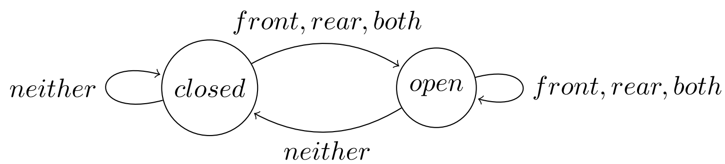
2 Lesson 2

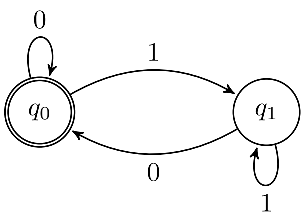
2.1 Finite Automaton

***e.g.* 2.1.** *Automatic Door*



	$front$	$rear$	$both$	$neither$
$front$	✓	×	✓	×
$rear$	×	✓	✓	×

***e.g.* 2.2.** $L = \{w \in \{0,1\}^* | w = w_1 w_2 \cdots w_n, w_n = 0\}$



***remark*.** q_0 *: accepted state*

$Q = \{q_0, q_1\}, \Sigma = \{0,1\}, \delta : Q \times \Sigma \to Q$

	0	1
q_0	q_0	q_1
q_1	q_0	q_1

***def* 2.1.** *(finite automaton)*

A ***finite automaton*** *is a 5-tuple* $(Q, \Sigma, \delta, q_0, F)$*, where:*

1. Q *is a finite set called the states*

2. Σ *is the alphabet*

3. $\delta : Q \times \Sigma \to Q$ *is the transition function*

4. q_0 *is the start state*

5. $F \subseteq Q$ *is the set of accept states*

***def* 2.2.** *Let* $M = (Q, \Sigma, \delta, q_0, F)$ *be a finite automaton, let* $w = w_1 w_2 \cdots w_n$ *be a string, where each* $w_i \in \Sigma$.

Then M *accept* w *if there is a sequence of states* $r_0, r_1, \cdots, r_n \in Q$, *such that:*

1. $r_0 = q_0$

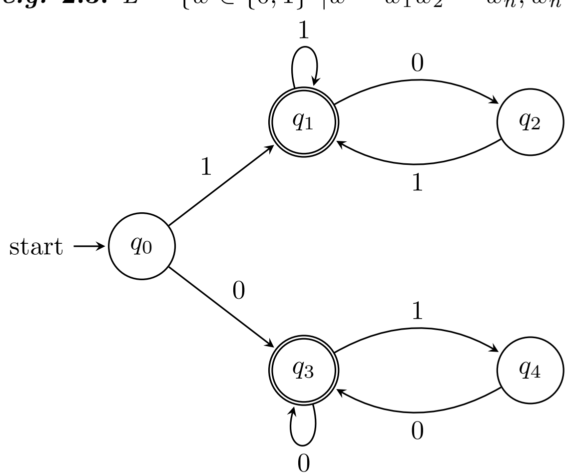
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, *for* $i = 0, 1, 2, \cdots, n-1$

3. $r_n \in F$

***def* 2.3.** *If L is the set of strings that M accepts, we say L is the language of M, and write L(M) = L, we say M recognizes/decides/accepts L.*

If M accepts no string, it recognizes one language namely, the empty language.

***e.g.* 2.3.** $L = \{w \in \{0,1\}^* | w = w_1 w_2 \cdots w_n, w_n = w_1\}$



2.2 Regular Language

def* 2.4.** *(regular language)* $L \subseteq \Sigma^*$ *is a* ***regular language *if there is a finite automaton that accepts* L

Let $A, B \subseteq \Sigma^*$, *define:*

- *(union)* $A \cup B = \{x \in \Sigma^* | x \in A \text{ or } x \in B\}$
- *(concatenation)* $AB = \{xy | x \in A, y \in B\}$

- *(star)* $A^* = \{x_1x_2\cdots x_k | k \geq 0, x_1, x_2, \cdots, x_k \in A\}$

***thm* 2.5.** *If A_1, A_2 are regular languages, so is $A_1 \cup A_2$*

Proof. *Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_{10}, F_1)$ accepts A_1, $M_2 = (Q_2, \Sigma_2, \delta_2, q_{20}, F_2)$ accepts A_2, construct $M = (Q, \Sigma, \delta, q_0, F)$:*

1. $Q = Q_1 \times Q_2 = \{(r_1, r_2) | r_1 \in Q_1, r_2 \in Q_2\}$
2. *$\delta : Q \times \Sigma \to Q$ is defined as for each $(r_1, r_2) \in Q$, and each $a \in \Sigma$, let $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$*
3. $q_0 = (q_{10}, q_{20})$
4. *$F = \{(r_1, r_2) | r_1 \in F_1$ or $r_2 \in F_2\}$*

$\square$

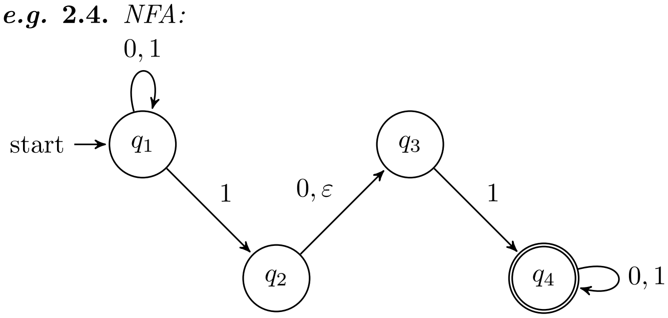
***remark*.** *so is $A \cap B$*

***thm* 2.6.** *If A_1, A_2 are regular languages, so is A_1A_2*

- ***DFA**: deterministic finite automaton*
- ***NFA**: nondeterministic...*

If at least one of these processes accepts, then the entire computation accepts.

***e.g.* 2.4.** *NFA:*



input: 010110

***e.g.* 2.5.** *Let A be the language consisting of all strings over $\{0, 1\}$ containing a 1 in the third position from the end (e.g., 000100 is in A but 0011 is not). The following four-state NFA recognizes A.*

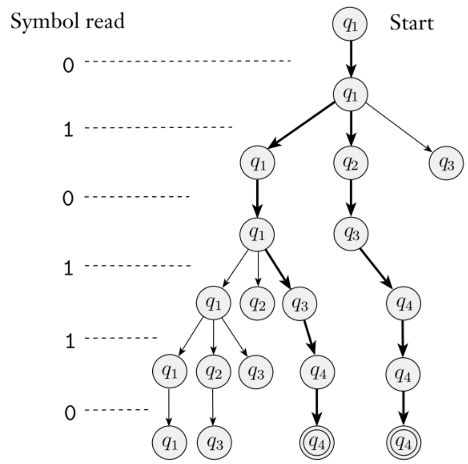


Figure 1: *The computation of NFA on input 010110*

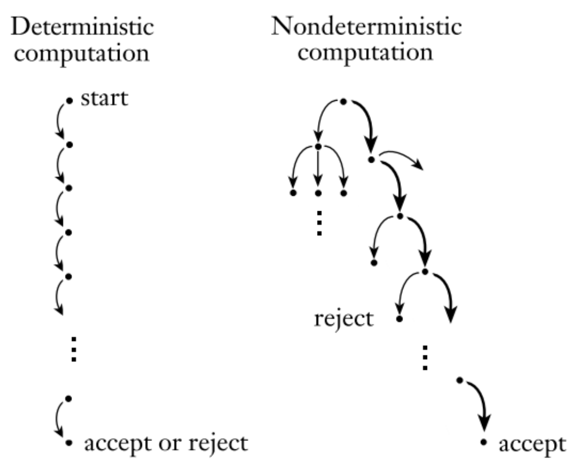
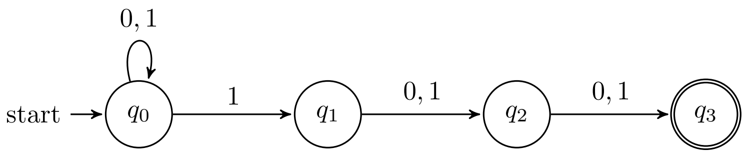
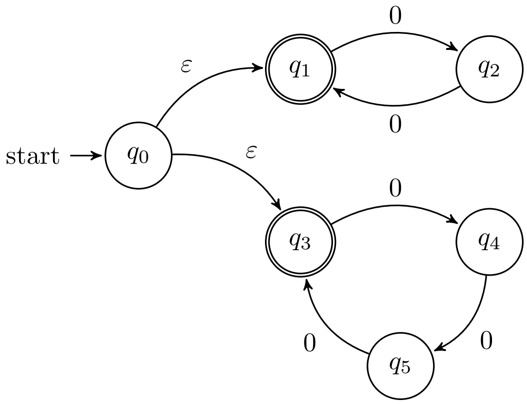


Figure 2: *Deterministic and nondeterministic computations with an accepting branch*



***e.g.* 2.6.** $L = \{0^k, 2|k \text{ or } 3|k\}$



***def* 2.7.** *(NFA)*

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

1. Q *is a finite set of states*
2. Σ *is the alphabet*
3. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to P(Q)$ *is the transive funtion*
4. $a_0 \in Q$ *is the start state*
5. $F \subseteq Q$ *is the set of accept states*

***def* 2.8.**

Let $N = (Q, \Sigma, \delta, q_0, F)$ *be an NFA, and let* $w \in \Sigma^*$*. Say* N *accepts* w *if we can write* $w = y_1 y_2 \cdots y_m$*, where* $y_i \in \Sigma \cup \{\varepsilon\}$*, and there exist* $r_0, r_1, \cdots, r_m \in Q$*, such that:*

1. $r_0 = q_0$
2. $r_{i+1} \in \delta(r_i, y_{i+1})$ *for* $i = 0, 1, \cdots, m-1$
3. $r_m \in F$

***thm* 2.9.** *Every NFA has an equivalent DFA*

Proof. □