BU CS 332 – Theory of Computation

Lecture 3:

- Nondeterminism
- Equivalence of NFAs and DFAs
- More on closure properties

Reading:

Sipser Ch 1.1-1.2

HW 1 out (due Monday)

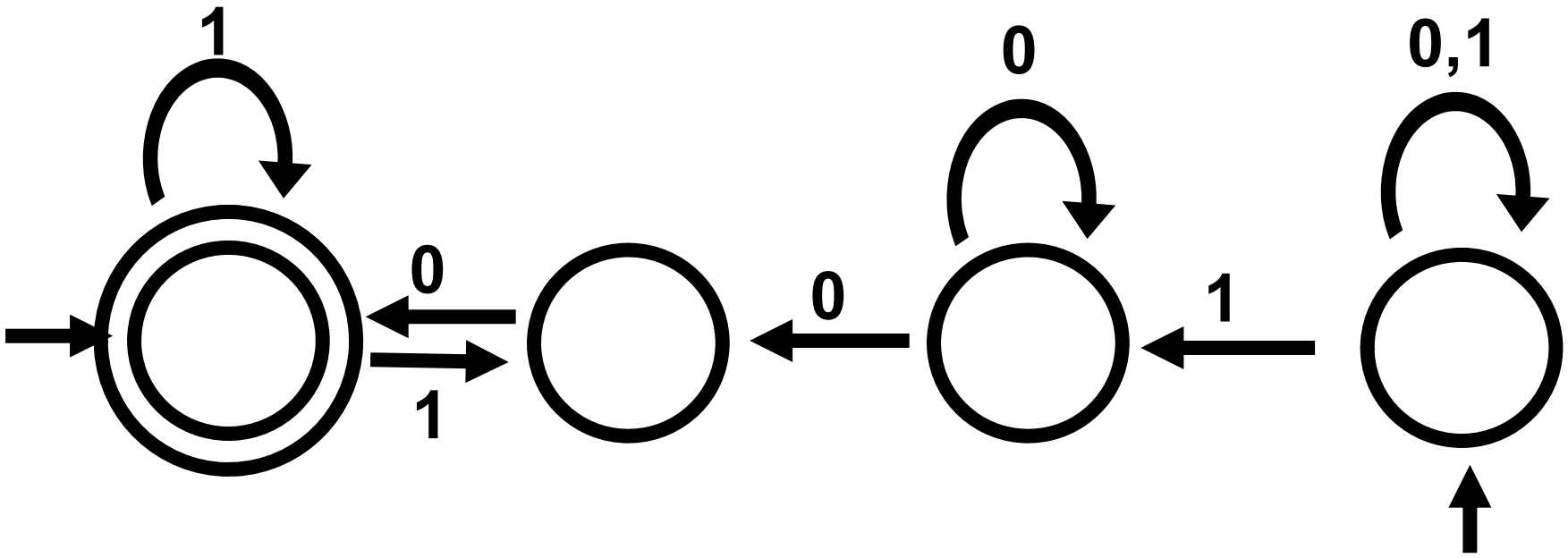
My OH: Today until 6

Nadya's OH: Fri 2:30-4:30

Mark Bun

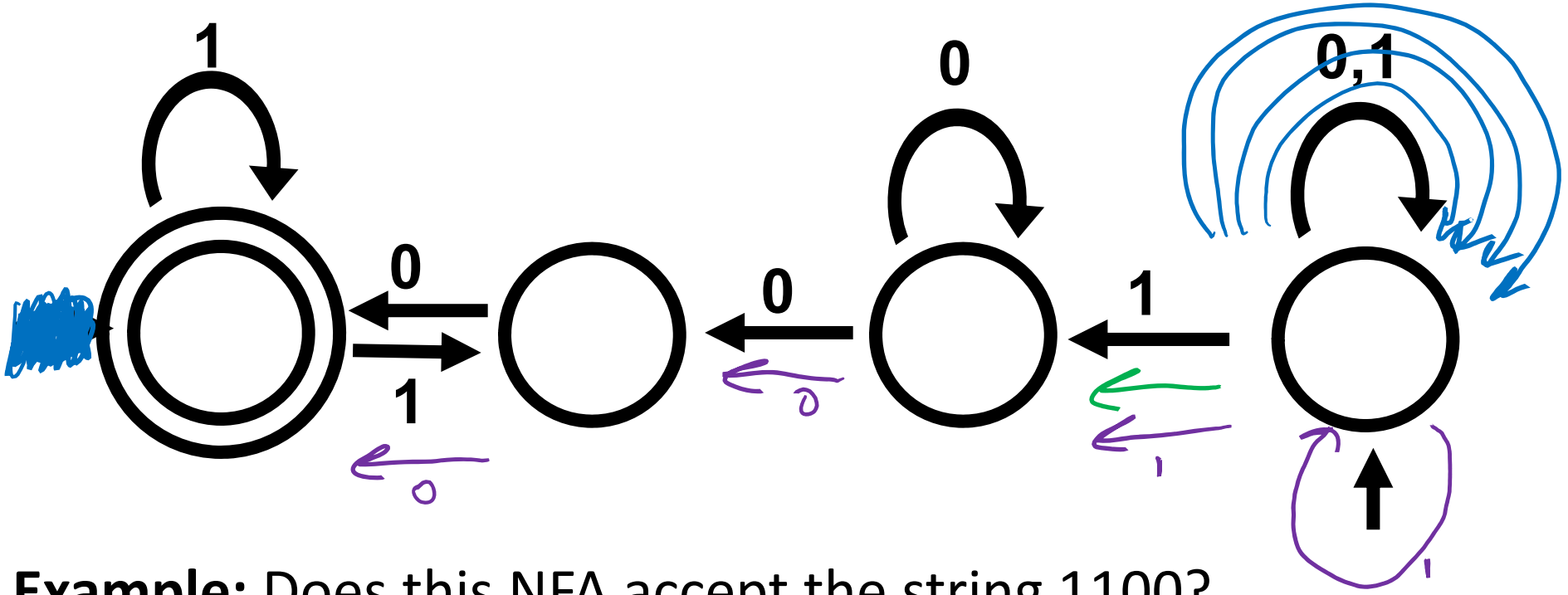
January 29, 2020

Nondeterminism



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

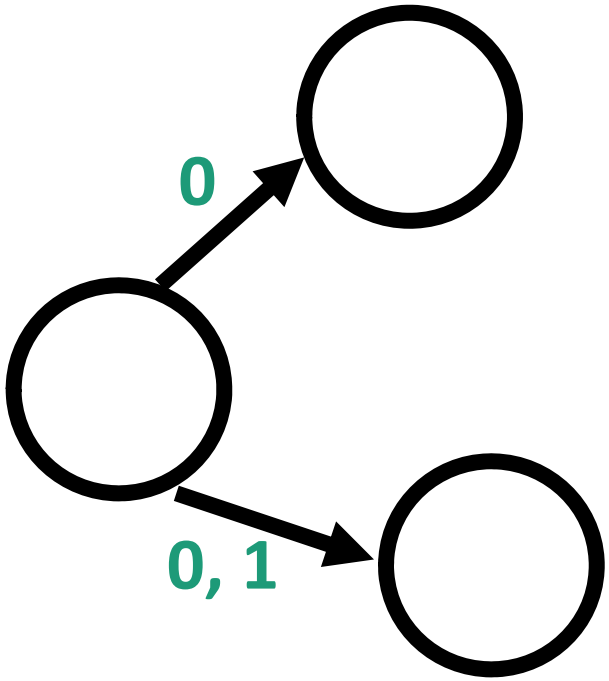
Nondeterminism

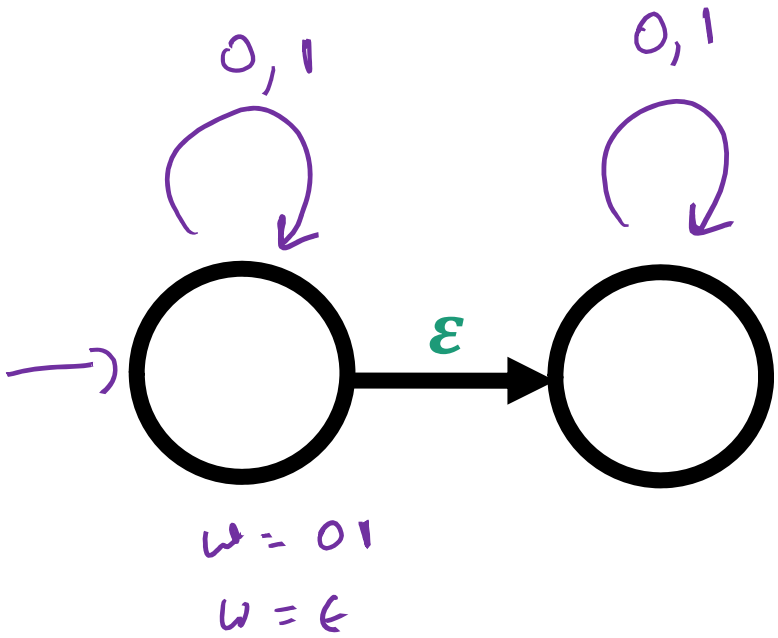


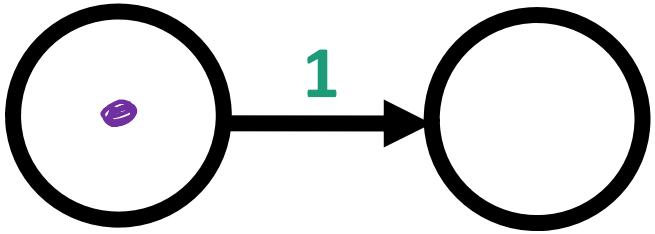
Example: Does this NFA accept the string 1100?

1100 is accepted

Some special transitions

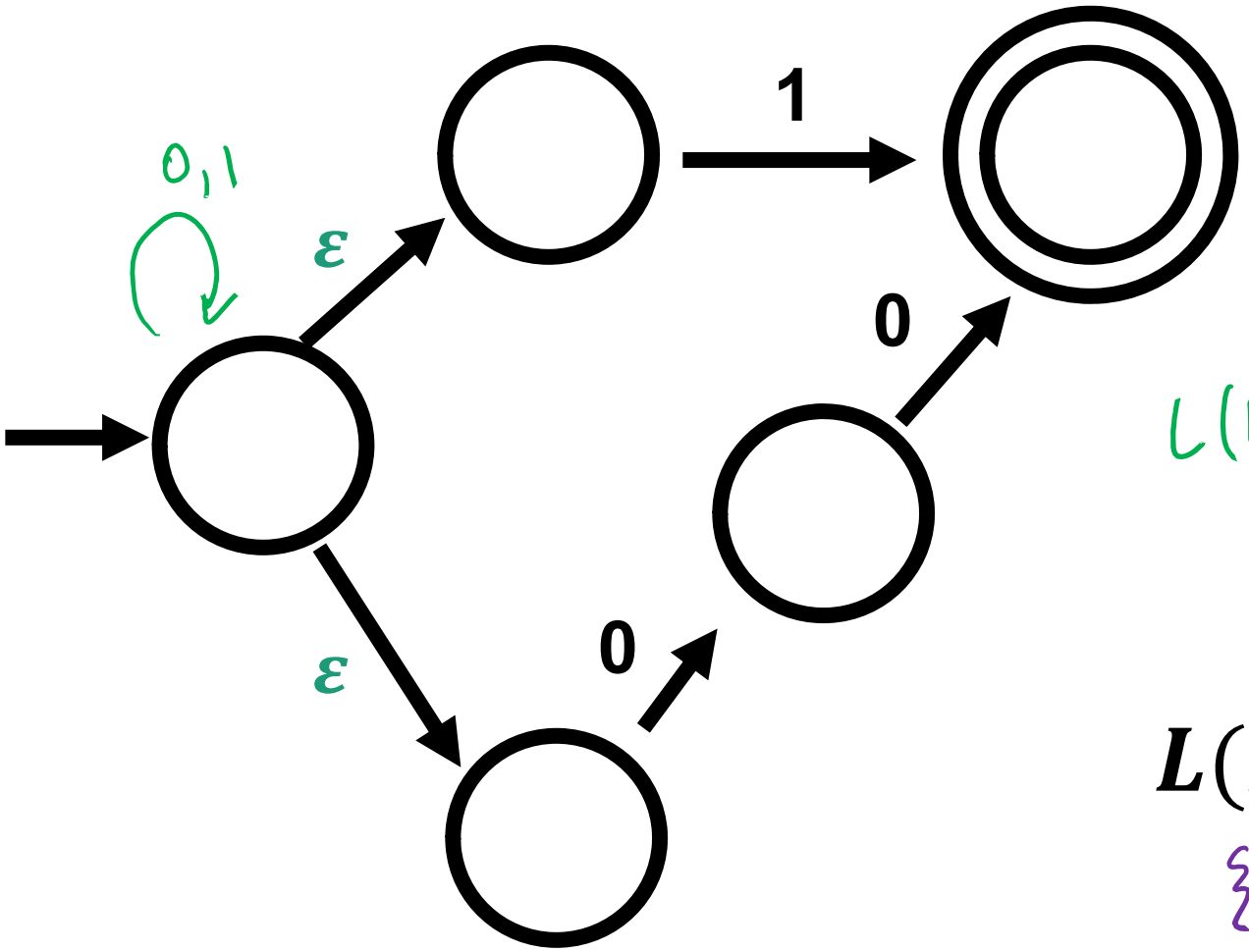






Example

$100 \notin L(M)$

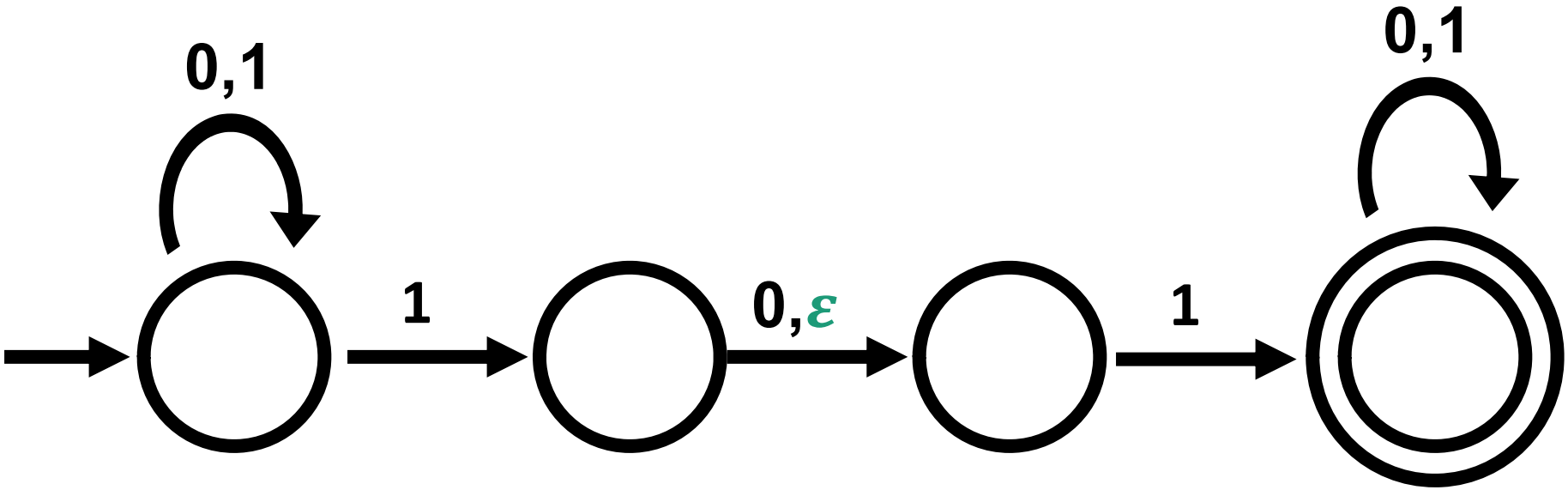


$L(M') = \{ w \mid w$ ends in 1 or w ends in $00 \}$

$$L(M) =$$

$\{ 1, 00 \}$

Example



$$L(M) = \{ w \mid w \text{ contains } 101 \text{ or } 11 \}$$

Formal Definition of a NFA

NFA: $\delta : Q \times \Sigma \to Q$

An NFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

Q is the set of states

$P(Q)$ = set of subsets of Q

Σ is the alphabet

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

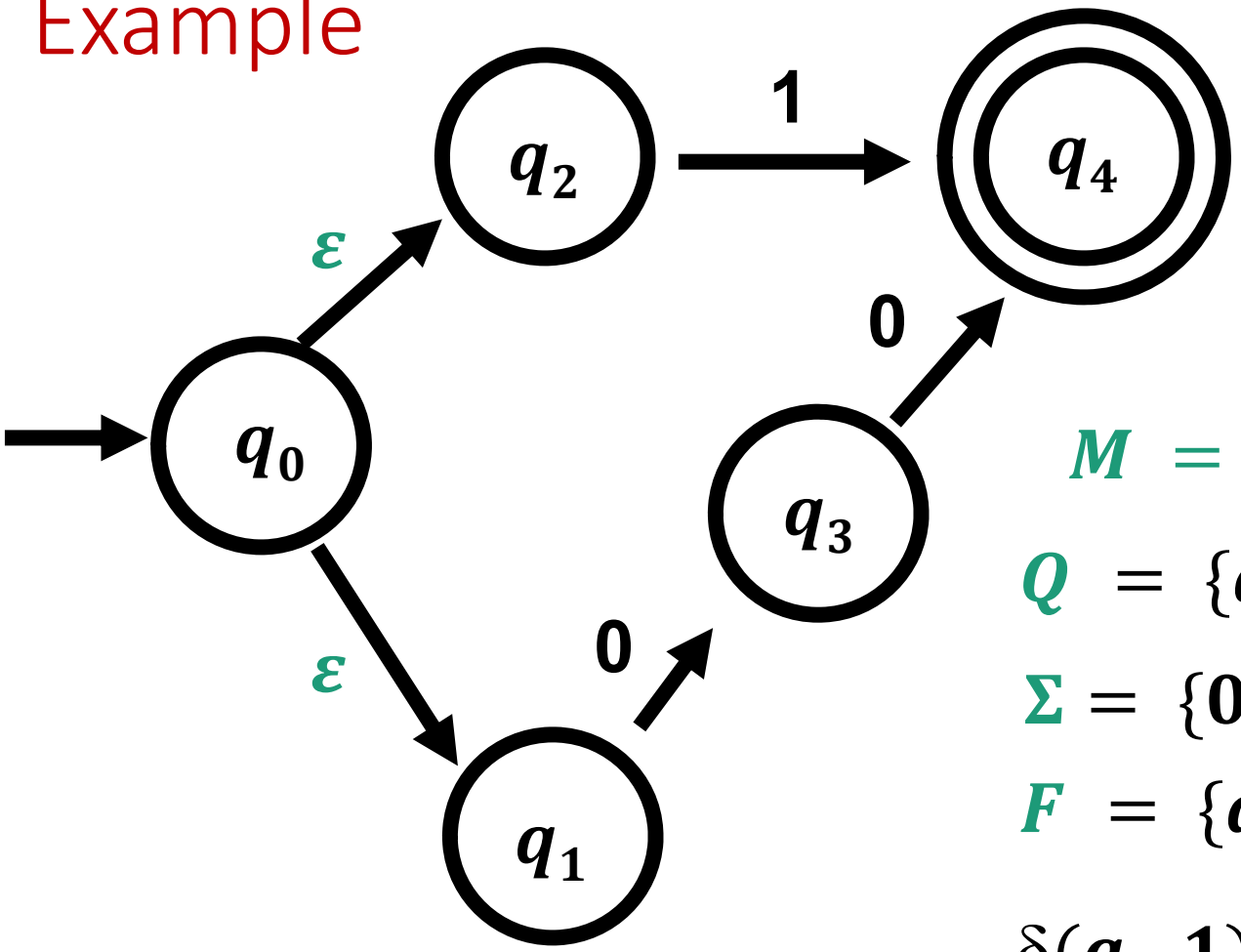
$\delta : Q \times \Sigma_\varepsilon \to P(Q)$ is the transition function

$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accept states

M ***accepts*** a string w if **there exists** a path from q_0 to an accept state that can be followed by reading w.

Example



$$M = (Q, \Sigma, \delta, Q_0, F)$$

$$Q = \{q_0, q_1, q_2, q_3, q_4\}$$

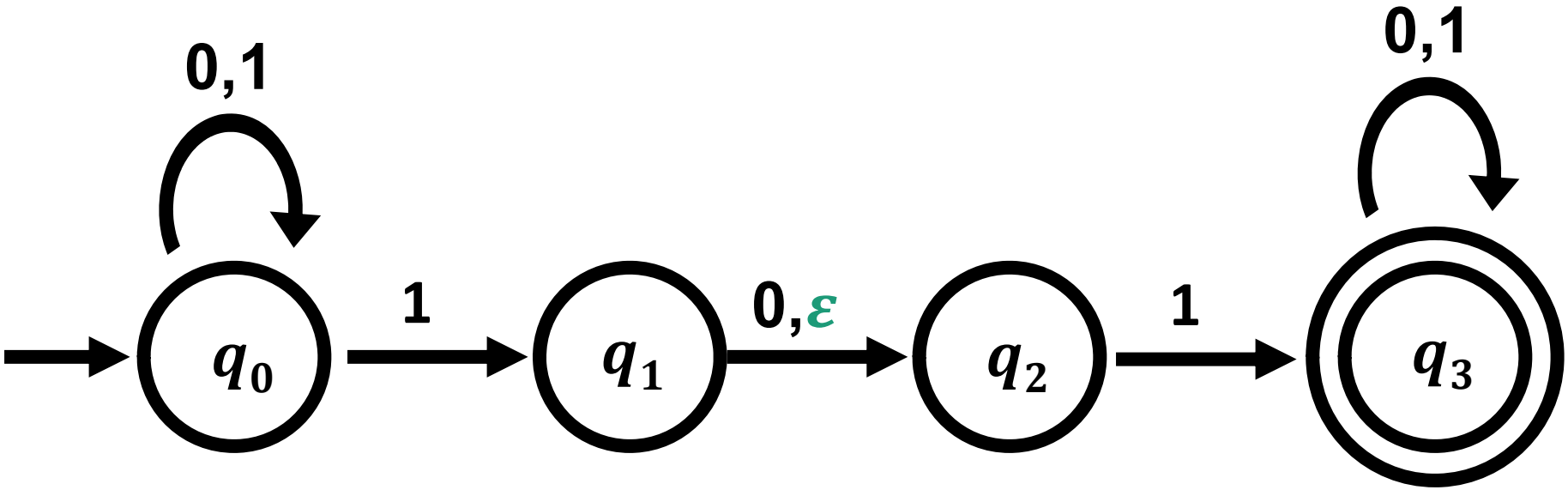
$$\Sigma = \{0, 1\}$$

$$F = \{q_4\}$$

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

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

Example



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

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

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

$$F = \{q_3\}$$

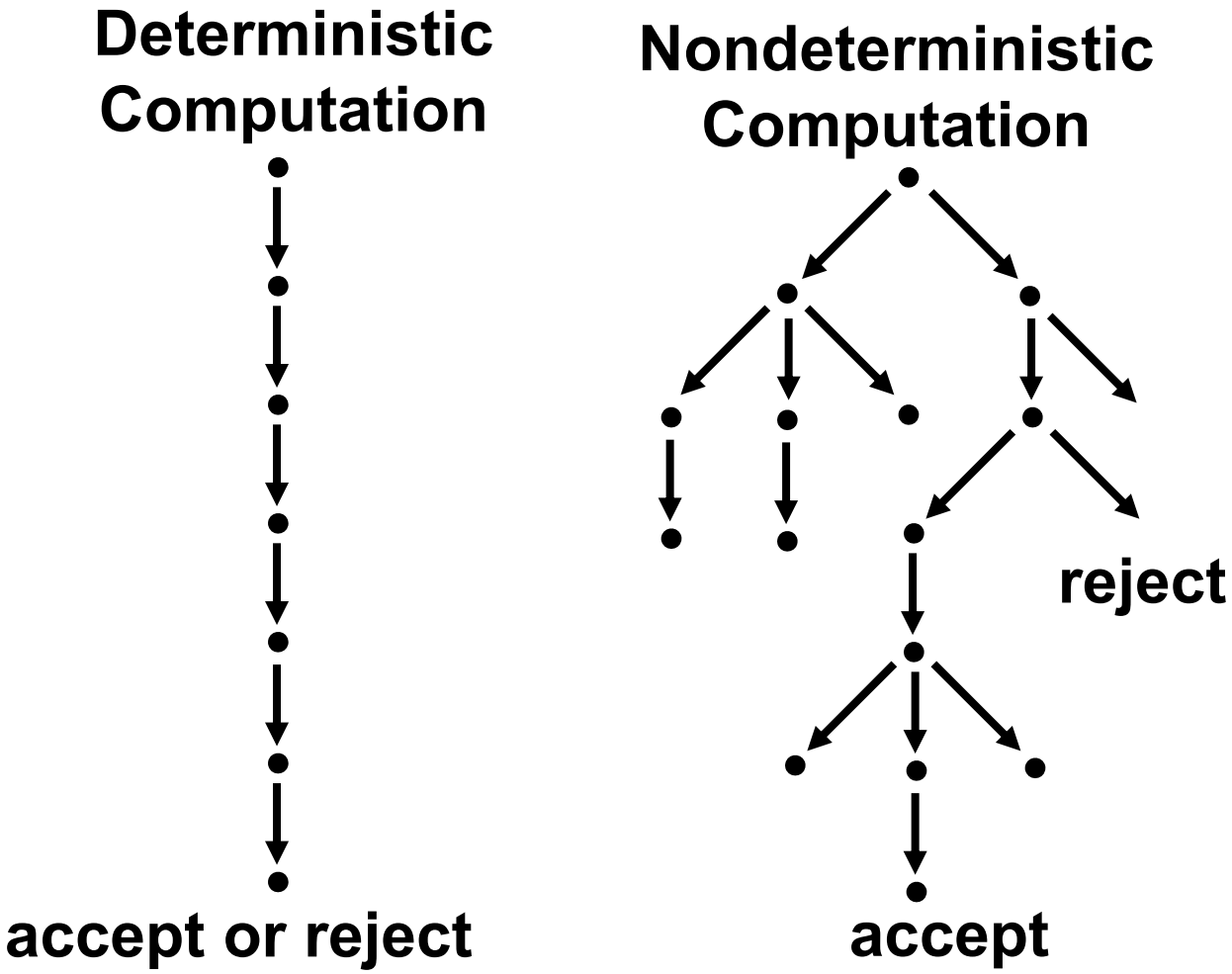
$$\delta(q_0, 0) = \{q_0\}$$

$$\delta(q_0, 1) = \{q_1, q_2, q_3\}$$

$$\delta(q_1, \varepsilon) = \{q_1, q_2\}$$

$$\delta(q_2, 0) = \phi$$

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

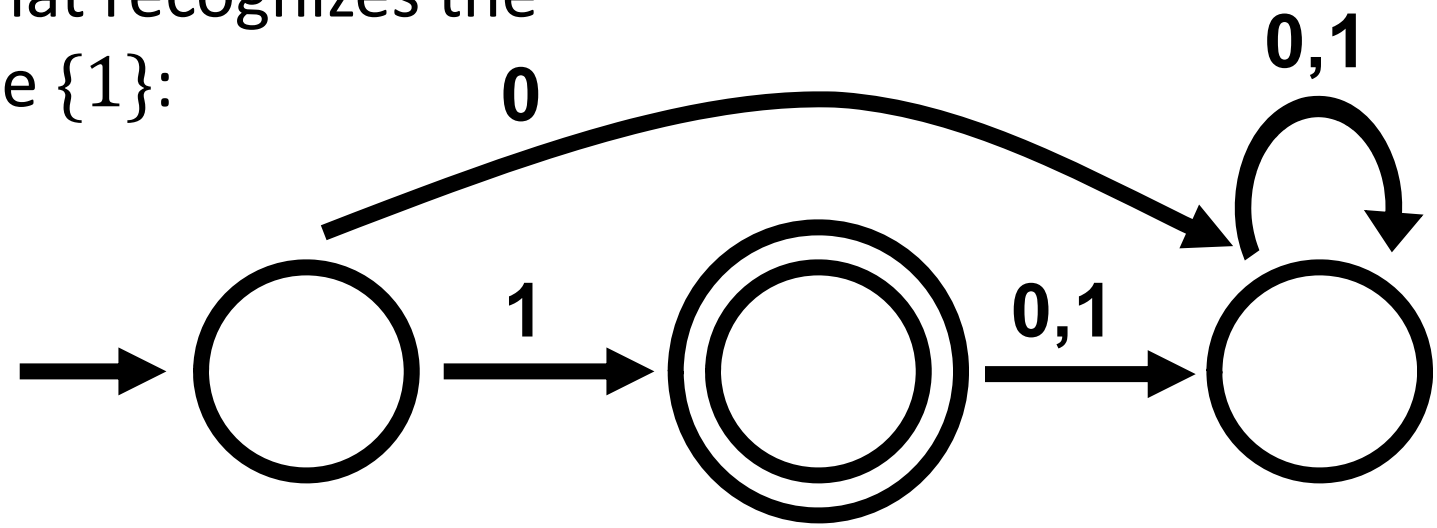
But:

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

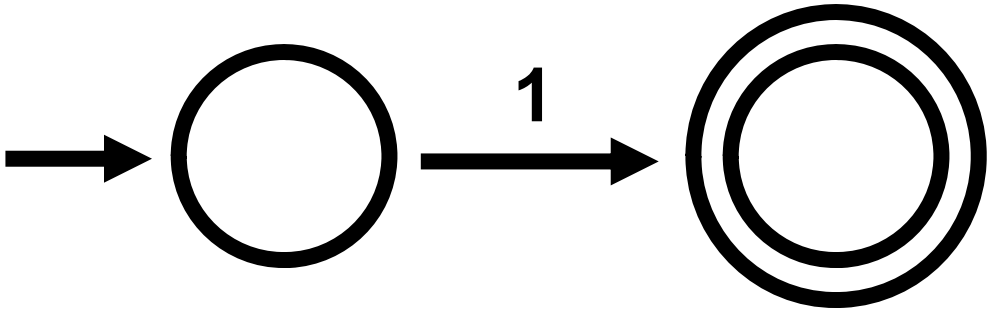
  (cf. P vs. NP)

NFAs can be simpler than DFAs

A DFA that recognizes the language $\{1\}$:



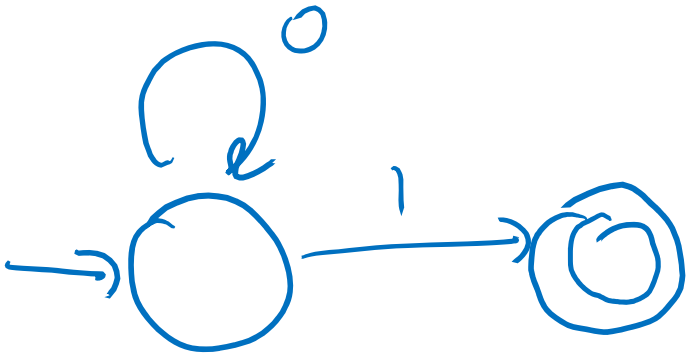
An NFA that recognizes the language $\{1\}$:

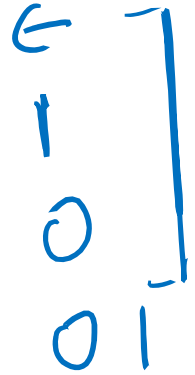


Sometimes DFAs **must** be larger

Theorem. Every DFA for the language {1} must have at least 3 states.

Proof: Contradiction: Suppose exists 2-state DFA





Any DFA that does the right thing on these inputs messes up on 01 ✗

⇒ no 2-state DFA for {1}

Equivalence of NFAs and DFAs

Equivalence of NFAs and DFAs

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

Regular languages $\subseteq$ languages recognizable by NFAs

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

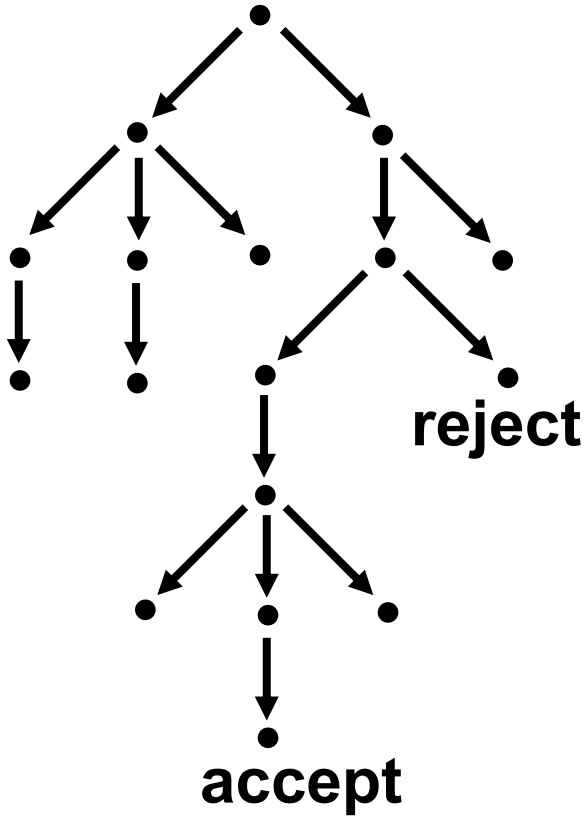
$\supseteq$

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

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

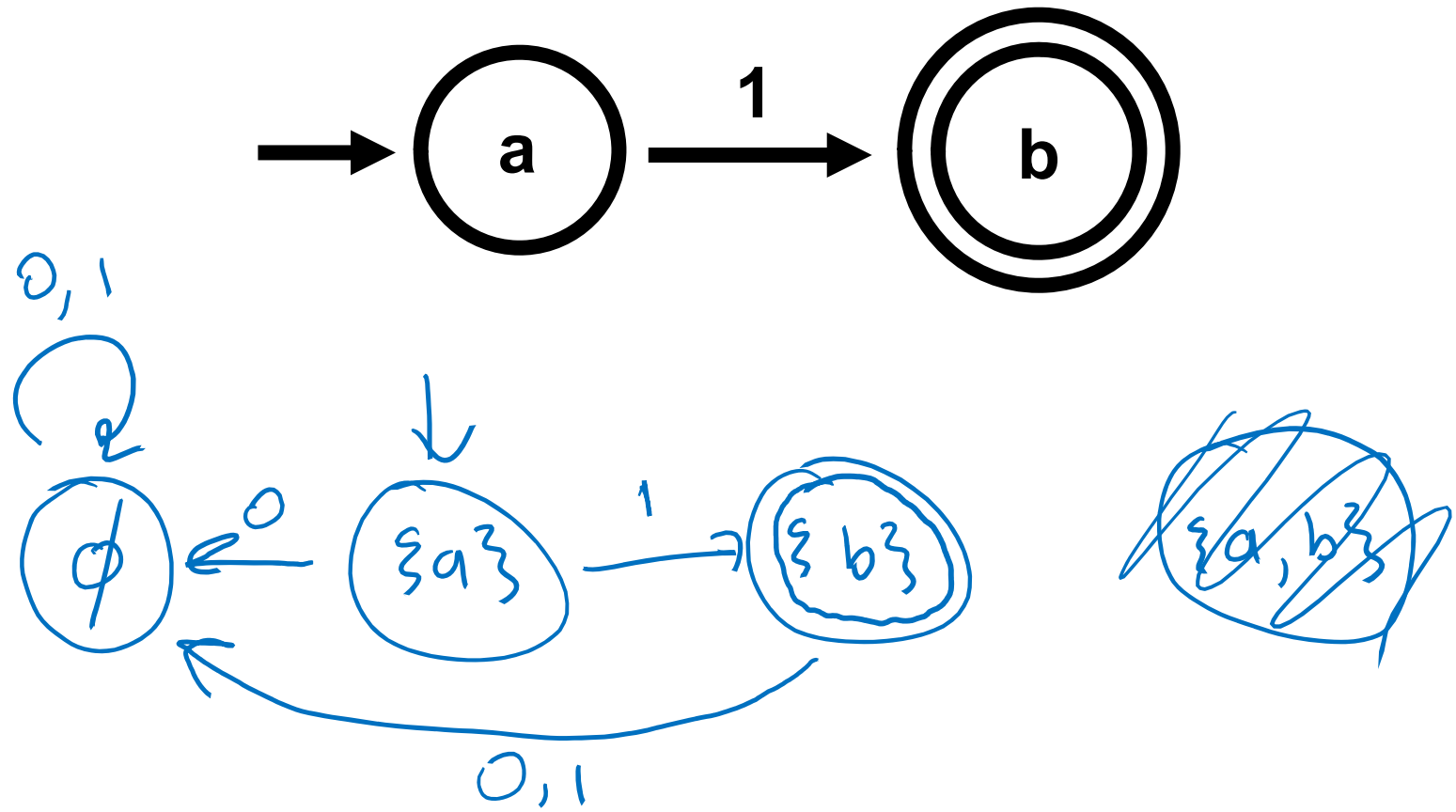


Intuition: Run all threads of N in parallel, maintaining the set of states where all threads are.

Formally: $Q' = P(Q)$

"The Subset Construction"

NFA -> DFA Example



Subset Construction (Formally)

Input: NFA $N = (Q, \Sigma, \delta, q_0, F)$

Output: DFA $M = (Q', \Sigma, \delta', q_0', F')$

$Q' = P(Q)$

$\delta' : Q' \times \Sigma \rightarrow Q'$

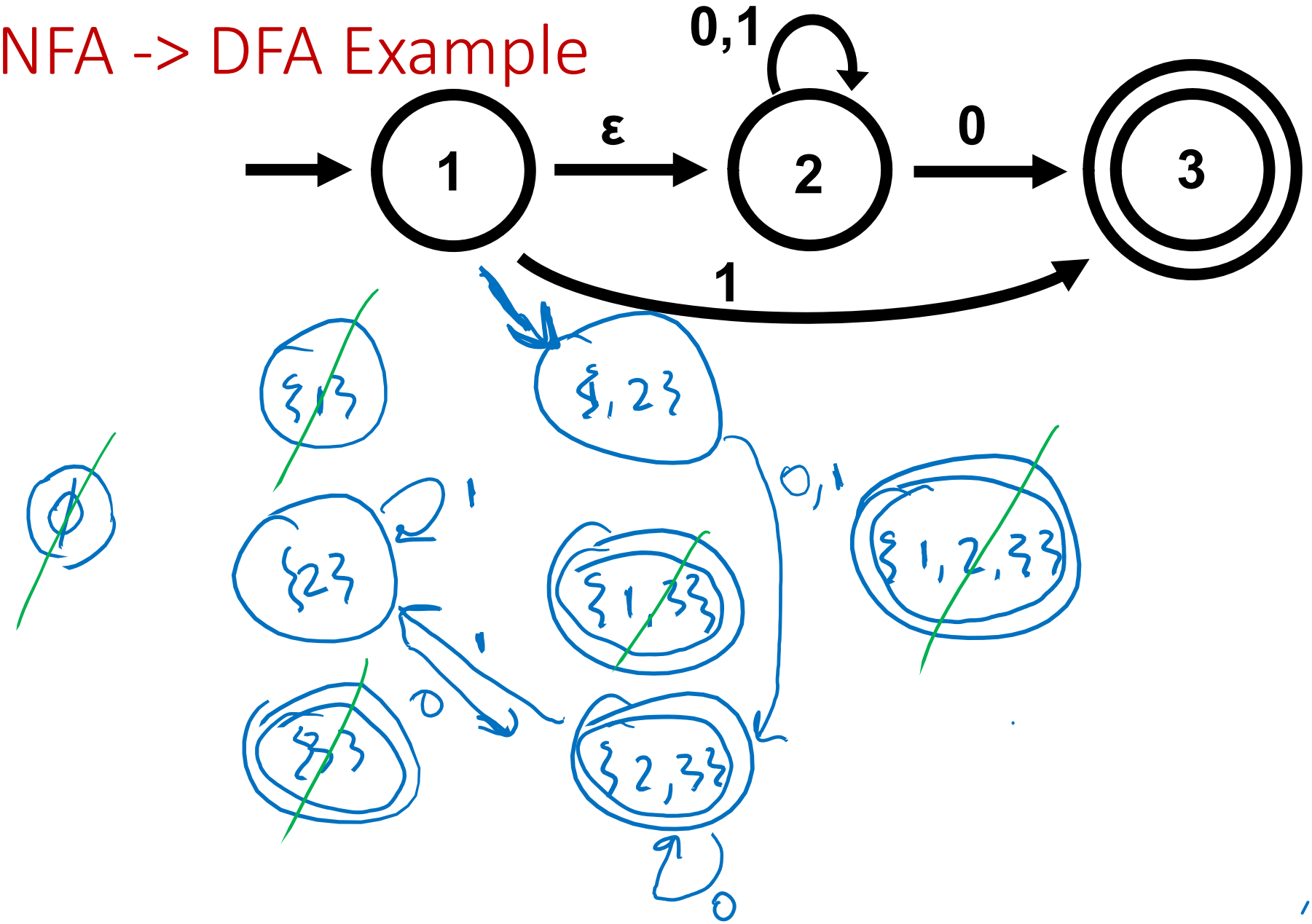
$\delta'(R, \sigma) = \bigcup_{r \in R} \delta(r, \sigma)$ for all $R \subseteq Q$ and $\sigma \in \Sigma$.

$q_0' = \{q_0\}$

$F' = \{R \in Q' \mid R \text{ contains a final state of } N\}$

$(\subseteq Q)$

NFA -> DFA Example



Subset Construction (Formally)

Input: NFA $N = (Q, \Sigma, \delta, q_0, F)$

Output: DFA $M = (Q', \Sigma, \delta', q_0', F')$

For $R \subseteq Q$, define $E(R)$ = set of states reachable from R using ε-moves

$Q' = P(Q)$

$\delta' : Q' \times \Sigma \to Q'$

$\delta'(R, \sigma) = \bigcup_{r \in R} E(\delta(r, \sigma))$ for all $R \subseteq Q$ and $\sigma \in \Sigma$.

$q_0' = E(\{q_0\})$

$F' = \{ R \in Q' \mid R \text{ contains some accept state of } N \}$

Proving the Construction Works

Claim: For every string w, running M on w leads to state

$$\{q \in Q | \text{There exists a computation of } N \text{ on input } w \text{ ending at } q\}$$

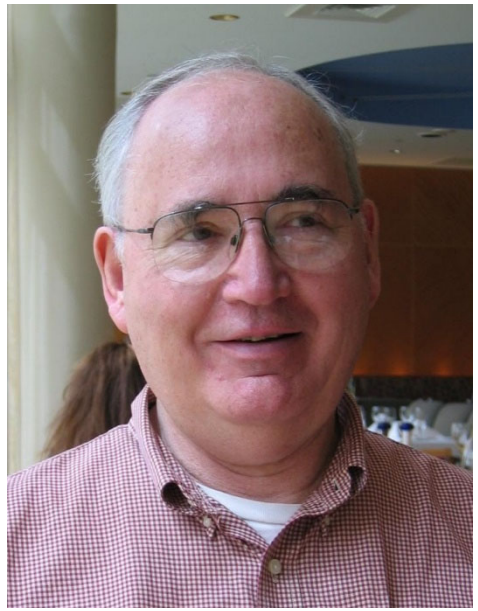
Proof idea: By induction on $|w|$

Historical Note

Subset Construction introduced in Rabin & Scott's 1959 paper "Finite Automata and their Decision Problems"

1976 ACM Turing Award citation

For their joint paper "Finite Automata and Their Decision Problem," which introduced the idea of nondeterministic machines, which has proved to be an enormously valuable concept. Their (Scott & Rabin) classic paper has been a continuous source of inspiration for subsequent work in this field.



Is this construction the best we can do?

Subset construction converts an n state NFA into a 2^n-state DFA

Could there be a construction that always produces, say, an n^2-state DFA?

Theorem: For every $n \geq 1$, there is a language L_n such that

1. There is an $(n+1)$-state NFA recognizing L_n.
2. There is no DFA recognizing L_n with fewer than 2^n states.

Conclusion: For finite automata, nondeterminism provides an exponential savings over determinism (in the worst case).