Non deterministic Finite Automata (NFA)

Till now, we have seen Deterministic FA.
Now we will see NFA. Let us see the major differences.

1) DFA has exactly one transition for each symbol and state.

   NFA can have 0, 1 or many.

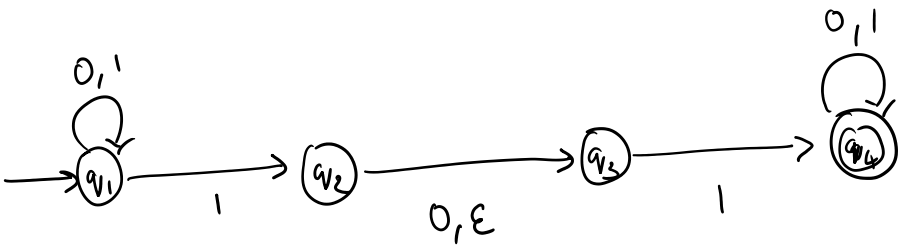
2) DFA has arrows labelled with symbols in the alphabet.

   NFA can have arrows marked with ε.
   → Can have 0, 1 or many arrows marked with ε.

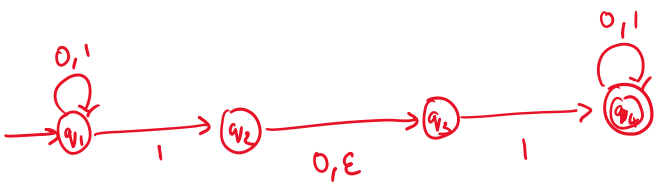
3) NFA can have multiple choices to be made at each state, possibly. It could have many possible computation paths.

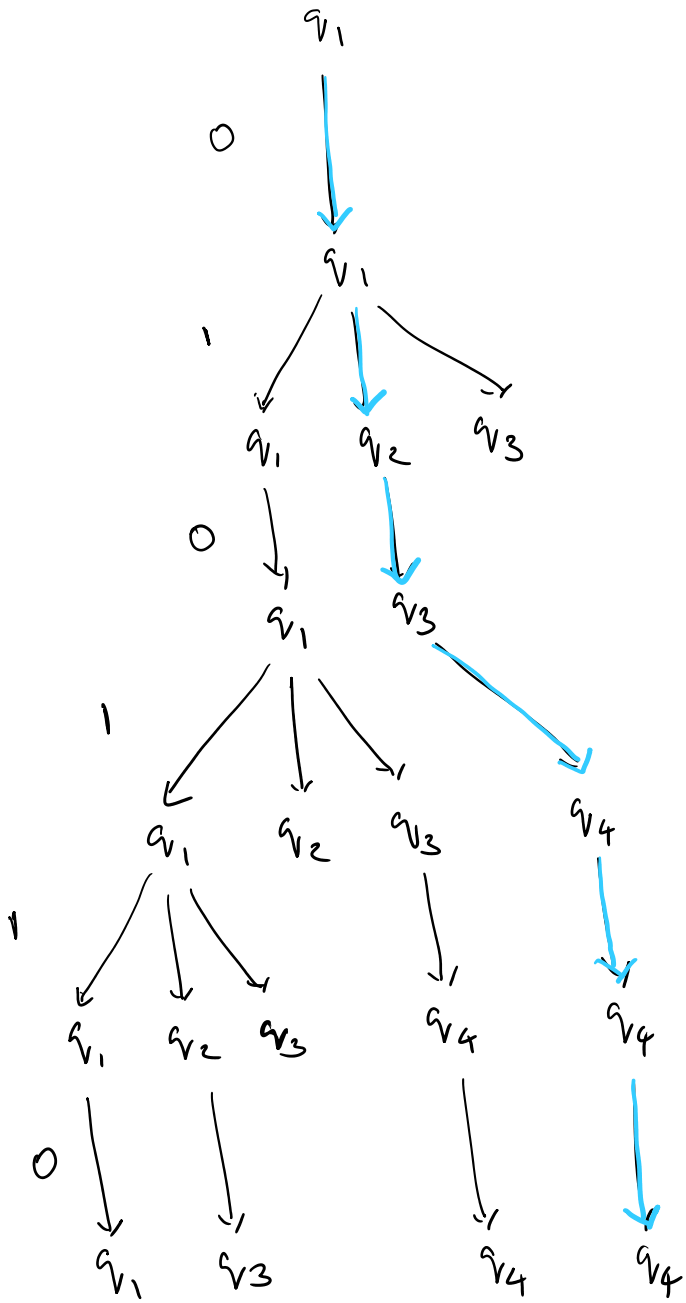
4) NFA accepts a string if there is at least one accepting computation path.

one accepting computation path.



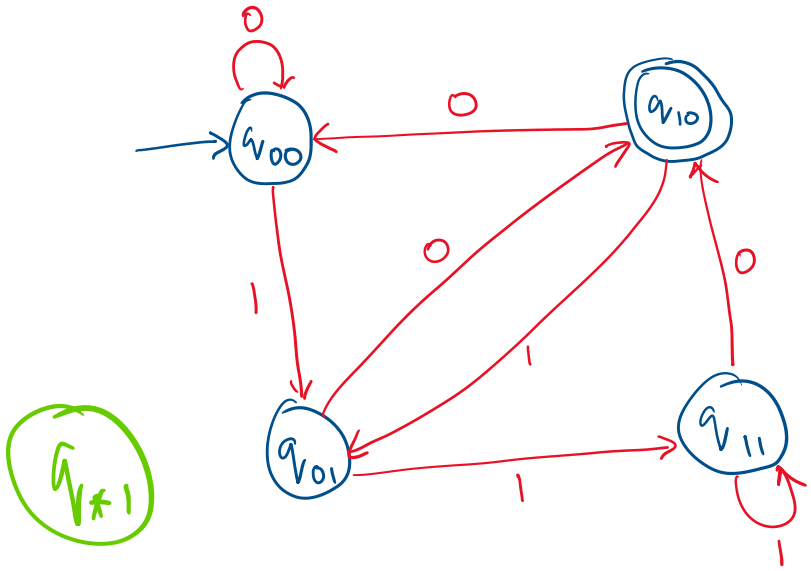
5) Nondeterminism can be viewed as guessing.



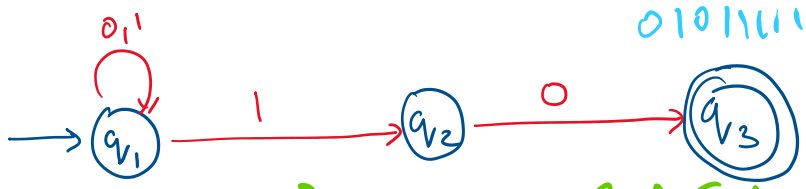


$A = \{ w \mid w \text{ ends in } 10 \}$

DFA for A



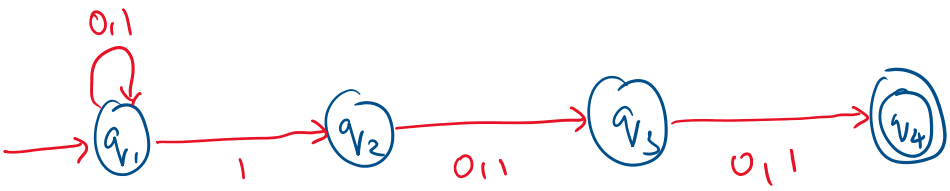
NFA for A:



01011111

Ex: $B = \{ w \mid w \text{ ends in } 101 \}$. DFA & NFA.

Example 1.30: language of strings where the third symbol from the end is a 1.



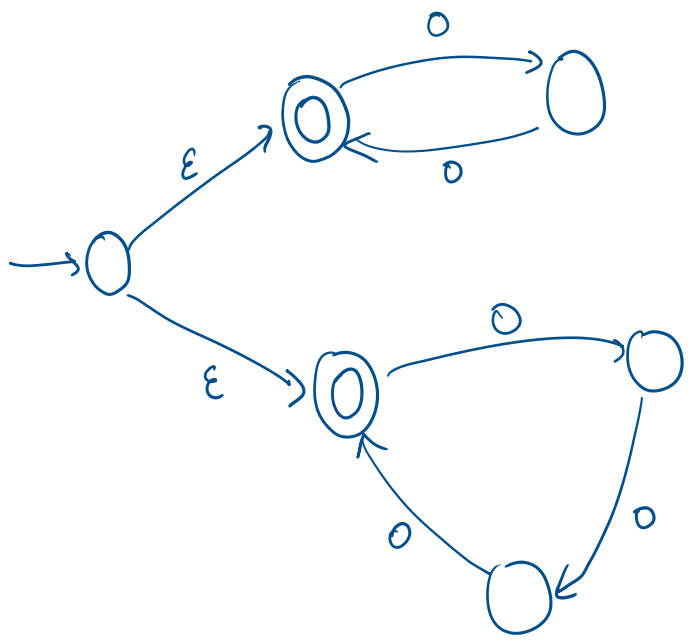
The DFA requires 8 states and is more involved.

Read the book and see the DFA.

Example 1.33: Unary language.

$\{ 0^k \mid k$ is a multiple of 2 or 3 $\}$

$A = \{ 0^k \mid k \text{ is a multiple of } 2 \text{ or } 3 \}$



<u>Def 1.37</u>: A nondeterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

(1) Q is a finite set of states

(2) Σ is a finite alphabet.

(3) $\delta : Q \times \Sigma_\varepsilon \to \underline{\mathcal{P}(Q)}$ is the transition function.

Power set of Q

(4) $q_0 \in Q$ is the start state

(5) $F \subseteq Q$ is the set of accepting states.

$\mathcal{P}(Q)$ = Power set of Q = Set of all subsets of Q.

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.

NFA computation is similar to DFA computation.

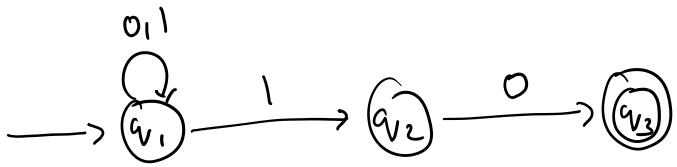
NFA N accepts w if we can write $w = y_1 y_2 \cdots y_m$ where $y_i \in \Sigma_\varepsilon$ and a sequence of states $r_0, r_1, \ldots r_m \in Q$ s.t.

(1) $r_0 = q_0$

(2) $\delta(r_i, y_{i+1}) \ni r_{i+1}$ for $i = 0, 1, \ldots m-1$

(3) $r_m \in F$.

<u>Example</u>:



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

$\Sigma = \{0, 1\}$

$F = \{q_3\}$

Start State $\Big\} = q_1$

$\delta(q_1, 0) = \{q_1\}$

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

$\delta(q_2, 0) = \{q_3\}$

$\delta(q_2, 1) = \phi$

$\delta(q_3, 0) = \delta(q_3, 1) = \phi$.

Read <u>Example 1.38</u>.

<u>Defn</u>: Two machines are <u>equivalent</u> if they

Defn: Two machines are equivalent if they recognize the same language.

Theorem 1.39: Every NFA has an equivalent DFA.

This implies that NFAs are only as powerful as DFAs.

Proof Idea: Keep track of all the set of states that you could possibly be in.

{Possible State} $\longrightarrow$ New State

Proof: Suppose there are no ε arrows. Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA recognizing some language A. We want to build an equivalent DFA $M = (Q', \Sigma, \delta', q_0', F')$.

1) $Q' = P(Q) = \{R \mid R \subseteq Q\}$

↳ Power set of Q.

2) $q_0' = \{q_0\}$.

3) $F' = \{R \in Q' \mid \exists r \in R \text{ s.t. } r \in F\}$

$= \{R \subseteq Q \mid R \cap F \neq \emptyset\}$.

$$= \{ R \subseteq Q \mid R \cap F \neq \phi \}.$$

4) $\delta'(R, a)$ where $R \in Q'$ and $a \in \Sigma$.

$$\delta'(R, a) = \{ q \in Q \mid q \in \delta(r, a) \text{ for some } r \in R \}.$$

$$= \bigcup_{r \in R} \delta(r, a).$$

This takes care of all the transitions except the ε transitions. To handle ε transitions, we define $E(R)$ for every $R \in Q'$.

$E(R) = \{ q \mid q$ can be reached from R by travelling along 0 or more ε transitions $\}$.

Notice that $R \subseteq E(R)$.

We just need to redefine δ' to account for ε transitions.

$$\delta'(R, a) = \{ q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R \}.$$

$$= \bigcup_{r \in R} E(\delta(r, a)).$$

$r \in R$

Also, we must change the start state to

$q_0' = E(\{q_0\})$.

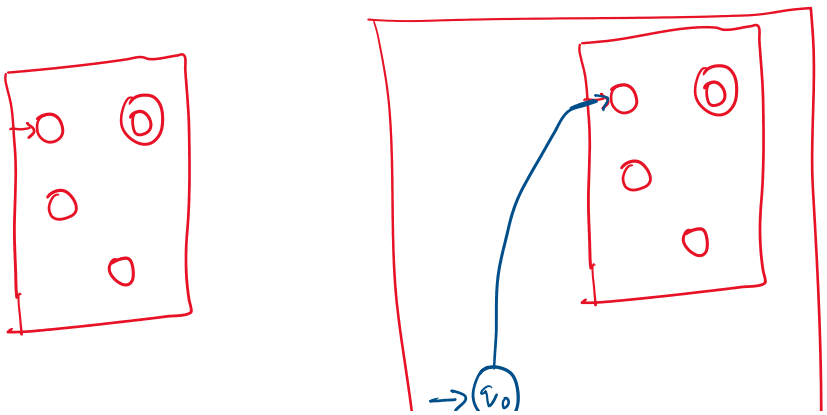
After reading any string, if N can possibly reach $R \subseteq Q$, then M will reach $R \in P(Q)$.

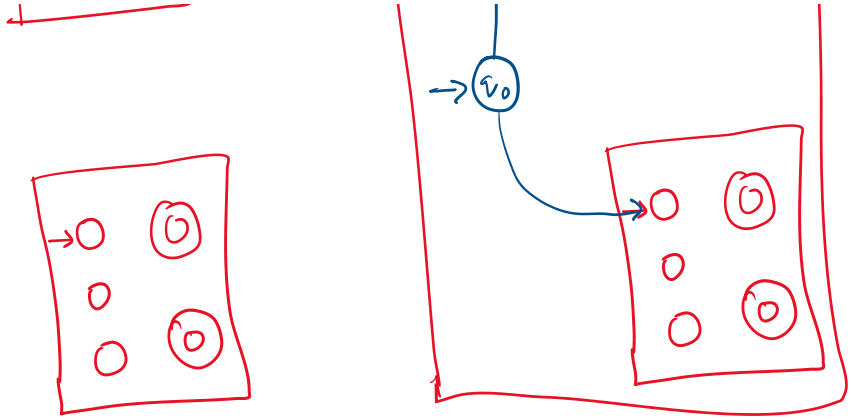
<u>Corollary 1.40</u>: A language is regular if and only if some nondeterministic finite automaton recognizes it.

Read Example 1.41. for an illustration of the construction in the proof of Theorem 1.39.

<u>Theorem 1.45</u>: The class of regular languages is closed under union.

<u>Proof</u>:





$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

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

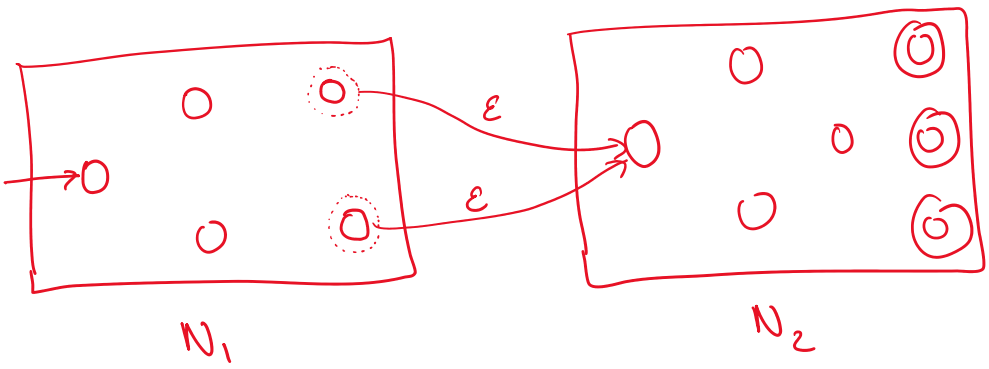
$Q = Q_1 \cup Q_2 \cup \{q_0\}$

Exercise: Read the details of this proof.

<u>Theorem 1.47</u>: The class of regular languages is closed under the concatenation operation.

<u>Proof Sketch</u>: A_1 is recognized by N_1.

A_2 is recognized by N_2.

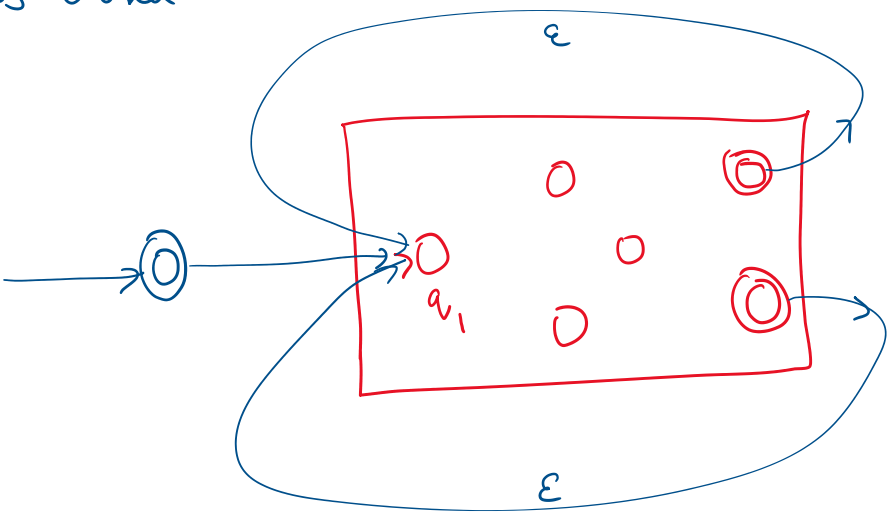


Every w such that $w = w_1 w_2$ where $w_1 \in A_1$ and $w_2 \in A_2$ will be accepted by

$w_1 \in A_1$ and $w_2 \in A_2$ will be accepted by the above machine.

Exercise: Read the details.

<u>Theorem 1.49</u>: The class of regular languages is closed under the star operation.



1) Add ε transitions from accepting states to start state.

2) Unless $q_1 \in F$, (1) does not ensure that ε is accepted. If we connect q_1 to an accepting state, then it may lead to acceptance of undesirable strings.

Suppose original machine is N, where

Suppose original machine is N_1 where

$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$. We will construct

$N = (Q, \Sigma, \delta, q_0, F)$, where ...

$Q = Q_1 \cup \{q_0\}$. q_0 is a new state.

$F = F_1 \cup \{q_0\}$.

$\delta(q, a) = \delta_1(q, a)$ if $q \notin F_1$ and $q \in Q_1$.

$\delta(q, a) = \delta_1(q, a)$ if $q \in F_1$ and $a \neq \varepsilon$.

$\delta(q, \varepsilon) = \delta_1(q, \varepsilon) \cup \{q_1\}$ if $q \in F_1$.

$\delta(q_0, a) = \{q_1\}$ if $a = \varepsilon$

$= \phi$ if $a \neq \varepsilon$.