Finite Automata (Finite State Machines)

- A finite automata is a mathematical model to recognize if a string is in a language.

Example:

- states: q0, q1, q2, q3
- input symbols: a, b
- transitions: edges
- start (initial) state: q0
- final (accepting) state(s): q3
Finite Automata

- A finite automaton (FA) $M$ is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots, q_n\}$ is a finite set of states
  - $\Sigma$ is a finite input alphabet
  - $\delta$ is a transition function mapping $Q \times \Sigma$ to $Q$.
  - $q_0 \in Q$ is the initial state.
  - $F \subseteq Q$ is the set of final states.
How A Finite Automaton Works

An FA works as follows. It is on the current state in Q, reading a string of symbols from $\Sigma$ written on a tape. In one move, it is in state $q$ and reading in the symbol $a$ and moves to the state $\delta(q, a)$ and moves its head to the next symbol.
Transition Function $\hat{\delta}$ on String

$\hat{\delta}$ is a transition function mapping $Q \times \Sigma^*$ to $Q$.

1. $\hat{\delta}(q, \varepsilon) = q$

2. For all strings $w = xa$, where $x$ is a string and $a$ is symbol in $\Sigma$
   $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$
Language Defined by A Finite Automaton

- A string $x$ is said to be accepted by a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ if $\hat{\delta}(q_0, x) = p$ for some $p$ in $F$.

- A language accepted by $M$, designated $L(M)$, is the set
  $\{x| \hat{\delta}(q_0, x) \text{ is in } F\}$
Algorithm for FA

q = initial state
s = the first symbol of the string
while (s != blank) do
    q = δ(q, s)
    s = next symbol

if q is in F then accept, otherwise reject.
Nondeterministic Finite Automata (NFA)

- A NFA $M$ is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_0, q_1, \ldots, q_n\}$ is a finite set of states
  - $\Sigma$ is a finite input alphabet
  - $\delta$ is a transition function mapping $Q \times \Sigma$ to $2^Q$.
    $2^Q$ is the power set of $Q$, the set of all subsets of $Q$.
  - $q_0 \in Q$ is the initial state.
  - $F \subseteq Q$ is the set of final states.
Transition Function $\hat{\delta}$ on String

$\hat{\delta}$ is a transition function mapping $Q \times \Sigma^*$ to $2^Q$.

1. $\hat{\delta}(q, \varepsilon) = \{q\}$

2. For all strings $w = xa$, where $x$ is a string and $a$ is symbol in $\Sigma$
   
   $\hat{\delta}(q, xa) = \bigcup_{i=1}^{k} \delta(p_i, a)$

   where $\hat{\delta}(q, x) = \{p_1, \ldots, p_k\}$
Language Defined by An NFA

- There may exist more than one path for a string $x$ in an NFA. This means that a string may end up at more than one state or no state.
  \[ \hat{\delta}(q_0, x) = P, \text{ where } P \text{ is a subset of } Q \]
- A string $x$ is said to be accepted by a nondeterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ if $\hat{\delta}(q_0, x) = P$ and $P$ contains some final state.
- A language accepted by NFA $M$, designated $L(M)$, is the set \[ \{x | \hat{\delta}(q_0, x) \text{ contains some state in } F\} \]
Example of NAF
Equivalence of DFA and NFA

• Theorem:
  Let L be a set accepted by a nondeterministic finite automaton. Then there exists a deterministic finite automaton that accepts L.

• Theorem:
  Let L be a set accepted by a deterministic finite automaton. Then there exists a nondeterministic finite automaton that accepts L.
Equivalence of DFA and NFA

• Theorem:
  Let $L$ be a set accepted by a nondeterministic finite automaton. Then there exists a deterministic finite automaton that accepts $L$.

• Theorem:
  Let $L$ be a set accepted by a deterministic finite automaton. Then there exists a nondeterministic finite automaton that accepts $L$. 