Chapter 2

Finite Automata

A Finite-State Machine

- **Finite automata** – is a machine which takes and reads an input string one by one and then after the input is completely read, it decides to whether the string is **accepted** or **not accepted**.
- The automaton consists of **states** and **transitions.**
- As the automaton sees a symbol of input it makes a transition or jump to another state according to its **transition function**.

- A **finite state machine** is a mathematical model of a system, with **discrete inputs** and **outputs**.

A Finite-State Machine…

- **Finite Automata FA:**
  - a finite set of states
  - a set of transitions (edges)
  - a start state
  - a set of final states

Defining a FA is a kind of programming.

- Problem definition
  - Includes defining possible actions & accepting condition.
- States ≈ structure of program
  - Includes designating which are <u>initial</u> & <u>final</u>.
- Transitions ≈ program

Types Finite Automata

Finite Automata : **A recognizer that takes an input string & determines whether it's a valid sentence of the language.**

Deterministic : **Has at most one action for every given input symbol.**

Non-Deterministic : **Has more than one (or no) alternative action for the same input symbol.**

Deterministic Finite Accepters (dfa)

Definition

A deterministic finite accepter or *dfa*

is defined by the quintuple

$$M = (Q, \Sigma, \delta, q_0, F)$$

where

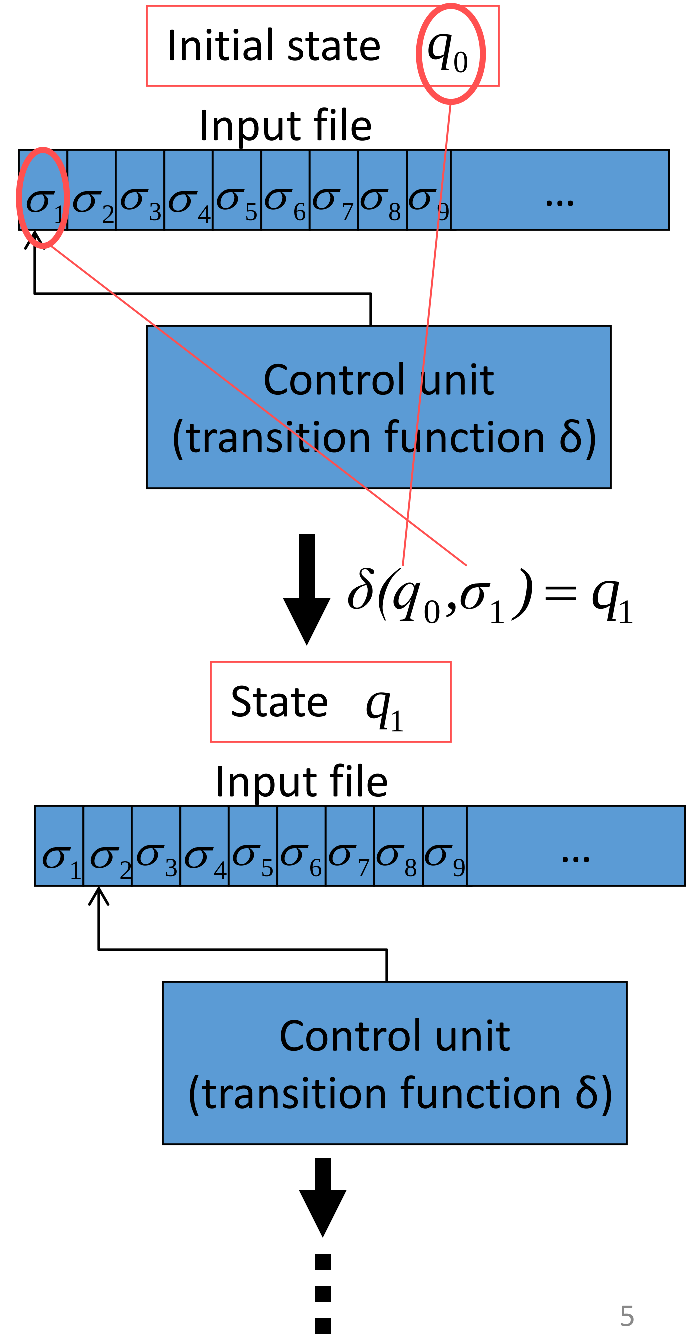
Q is a finite set of internal states,

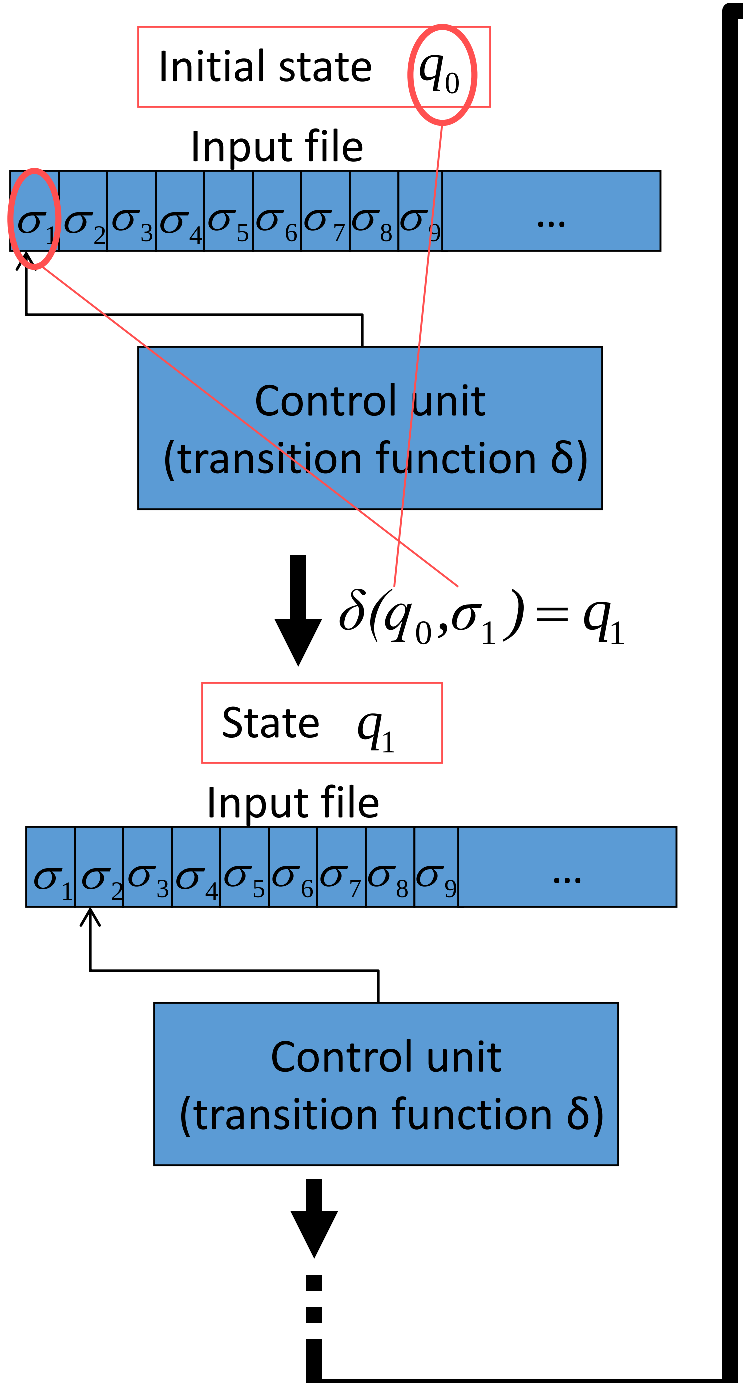
Σ is a finite set of symbols called the input alphabet,

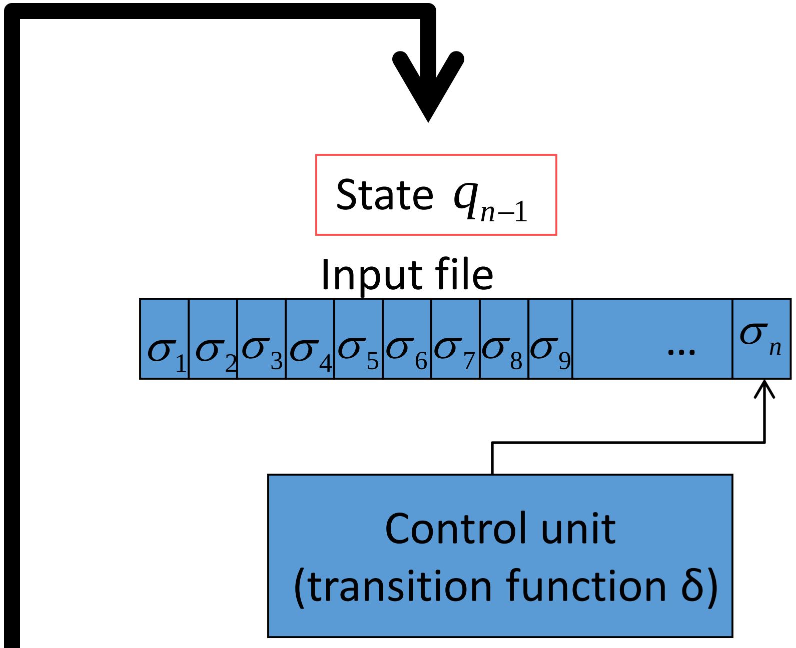
$\delta : Q \times \Sigma \rightarrow Q$ is a function called the transition function,

$q_0 \in Q$ is the initial state,

$F \subseteq Q$ is a set of final states.







If the end of string is reached and $\delta(q_{n-1},\sigma_n) = q_n$ is a final state, then the string is accepted. Otherwise, the string is rejected.

Deterministic Finite Automata

- DFA M = (Q, Σ, δ, q_0, F)
  - Q = a finite set of states
  - Σ = a finite set called the *alphabet*
  - δ = *transition function*
    - total function $Q \times \Sigma \rightarrow Q$
  - q_0 = start state $q_0 \in Q$
  - F = *final* or *accepting* states $F \subseteq Q$

Deterministic Finite Automata…

- DFA M
  - $Q = \{\mathbf{q_0}, q_1\}$
  - $\Sigma = \{a, b\}$
  - $F = \{q_1\}$
  - The transition function δ is given in a tabular form called the **transition table**
  - $\delta(q_0, \mathbf{a}) = q_1$ $\delta(q_0, \mathbf{b}) = q_0$
  - $\delta(q_1, \mathbf{a}) = q_1$ $\delta(q_1, \mathbf{b}) = q_0$

Deterministic Finite Automata…

- A **DFA** M can be considered to be a language acceptor.

- The **language** of M, **L(M)**, is the set of strings Σ^* accepted by M.

- A **DFA** M reads an input string from left to right.

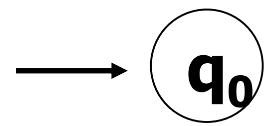
- The **next state** depends on the **current state** and the **unread (unprocessed) symbol**.

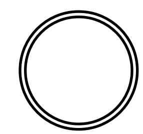
Deterministic Finite Automata…

- The DFA **M** accepts the set of strings over {a, b} that contain the substring **bb.**
  - M : $Q = \{\mathbf{q_0}, q_1, q_2\}$, $\Sigma = \{a, b\}$, $F = \{q_2\}$
  - The transition function **δ** is given in a tabular form called the **transition table.**
  - $\delta(q_0, \mathbf{a}) = q_0$ $\delta(q_0, \mathbf{b}) = q_1$
  - $\delta(q_1, \mathbf{a}) = q_0$ $\delta(q_1, \mathbf{b}) = q_2$
  - $\delta(q_2, \mathbf{a}) = q_2$ $\delta(q_2, \mathbf{b}) = q_2$
- Is **abba** $\in$ L(M)? Yes, since the computation halts in state q_2, which is a final state.
- Is **abab** $\in$ L(M)? No, since the computation halts in state q_1, which is **NOT** a final state.

State Diagrams and Examples

- The state diagram of a DFA $M = (Q, \Sigma, \delta, q_0, F)$ is a labeled graph **G** defined by the following conditions:
  - The nodes of **G** are the elements of **Q**
  - The labels on the arcs of **G** are elements of $\boldsymbol{\Sigma}$

    $\xrightarrow{\mathbf{a}}$
  - $\mathbf{q_0}$ is the start node, denoted by: 

  - **F** is the set of accepting nodes, denoted by: 
  - There is an arc from node $\mathbf{q_i}$ to $\mathbf{q_j}$ labeled $\boldsymbol{a}$ if $\delta(\mathbf{q_i}, \boldsymbol{a}) = \mathbf{q_j}$
  - For every node $\mathbf{q_i}$ and symbol $\boldsymbol{a} \in \Sigma$, there is exactly **one** arc labeled $\boldsymbol{a}$ leaving $\mathbf{q_i}$

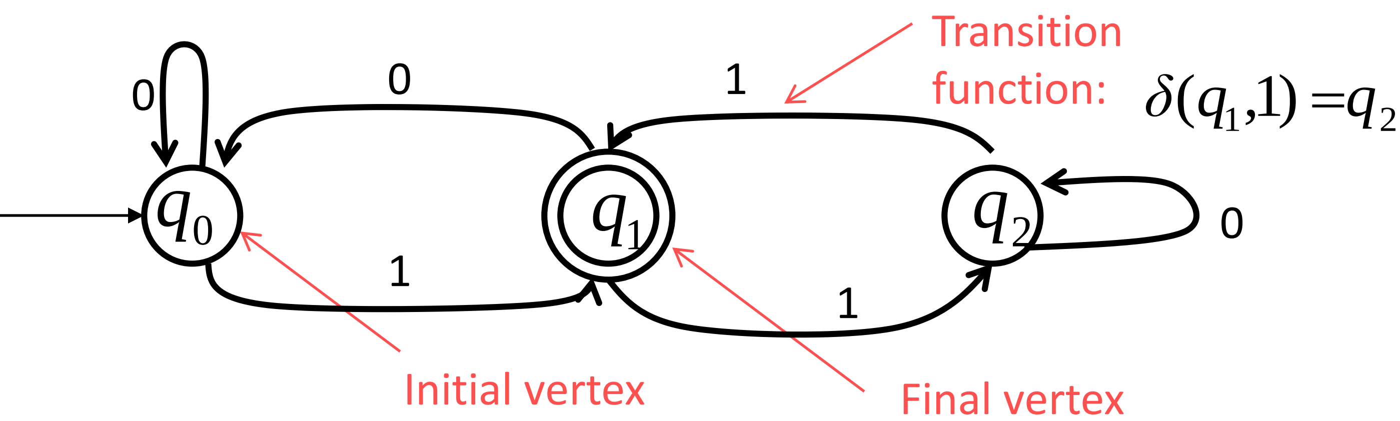
State Diagrams and Examples…

- **Deterministic Finite Automata DFA**
  - all outgoing edges are labelled with an input character
    - no state has **ε- transition, transition on input ε**
  - *no* two edges leaving a given state have the *same* label
    - for each state *s* and input symbol *a*, there is at most one edge label *a* leaving *s.*
  - **Therefore**: the next state can be *determined* uniquely, given the current state and the current input character.

- **Transition graphs**

Deterministic finite accepter

$M = (Q, \Sigma, \delta, q_0, F)$



Example

The above transition graph represents the dfa

$M = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\}),$

$where\delta$ is given by

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

$\delta(q_1, 0) = q_{0,}\ \ \delta(q_1, 1) = q_{2,}$

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

It accepts 01, 101,0111,11001,….

But not 00, 100,1100,….

State Diagrams and Examples…

Definition

The language accepted by

a dfa $M = (Q,\Sigma,\delta,q_0 F)$ is

the set of all string on Σ

accepted by M. In formal notation,

$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}.$$

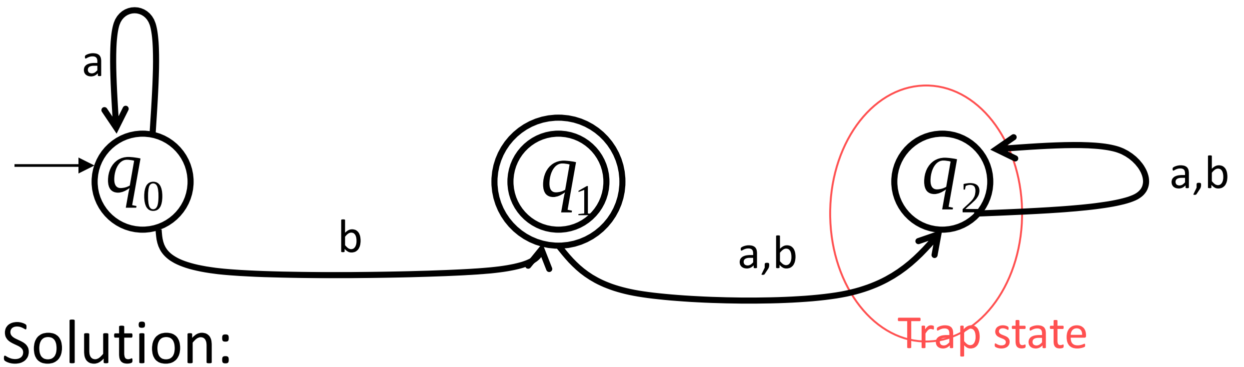
Let $w = a_1 a_2 \cdots a_n$.

If $\delta(\mathrm{q}_1, a_1) = q_2, \delta(q_2, a_2) = q_3, \cdots, \delta(q_n, a_n) = q_{n+1}$,

then we write $\delta^*(q_1, w) = q_{n+1}$.

Especially, $\delta^*(q,\lambda) = q$.

Example: Consider the dfa in the following transition graph.

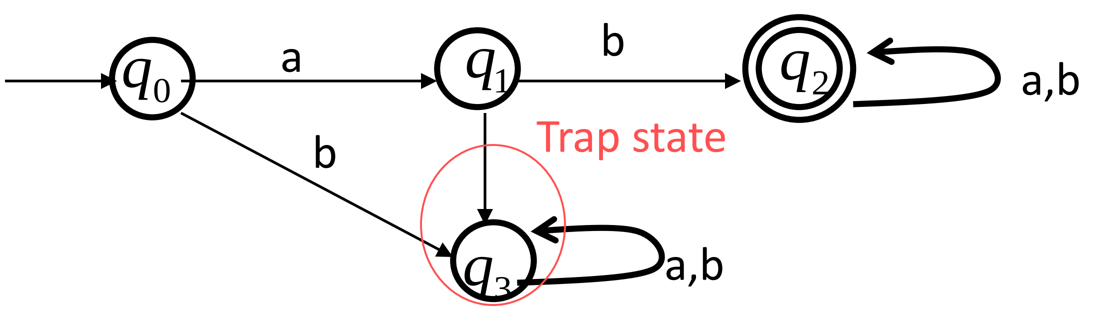


Solution:

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

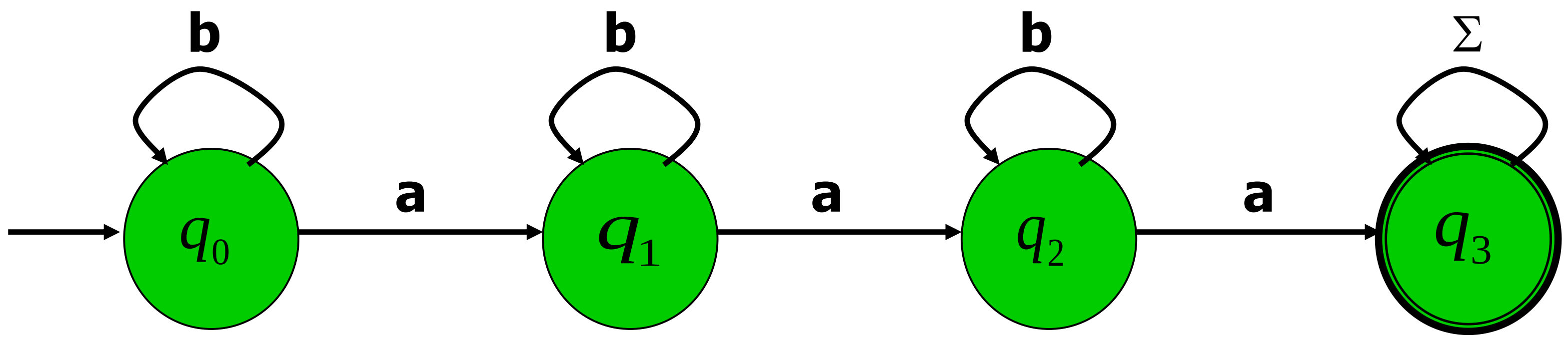
Example: Find a dfa that recognizes the set of all strings on $\Sigma = \{a, b\}$ starting with the prefix ab.

Solution:



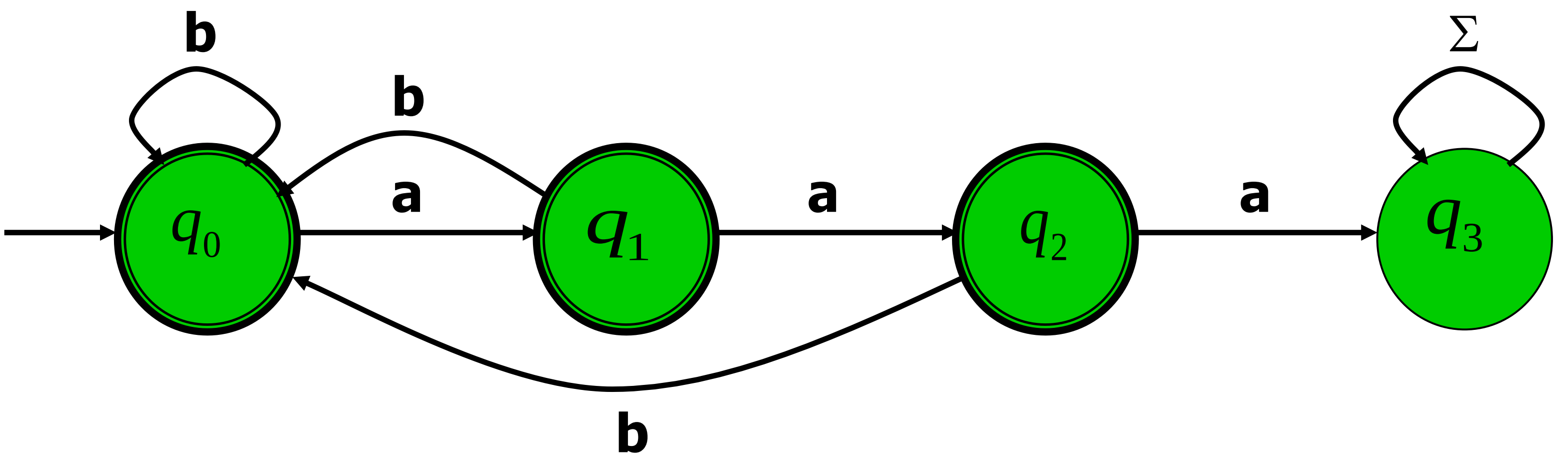
State Diagrams and Examples…

- strings over {**a**,**b**} with at **least 3 a**’s



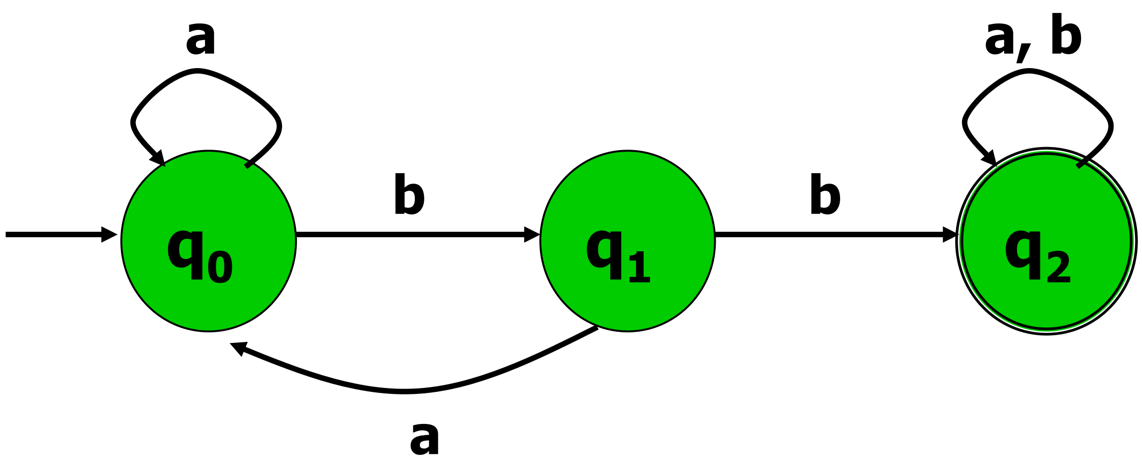
State Diagrams and Examples….

- strings over {**a**,**b**} ***without*** **3** consecutive **a**'s



State Diagrams and Examples…

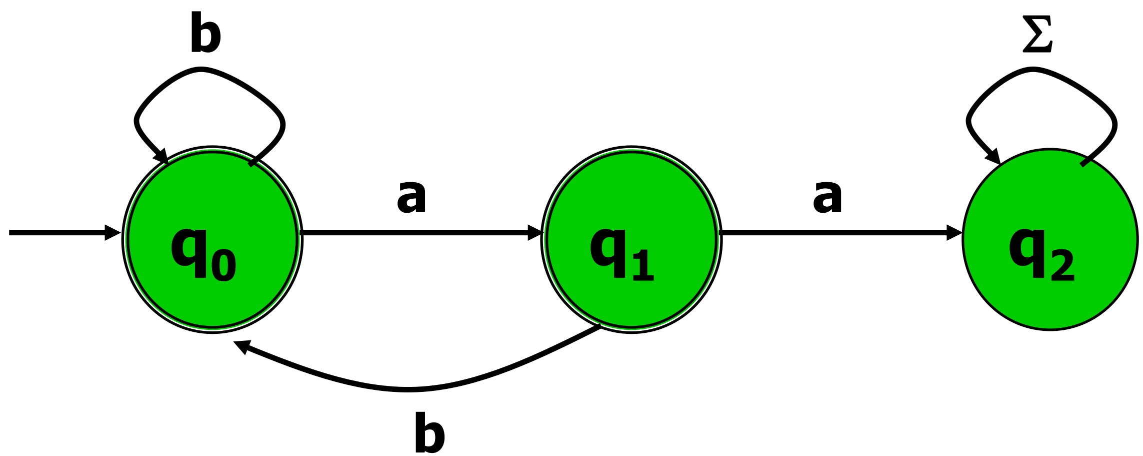
- Draw a state diagram for DFA **M** that accepts the set of strings over {a, b} that contain the substring **bb**



- The string **ababb** is accepted since the halting state is the accepting state $\mathbf{q_2}$

State Diagrams and Examples…

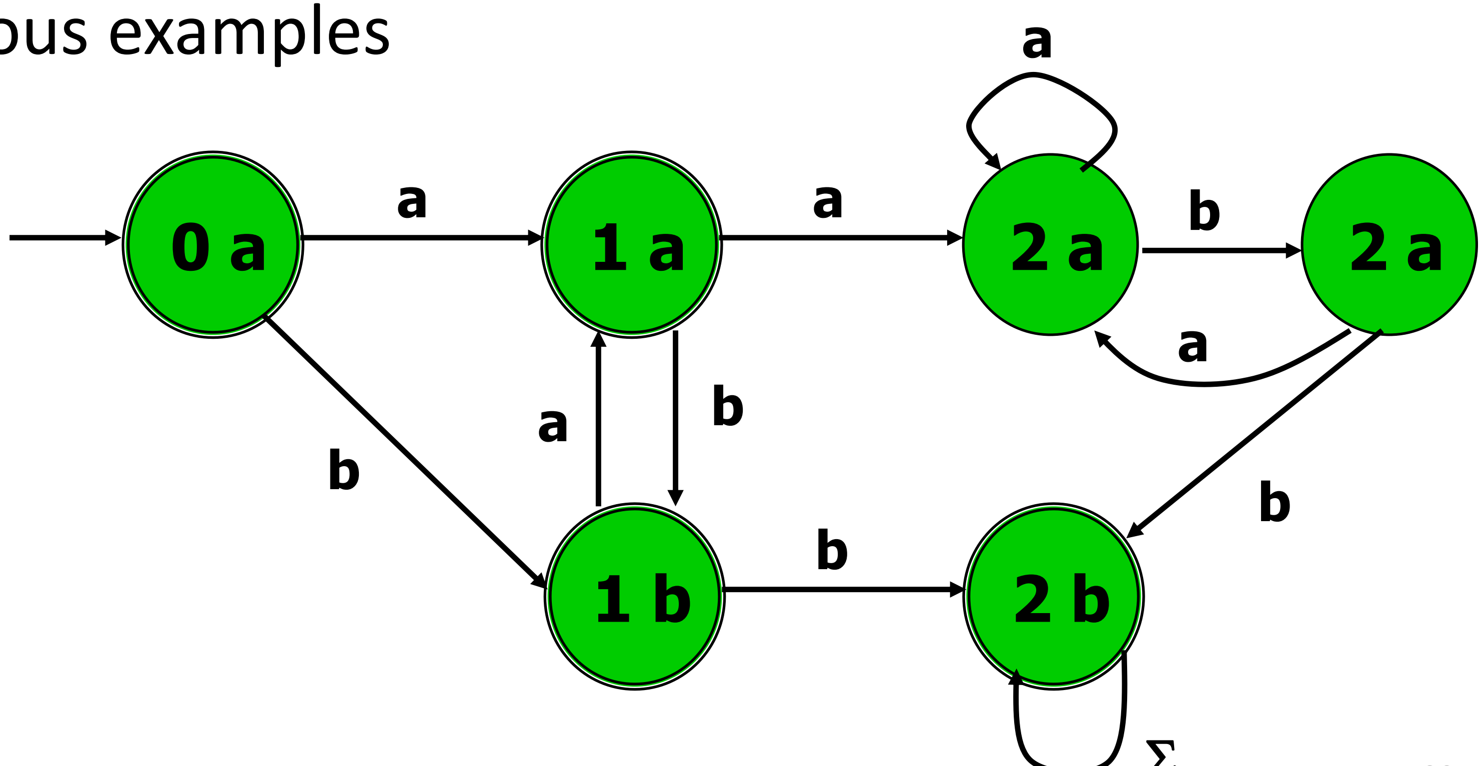
- The DFA



- accepts $(\mathbf{b}|\mathbf{ab})^*(\mathbf{a}|\varepsilon)$
- the set of strings over {a, b} that **do not** contain the substring **aa**

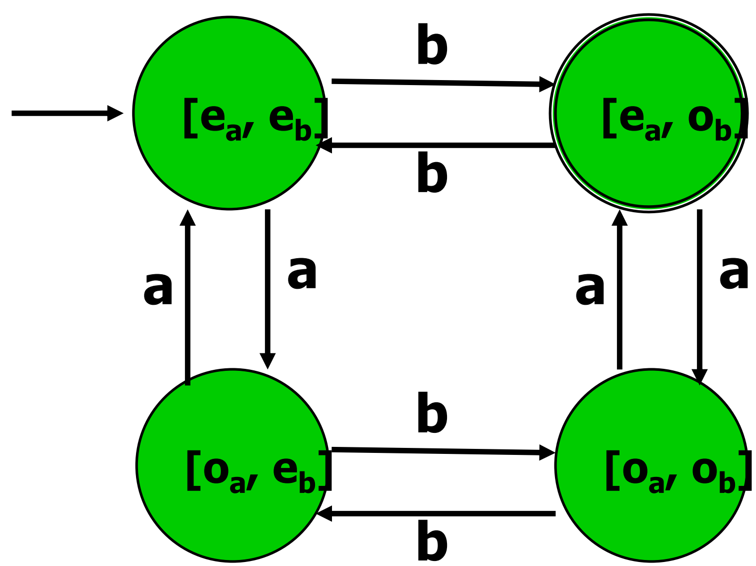
State Diagrams and Examples…

- strings over {a, b} that contain the substring **bb** <u>OR</u> **do not** contain the substring **aa**
- This language is the union of the languages of the previous examples



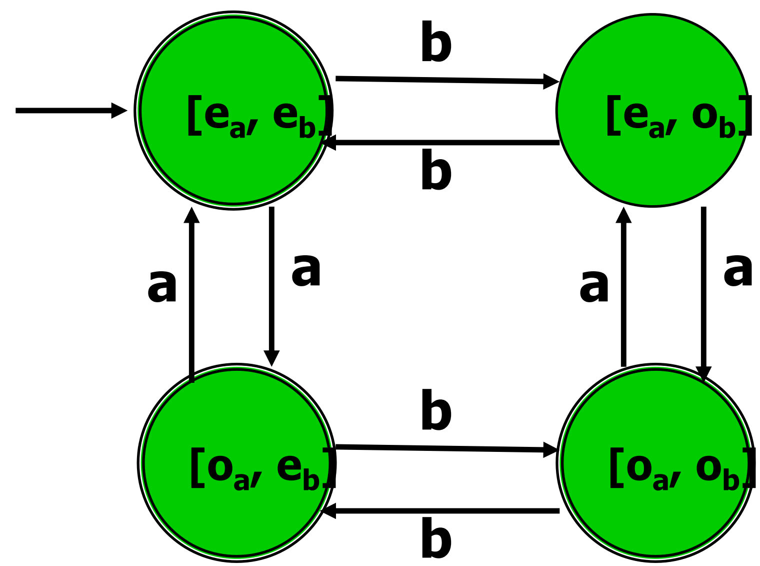
State Diagrams and Examples…

- strings over {a, b} that contain an **even** number of **a**'s **AND** an **odd** number of **b**'s



State Diagrams and Examples…

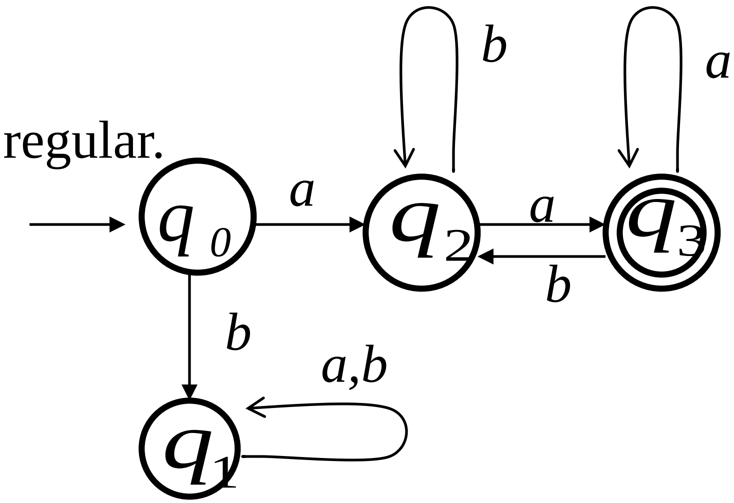
- Let **M** be the DFA previous slide
- A DFA **M’** that accepts all strings over {a, b} that **do not** contain an **even** number of **a**’s **AND** an **odd** number of **b**’s is shown below
  - L(**M’**) = {**a**, **b**}* - L(**M**) = Σ^* - L(**M**)
- Any string accepted by **M** is rejected by **M’** and vice versa



Definition: A language *L* is called regular if and only if there exists some deterministic finite accepter *M* such that *L = L(M)*.

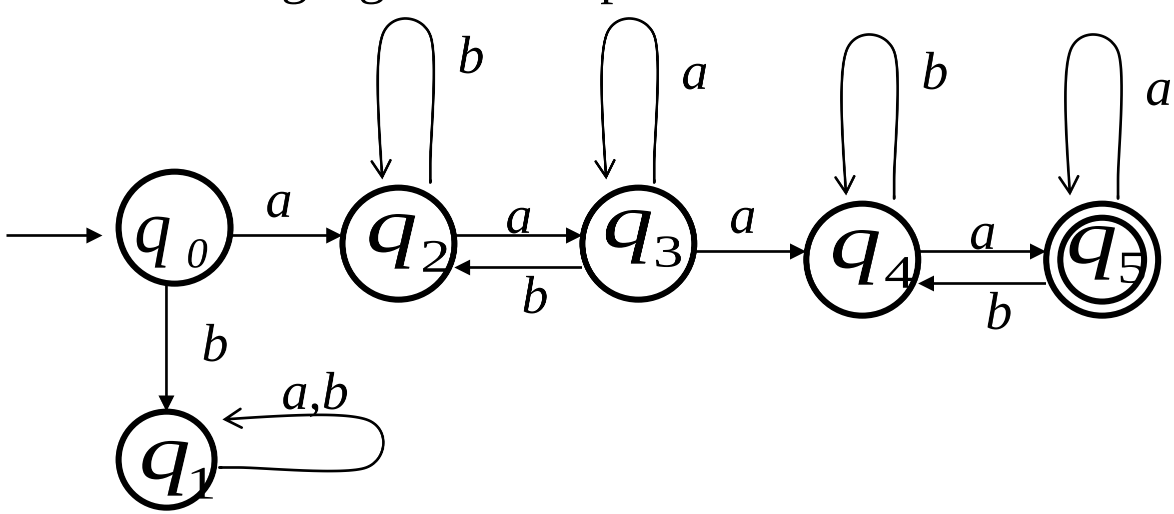
Example

Show that language $L=\{awa: w \in \{a,b\}^*\}$ is regular.



Example

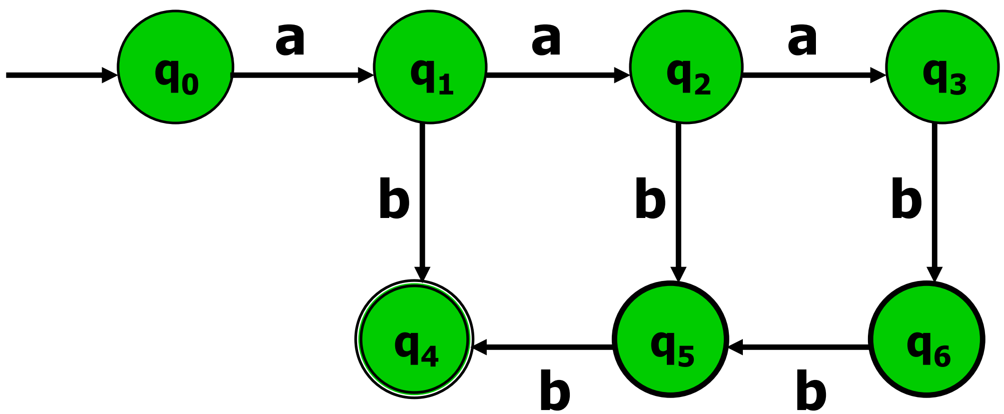
Let L be the language in Example above. Show that $L^2 = \{aw_1aaw_2a\}$ is regular.



State Diagrams and Examples...

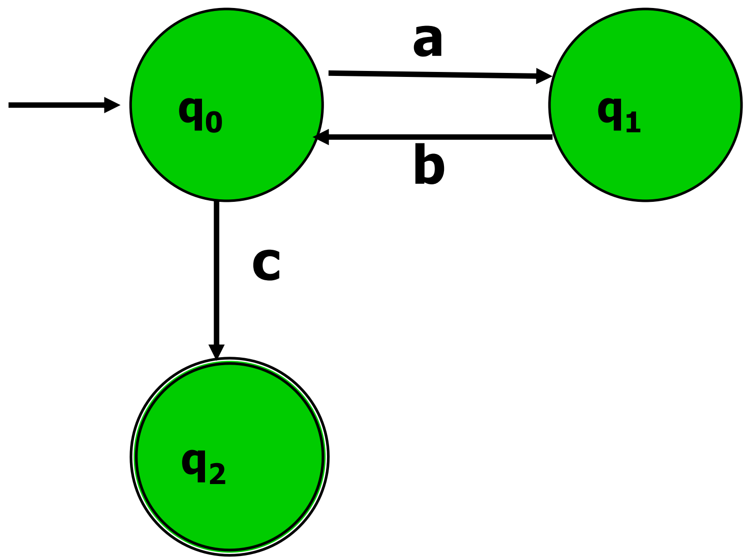
- The language {$\mathbf{a}^n\mathbf{b}^n$, $\boldsymbol{n}\geq 0$} is ***not regular***, so we can not build a DFA that accept this language.
- It needs an infinite number of states.
- But {$\mathbf{a}^n\mathbf{b}^n$, $1 \leq n \leq 3$} is ***regular*** and its DFA is:

This DFA is NOT Complete



State Diagrams and Examples…

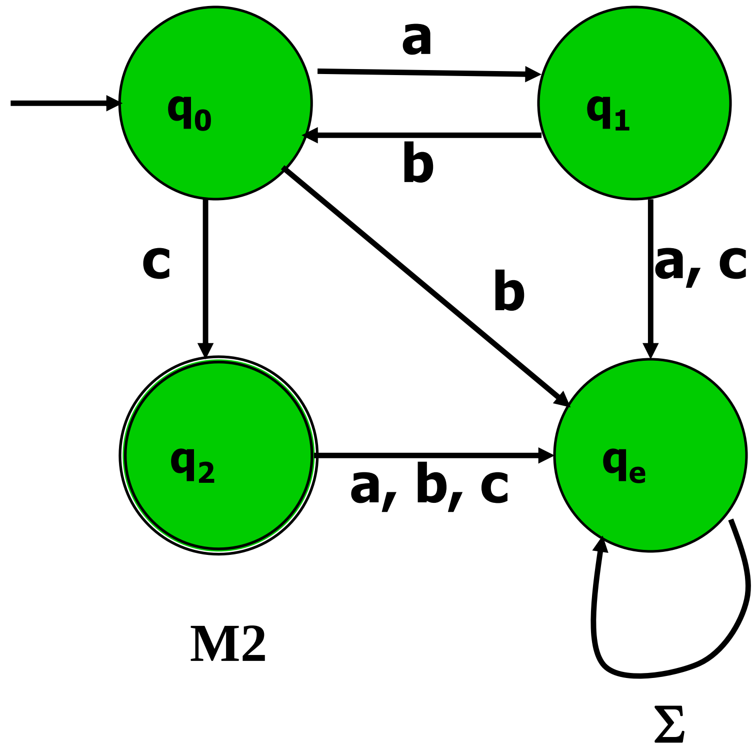
- DFA **M1** accepts **(ab)*c**
- **M1** is *incomplete determinism.*
- The string **abcc** is *rejected* since **M1** is unable to process the final **c** from state $\mathbf{q_2}$



M1

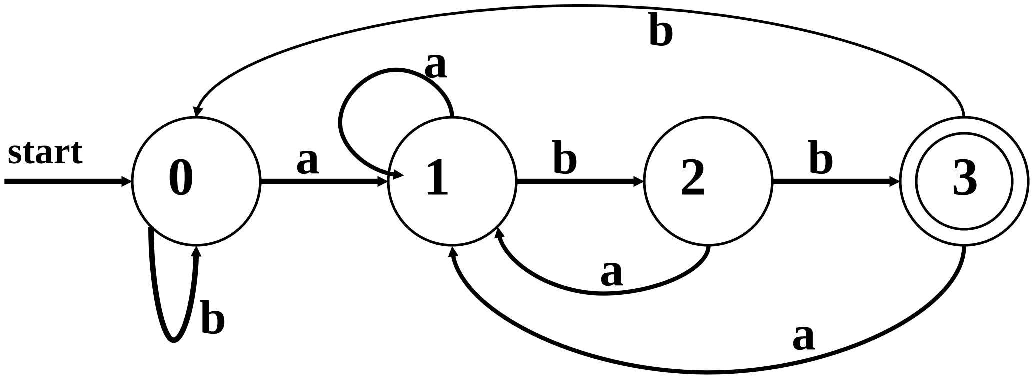
State Diagrams and Examples…

- **M2** accepts the same language as **M1** in previous example **$(ab)^*c$**
- The state **q_e** is the error state (dead end).



M2

State Diagrams …Exercise



What Language is Accepted?

Deterministic Finite Automata…

- **Extended transition function** δ^* of a DFA with transition function δ is a function from $Q \times \Sigma^* \rightarrow Q$ defined recursively on the length of the input string ***w.***
  - Basis: $|\boldsymbol{w}| = 0$. Then $\boldsymbol{w} = \boldsymbol{\varepsilon}$ and $\delta^*(q_i, \boldsymbol{\varepsilon}) = q_i$
  - Recursive step: Let $|\boldsymbol{w}| >= 1$. Then
    - $\delta^*(q_i, \boldsymbol{av}) = \delta^*(\delta(q_i, \boldsymbol{a}), \boldsymbol{v})$
  - $\forall\ q_i \in Q, \forall\ \boldsymbol{a} \in \Sigma, \forall\ \boldsymbol{v} \in \Sigma^*$

Deterministic Finite Automata…

- A string $\boldsymbol{w}$ is accepted if $\boldsymbol{\delta}^*(q_0, \boldsymbol{w}) \in F$.

- The language of a DFA **M** is
  - $L(M) = \{\boldsymbol{w} \in \Sigma^* \mid \boldsymbol{\delta}^*(q_0, \boldsymbol{w}) \in F\}$

- DFA $M = (Q, \Sigma, \boldsymbol{\delta}, q_0, F)$ accepts $\boldsymbol{w} \in \Sigma^* \leftrightarrow$
  - $\boldsymbol{\delta}^*(q_0, \boldsymbol{w}) \in F$

Deterministic Finite Automata…

- Two possibilities for DFA M running on ***w*.**
  - M accepts ***w***
    - M accepts ***w*** iff the computation of M on ***w*** ends up (halts) in an accepting configuration.
    - $\delta^*(q_0, \boldsymbol{w}) \in F$
  - M rejects ***w***
    - M rejects ***w*** iff the computation of M on ***w*** ends up (halts) in a rejecting configuration.
    - $\delta^*(q_0, \boldsymbol{w}) \notin F$

Nondeterministic Finite Automata

NFA: Formal Definition

- NFA $M = (Q, \Sigma, \delta, q_0, F)$
  - Q = a finite set of states
  - Σ = a finite set alphabet
  - δ = transition function
    - total function $Q \times \Sigma \rightarrow \mathrm{P}(Q) = 2^Q$ - power set of Q
  - q_0 = initial/starting state $\quad q_0 \in Q$
  - F = final or accepting states $\quad F \subseteq Q$

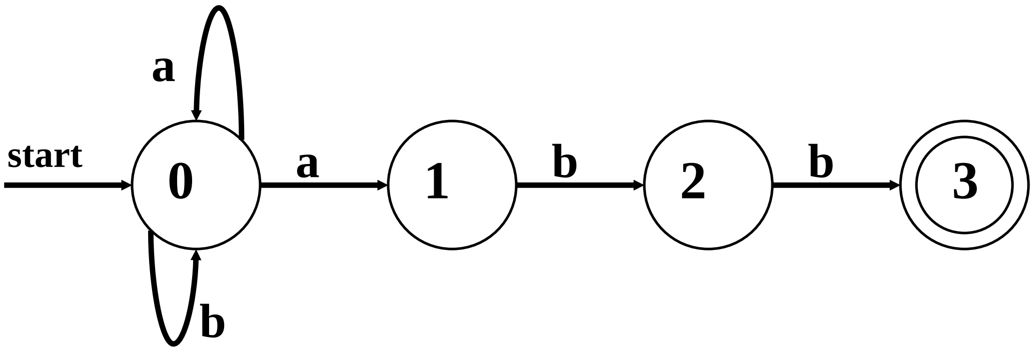
Nondeterministic Finite Automata…

$Q = \{ 0, 1, 2, 3 \}$

$q_0 = 0$

$F = \{ 3 \}$

$\Sigma = \{ a, b \}$



What Language is defined ?

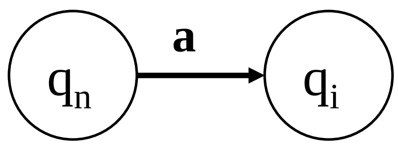
What is the Transition Table ?

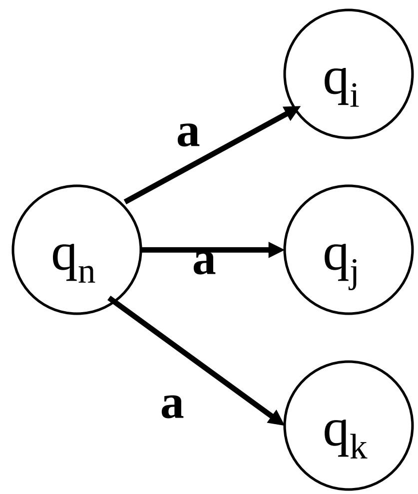
input

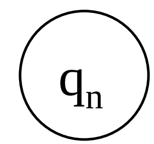
state	a	b
0	{ 0, 1 }	{ 0 }
1	$\varnothing$	{ 2 }
2	$\varnothing$	{ 3 }

Nondeterministic Finite Automata…

- Change in δ
  - For an **DFA** M, δ(**q, a**) results in one and only one state for all states **q** and alphabet **a.**
  - For an **NFA** M, δ(**q, a**) can result in a set of states, zero, one, or more states:





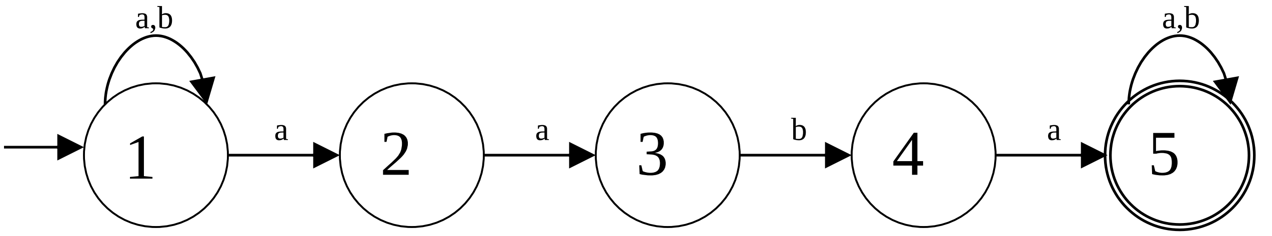


$\delta(q_n, a) = \{q_i\}$

$\delta(q_n, a) = \{q_i, q_j, q_k\}$

$\delta(q_n, a) = \{\} = \varnothing$

Nondeterministic Finite Automata…



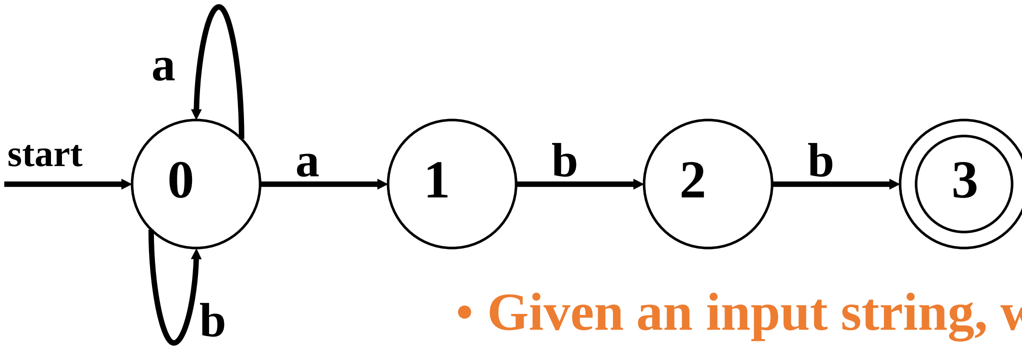
- Why is this only an NFA and not a DFA?

Nondeterministic Finite Automata…

Computing with NFA's

- Computations are different
- We always start from *start state*. Call it the ***root*** of the computation.
- Then we might go to different states on ***one*** symbol.
- Then from those states to new sets of states, creating a **tree-like** computation.
- If one path ends up in a final state, then ACCEPT, else REJECT.

Nondeterministic Finite Automata…



- Given an input string, we trace moves
- If no more input & in final state, ACCEPT

EXAMPLE:

Input: **ababb**

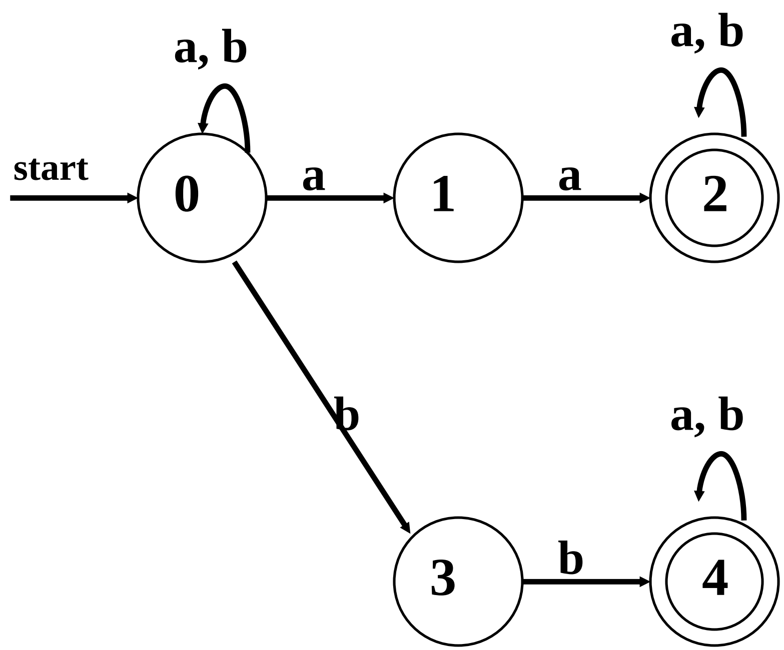
```
Path 1: 0 -> 0 -> 0 -> 0 -> 0 -> 0
(REJECT)
Path 2: 0 -> 0 -> 0 -> 1 -> 2 -> 3
(ACCEPT)
```

Nondeterministic Finite Automata...

- **Extended transition function** δ^* of a NFA with transition function δ is a function from $Q \times \Sigma^* \rightarrow 2^Q$ (power set) defined recursively on the length of the input string ***w***
  - Basis: $|\boldsymbol{w}| = 0$. Then $\boldsymbol{w} = \varepsilon$ and $\delta^*(q_i, \varepsilon) = \{q_i\}$
  - Recursive step: Let $|\boldsymbol{w}| >= 1$. Then
    - $\delta^*(q_i, \boldsymbol{av}) = \cup \delta^*(q_j, \boldsymbol{v}), q_j \in \delta(q_i, \boldsymbol{a})$
  - $\forall\ q_i \in Q, \forall\ q_j \in Q, \forall\ \boldsymbol{a} \in \Sigma, \forall\ \boldsymbol{v} \in \Sigma^*$
- The language of a NFA **M** is
  - $L(M) = \{\boldsymbol{w} \in \Sigma^* \mid \delta^*(q_0, \boldsymbol{w}) \cap F \neq \emptyset\}$
  - The language consists of all strings *w* for which there is a walk labeled *w* from the initial vertex of the transition graph to some final vertex.

NFA…

- An NFA that accepts string over {a, b} with substring **aa** or **bb.**



- There are 2 distinct acceptance paths for the string **abaaabb**

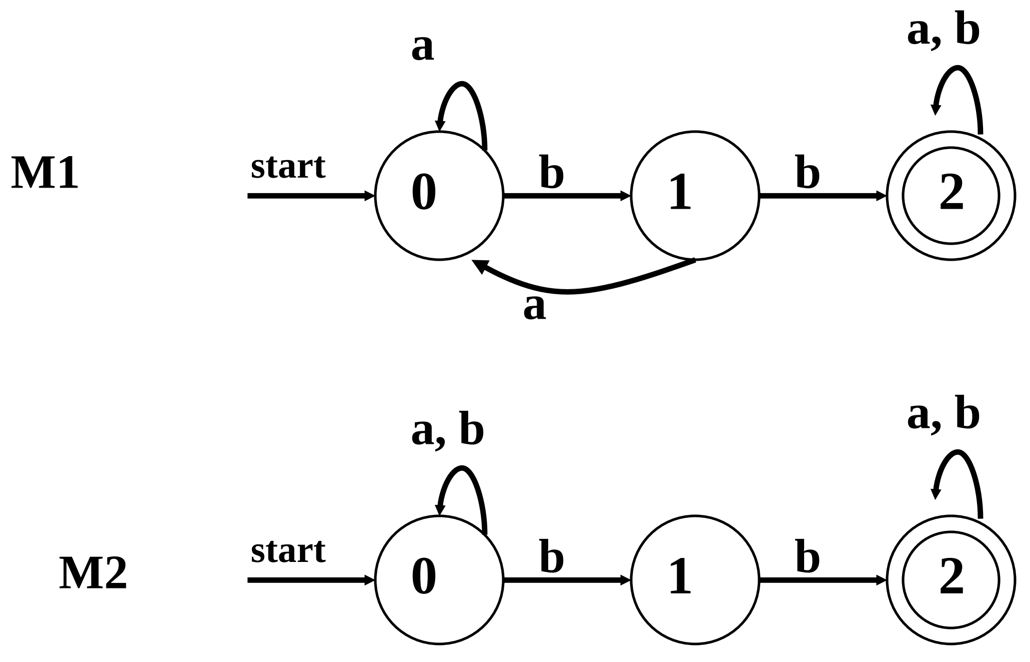
Equivalence of DFA & NFA

- Is NFA more powerful than DFA? NO!
  - NFA is inefficient to implement directly, so convert to a DFA that recognizes the same strings.
- Is there a language accepted by an NFA that is not accepted by any DFA? No
- There is an equivalent DFA for any NFA.
  - Each state in DFA corresponds to a SET of states of the NFA.

- Two finite accepters M_1 and M_2 are said to be **equivalent** if $L(M_1)=L(M_2)$. That is, if they accept the same language.

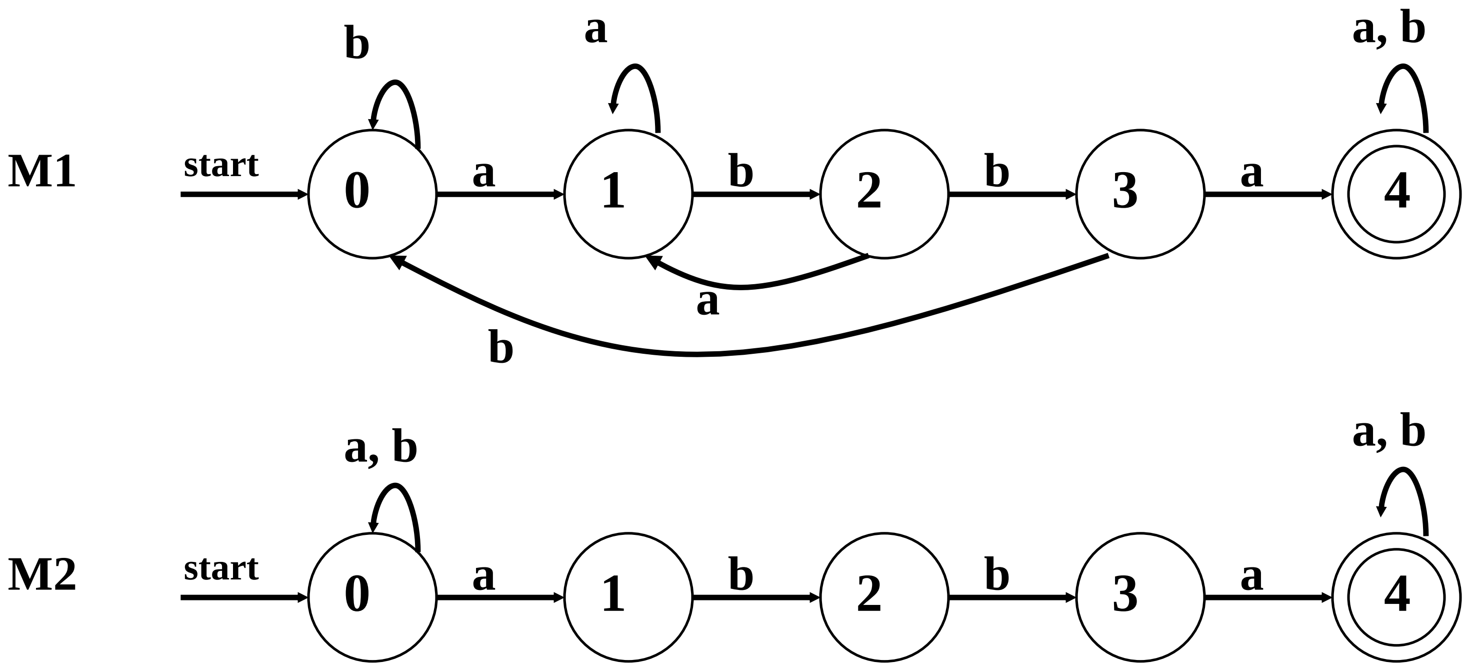
Equivalence of DFA & NFA…

- The state diagram **DFA** M1 and **NFA** M2 accepts **$(a|b)^*bb(a|b)^*$**



Equivalence of DFA & NFA…

- The state diagram **DFA** M1 and **NFA** M2 accepts **(a|b)*abba(a|b)***



Equivalence of DFA & NFA…

- A DFA can be turned into an NFA that accepts the same language.
- If $\delta_D(q, a) = p$, let the NFA have $\delta_N(q, a) = \{p\}$.
- Then the NFA is always in a set containing exactly one state – the state the DFA is in after reading the same input.

Equivalence of DFA & NFA…

- Surprisingly, for any NFA there is a DFA that accepts the same language.
- Proof is the *subset construction*.
- The number of states of the DFA can be exponential in the number of states of the NFA.
- Thus, NFA’s accept exactly the regular languages.

Subset Construction

- Given an NFA with states Q, inputs Σ, transition function δ_N, state state q_0, and final states F, construct equivalent DFA with:
  - States 2^Q (Set of subsets of Q).
  - Inputs Σ.
  - Start state $\{q_0\}$.
  - Final states = all those with a member of F.

Subset Construction…

- The DFA states have *names* that are sets of NFA states.
- But as a DFA state, an expression like {p,q} must be read as a single symbol, not as a set.
- Analogy: a class of objects whose values are sets of objects of another class.

Subset construction…

- The transition function δ_D is defined by:

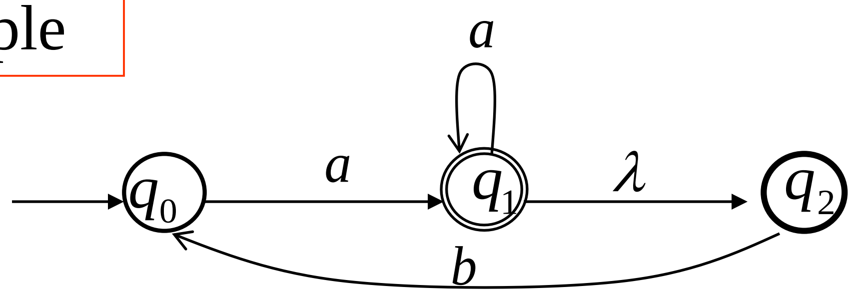
$\delta_D(\{q_1,\ldots,q_k\}, a)$ is the union over all $i = 1,\ldots,k$ of $\delta_N(q_i, a)$.

- Example:
  - Determine a deterministic Finite State Automaton from the given Nondeterministic FSA.
    M = $(\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_1\})$
    with the state table diagram for δ given below.

δ	a	b
q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1	$\varnothing$	$\{q_0, q_1\}$

Convert *nfa* to *dfa*

Example



nfa

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

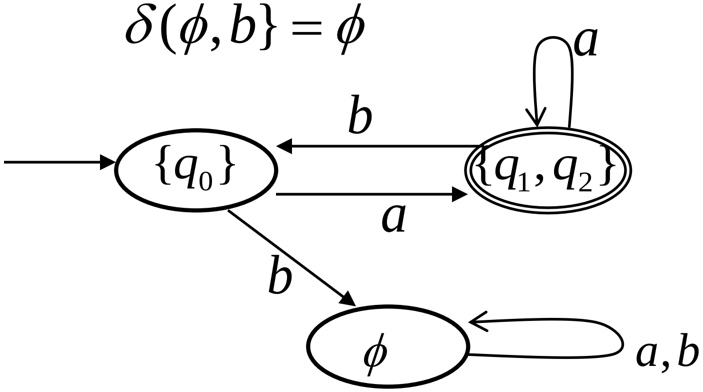
$$\delta(\{q_0\},b\}=\phi$$

$$\delta(\{q_{1,}q_2\},a\}=\{q_1,q_2\}$$

$$\delta(\{q_{1,}q_2\},a\}=\{q_0\}$$

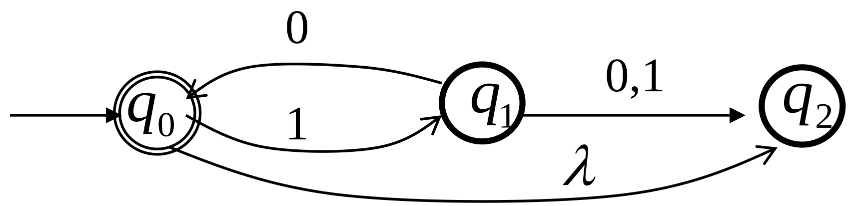
$$\delta(\phi,b\}=\phi$$

$$\delta(\phi,b\}=\phi$$



dfa

Example



nfa

$$\delta(\{q_0\},0\} = \phi$$

$$\delta(\{q_0\},1\} = \{q_1\}$$

$$\delta(\{q_1\},0\} = \{q_{0,} q_2\}$$

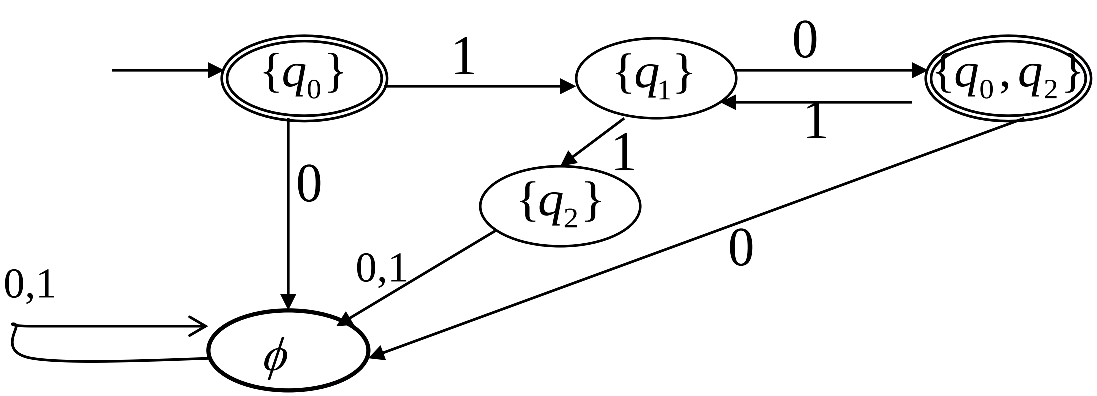
$$\delta(\{q_1\},1\} = \{q_2\}$$

$$\delta(\{q_0, q_2\},0\} = \phi$$

$$\delta(\{q_0, q_2\},1\} = \{q_1\}$$

$$\delta(\{q_2\},0\} = \phi$$

$$\delta(\{q_2\},1\} = \phi$$



dfa

Conversion from NFA to DFA

<u>Procedure to convert nfa to dfa</u>

1. Create a graph G(D) with vertex {q0}. Identify this vertex as the initial vertex.

2. Repeat the following steps until no more edges are missing.

   take any vertex{qi,qj,…,qk} of G(D) that has no outgoing edge for some $a \in \Sigma$. Compute $\delta_N^*(q_i, a), \delta_N^*(q_j, a), ..., \delta_N^*(q_k, a)$.

   If $\delta_N^*(q_i, a) \cup \delta_N^*(q_j, a) \cup ... \cup \delta_N^*(q_k, a) = \{ql, qm, ..., qn\}$,

   create a vertex for G(D) labeled {ql,qm,…,qn} if it is not already exist. Add to G(D) an edge from {qi,qj,…,qk} to {ql,qm,…,qn} and label it with a.

3. Every state of D(D) whose label contains any $q_f \in F_N$ is identified as a final vertex.

4. If M_N accepts λ, the vertex {q0} in G(D) is also made a final vertex.

Reading (Self Study)

- NFA with ε-Transitions
- Reduction of Number of States in Finite Automata
- Two way Finite Automata