ldots, q_k\}$ to $\{q_l, q_m, \ldots, q_t\}$ with a label a.
3. Every state of G_D whose label contains any $q_f \in F_N$ is identified as a final vertex.

- The G_D has at most $2^{|Q_n|}| \sum|$ edges. Thus the NFA to DFA algorithm should always terminates at some point.

Example : Convert the following nfa to an equivalent dfa.

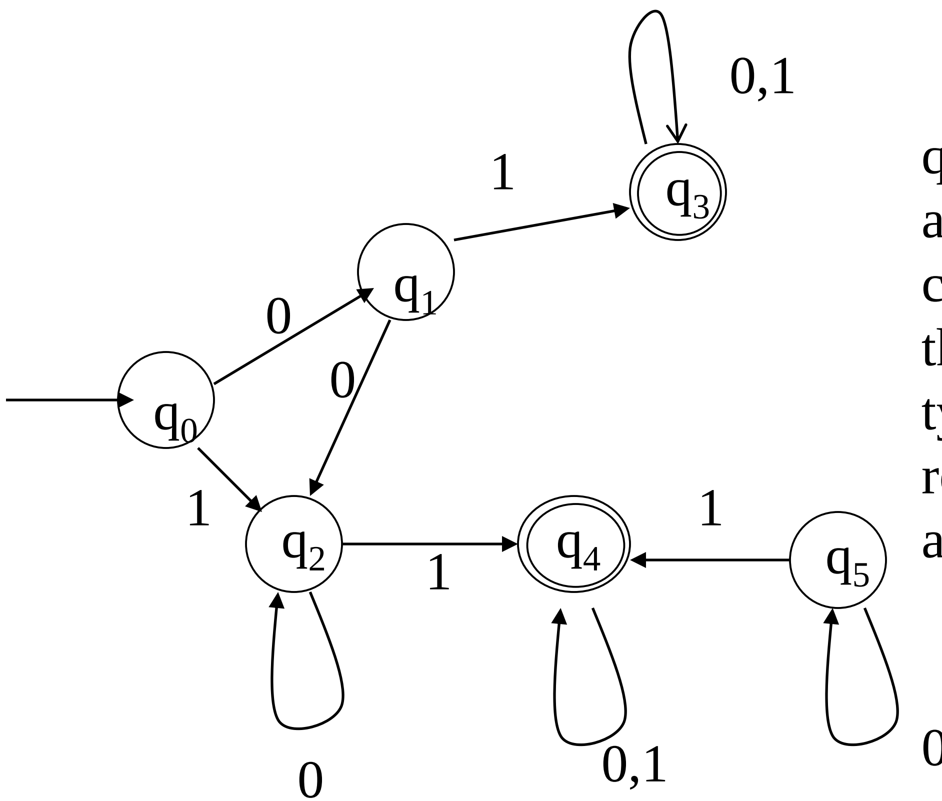



{q0}
0
1
{q0,q1}
{q1}
0
1
0, 1
0, 1
1
{q0,q1,q2}
{q1,q2}
{q2}
1
0
ϕ
0
0, 1

Reduction of the number of states in Finite Automata

For storage efficiency, it is desirable to reduce the number of states as far as possible.

- The process will
  - Remove states that cannot be reached from the start state.
  - recognize and merge equivalent states.



q_5 plays no role in the automata since it cannot be reached from the initial state. These types of states can be removed without affecting the automata.

Reduction of the number of states in Finite Automata …

How to find equivalent states?

- Assume that all states are equivalent and then separate states only if they can be proved as different.

Indistinguishable States

- In an automata there may be states that can be considered as equivalent with respect to the outcome of the automata.
  – **indistinguishable (equivalent) states.**

- How to identify such states?

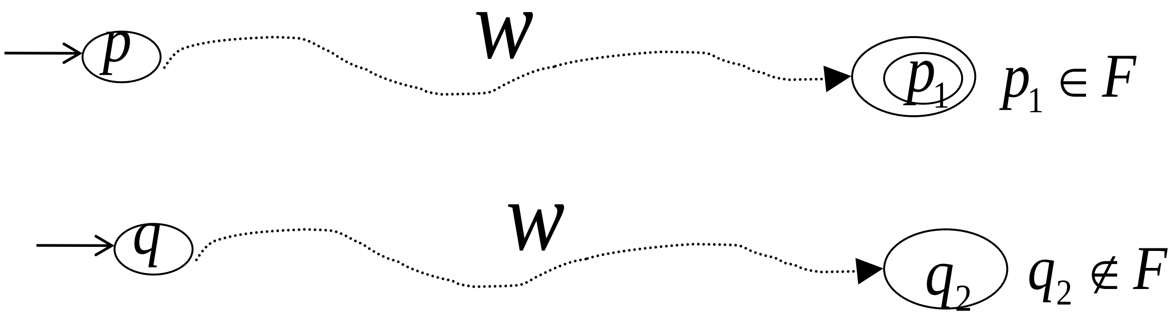
Indistinguishable (Equivalent) states

- Definition : Two states p and q of a dfa are called **indistinguishable** (equivalent) iff

$$\forall w,\ \delta^*(p,w) \in F \Leftrightarrow \delta^*(q,w) \in F$$

This means that the machine does the same thing (with respect to all possible strings) when started in either state.

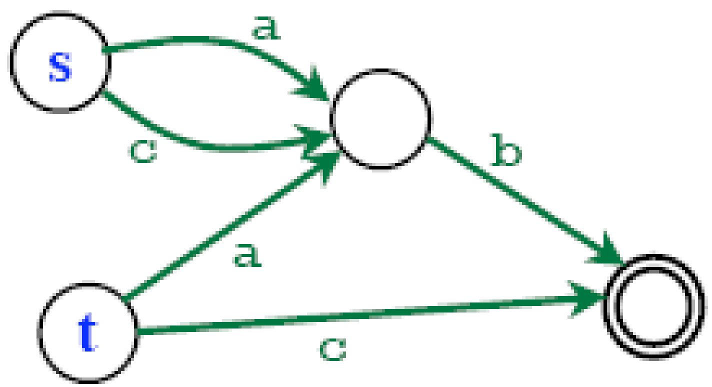
- If there exists some string $w \in \Sigma^*$ such that $\delta^*(p,w) \in F$ and $\delta^*(q,w) \notin F$ or vice versa, then the states p and q are said to be **distinguishable** by a string w.



-

- If two states are *different* then they cannot be merged.

Example:



"ab" does not distinguish s and t.
But "c" distinguishes s and t.
Therefore, s and t cannot be merged.

s and t are **distinguishable**

- Non-distinguishability is an equivalence relation.
  - If p and q are non-distinguishable and q and r are non-distinguishable, then p and r are also non-distinguishable
    - all three states are non-distinguishable.

Reduction of the number of states in Finite Automata …

The following rules can be used for this:

- An accepting state is not equivalent to a non-accepting state (**distinguishable)**.
- If two states s1 and s2 have transitions on the same symbol c to states t1 and t2 that we have already proven to be different, then s1 and s2 are different. This also applies if only one of s1 or s2 have a defined transition on c.

DFA Minimization Algorithm

- Given a DFA D over the alphabet Σ with states Q where $F \subseteq Q$ is the set of the accepting states, the construction of minimal DFA D' where each state is of D' is a group of states from D. The groups in the minimal DFA are consistent: For any pair of states s1, s2 in the same group G and any symbol c in Σ, move(s1, c) is in the same group G' as move(s2, c) or both are undefined.

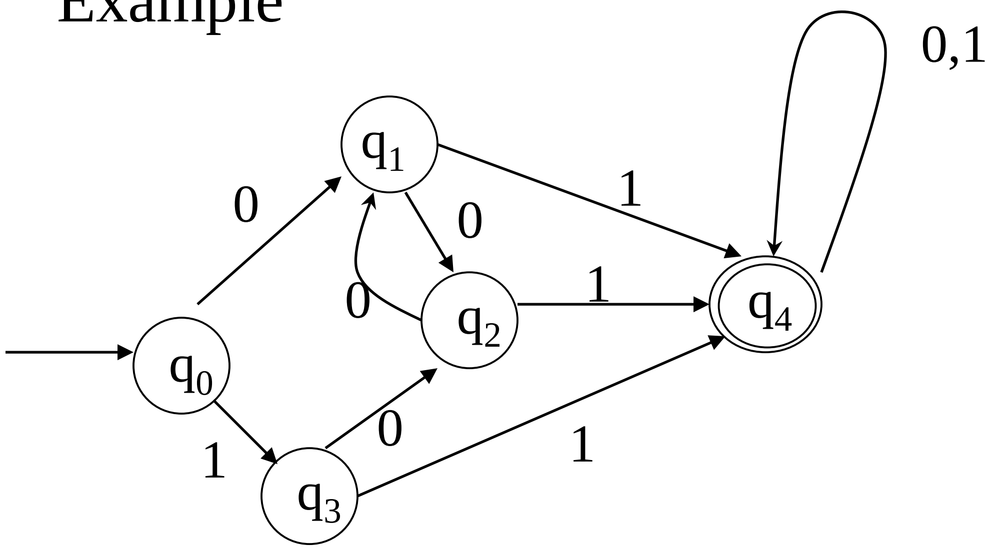
DFA Minimization Algorithm …

- Start with two groups: F and Q \ F. Take them as unmarked.
- Pick any unmarked group G and check if it is consistent.
  - If it is consistent, remove it from the process (mark it). If it is not consistent, we split it into maximal consistent subgroups and replace G by these subgroup. Take all these sub-groups as unmarked.
- If there are no unmarked groups left, terminate the process and the remaining groups are the states of the minimal DFA. Otherwise, go back to the previous step.

DFA Minimization Algorithm …

- The starting state of the minimal DFA is the group that contains the original starting state and any group of accepting states is an accepting state in the minimal DFA.
- A group that consists of a single state need never be split, and the process can stop when all unmarked groups are singletons.

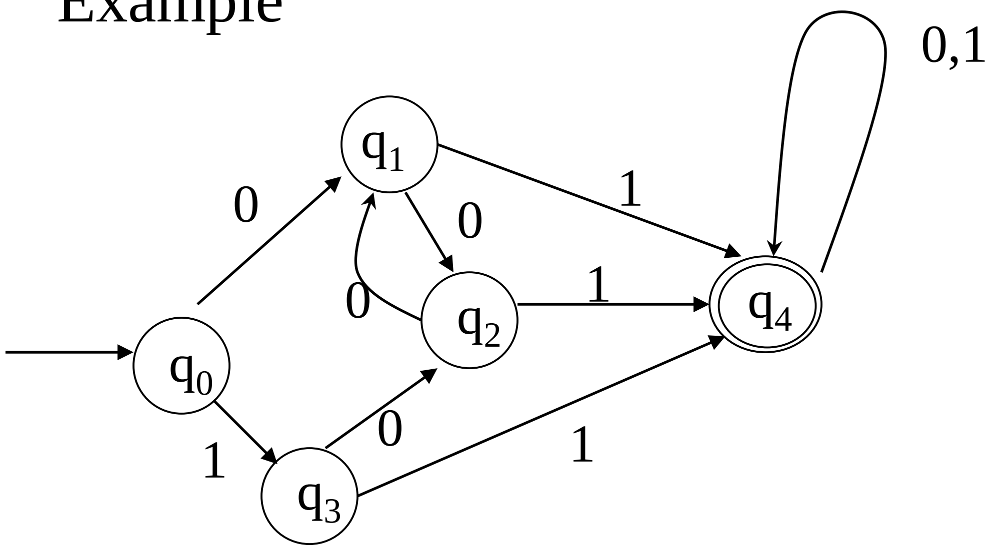
Example



Initial Groups G1 = $\{q_4\}$ and G2 = $\{q_0, q_1, q_2, q_3\}$

Remove Group G1 from the process as it is singleton.

Example



$G1 = \{q_4\}$

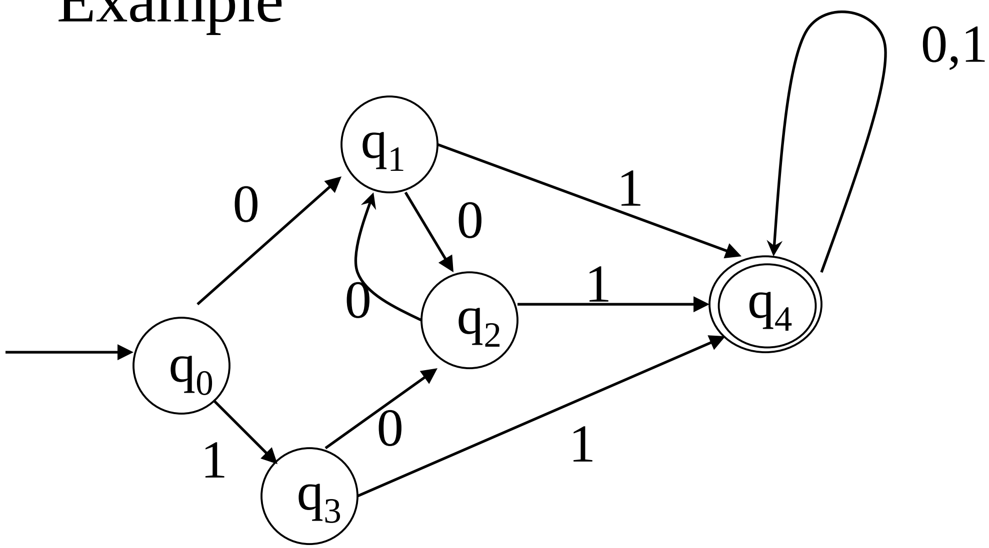
$G2 = \{q_0, q_1, q_2, q_3\}$ replace this with the new Group G2

$G2 = \{q_0\}$

$G3 = \{q_1, q_2, q_3\}$

G2	0	1
q_0	G2	G2
q_1	G2	G1
q_2	G2	G1
q_3	G2	G1

Example



$G1 = \{q_4\}$

$G2 = \{q_0\}$

$G3 = \{q_1, q_2, q_3\}$ consistent

G2	0	1
q_1	G3	G1
q_2	G3	G1
q_3	G3	G1

Reducing the number of states in a DFA

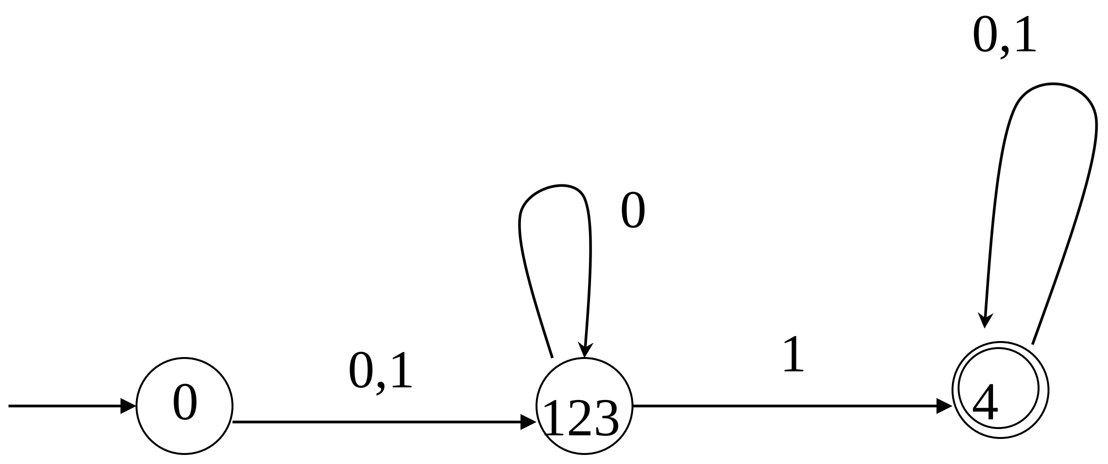
Given a DFA $M = (Q, \sum, \delta, q_0, F)$, a reduced DFA $M' = (Q', \sum, \delta', q_0', F')$ can be constructed as follows.

Produce : Reduce

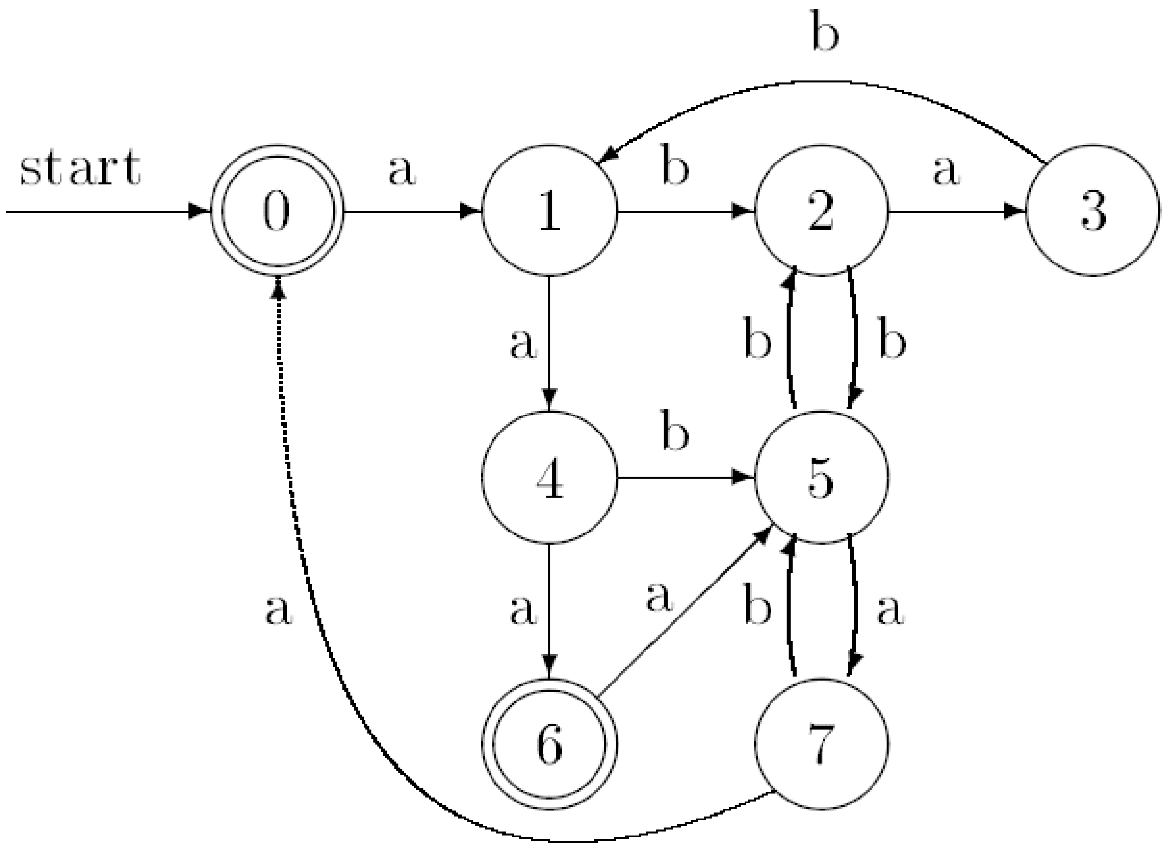
1. Use mark to find all pairs of distinguishable states. Then by using the result partition the set of states into a set of mutually disjoint states.
2. For each set $G_i = \{q_i, q_j, \ldots, q_k\}$ of non-distinguishable states, create a state labeled ij…k (or G_i) in the final automata.
3. For each transition rule $\delta(q_r, a) = q_p$ in the original automata find the sets to which q_r and q_p belong. If $q_r \in \{q_i, q_j, \ldots, q_k\} = G_s$ and $q_p \in \{q_l, q_m, \ldots, q_n\} = G_t$ then add a rule $\delta'(ij\ldots k, a) = lm..n$ (or $\delta'(G_s) = G_t$)to the final automata.
4. The initial state q’ is that state of M’ whose label includes 0.
5. F’ is the set of all states whose label contains i such that $q_i \in F$

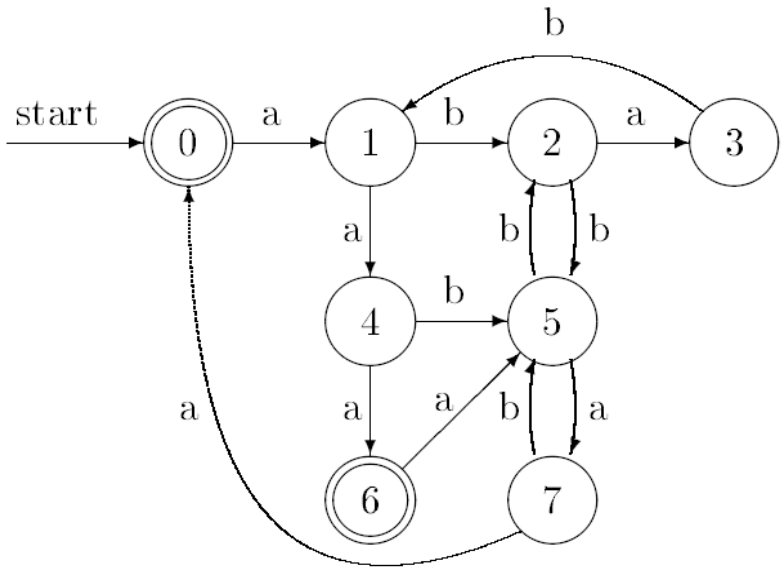
This result partitions the set of states to the subsets $\{q_0\},\{q_1,q_2,q_3\},\{q_4\}$.

The reduced DFA



Example





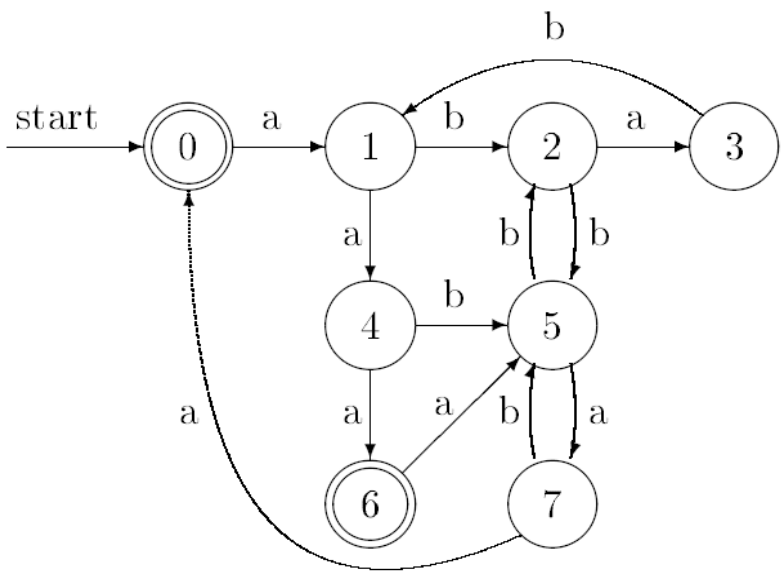
G1 = {0,6}

G2 = {1,2,3,4,5,7}

G1	a	b
0	G2	-
6	G2	-

G1 Consistent remove it from the process

G2	a	b
1	G2	G2
2	G2	G2
3	-	G2
4	G1	G2
5	G2	G2
7	G1	G2



G3	a	b
1	G5	G3
2	G4	G3
5	G5	G3

G1 = {0,6}

G3 = {1,2,5}

G4 = { 3}

G5 = {4,7}

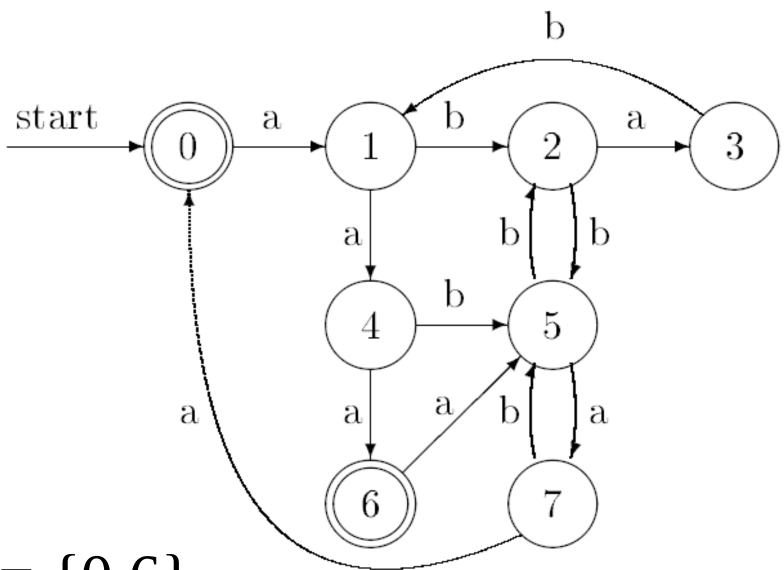
G1 = {0,6}

G6 = {1,5}

G7 = {2}

G4 = { 3}

G5 = {4,7}



G1 = {0,6}

G6 = {1,5}

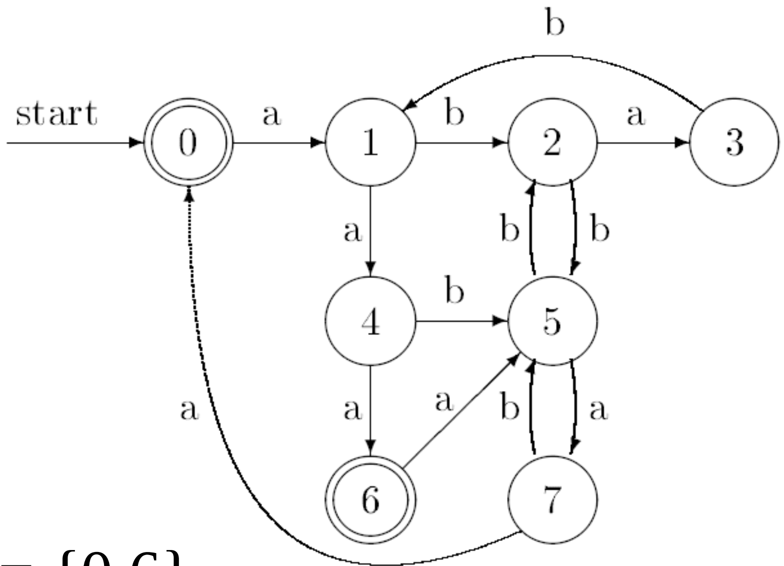
G7 = {2}

G4 = { 3}

G5 = {4,7}

G5	a	b
4	G1	G6
7	G1	G6

G6	a	b
1	G5	G7
5	G5	G7



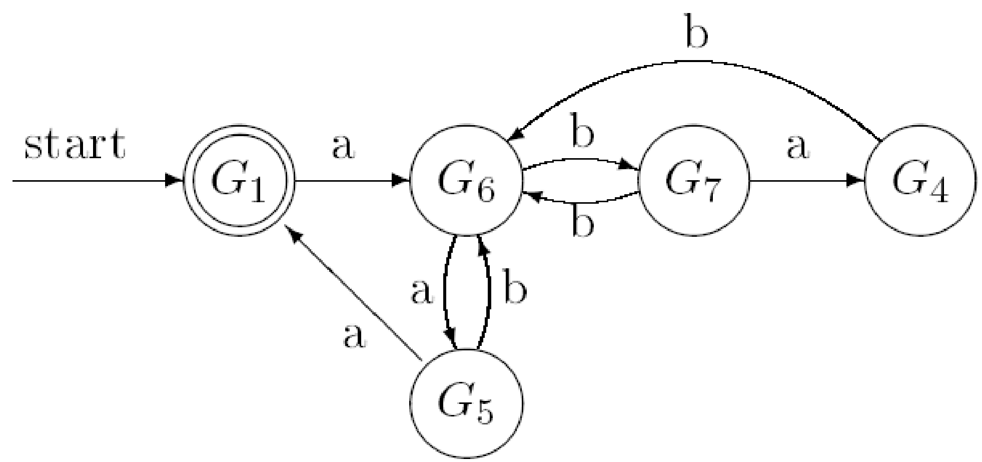
G1 = {0,6}

G6 = {1,5}

G7 = {2}

G4 = { 3}

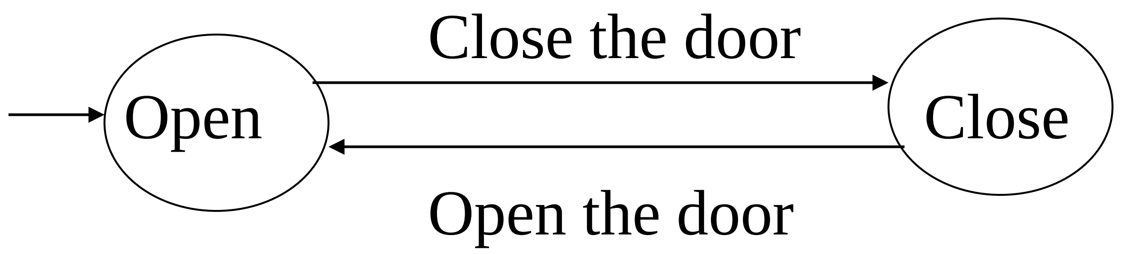
G5 = {4,7}



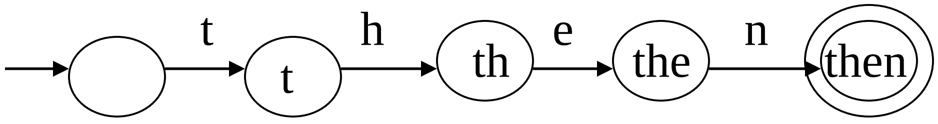
Minimal DFA

- For any given DFA there is a unique minimal DFA.
- Equivalence of DFAs can be checked by converting the DFAs to minimal DFAs and comparing the results.

Example : Finite state machine modeling the states of a door.



Example : Finite state machine recognizing the string **then**.



JFLAP

- **JFLAP** is a package of graphical tools which can be used as an aid in learning the basic concepts of Formal Languages and Automata Theory. (**http://www.cs.duke.edu/csed/jflap/**)
- Slight difference in the definition of DFA

$$\delta : D \rightarrow 2^Q$$

where $D \subseteq Q \times \Sigma^*$