Nondeterministic Finite Automata (NFA)

Lecturer: Manar Elkady, Ph.D

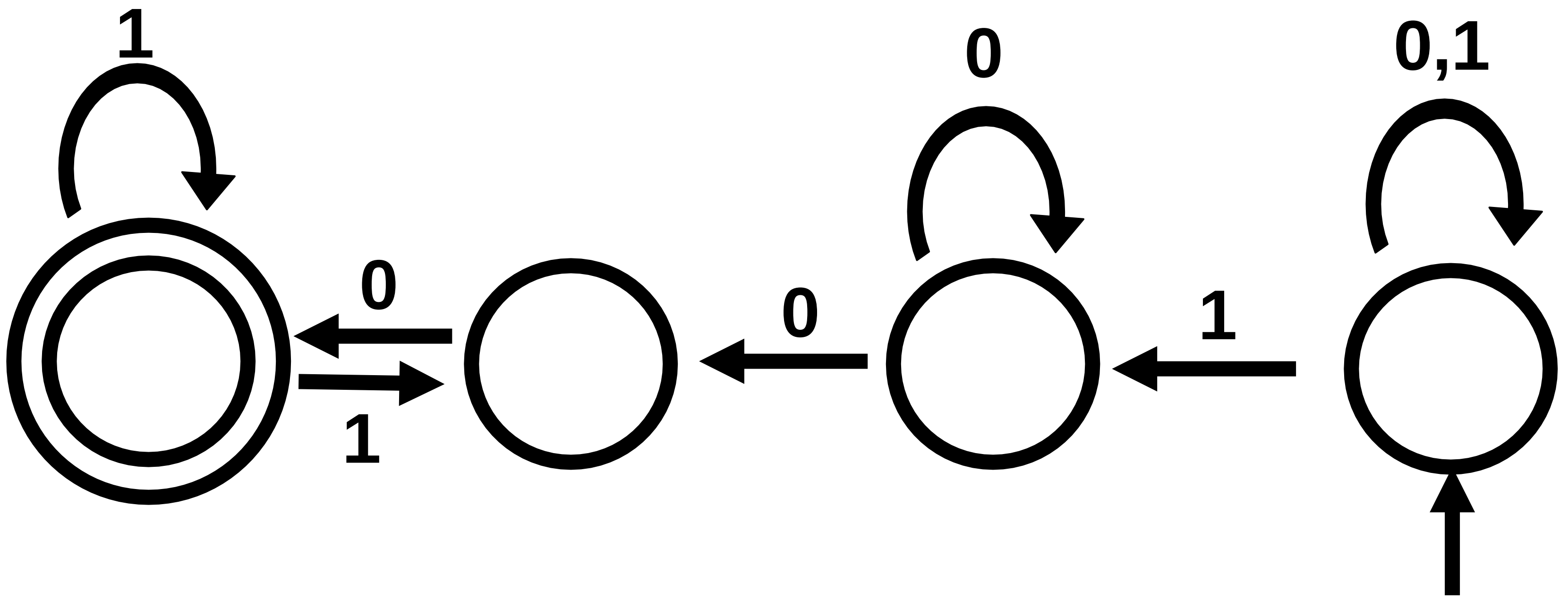
Nondeterminism

In a DFA, the machine is always in exactly one state upon reading each input symbol

In a nondeterministic FA, the machine can try out many different ways of reading the same string

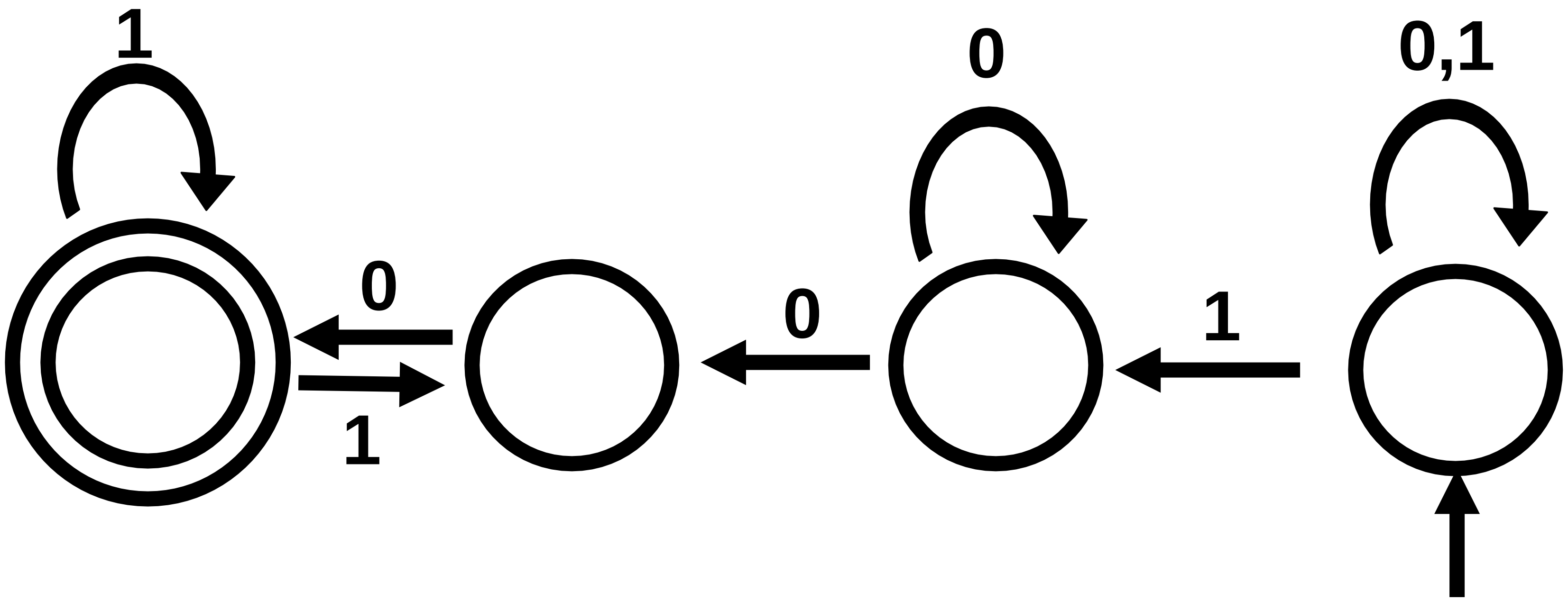
- Next symbol may cause an NFA to “branch” into multiple possible computations
- Next symbol may cause NFA’s computation to fail to enter any state at all

Nondeterminism



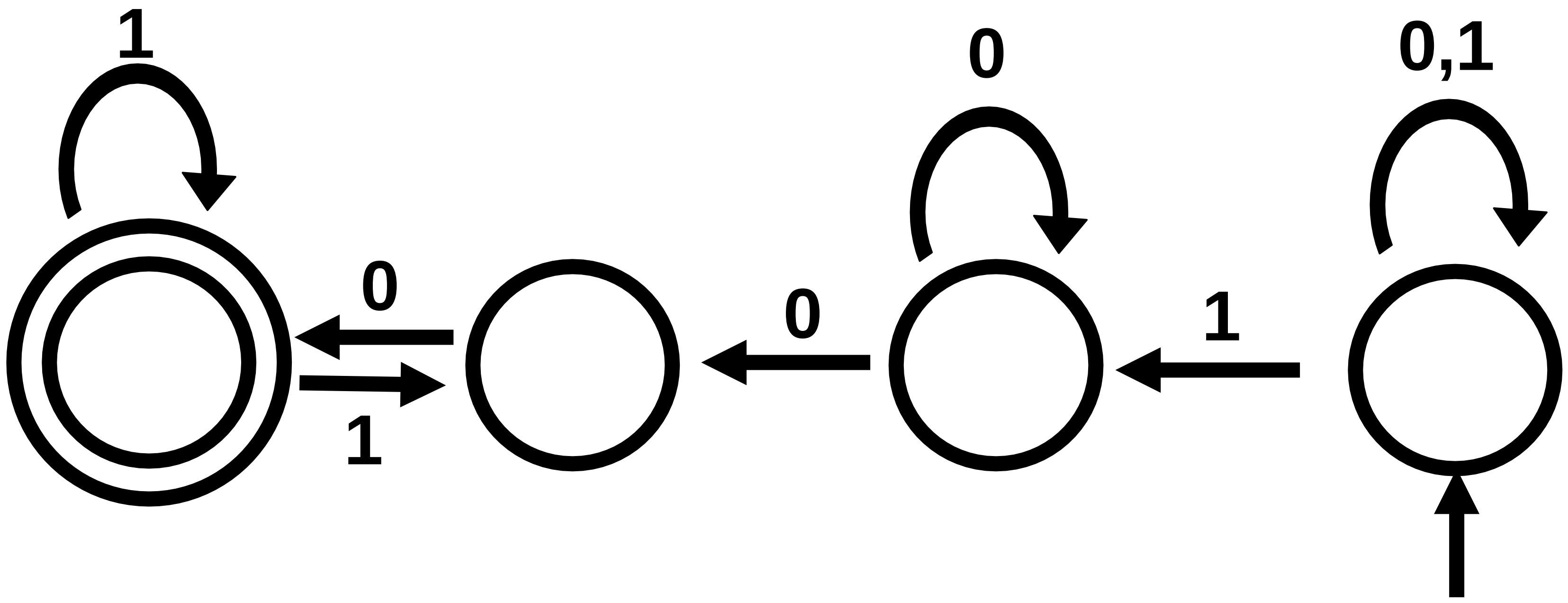
A Nondeterministic Finite Automaton (NFA) accepts if there ***exists*** a way to make it reach an accept state.

Nondeterminism



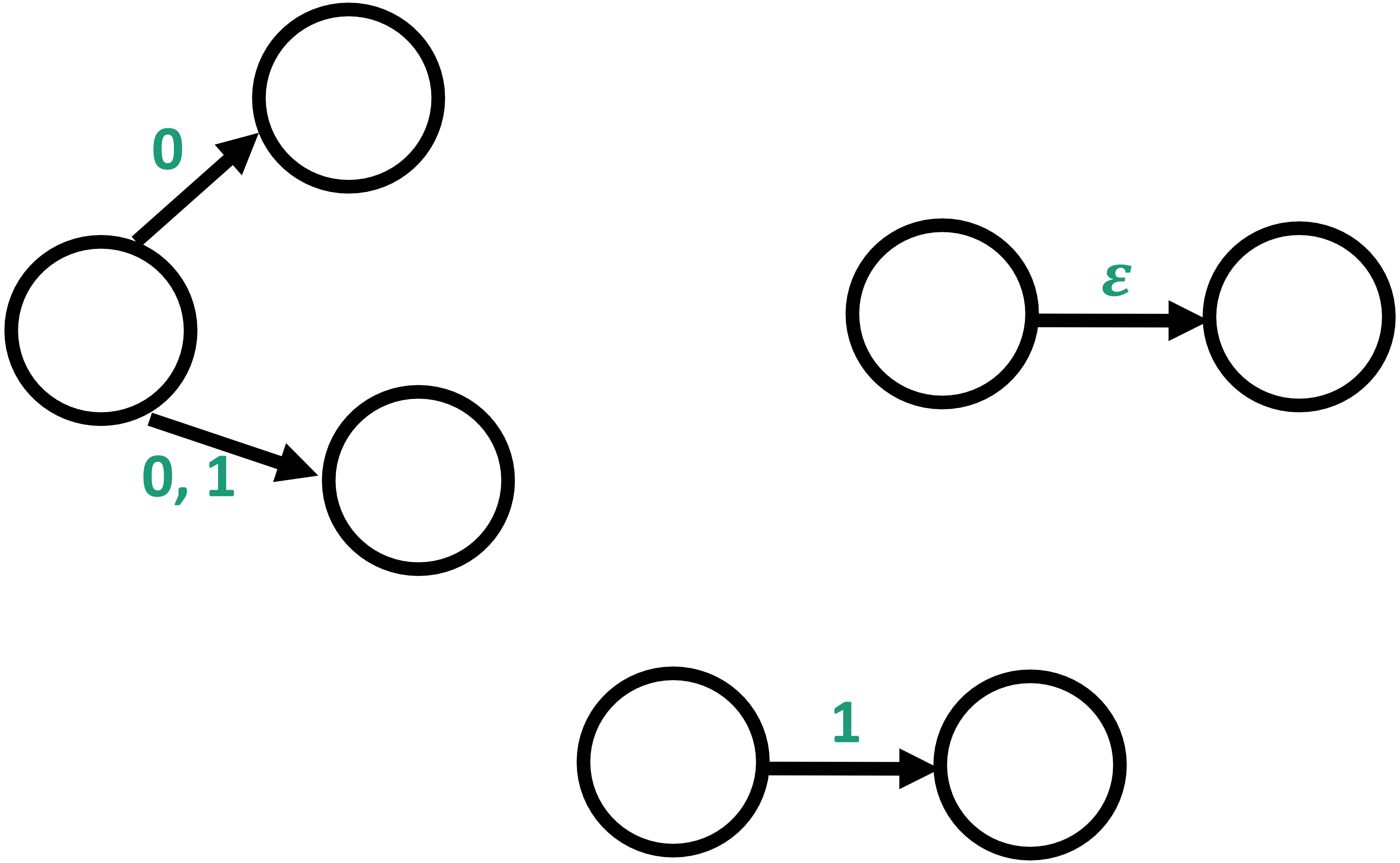
Example: Does this NFA accept the string 1100?

Nondeterminism

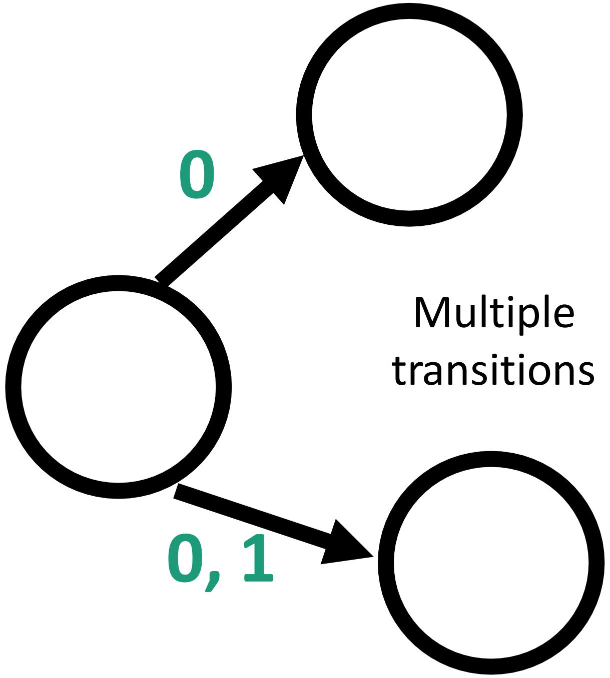


Example: Does this NFA accept the string 11?

Some special transitions

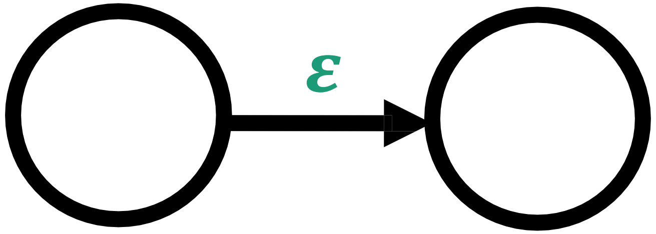


Some special transitions

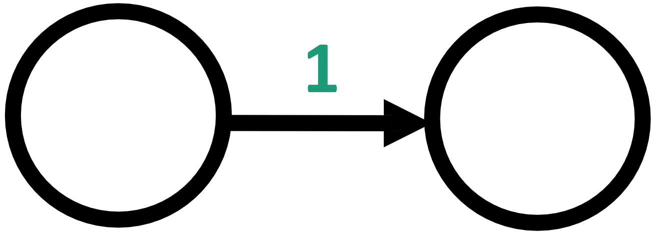


Multiple transitions

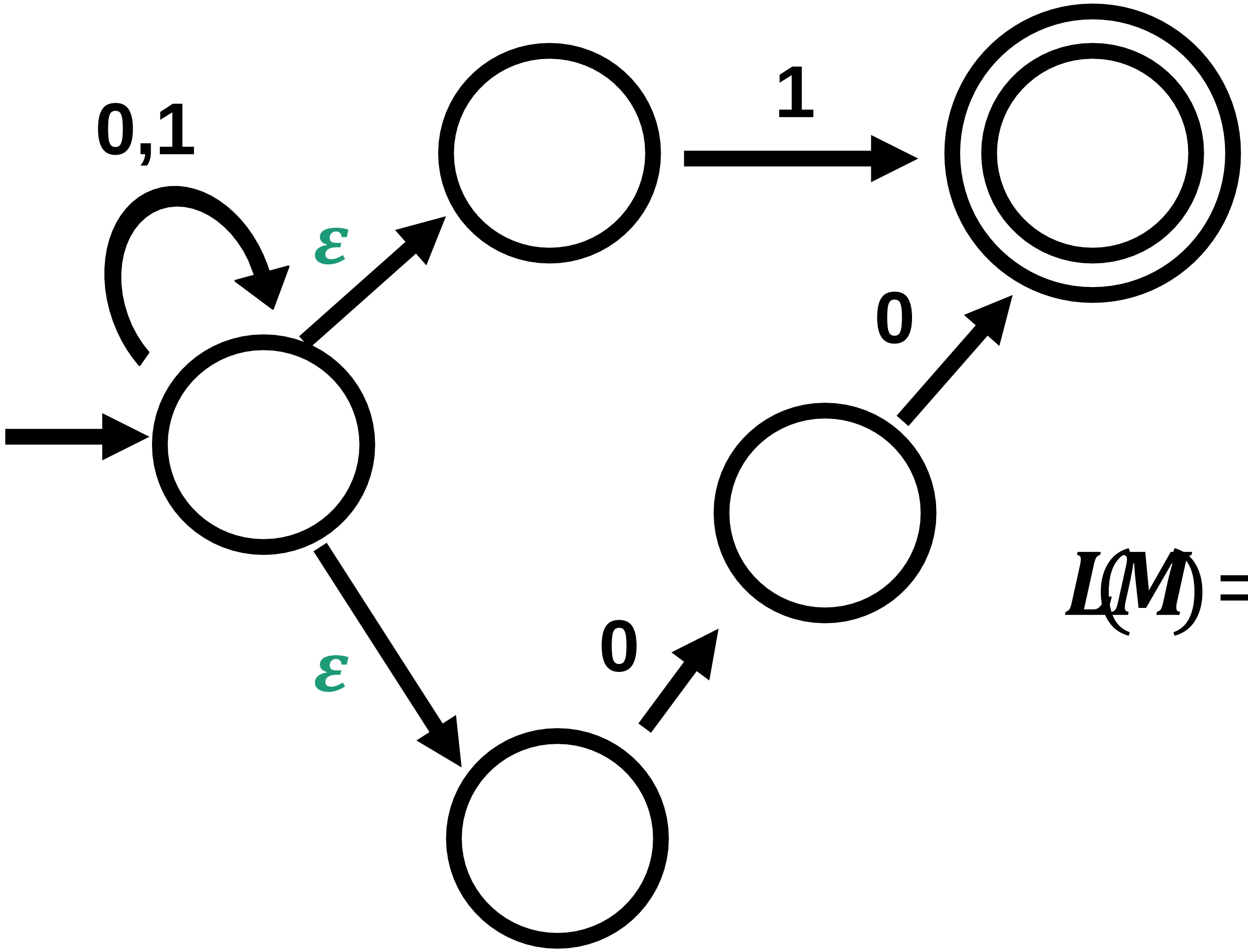
ε-transitions
(don't consume a symbol)



No transition

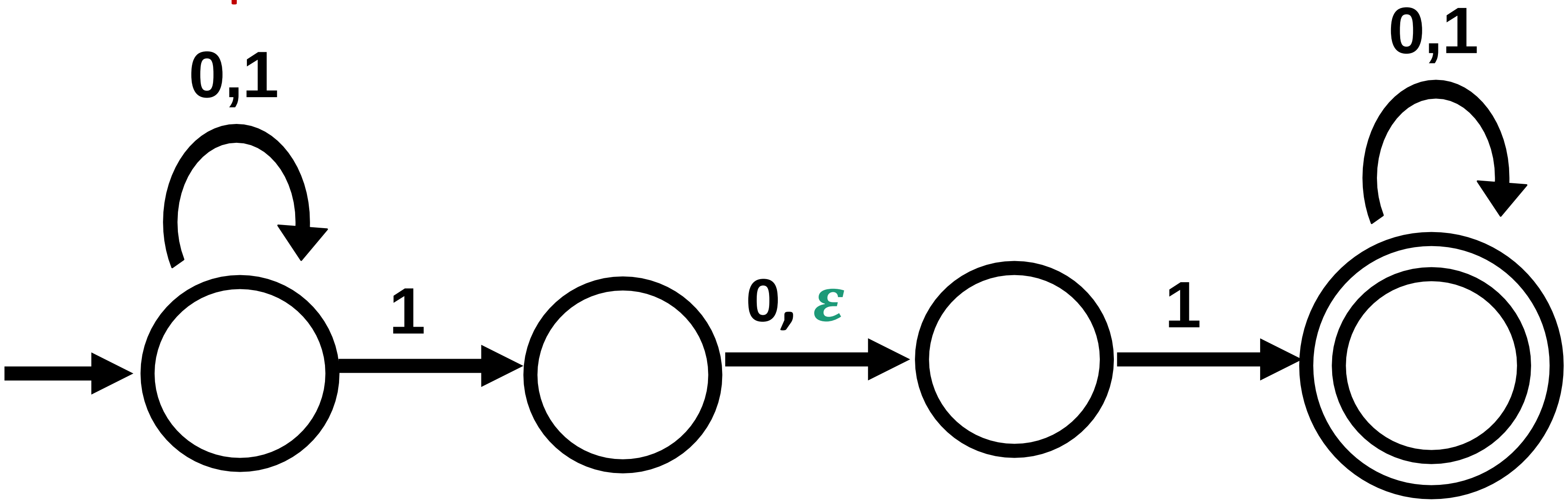


Example



$L(M) =$

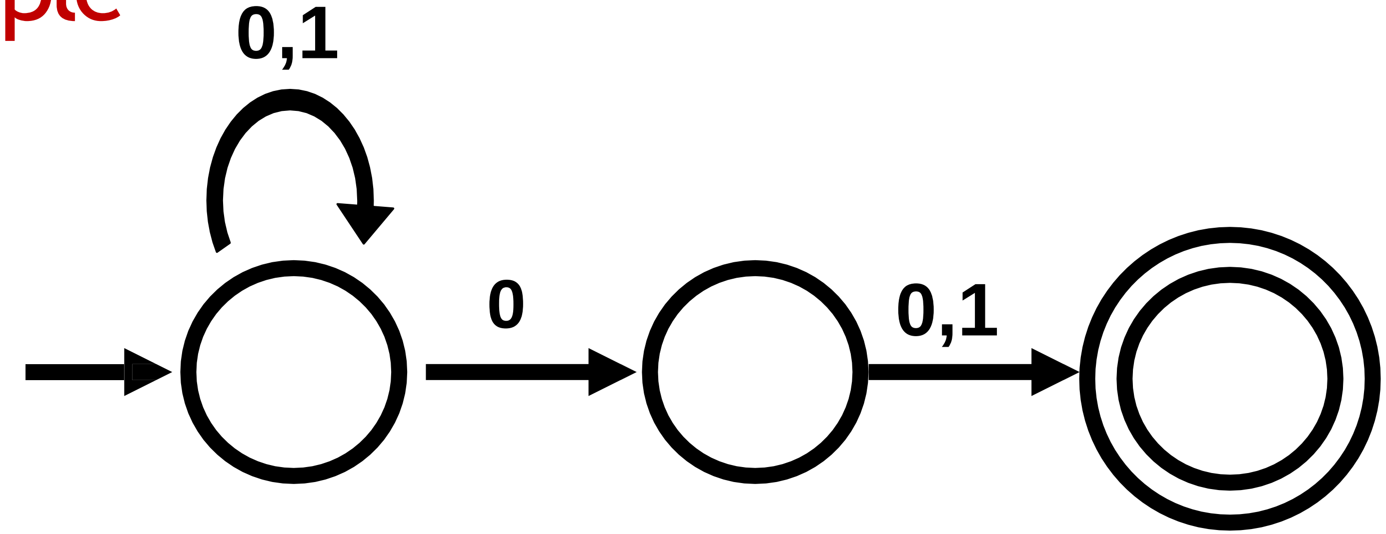
Example



$L(N) =$

a) $\{w \mid w \text{ ends with } 101\}$
b) $\{w \mid w \text{ ends with } 11 \text{ or } 101\}$
c) $\{w \mid w \text{ contains } 101\}$
d) $\{w \mid w \text{ contains } 11 \text{ or } 101\}$

Example



$L(N) =$

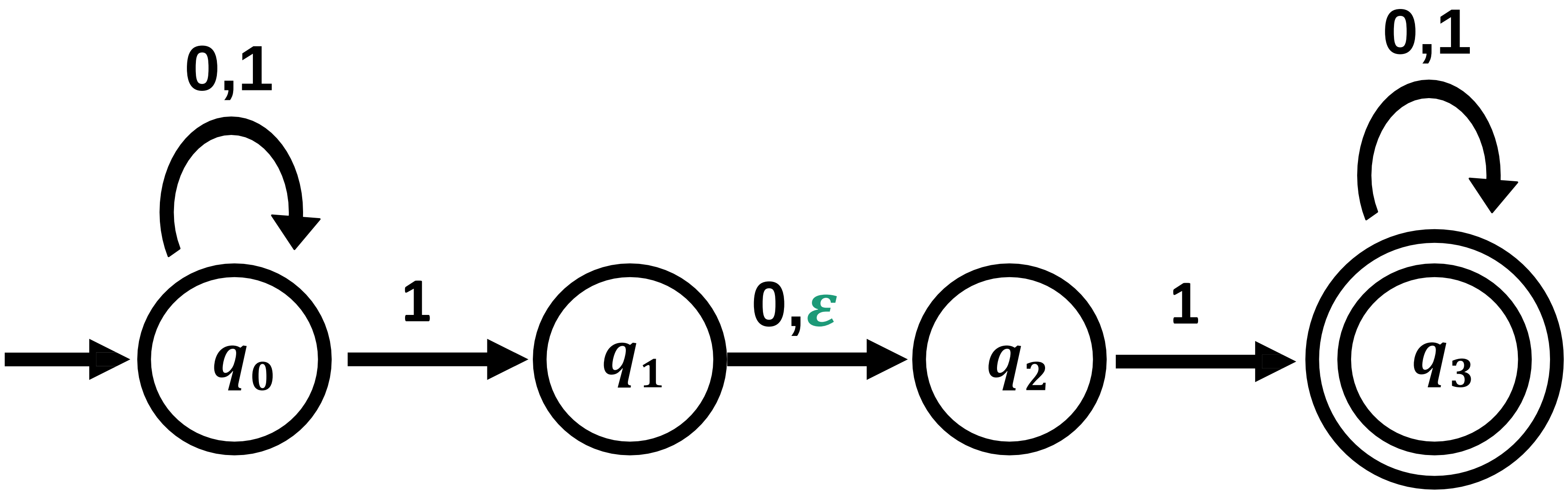
a) $\{w \mid w$ contains 00 or 01$\}$

b) $\{w \mid$ the second to last symbol of w is $\{0\}\mid$

c) $\{w \mid w$ starts with 00 or 01$\}$

d) $w \quad w$ ends with 001$\}$

Example



$= \ (Q, \Sigma, \delta, q_0, F)$

$Q \ = \ \{q_0, q_1, q_2, q_3\}$

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

$F \ = \ \{q_3\}$

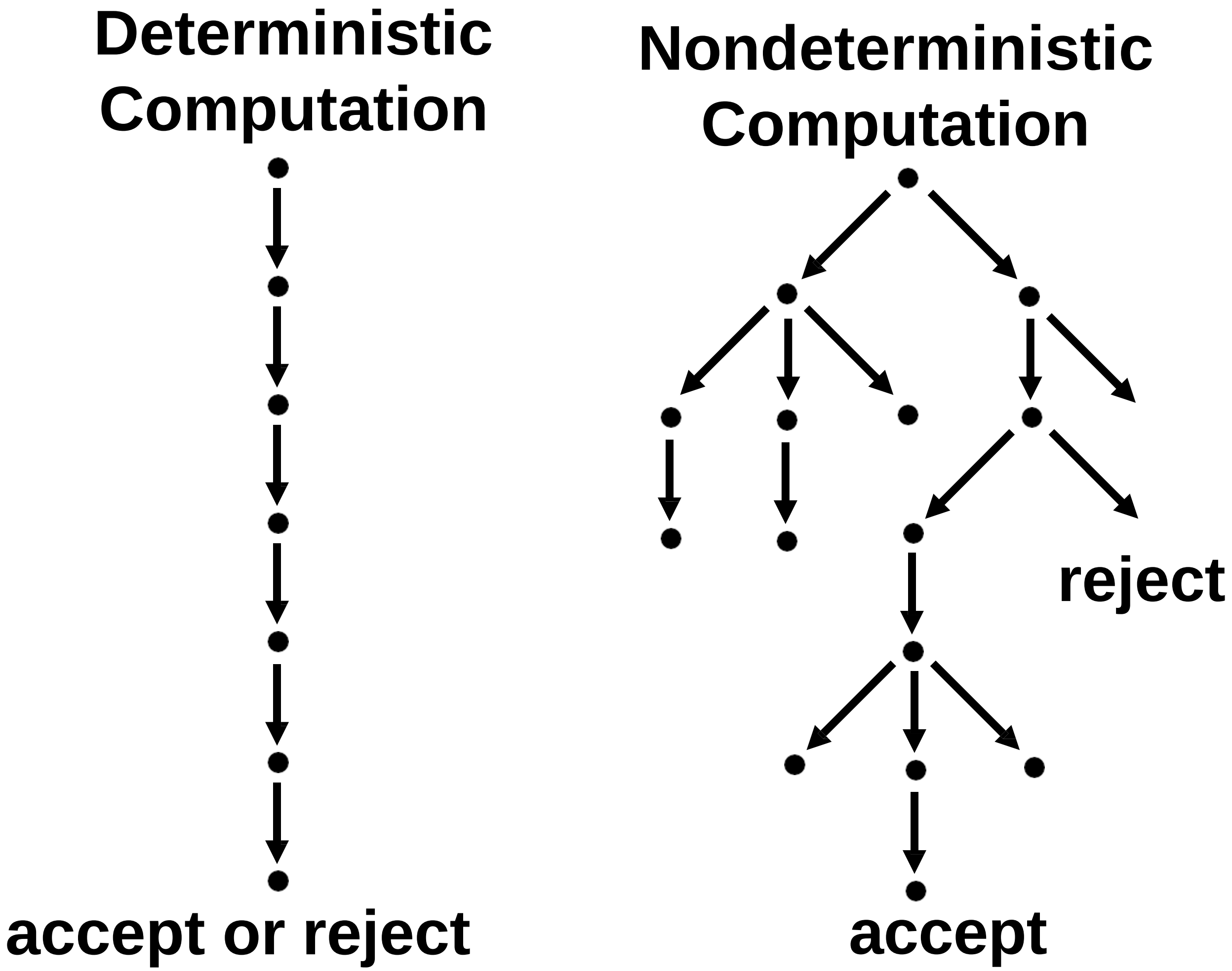
$\delta(q_0, 0) \ =$

$\delta(q_0, 1) \ =$

$\delta(q_1, \varepsilon) \ =$

$\delta(q_2, 0) \ =$

Nondeterminism



Ways to think about nondeterminism

- (restricted) parallel computation
- tree of possible computations
- guessing and verifying the "right" choice

Why study NFAs?

- Not really a realistic model of computation: Real computing devices can't really try many possibilities in parallel

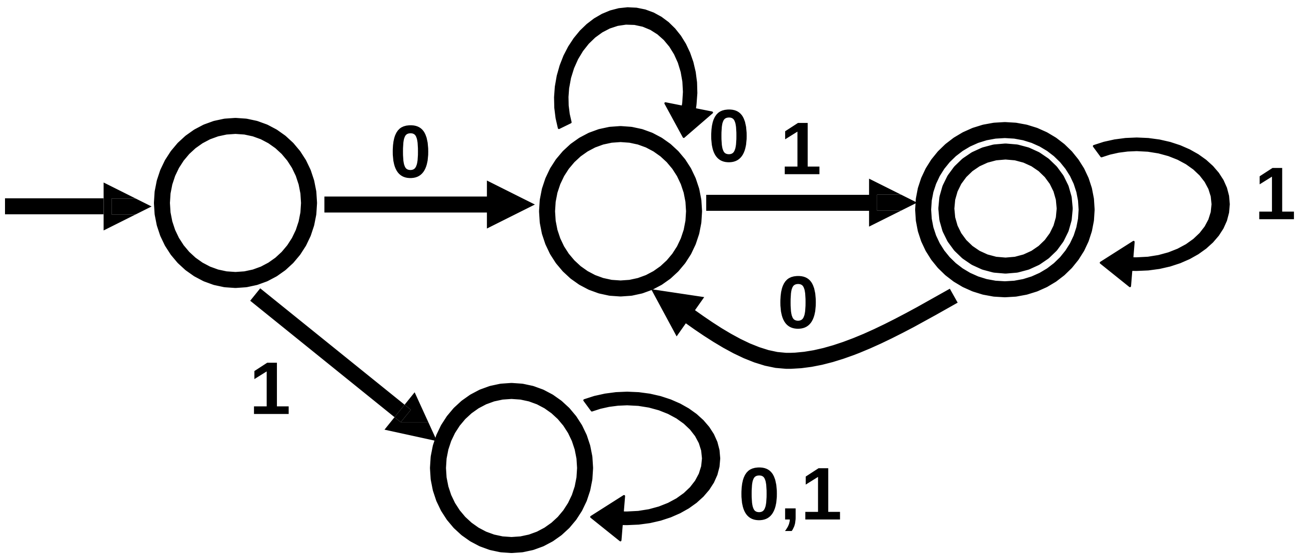
But:

- Useful tool for understanding power of DFAs/regular languages
- NFAs can be simpler than DFAs
- Lets us study "nondeterminism" as a resource

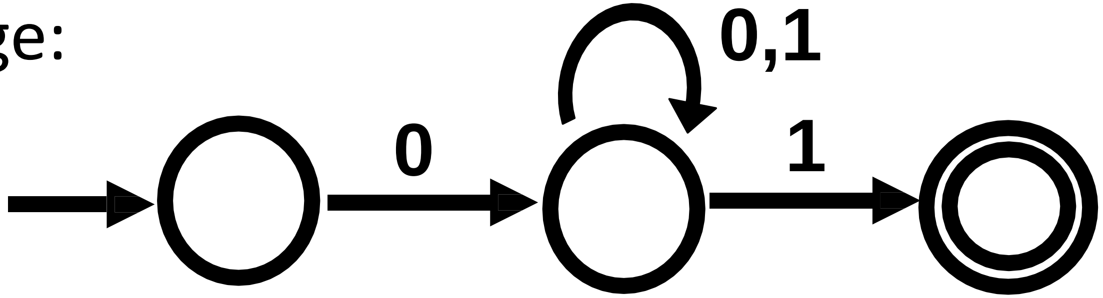
  (cf. P vs. NP)

NFAs can be simpler than DFAs

A DFA that recognizes the language $\{w \mid w$ starts with 0 and ends with 1$\}$:



An NFA for this language:



Equivalence of NFAs and DFAs

Equivalence of NFAs and DFAs

Every DFA *is* an NFA, so NFAs are *at least* as powerful as DFAs

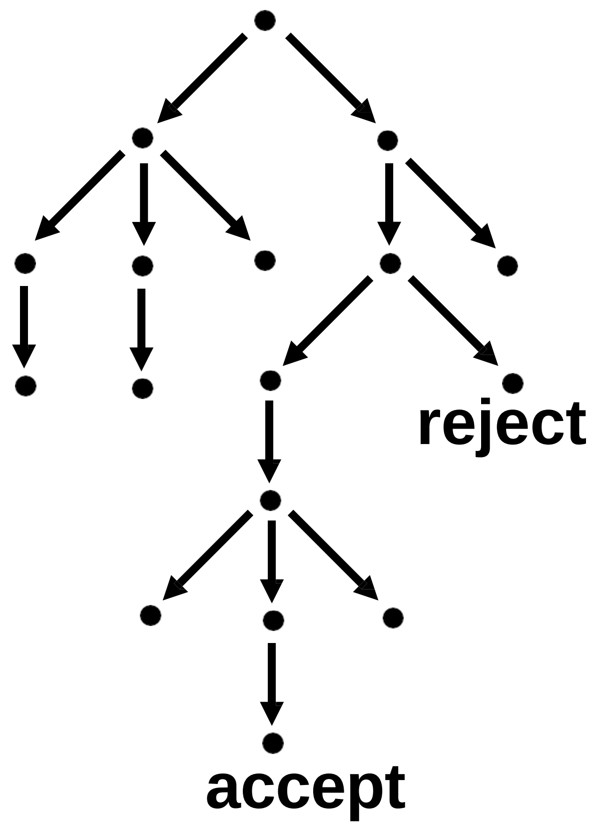
Theorem: For every NFA N, there is a DFA M such that $L(M) = L(N)$

Corollary: A language is regular if and only if it is recognized by an NFA

Equivalence of NFAs and DFAs (Proof)

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA

<u>Goal:</u> Construct DFA $M = (Q', \Sigma, \delta', q_0', F')$ recognizing $L(N)$



Intuition: Run all threads of N in parallel, maintaining the set of states where all threads are.

Formally: $Q' = P(Q)$

"The Subset Construction"