

BU CS 332 – Theory of Computation

<https://forms.gle/2CcC2ZVTBd12ziwX6>



Lecture 4:

- More on NFAs
- NFAs vs. DFAs
- Closure Properties?

Reading:

Sipser Ch 1.1-1.2

HW 1 due tonight

Alexander Poremba & Mark Bun

January 29, 2026

Last Time

- Deterministic Finite Automata (DFAs)
 - Informal description: State diagram
 - Formal description: What are they?
 - Formal description: How do they compute?
 - A language is **regular** if it is recognized by a DFA
- Intro to Nondeterministic Finite Automata (NFAs)

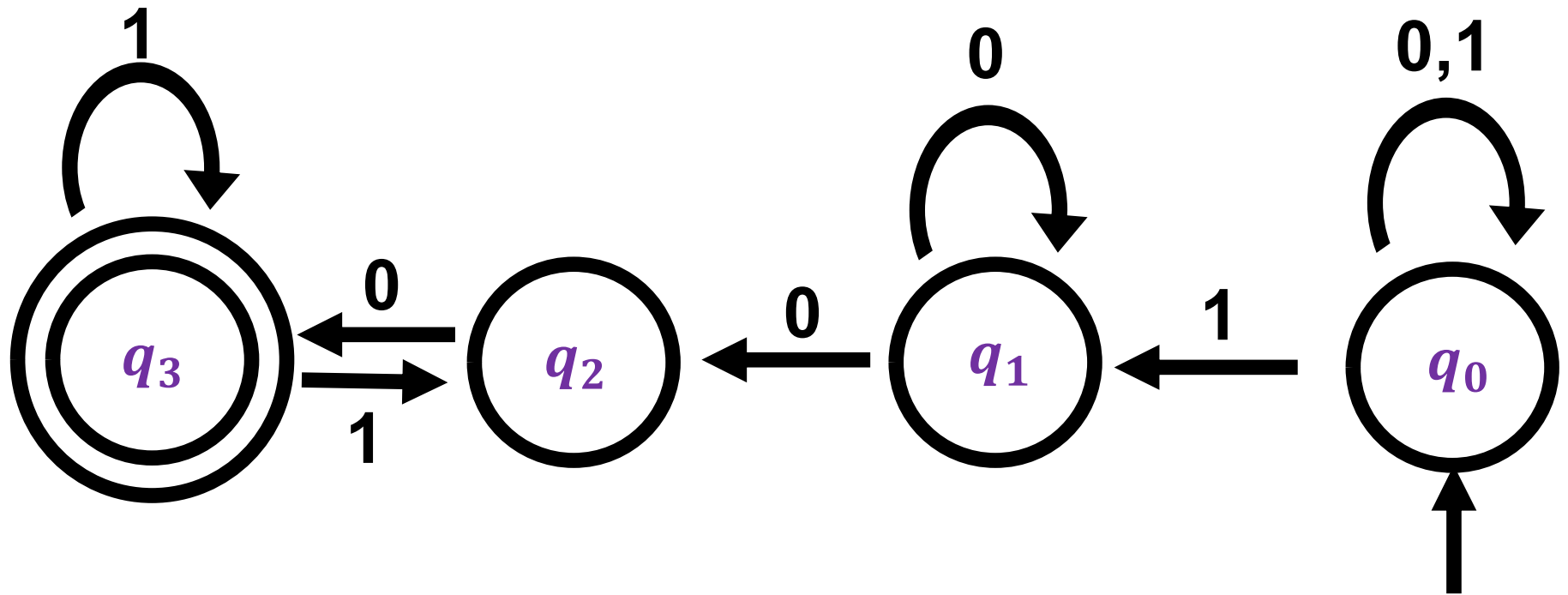
Nondeterminism

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



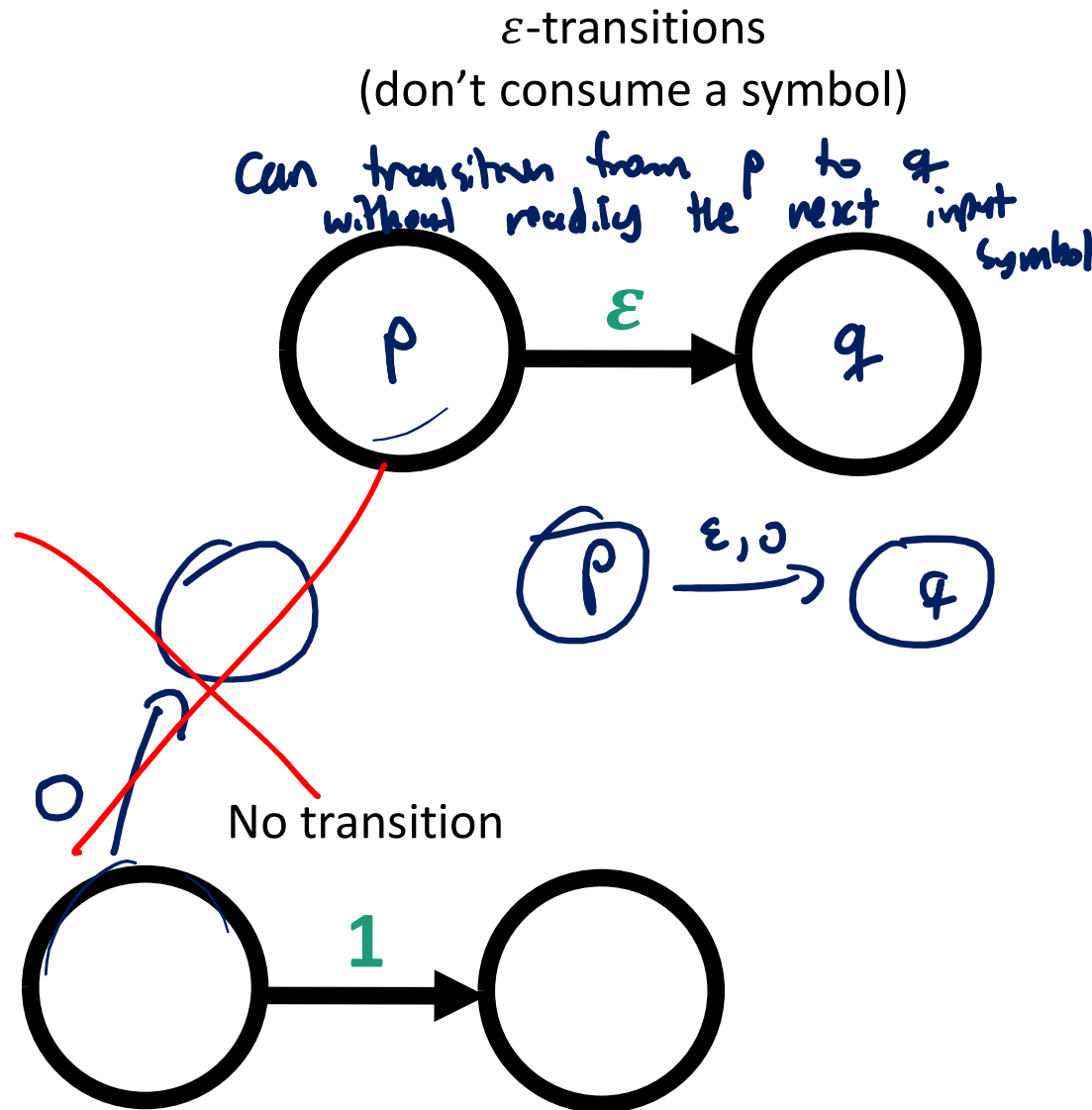
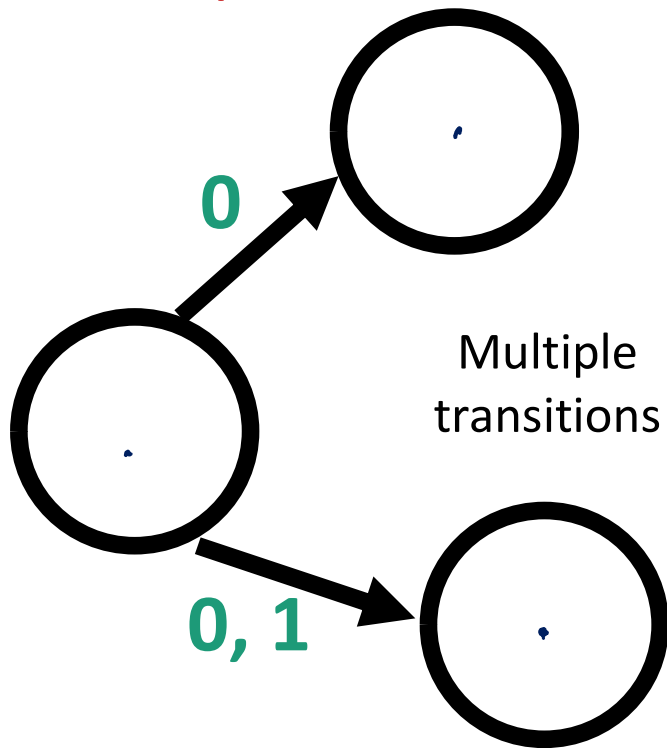
A **Nondeterministic Finite Automaton** (NFA) accepts if there **exists** a way to make it reach an accept state.

Ex. This NFA accepts input 1100, but does not accept input 11

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

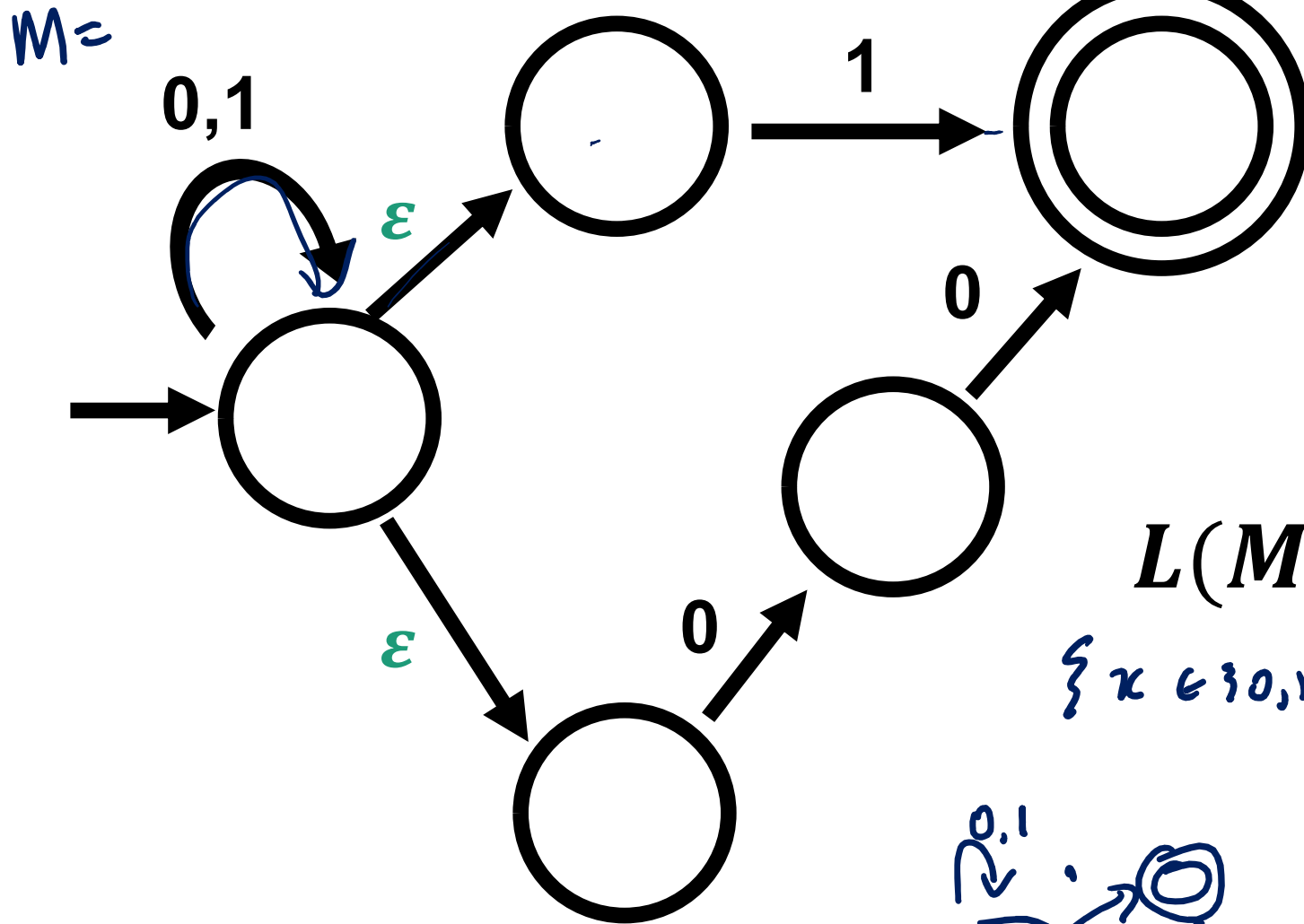
There is no way to get to an accept state when processing 11

Some special transitions



Example

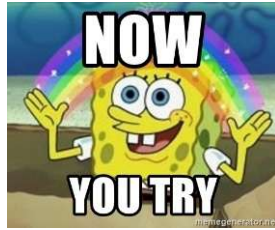
What language does M recognize, i.e. what is the set of strings that M accepts?



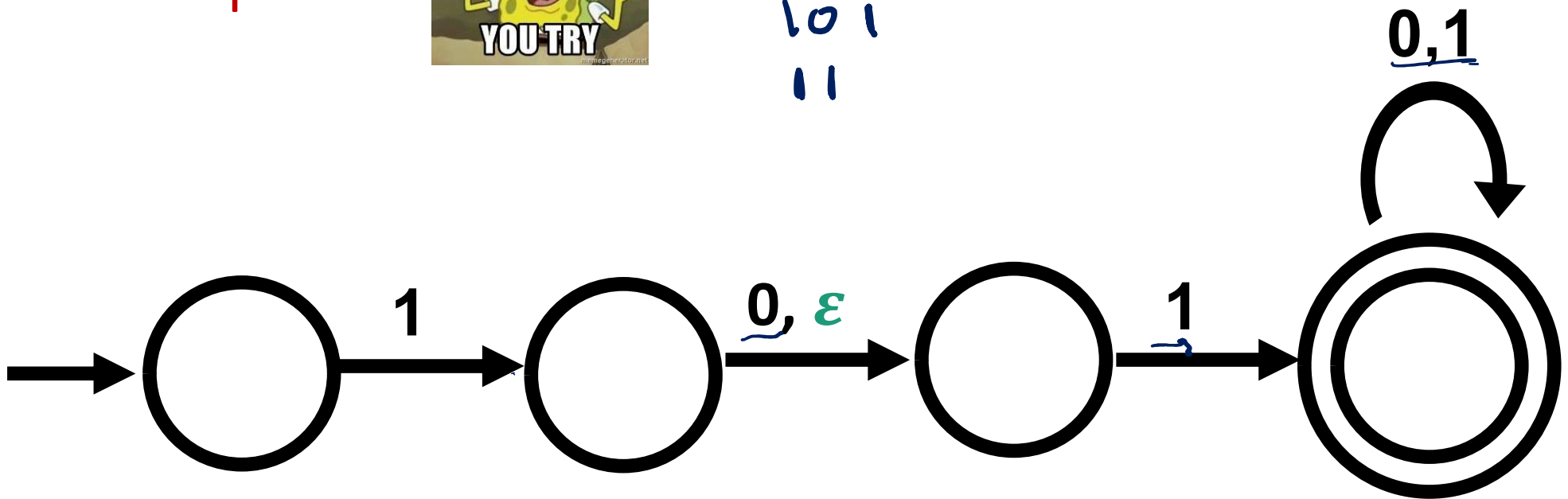
$$L(M) = \{ x \in \{0,1\}^* \mid \begin{array}{l} x \text{ ends in } 1 \\ \text{or} \\ x \text{ ends in } 00 \end{array} \}$$



Example



101
11



$L(N) =$

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



Formal Definition of a NFA

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

Q is the set of states

$P(Q) = \text{power set of } Q = \{R \mid R \subseteq Q\}$

Σ is the alphabet

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

δ is the transition function

$\delta: Q \times \Sigma_\epsilon \rightarrow P(Q)$ is the transition function

Set of possible states NFA could enter

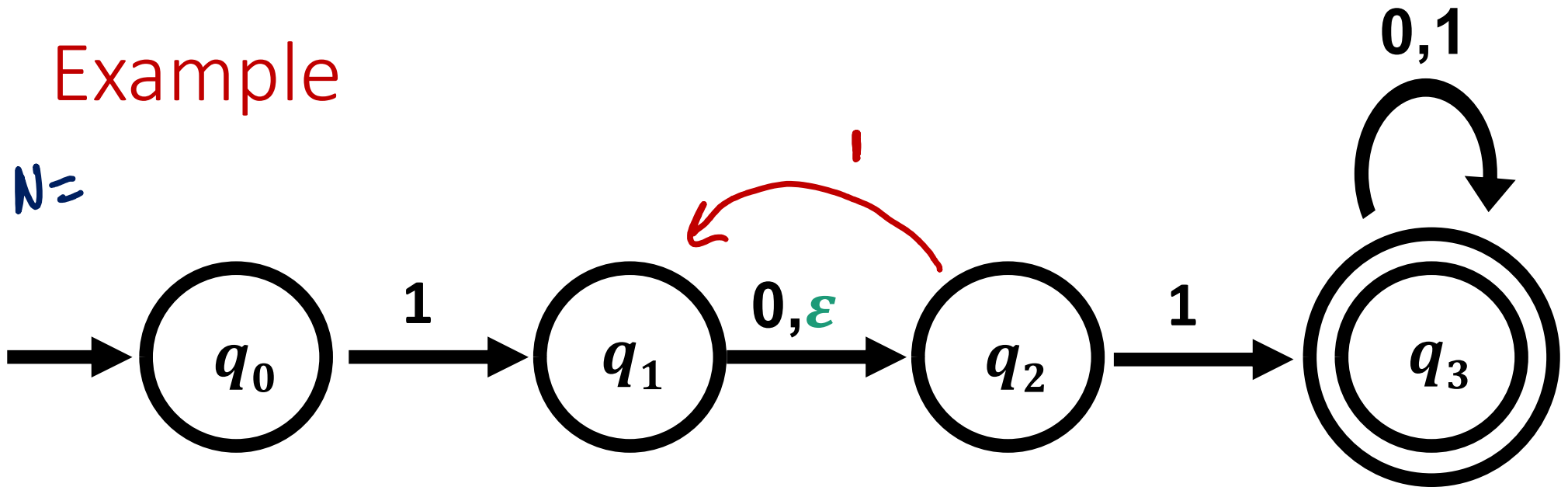
$q_0 \in Q$ is the start state

$F \subseteq Q$ 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

$N =$



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

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

$$\Sigma = \{0, 1\} \quad \Sigma_{\epsilon} = \{0, 1, \epsilon\}$$

$$F = \{q_3\}$$

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

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

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

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

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

\vdots

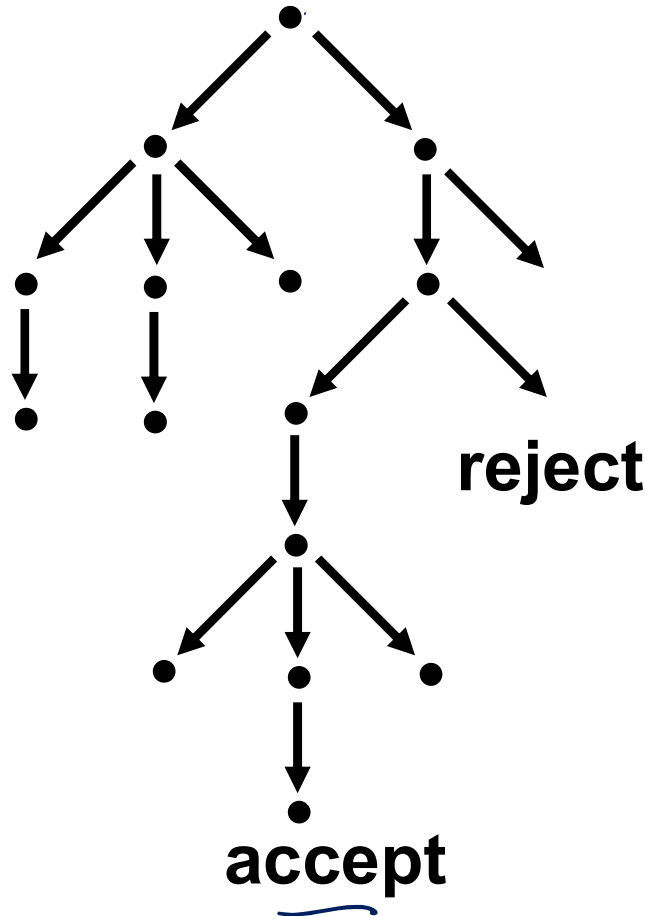
Nondeterminism

Deterministic Computation



accept or reject

Nondeterministic Computation



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

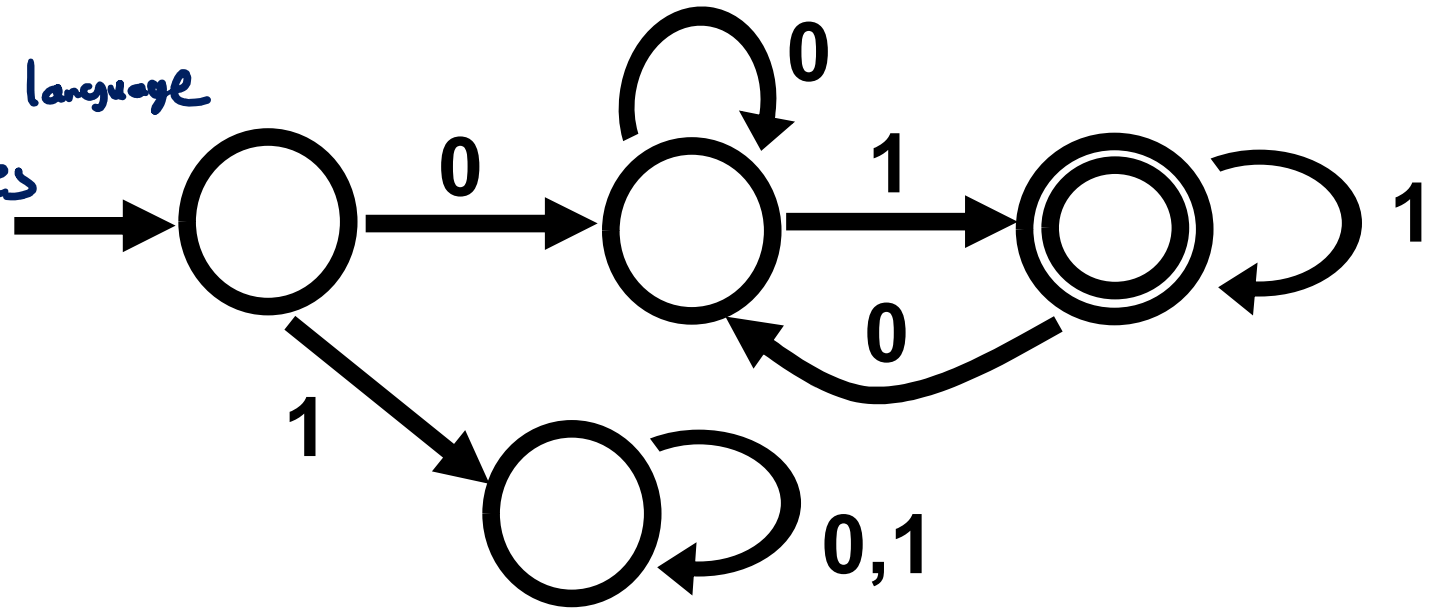
But:

- NFAs can be simpler than DFAs
- Useful for understanding power of DFAs/regular languages
- Lets us study “nondeterminism” as a resource
(cf. P vs. NP)

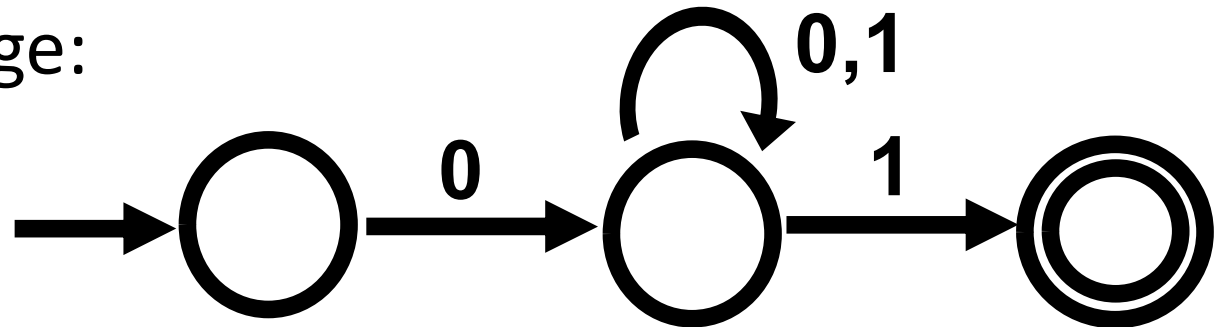
NFAs can be simpler than DFAs

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

Every DFA for this language 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

Class of regular languages = class of languages recognizable by DFAs
 \subseteq class of languages recognizable by NFAs

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

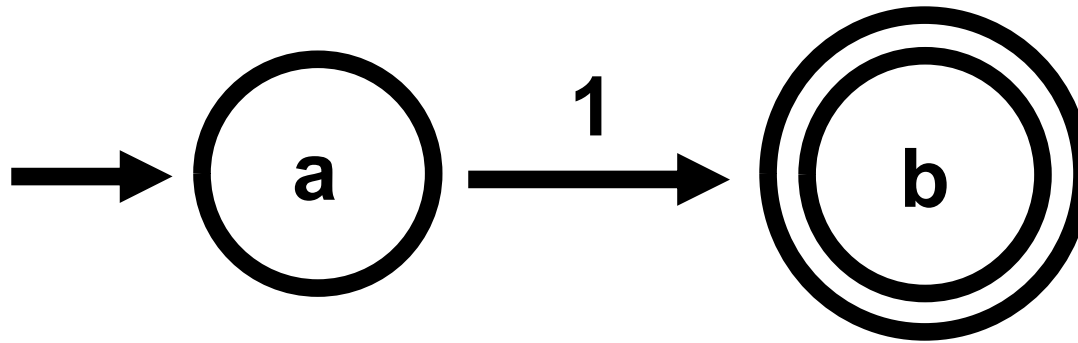
class of langs. recognizable by NFAs
 \subseteq class of langs. recognizable by DFAs

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

NFA \rightarrow DFA Example

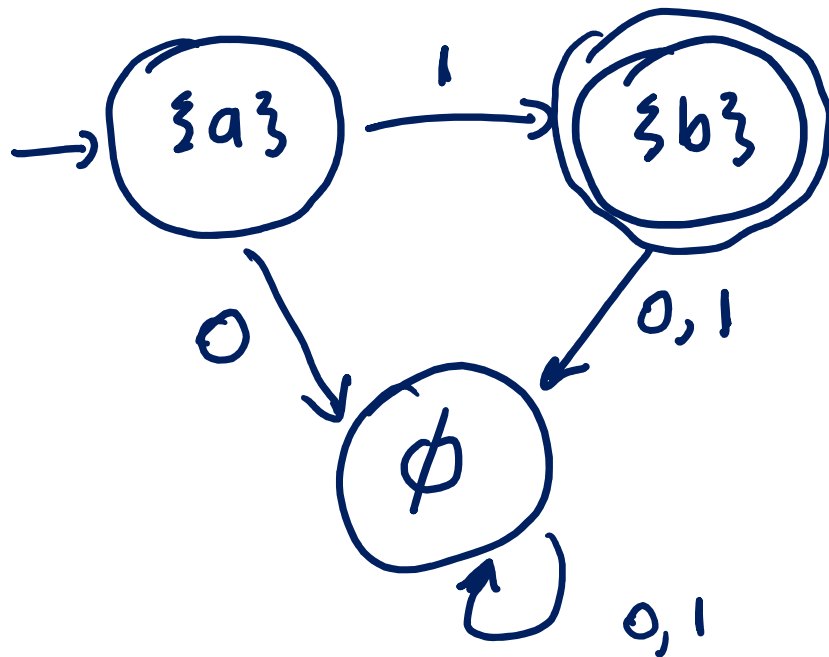
$$\Sigma = \{0, 1\}$$

$N =$



$$L(N) = L(D) = \{1\}$$

$D =$



Subset Construction (Formally, first attempt)

Input: NFA $N = (Q, \Sigma, \delta, q_0, F)$ Does not address 2-transitions

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

$$Q' = P(Q) = \{R \mid R \subseteq Q\} \quad (\text{power set} = \text{set of all subsets of } Q)$$

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

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

$=$ set of possible states N could be in if started from q_0 and read symbol σ .
 $q_0' = \{q_0\}$ set of states N could reach when reading σ starting from some state in R and read symbol σ .

$$F' = \{R \subseteq Q \mid R \text{ contains an accept state of } N\} = \{R \subseteq Q \mid R \cap F \neq \emptyset\}$$

$=$ set of states of M (i.e., sets of states of N) containing some accept state of N

Subset Construction (Formally, for real)



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

$R = \{q_0, q_1\}$ $E(R) = \{q_0, q_1, q_2, q_3\}$

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

Def. For a set R of NFA states, define $E(R)$ = set of states reachable from R by following ϵ -transitions 0 or more

$$Q' = P(Q)$$

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

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

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

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

Proving the Construction Works

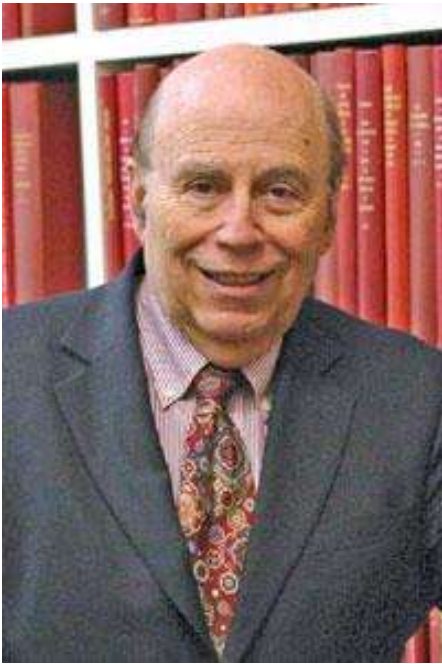
Claim: For every string w , running M on w leads to state

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

Proof idea: By induction on $|w|$

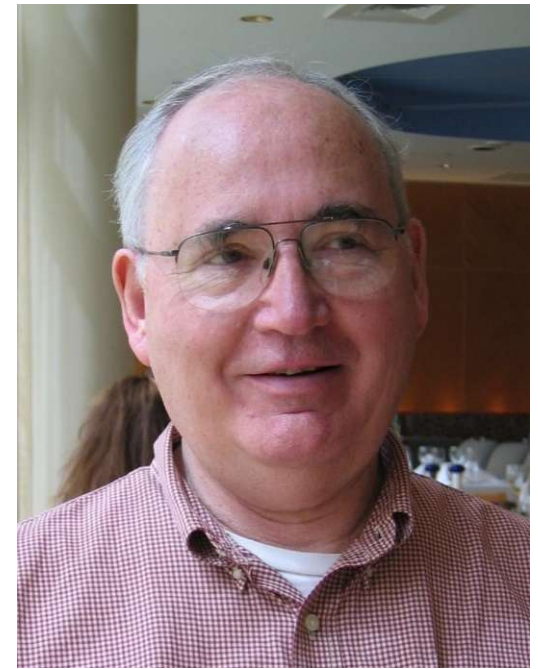
Historical Note

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.



NFA \rightarrow 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.)

$$|P(Q)| = 2^s$$

a) s

b) s^2

c) 2^s

d) None of the above

Is this construction the best we can do?

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

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

Theorem: For every $s \geq 1$, there is a language L_s such that

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

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