Non-Deterministic Finite Automata
At each state, we can have *any* number of out arrows for some letter $\sigma \in \Sigma$, including $\varepsilon$.

Set of strings accepted by this NFA = \{w \mid w \text{ contains a } 0\}
Multiple Start States

We allow *multiple* start states for NFAs, and Sipser allows only one.

Can easily convert NFA with many start states into one with a single start state:
Yet Another Example of an NFA

\[ L(M) = \{1, 00\} \]
A non-deterministic finite automaton (NFA) is a 5-tuple \( N = (Q, \Sigma, \delta, Q_0, F) \) where

- \( Q \) is the set of states
- \( \Sigma \) is the alphabet
- \( \delta : Q \times \Sigma_\epsilon \rightarrow 2^Q \) is the transition function
- \( Q_0 \subseteq Q \) is the set of start states
- \( F \subseteq Q \) is the set of accept states

- \( 2^Q \) is the set of all possible subsets of \( Q \)
- \( \Sigma_\epsilon = \Sigma \cup \{\epsilon\} \)
Def. Let \( w \in \Sigma^* \). Let \( N \) be an NFA. \( N \) accepts \( w \) if there’s a sequence of states \( r_0, r_1, ..., r_k \in Q \) and \( w \) can be written as \( w_1 ... w_k \) with \( w_i \in \Sigma \cup \{\varepsilon\} \) s.t.

1. \( r_0 \in Q_0 \)
2. \( r_{i+1} \in \delta(r_i, w_{i+1}) \) for all \( i = 0, ..., k-1 \), and
3. \( r_n \in F \)

\[
\text{L}(N) = \text{the language recognized by N} = \text{set of all strings machine N accepts}
\]

Language \( L' \) is recognized by an NFA \( N \) if \( L' = \text{L}(N) \).
$N = (Q, \Sigma, \delta, Q_0, F)$

$Q = \{q_1, q_2, q_3, q_4\}$

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

$Q_0 = \{q_1, q_2\}$

$F = \{q_4\} \subseteq Q$

$\delta(q_2, 1) = \{q_4\}$

$\delta(q_3, 1) = \emptyset$

$\delta(q_1, 0) = \{q_3\}$

$00 \in L(N)$?

$01 \in L(N)$?
NFAs are generally simpler than DFAs

A DFA recognizing the language \{1\}

An NFA recognizing the language \{1\}
Deterministic Computation

Non-Deterministic Computation

Are these equally powerful???
Parting thoughts:
When life hands you ambiguity define
Nondeterministic Finite Automata

Is verifying easier than computing (I)?