Nondeterministic Finite Automata (NFA)

• When an NFA receives an input symbol $a$, it can make a transition to a number (including 0) of states (each state can have multiple edges labeled with the same symbol).

• An NFA accepts a string $w$ iff there exists a path labeled $w$ from the initial state to one of the final states.
Example N1

- The language of the following NFA consists of all strings over \( \{0, 1\} \) whose 3\(^{rd}\) symbol from the right is 0.

- Note \( Q_0 \) has multiple transitions on 0.
Example N2

• The NFA $N_2$ accepts strings beginning with 0.

• Note $Q_0$ has no transition on 1
NFA Processing

Suppose $N_1$ receives the input string $0011$. There are three possible execution sequences:

- $q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_0$
- $q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3$
- $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3$

Only the second finishes in an accept state. The third even gets stuck (cannot even read the fourth symbol).
Implementation

• Implementation of NFAs has to be deterministic, using some form of backtracking to go through all possible executions.

• Any thoughts on how this might be accomplished?
Formal Definition

• An NFA is a quintuple $A = (Q, \Sigma, s, F, \delta)$, where the first four components are as in a DFA, and the transition function takes values in $P(Q)$ instead of $Q$. Thus
  $\delta : Q \times \Sigma \rightarrow P(Q)$

• The extension $\delta : Q \times \Sigma^* \rightarrow P(Q)$ is defined by
  $\delta(q, \epsilon) = \{q\}$
  $\delta(q, ua)$ is the union of the sets $\delta(p, a)$, where $p$ varies over all states in $\delta(q, u)$
  $\bigcup_{p \in \delta(q, u)} \delta(p, a),$
NFA Acceptance

• An NFA accepts a string $w$ iff $\delta(s, w)$ contains a final state. The language of an NFA $N$ is the set $L(N)$ of accepted strings:

$$L(N) = \{w \mid \delta(s, w) \cap F \neq \emptyset\}$$
compute $\delta(q_0, 000)$

- $\delta(q, ua) = \bigcup_{p \in \delta(q, u)} \delta(p, a)$
- $\delta(q_0, 000) = \bigcup_{x \in \delta(q_0, 00)} \delta(x, 0)$
- $\delta(q_0, 00) = \bigcup_{y \in \delta(q_0, 0)} \delta(y, 0)$
- $\delta(q_0, 00) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\delta(q_0, 00) = \bigcup_{y \in \{q_0, q_1\}} \delta(y, 0)$
- $\delta(q_0, 00) = \{q_0, q_1\} \cup \{q_2\} = \{q_0, q_1, q_2\}$
- $\delta(q_0, 000) = \bigcup_{x \in \{q_0, q_1, q_2\}} \delta(x, 0)$
- $\delta(q_0, 000) = \{q_0, q_1\} \cup \{q_2\} \cup \{q_3\}$
- $\delta(q_0, 000) = \{q_0, q_1, q_2, q_3\}$
Intuition

- At any point in the walk over a string, such as “000” the machine can be in a set of states.

- To take the next step, on a character ‘c’, we create a new set of states. Those reachable from the old set on a single ‘c’
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Q0}</td>
<td>{Q0,Q1}</td>
<td>{Q0}</td>
</tr>
<tr>
<td>{Q0,Q1}</td>
<td>{Q0,Q1,Q2}</td>
<td>{Q0,Q2}</td>
</tr>
<tr>
<td>{Q0,Q2}</td>
<td>{Q0,Q1,Q3}</td>
<td>{Q0,Q3}</td>
</tr>
<tr>
<td>{Q0,Q1,Q3}</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>{Q0,Q3}</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>