BU CS 332 – Theory of Computation

https://forms.gle/YF1mHK5aSss1Phtn6

Lecture 4:

- Nondeterministic Finite Automata
- NFAs vs. DFAs
- Closure Properties

Reading:

Sipser Ch 1.1-1.2

Mark Bun

September 15, 2022

Last Time

- Deterministic Finite Automata (DFAs)
  - Informal description: State diagram
  - Formal description: What are they? $M = (Q, \Sigma, \delta, q_0, F)$
  - Formal description: How do they compute?

  - A language is regular if it is recognized by a DFA

$L \ (\subseteq \Sigma^*)$ is regular if

$\exists$ a DFA M s.t. L is the language recognized by M

$$\Longleftrightarrow \quad L = \{ x \in \Sigma^* \mid M \text{ accepts } x \}$$

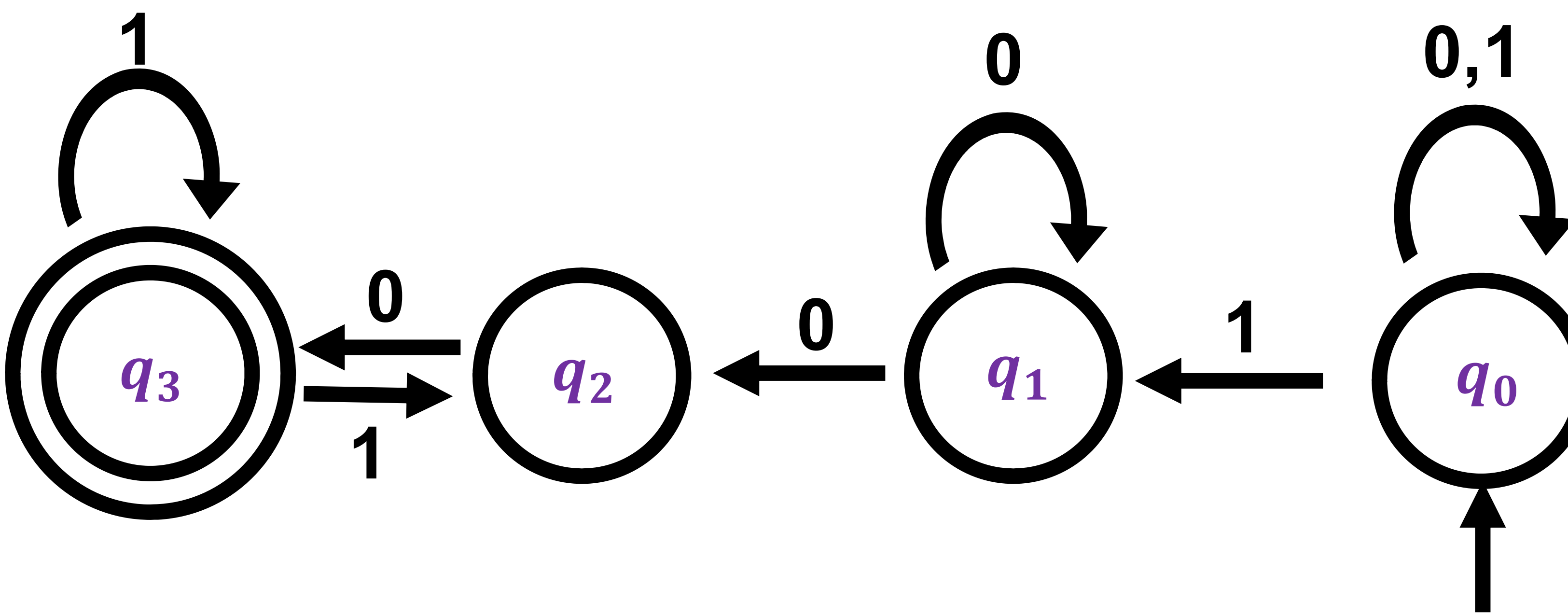
Nondeterminism

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

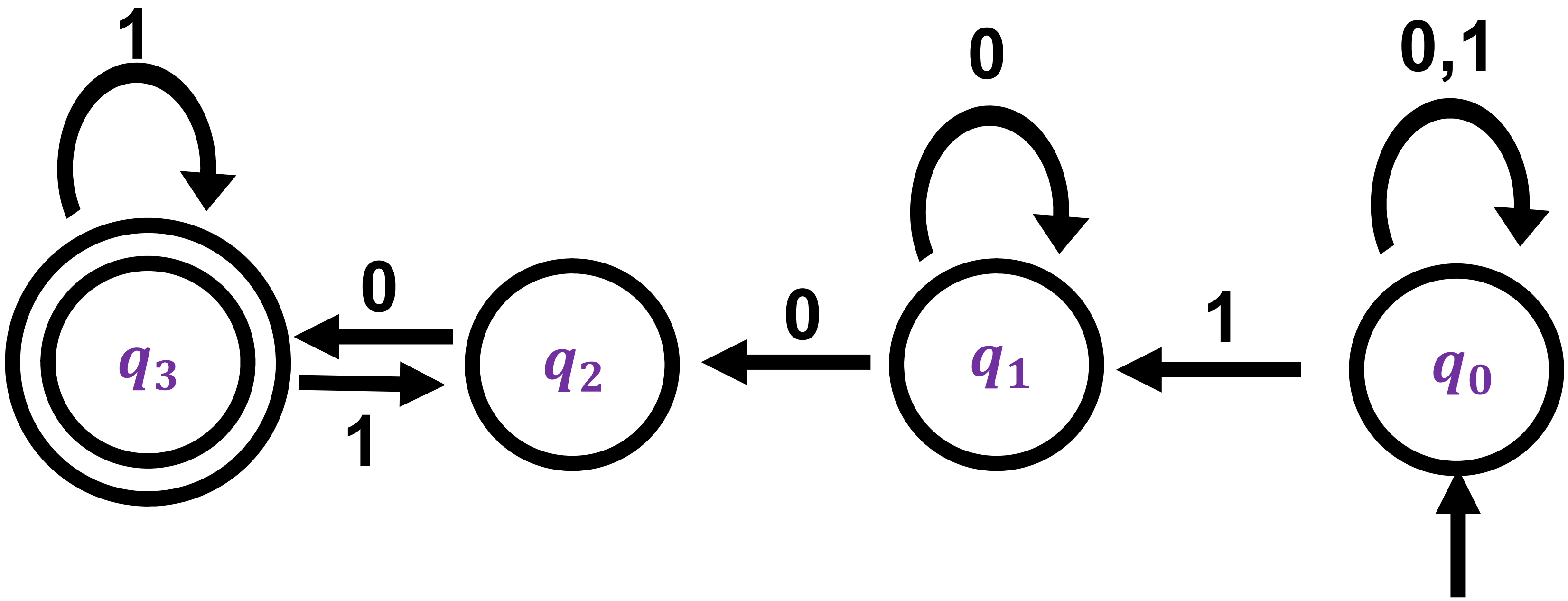
- Next symbol may cause an NFA to “branch” into multiple possible computations
- Next symbol may cause NFA’s computation to fail to enter any state at all

Nondeterminism



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

Nondeterminism

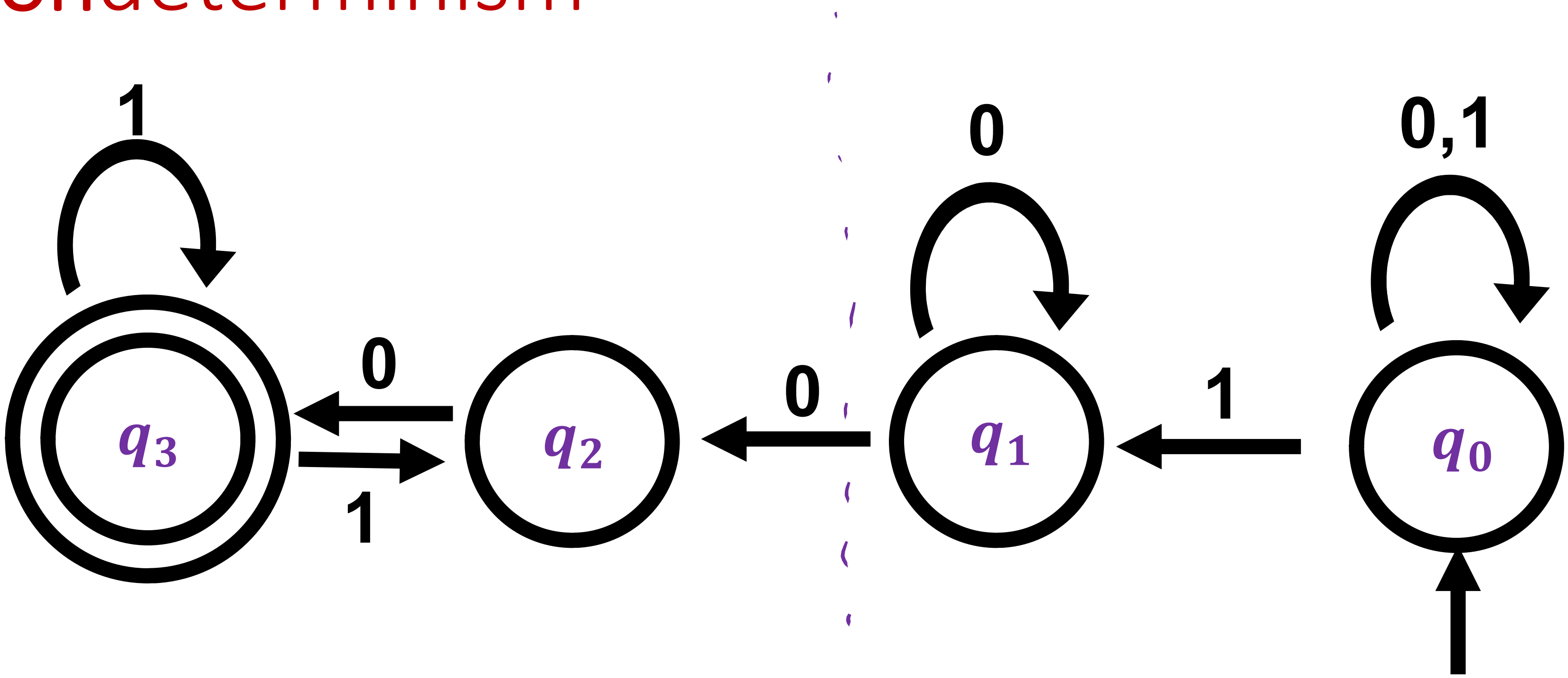


Example: Does this NFA accept the string 1100? $\boxed{\text{YES}}$

$q_0 \xrightarrow{1} q_1 \xrightarrow{1} \perp$

$q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{0} q_3$ accept

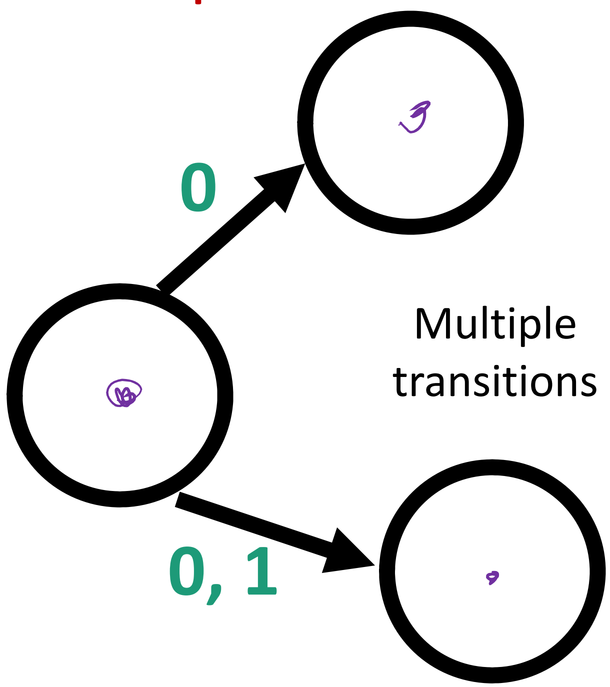
Nondeterminism



Example: Does this NFA accept the string 11?

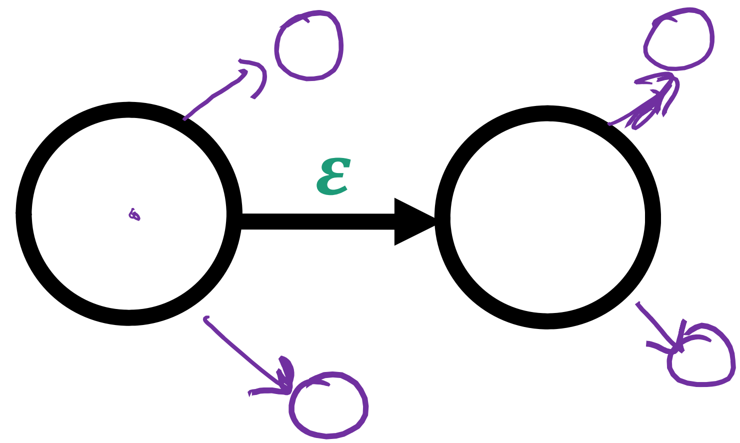
$\boxed{\text{No}}$ No computation paths lead to acceptance

Some special transitions

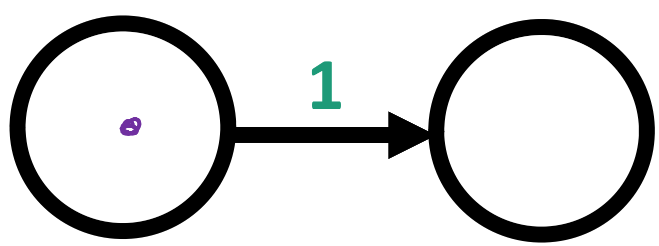


Multiple transitions

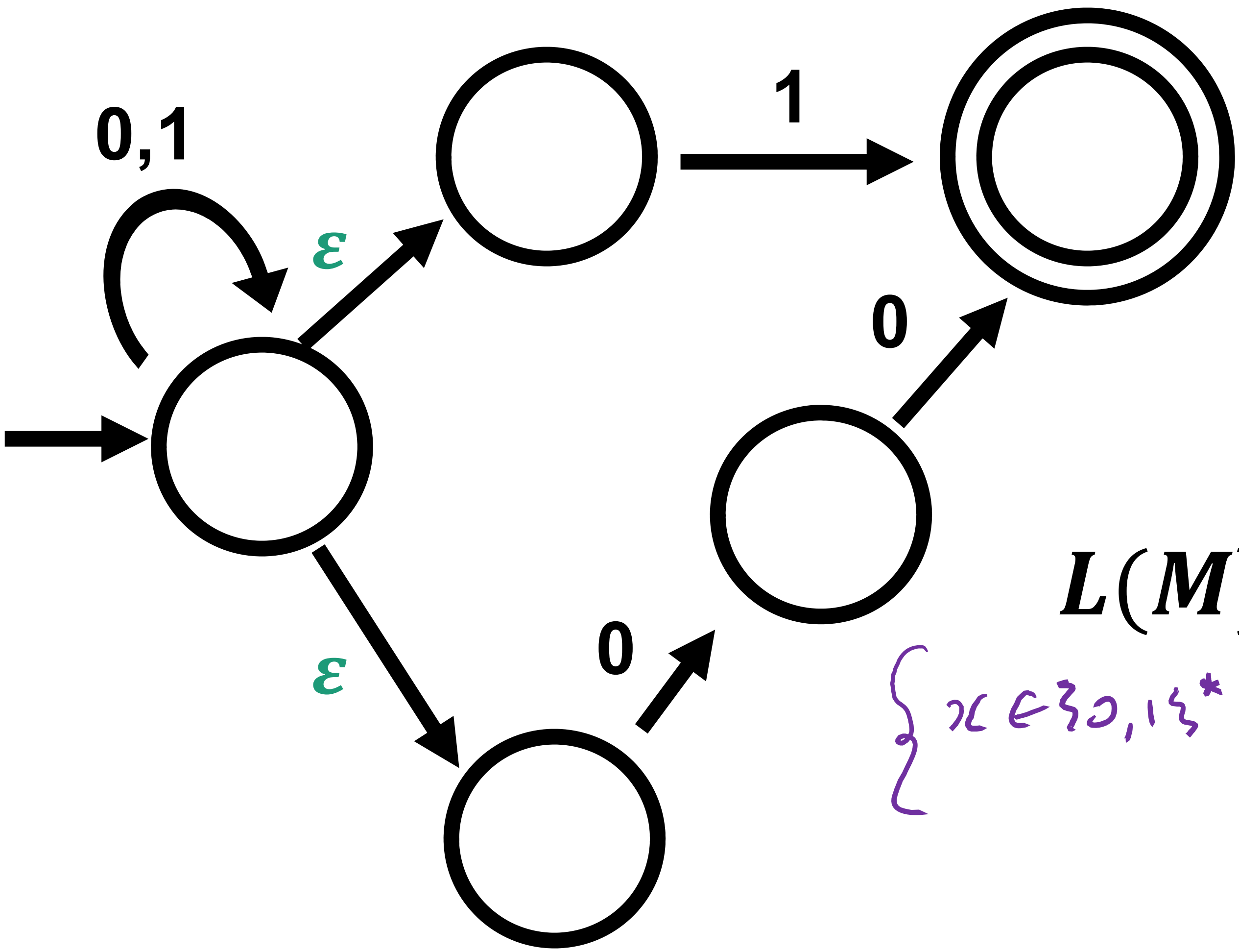
ε-transitions
(don't consume a symbol)



No transition



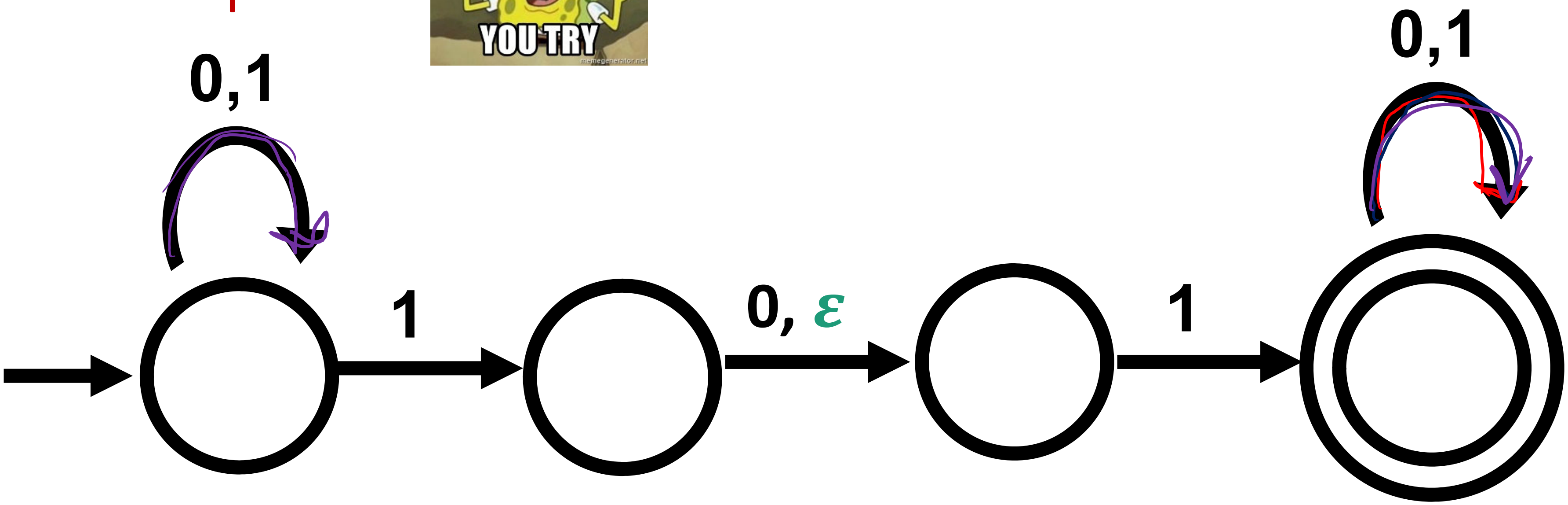
Example



$$L(M) =$$

$\{x \in \{0,1\}^* \mid x \text{ ends in } 1$ OR $x \text{ ends in } 00\}$

Example



$\boldsymbol{L(N)} =$

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) $\{w \mid w \text{ contains } 11 \text{ or } 101\}$

Formal Definition of a NFA

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

Q is the set of states

$P(Q) = \{ R \mid R \subseteq Q \}$

"power set of Q"

Σ is the alphabet

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

old state next symbol set of possible next states

$\delta: Q \times \boxed{\Sigma_\varepsilon} \to \boxed{P(Q)}$ 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

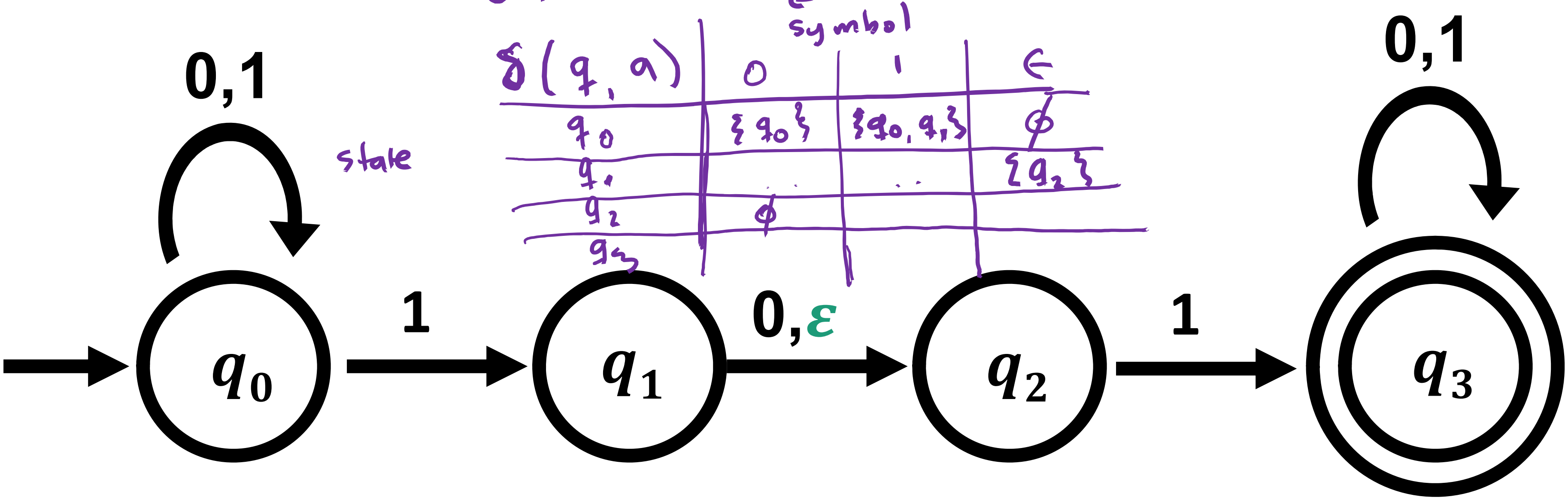
$\Sigma_\varepsilon = \{0, 1, \varepsilon\}$

$\delta : Q \times \Sigma_\varepsilon \to P(Q)$

symbol

state

$\delta(q, a)$	0	1	ε
q_0	$\{q_0\}$	$\{q_0, q_1\}$	$\emptyset$
q_1	. . .	. .	$\{q_2\}$
q_2	$\emptyset$		
q_3			



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

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

$\Sigma = \{0, 1\}$ $\quad \Sigma^* = \{0,1\}^*$

$= \{$ all 0-1 strings $\}$

$F = \{q_3\}$

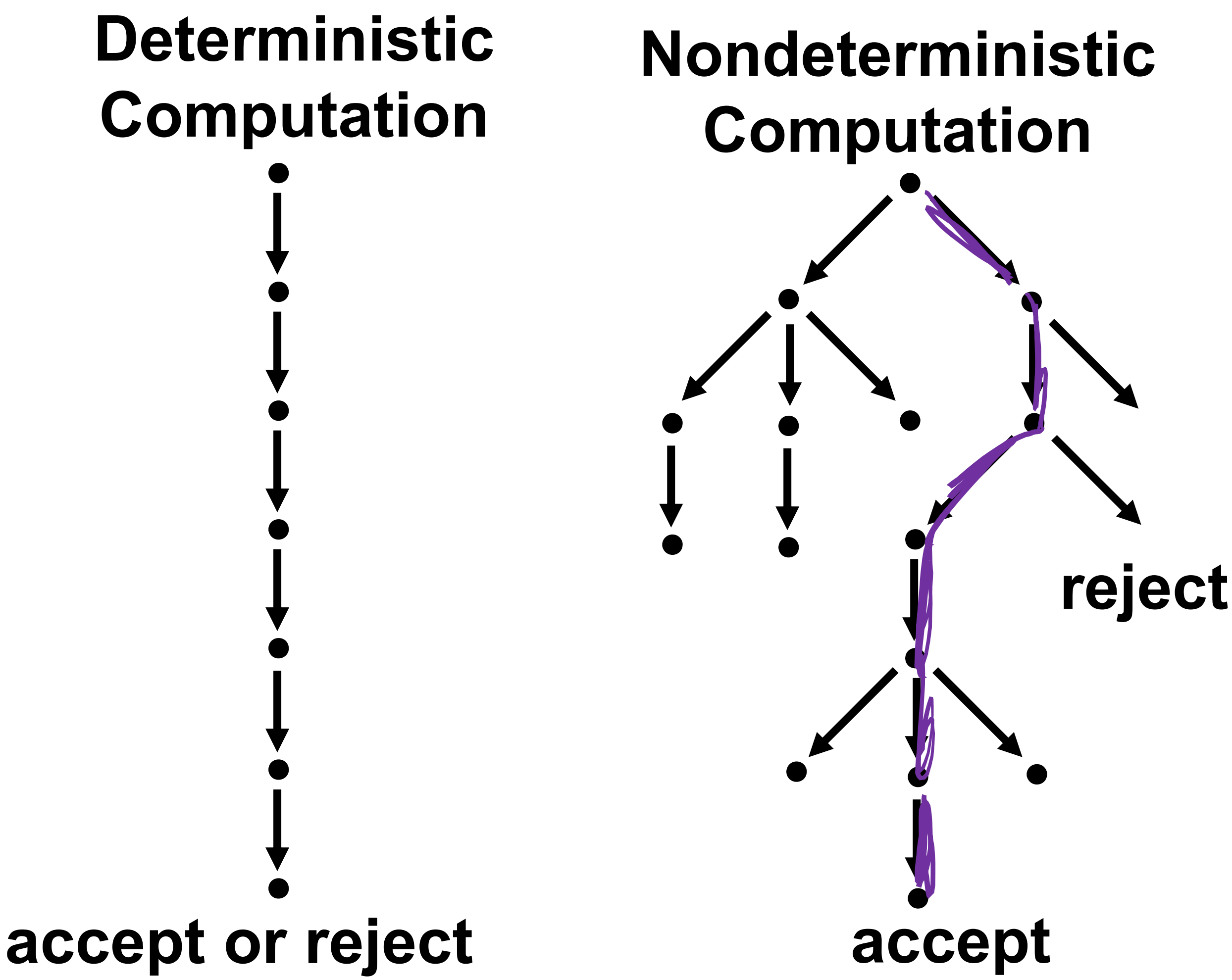
$\delta(q_0, 0) = \{q_0\}$

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

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

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

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:

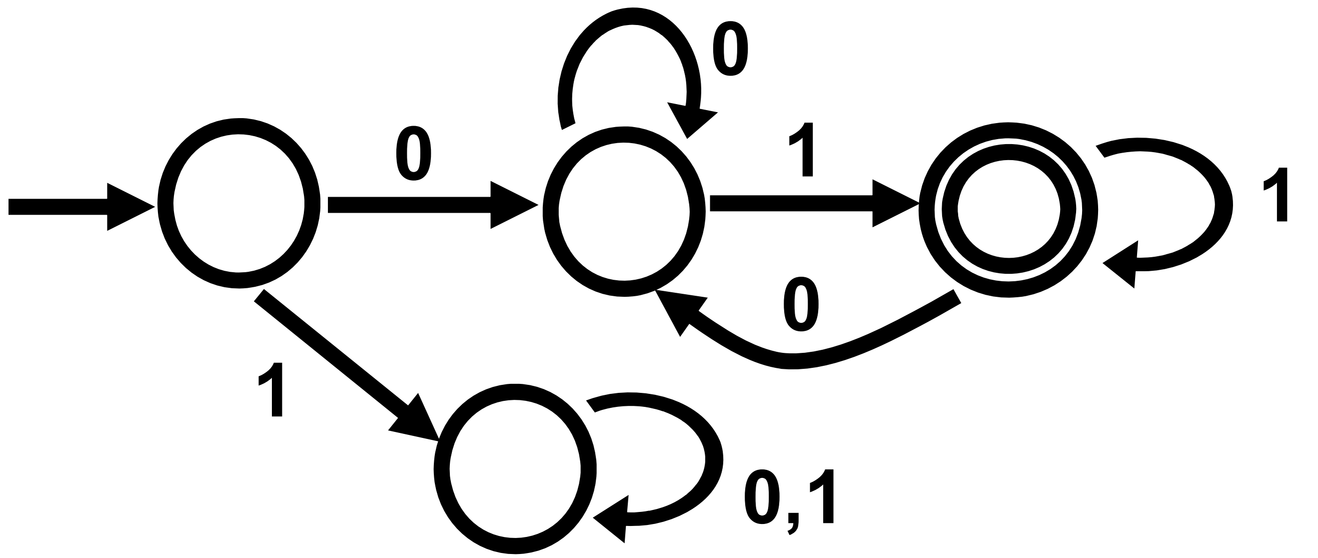
- Useful tool for understanding power of DFAs/regular languages
- NFAs can be simpler than DFAs
- Lets us study "nondeterminism" as a resource

  (cf. P vs. NP)

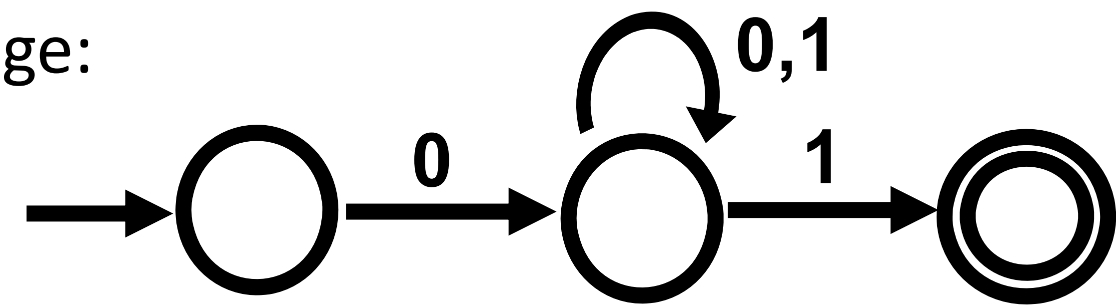
NFAs can be simpler than DFAs

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

Every DFA requires ≥ 4 states



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

Regular languages $\subseteq$ Langs. recog. by NFAs
(i.e. langs recognizable by DFAs)

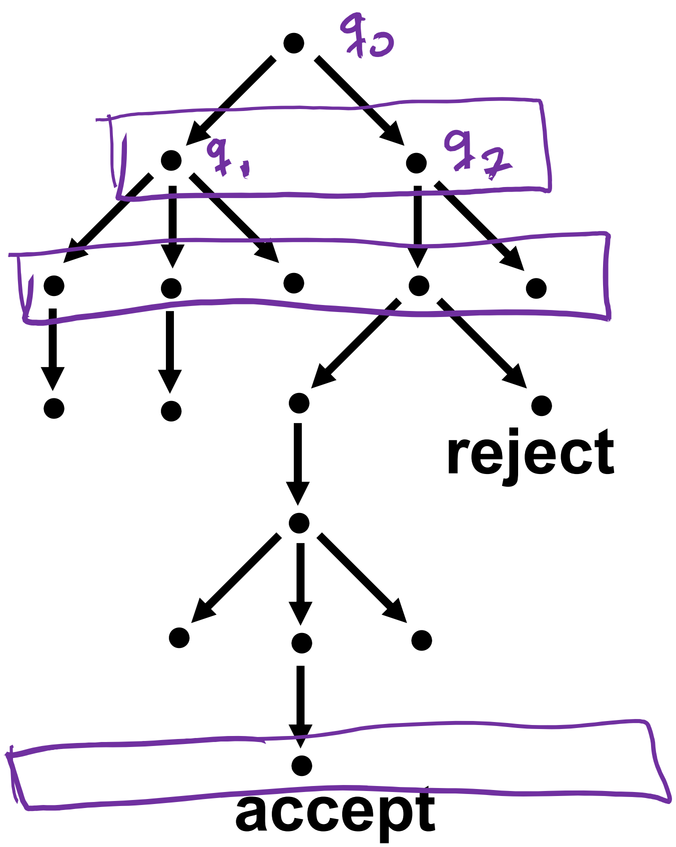
Theorem: For every NFA N, there is a DFA M such that $L(M) = L(N)$ $\Rightarrow$ regular langs $\supseteq$ langs recog. by NFAs

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 $L(N)$



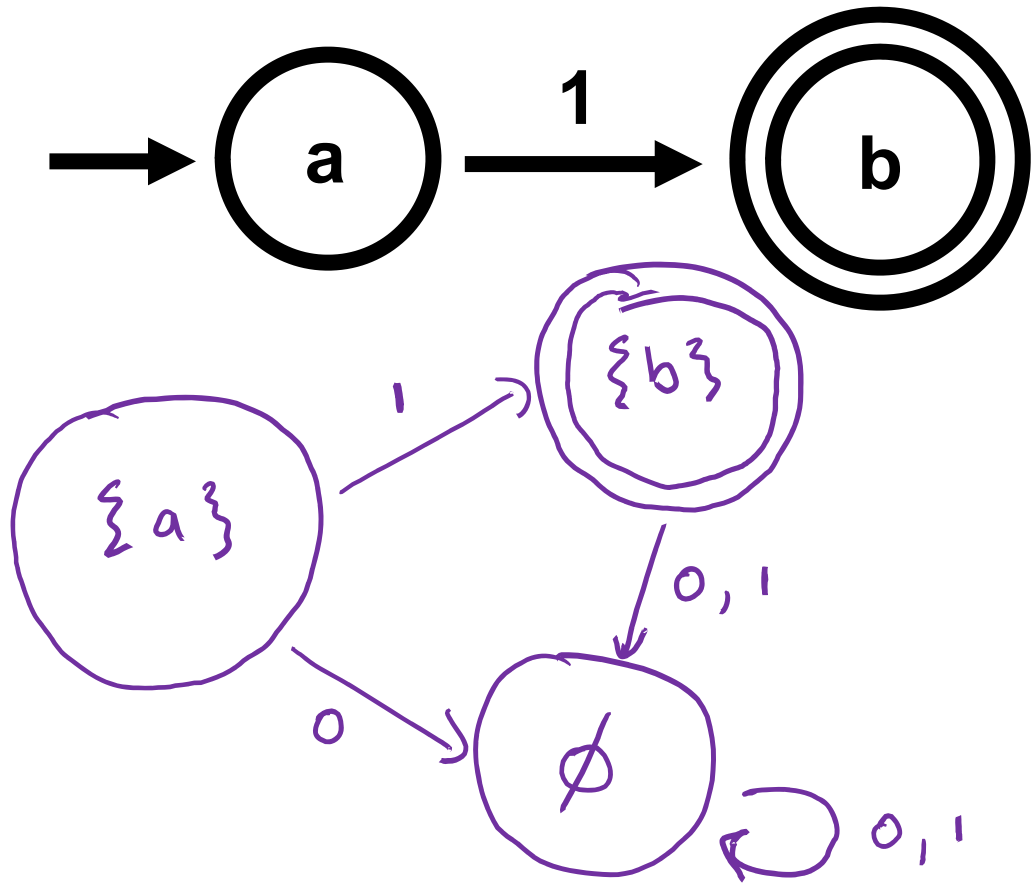
Intuition: Run all threads of N in parallel, maintaining the set of states where all threads are.

Formally: $Q' = P(Q)$

"The Subset Construction"

NFA -> DFA Example

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



Subset Construction (Formally, first attempt)

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

$Q' = P(Q) = \{R \mid R \subseteq Q\}$

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

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

$q_0' = \{q_0\}$

$F' = \{R \mid R \subseteq Q, \exists q \in F \text{ s.t. } q \in R\}$

Subset Construction (Formally, for real)

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

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

$Q' = P(Q)$

$E(R)$ = set of states reachable via 1 or more ϵ transition from a state in R

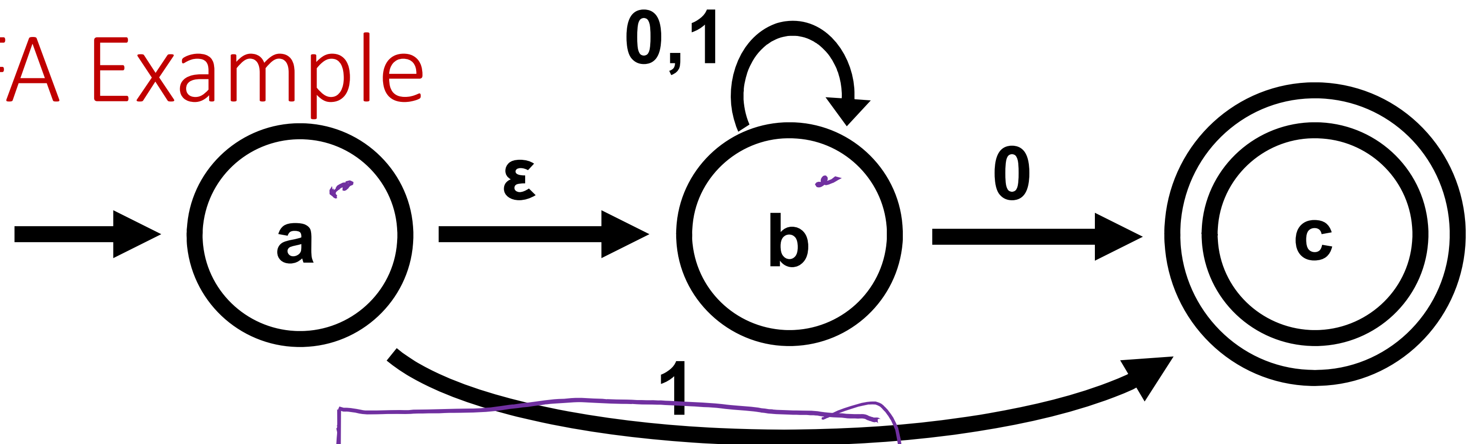
$\delta' : Q' \times \Sigma \rightarrow Q'$

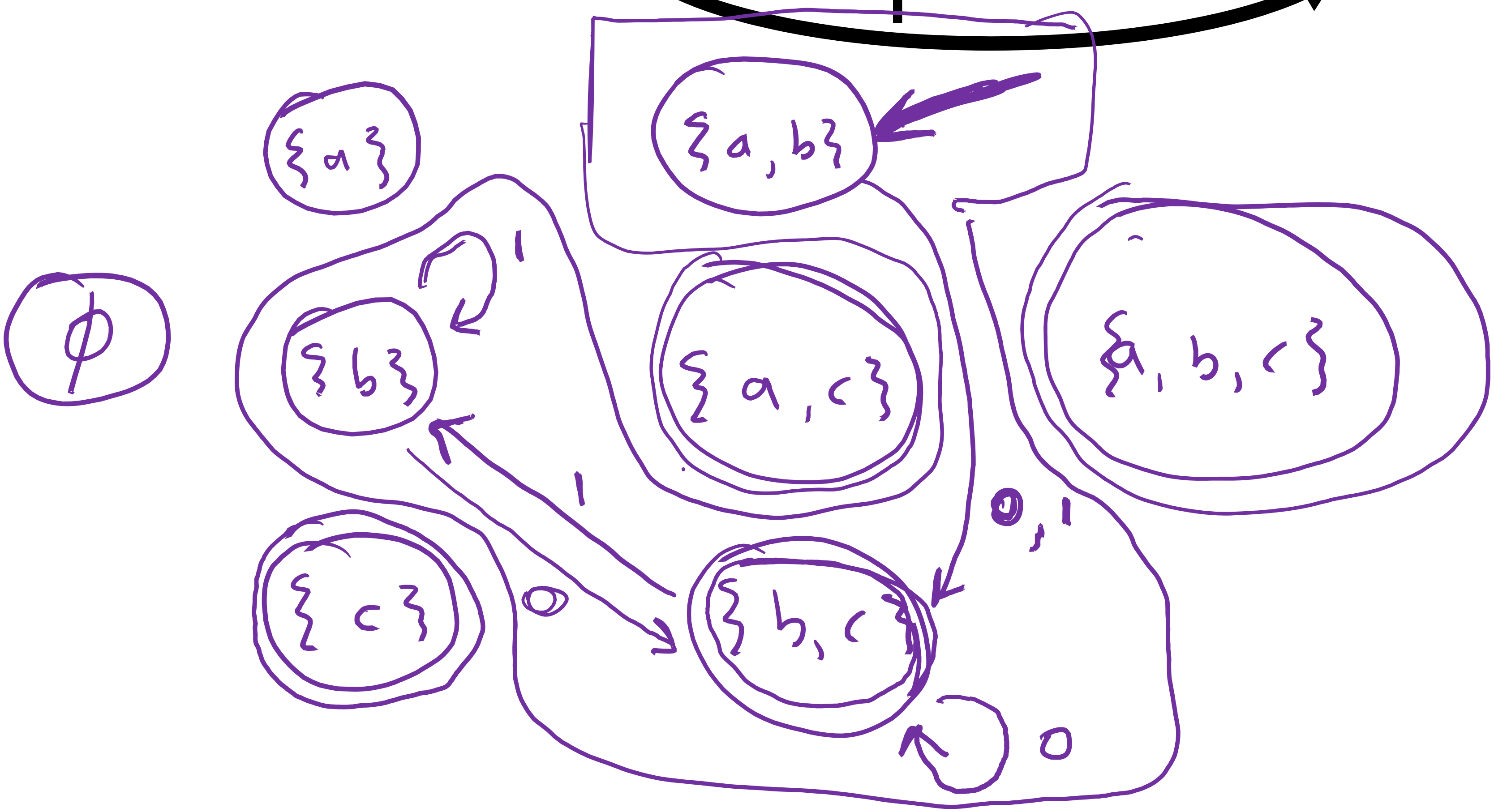
$\delta'(R, \sigma) = \bigcup_{r \in R} E(\delta(r, \sigma))$ for all $R \subseteq Q$ and $\sigma \in \Sigma$.

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

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

NFA -> DFA Example





Proving the Construction Works

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

(of the DFA)

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

= set of states of the NFA

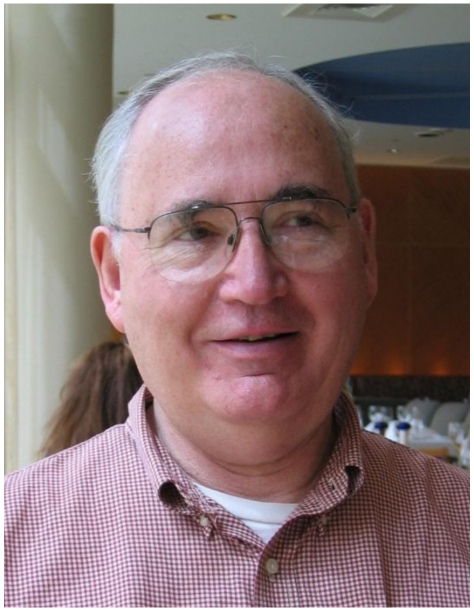
Proof idea: By induction on $|w|$

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.



NFA -> DFA: The Catch

If N is an NFA with s states, how many states does the DFA obtained using the subset construction have? (In the worst case.)

$$Q' = P(Q)$$

$$|Q'| = |P(Q)| = 2^{|Q|} = 2^s$$

a) s

b) s^2

c) $\boxed{2^s}$

d) None of the above

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).