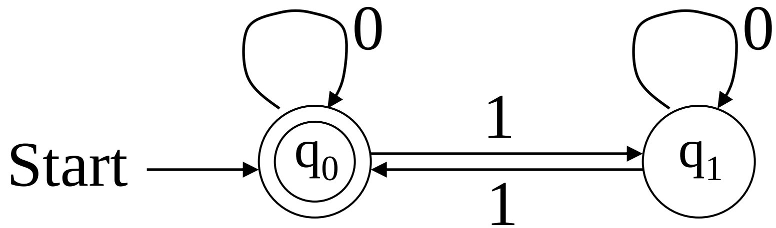
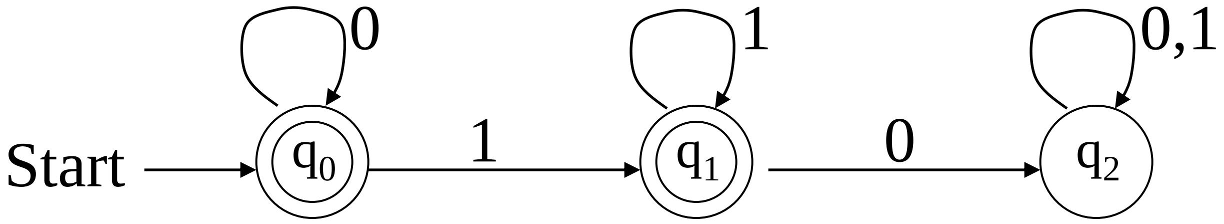
Class Discussion





What are the languages accepted by these DFA?

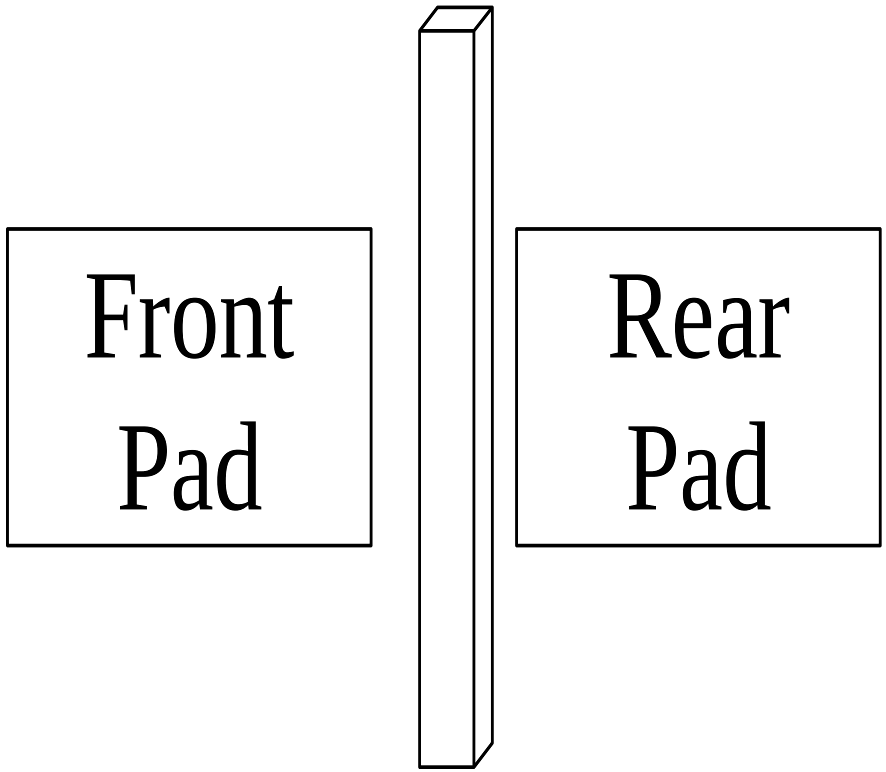
Class Discussion

Construct a DFA that accepts a language L over $\Sigma = \{0, 1\}$ such that L is the set of all the strings:

a) which starts with a ‘0’ and ends with ’10’.
b) starting with “11”.
c) having “00” as substring.
d) containing odd number of 0’s.

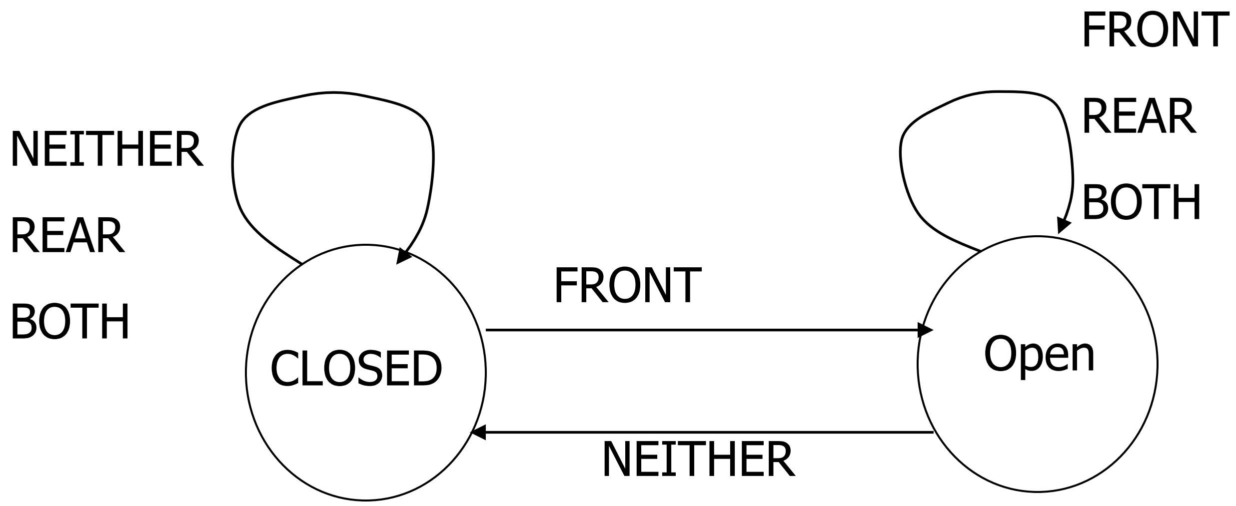
Example – Automatic Door (1 way door)

- Assumptions :
  - Two pads front and rear pad ,that can sense when someone is standing on them.
  - We want people to walk through the front and toward the rear, but not allow someone to walk the other direction.
  - Open when person approaches
  - Hold open until person clears



The Automatic Door as DFA

We can design the following automaton so that the door doesn't open if someone is still on the rear pad and hit them



States: Open, Closed

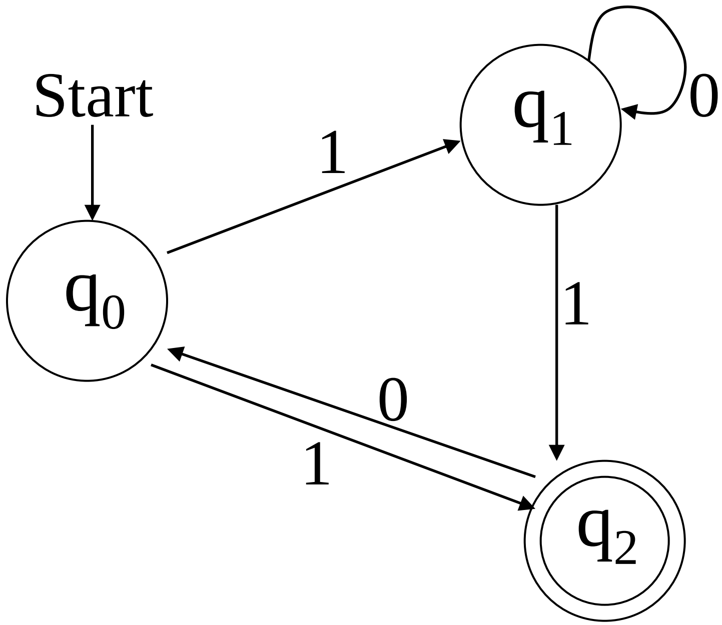
Sensor: Front, Rear, Both, Neither

Non-deterministic FA (NFA)

- For each state, zero, one or more transitions are allowed on the same input symbol.
- An input is accepted if there is a path leading to a final state.

An Example of NFA

In this NFA (Q,Σ,δ,q_0,F), $Q = \{q_0,q_1,q_2\}$, $\Sigma = \{0,1\}$, $F = \{q_2\}$ and δ:



OR

δ	0	1
$\rightarrow q_0$	ϕ	$\{q_1,q_2\}$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\{q_0\}$	ϕ

Note that each transition can lead to <u>a set of states</u>,
which can be empty.

Definition of NFA

An NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

Q: Finite set of states

Σ: Finite set of input alphabets

δ: Transition function mapping $Q \times \Sigma \longrightarrow 2^Q$

$q_0 \in Q$: Initial state (only one)

$F \subseteq Q$: Set of final states (zero or more)

An NFA accepts a string

- All the input is consumed and the automaton is in a final state

An NFA rejects a string:

- All the input is consumed and the automaton is in a non final state
- The input cannot be consumed

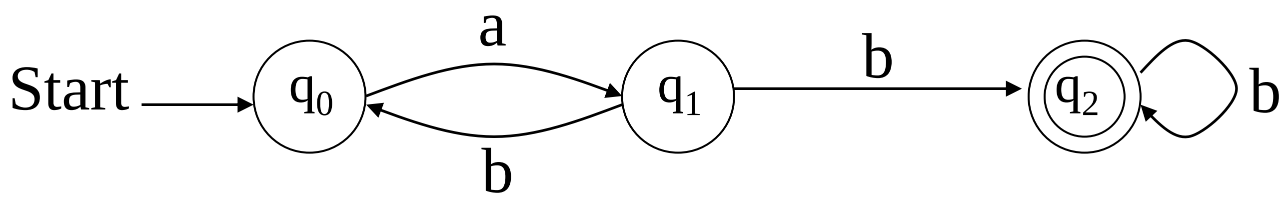
Language of an NFA

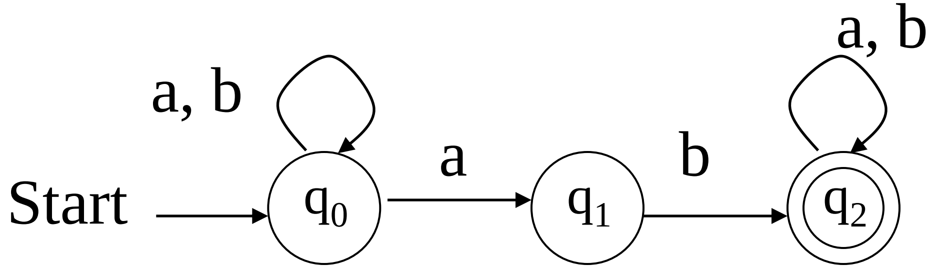
Given an NFA M, the language recognized by M is the set of all strings that, starting from the initial state, has at least one path reaching a final state after the whole string is read.

Consider the previous example:

For input "101", one path is $q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_2$ and the other one is $q_0 \rightarrow q_2 \rightarrow q_0 \rightarrow q_1$ and one more path is $q_0 \rightarrow q_2 \rightarrow q_0 \rightarrow q_2$. Since q_2 is a final state, so "101" is accepted and the paths lading to q_2 are valid paths. For input "1010", none of its paths can reach a final state, so it is rejected.

More Examples of NFA





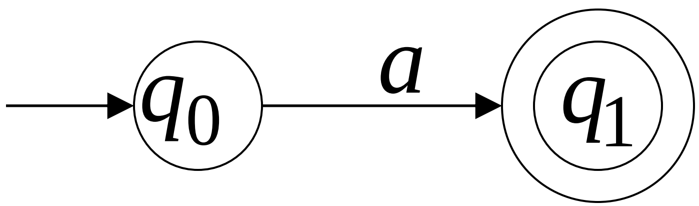
Class Discussion

Draw the NFA for the language L that consists of all the strings over $\Sigma = \{a, b\}$ such that :

- starting with “a”
- containing substring “aa”
- the 3rd last symbol of string is “b”

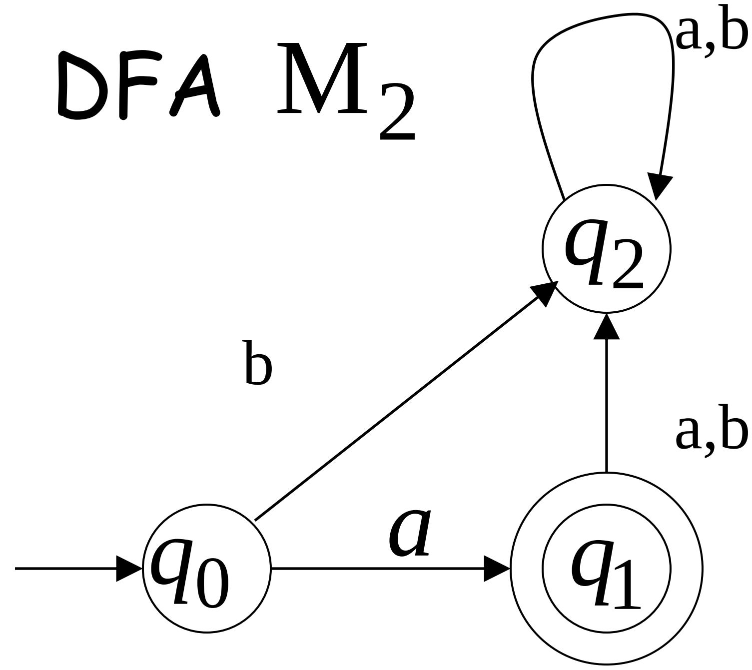
NFAs are interesting because we can express languages easier than DFAs

NFA M_1



$$L(M_1) = \{a\}$$

DFA M_2



$$L(M_2) = \{a\}$$

NFA with ε-Transitions (ε-NFA)

- There exist ε-transitions that allow state changes without consuming any input symbol.
- Similar to NFA, an input is accepted if there is a path leading from the start state to a final state after the whole string is read.

Definition of ε-NFA

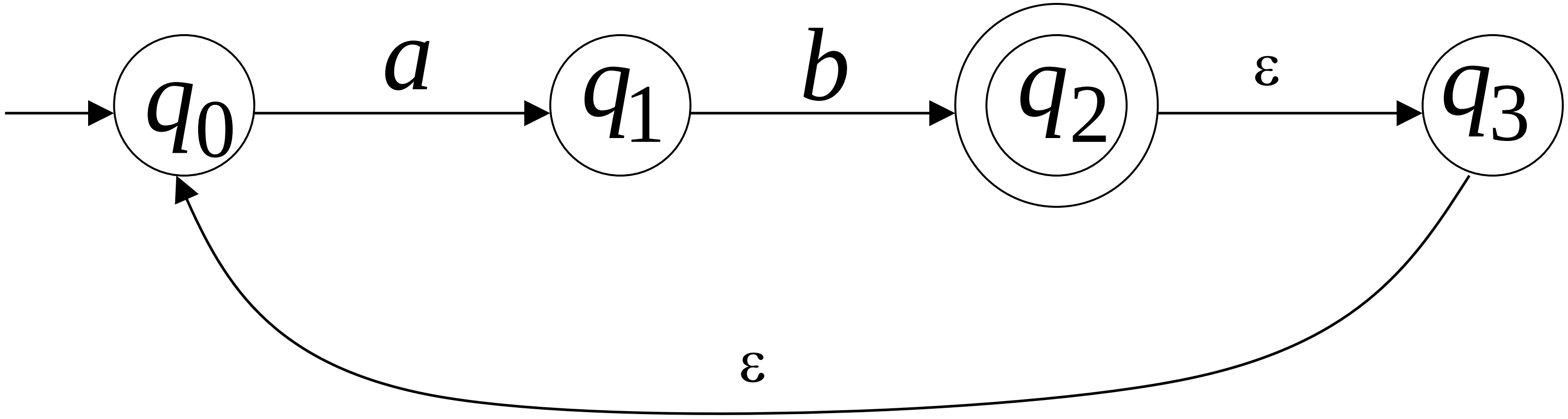
An ε- NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q: Finite set of <u>states</u>
- Σ: Finite set of input <u>alphabets</u>
- δ:T<u>ransition function</u> mapping $Q \times (\Sigma \cup \varepsilon) \longrightarrow 2^Q$
- $q_0 \in Q$:<u>Initial state</u> (only one)
- $F \subseteq Q$:Set of <u>final states</u> (zero or more)

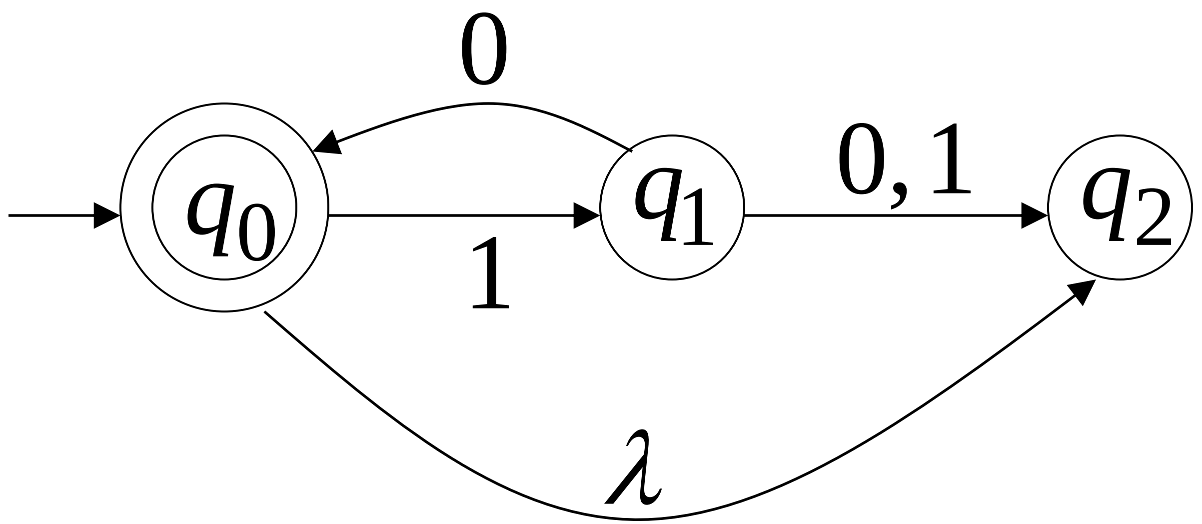
Example

$$L = \{ab,\ abab,\ ababab,\ \ldots\}$$

$$= \{ab\}^{+}$$



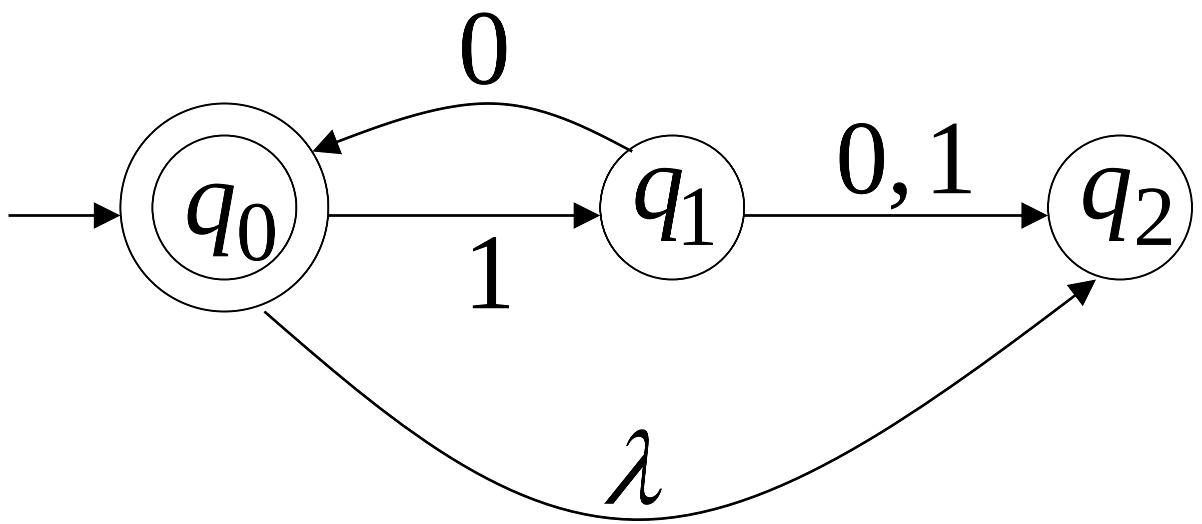
Another NFA Example



Language accepted

$$L(M) = \{\lambda,\ 10,\ 1010,\ 101010,\ \ldots\}$$

$$= \{10\}^*$$

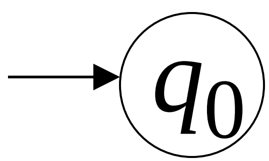


Remarks:

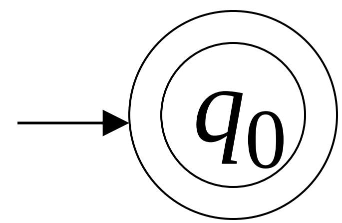
- The λ symbol never appears on the input tape

- draw the automata for empty language and the language which contains λ :

M_1



M_2



$L(M_1) = \{\}$

$L(M_2) = \{\lambda\}$

Equivalence of NFAs and DFAs

Question: NFAs = DFAs ?

Same power?

Accept the same languages?

Equivalence of NFAs and DFAs

Question: NFAs = DFAs ? **YES!**

Same power?

Accept the same languages?

We will prove:

$$\left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by NFAs}\end{array}\right\} = \left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by DFAs}\end{array}\right\}$$

NFAs and DFAs have the same computation power

Step 1

$$\left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by NFAs}\end{array}\right\} \supseteq \left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by DFAs}\end{array}\right\}$$

Proof: Every DFA is trivially an NFA

A language accepted by a DFA
is also accepted by an NFA

Step 2

$$\left\{\begin{array}{l}\text{Languages}\\\text{accepted}\\\text{by NFAs}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Languages}\\\text{accepted}\\\text{by DFAs}\end{array}\right\}$$

Proof: Any NFA can be converted to an equivalent DFA

A language accepted by an NFA is also accepted by a DFA

Constructing DFA from NFA

Given any NFA $M=(Q,\Sigma,\delta,q_0,F)$ recognizing a language L over Σ, we can construct a DFA $M'=(Q',\Sigma,\delta',q_0',F')$ which also recognizes L

If the NFA has states

$$q_0, q_1, q_2, \ldots$$

- the DFA has states in the power set

$$\emptyset, \{q_0\}, \{q_1\}, \{q_1, q_2\}, \{q_3, q_4, q_7\}, \ldots.$$

Procedure NFA to DFA

1. Initial state of NFA:

$$q_0$$

Initial state of DFA:

$$\{q_0\}$$

Procedure NFA to DFA

2. For every DFA's state $\{q_i, q_j, \ldots, q_m\}$

Compute in the NFA

$$\left.\begin{array}{r}\delta^*(q_i, a), \\ \delta^*(q_j, a), \\ \ldots\end{array}\right\} = \{q'_i, q'_j, \ldots, q'_m\}$$

$$\delta(\{q_i, q_j, \ldots, q_m\}, a) = \{q'_i, q'_j, \ldots, q'_m\}$$

Add transition to DFA

Procedure NFA to DFA

Repeat Step 2 for all letters in alphabet,

until

no more transitions can be added.

Procedure NFA to DFA

3. For any DFA state $\{q_i, q_j, \ldots, q_m\}$

If some q_j is a final state in the NFA

Then, $\{q_i, q_j, \ldots, q_m\}$
is a final state in the DFA

Informal Proof of Correctness

- Each state in the DFA represents a set of states in the original NFA.
- After reading an input string ω, the DFA is in a state that represents the set of all states the original NFA could be in after reading ω.
- Since any state in the DFA that includes a final state of the NFA is a final state, the DFA and the NFA will accept the same set of strings.

We have proven

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} = \left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by DFAs} \end{array} \right\}$$

Regular Languages

Regular Languages

We have proven

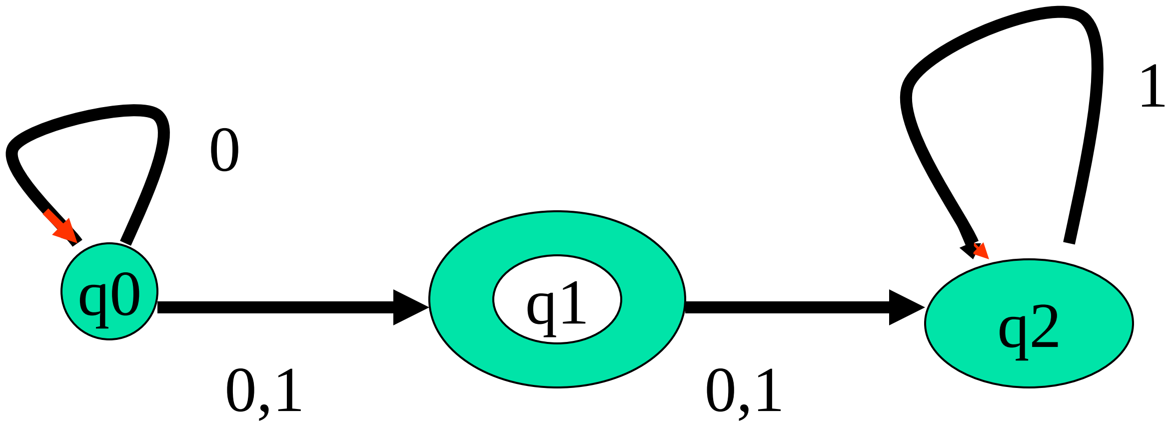
$$\left\{\begin{array}{l}\text{Languages} \\ \text{accepted} \\ \text{by NFAs}\end{array}\right\} = \left\{\begin{array}{l}\text{Languages} \\ \text{accepted} \\ \text{by DFAs}\end{array}\right\}$$

Regular Languages

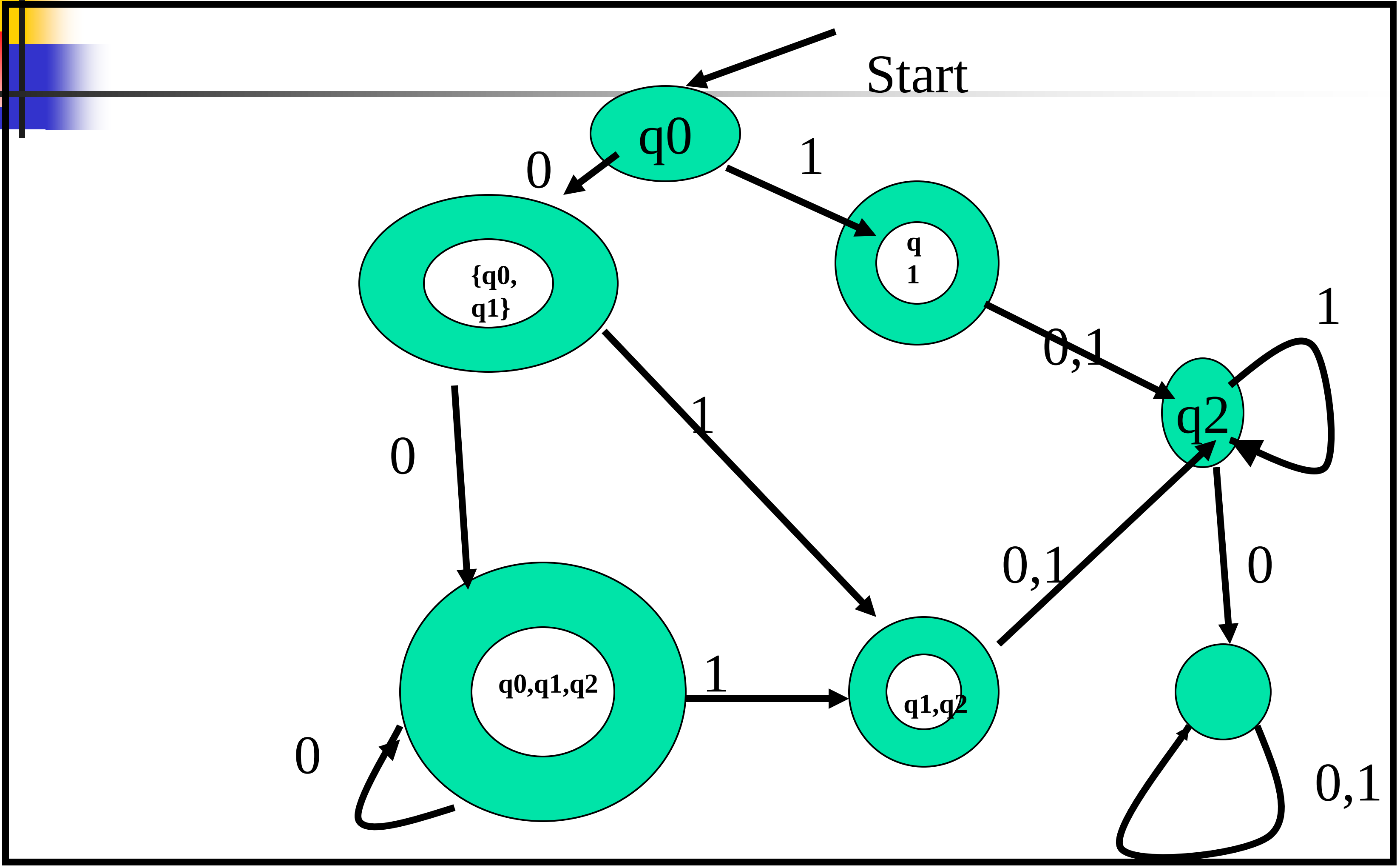
Regular Languages

Thus, NFAs accept the regular languages

Example

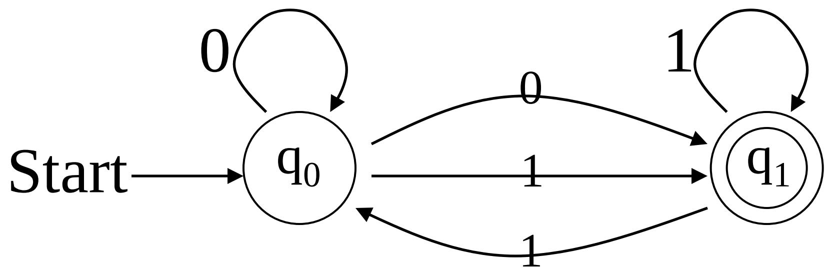


Equivalent DFA

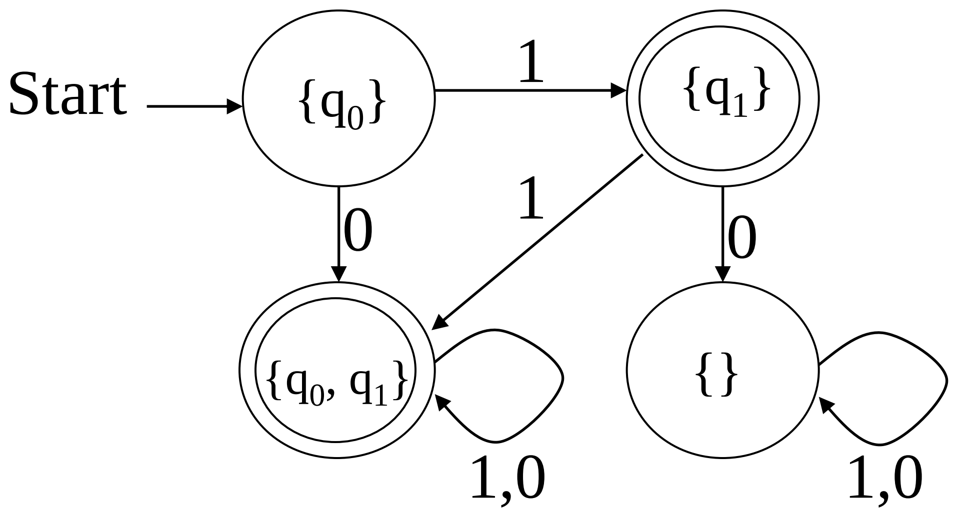


Another Example of NFA → DFA

NFA



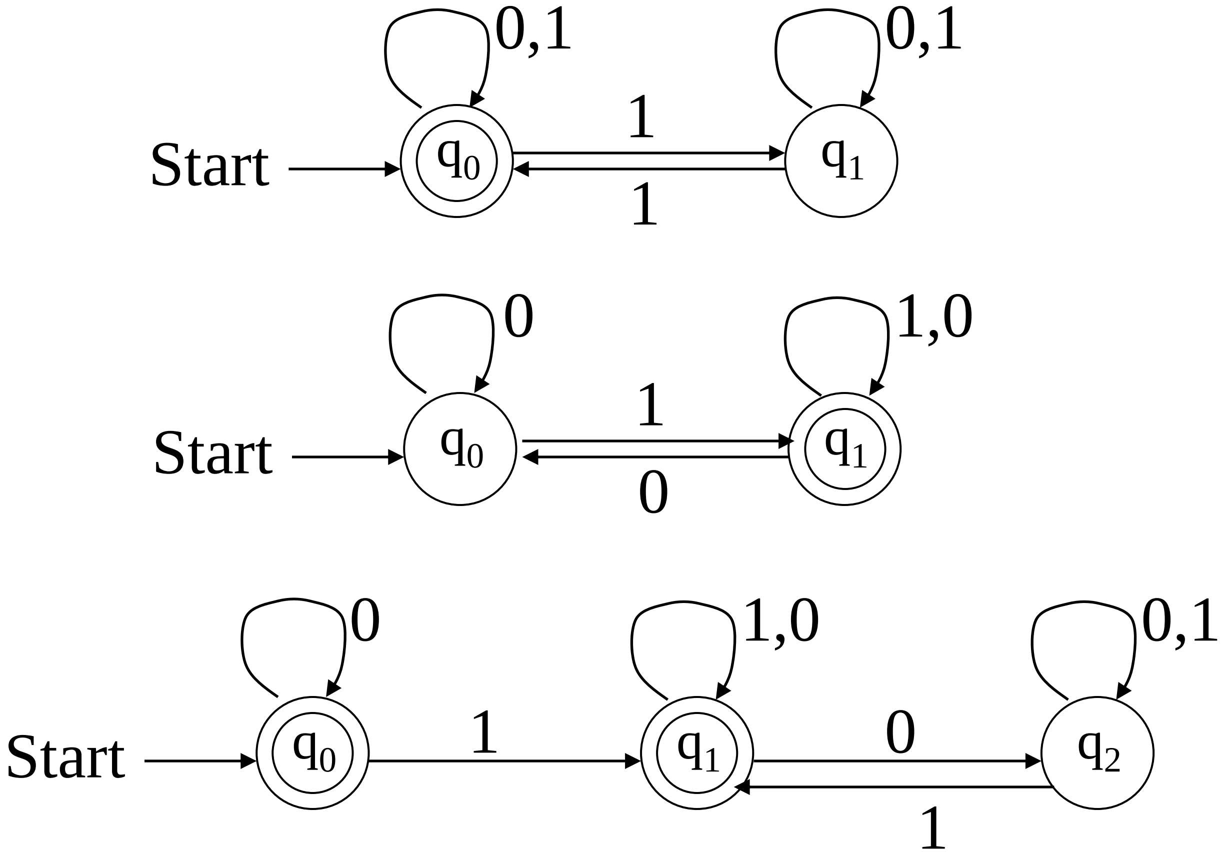
Equivalent DFA



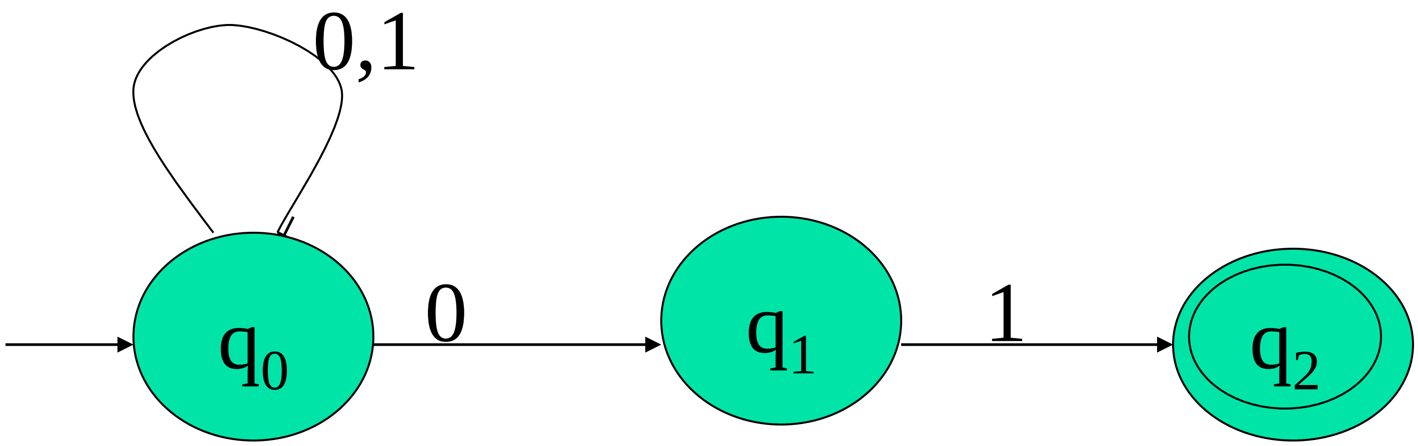
Note on NFA $\rightarrow$ DFA

Sometimes we do not need to consider all possible subsets of the states in the original NFA (especially when the original NFA is complicated). We can construct the states in the DFA one by one whenever needed, starting from the initial state $\{q_0\}$ where q_0 is the initial state of the original NFA.

Class Discussion (NFA → DFA)



Convert an NFA to a DFA: An example



Solution