Lecture 5: Nondeterministic Finite Automata

- In a DFA,
  - *each* symbol read causes a transition to the next state, which is *completely* determined by the current state and current symbol (i.e., there is exactly one next state).
- In an NFA,
  - some state may have more than one outgoing edge labeled with the same symbol
  - some edges may be labeled with e, the empty word.

Therefore, an input string may drive the automaton through more than one path.

In addition, some state may not have any outgoing edge labeled with a particular symbol (can get stuck).

Nondeterministic finite automata (NFA)

NFA are not realistic models of computers.

They are only a convenient specification of finite automata – NFA are much easier to design than DFA.

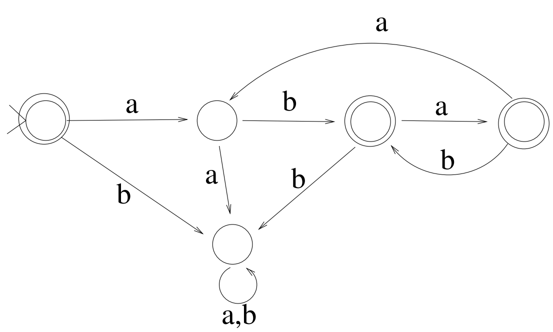
We will show in the next lecture that every NFA can be converted into an equivalent DFA; that is, non-determinism does not provide additional computation power.

Nondeterministic finite automata (NFA)

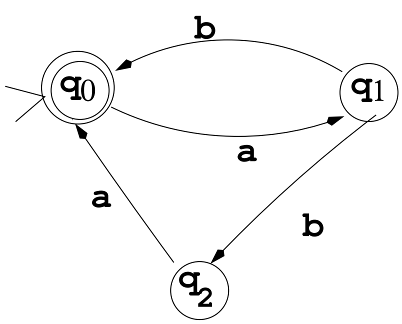
Example:

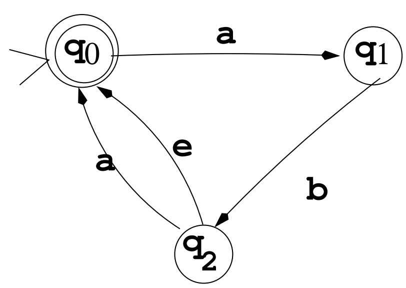
$L = (ab \cup aba)^*$

DFA:



NFAs:





Nondeterministic finite automata (NFA)

Formally, an NFA is a quintuple $M = (K, \Sigma, \Delta, s, F)$, where

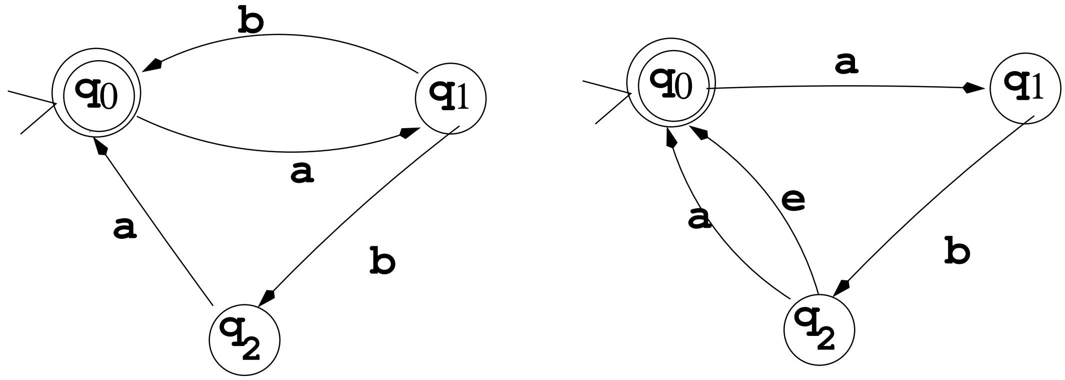
1. K — a finite set of states,
2. Σ — an alphabet,
3. $s \in K$ — the initial state,
4. $F \subseteq K$ — the set of final states,
5. Δ — the transition relation:
   - $\Delta \subseteq K \times (\Sigma \cup \{e\}) \times K$.

The only difference with DFA is Δ.

Note that a DFA is also an NFA – define $\Delta = \{(q, \sigma, p) : \delta(q, \sigma) = p\}$.

Nondeterministic finite automata (NFA)

Examples:



- Left NFA:

  1. $K = \{q_0, q_1, q_2\}$, $s = q_0$, $F = \{q_0\}$, $\Sigma = \{a, b\}$;
  2. $\Delta = \{(q_0, a, q_1), (q_1, b, q_0), (q_1, b, q_2), (q_2, a, q_0)\}$

     e.g., (q_1, b, q_0) means that, when in state q_1, the machine **may** consume b and enter state q_0.

- Right NFA:

  1. $K = \{q_0, q_1, q_2\}$, $s = q_0$, $F = \{q_0\}$, $\Sigma = \{a, b\}$;
  2. $\Delta = \{(q_0, a, q_1), (q_1, b, q_2), (q_2, a, q_0), (q_2, e, q_0)\}$

Nondeterministic finite automata (NFA)

- Like for DFAs, a configuration for NFA is an element of $K \times \Sigma^*$.
- we write $(q, w) \vdash_M (q', w')$ iff there exists $u \in \Sigma \cup \{e\}$ such that $w = uw'$ and $(q, u, q') \in \Delta$.
- A string w is **accepted** by M iff

  $$(s, w) \vdash_M^* (q, e) \text{ and } q \in F$$

  Although the acceptance condition looks like the one for DFA, the definition of $\vdash_M$ is different. Here, a string is rejected only if all possible computation sequences do not end in an accepting state. It is accepted if one or more computation sequences end in an accepting state.
- The **language accepted by M**:

  $$L(M) = \{w : w \text{ accepted by } M\}.$$

Nondeterministic finite automata (NFA)

Examples of computation by NFA.
Let $M = (K, \Sigma, \Delta, s, F)$ where
$K = \{q_0, q_1, q_2\}$, $s = q_0$, $F = \{q_0\}$, $\Sigma = \{a, b\}$
$\Delta = \{(q_0, a, q_1), (q_1, b, q_0), (q_1, b, q_2), (q_2, a, q_0)\}$

- One possible computation:

$$\begin{aligned} (q_0, aba) &\vdash_M (q_1, ba) \\ &\vdash_M (q_0, a) \\ &\vdash_M (q_1, e) \end{aligned}$$

$$(q_0, aba) \vdash_M^* (q_1, e), q_1 \text{ not an accepting state}$$

- Another possible computation:

$$\begin{aligned} (q_0, aba) &\vdash_M (q_1, ba) \\ &\vdash_M (q_2, a) \\ &\vdash_M (q_0, e) \end{aligned}$$

$$(q_0, aba) \vdash_M^* (q_0, e), q_0 \text{ is an accepting state}$$

  Hence aba is in $L(M)$.

- All possible computation sequences of aba can be represented as a computation tree.