BU CS 332 – Theory of Computation

Lecture 4:

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

Reading:

Sipser Ch 1.1-1.2

Mark Bun

February 3, 2021

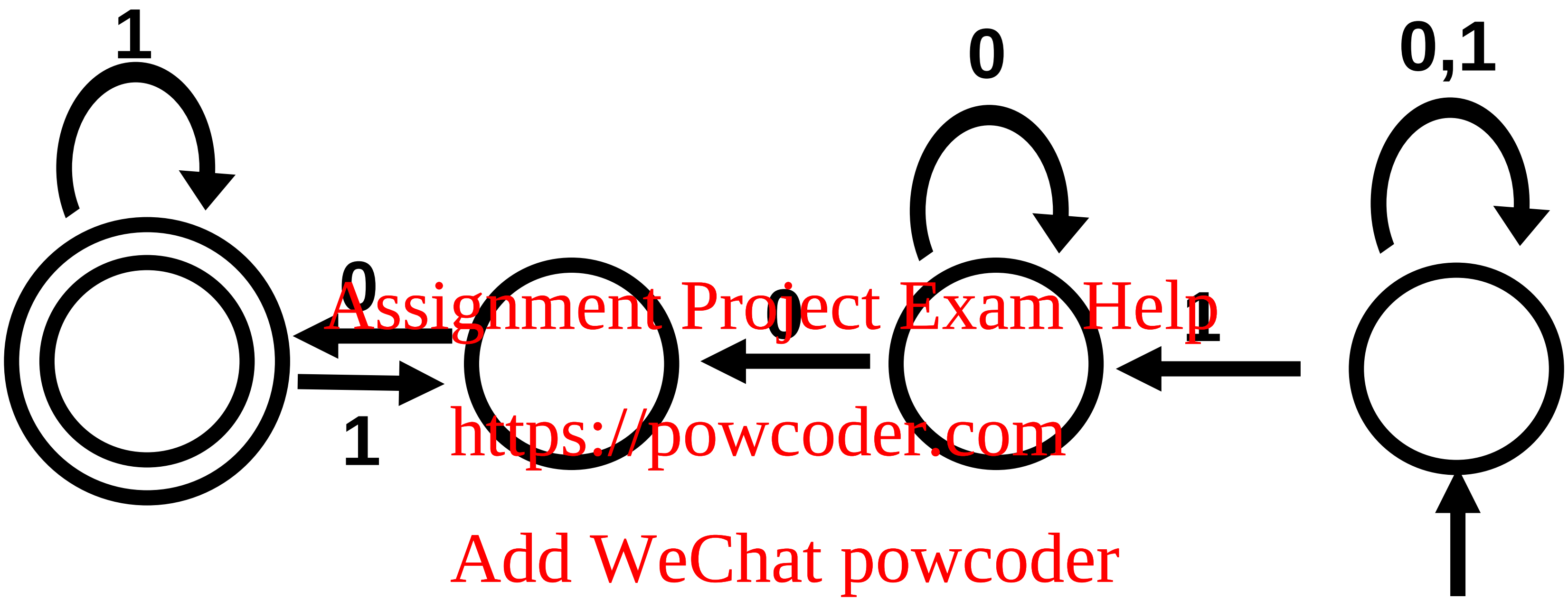
Last Time

- 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 FAs

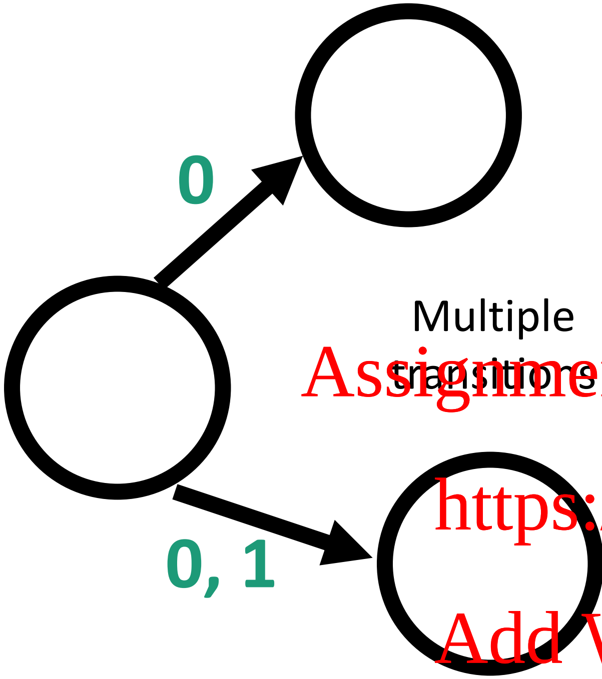
Nondeterminism



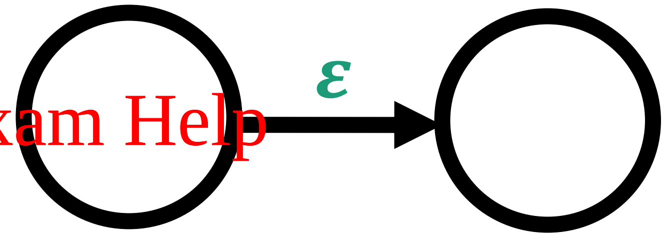
A Nondeterministic Finite Automaton (NFA) accepts if there *exists* a way to make it reach an accept state.

Some special transitions

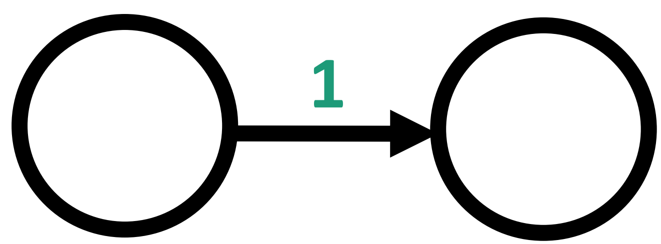
Multiple transitions



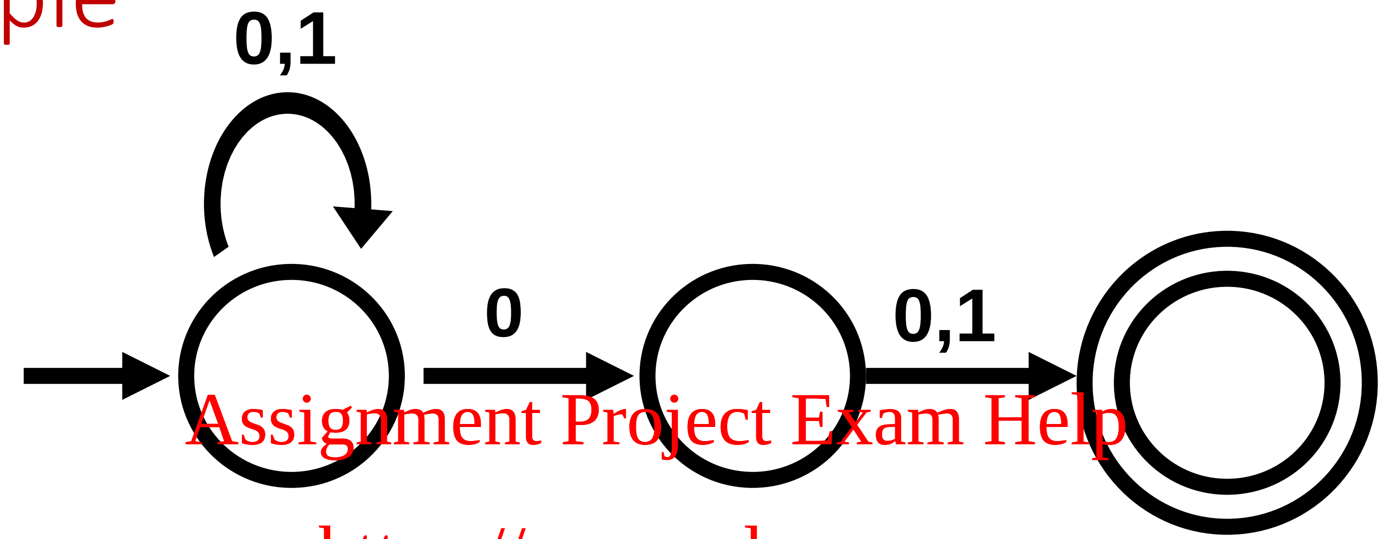
ε-transitions
(don't consume a symbol)



No transition



Example



$\boldsymbol{L(N)} =$

a) $\{w \mid w$ contains 00 or 01$\}$
b) $\{w \mid$ the second to last symbol of w is 0$\}$
c) $\{w \mid w$ starts with 00 or 01$\}$
d) $\{w \mid w$ ends with 001$\}$

Formal Definition of a NFA

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

Q is the set of states

Σ is the alphabet

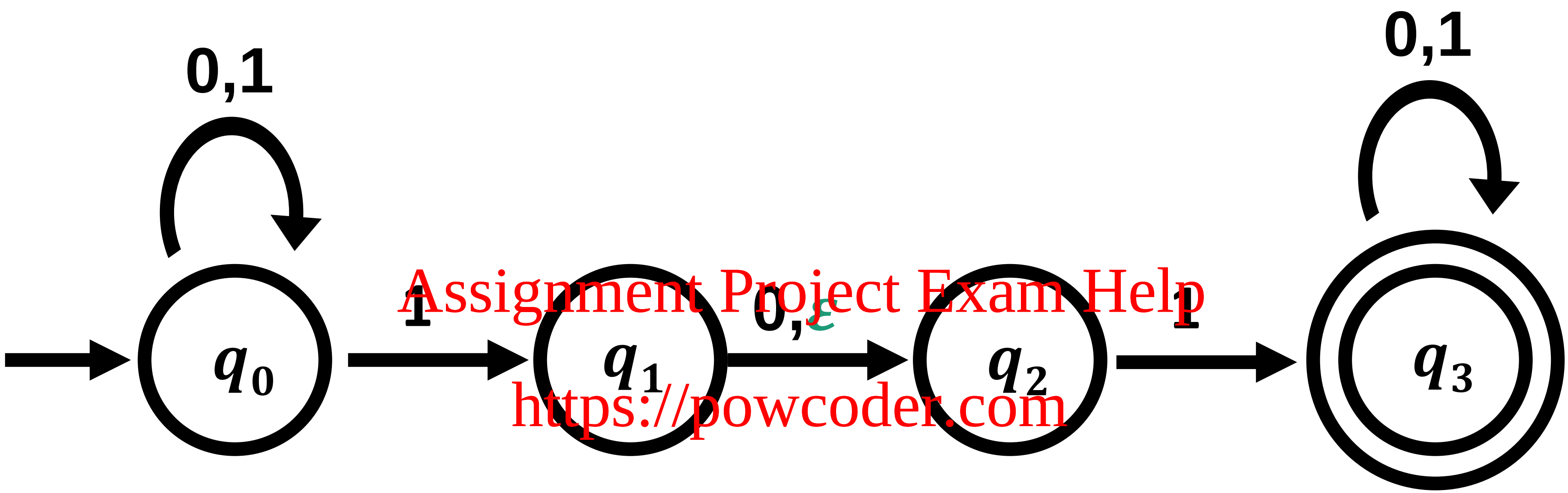
$\delta: Q \times \Sigma_\varepsilon \rightarrow P(Q)$ is the transition function

$q_0 \in Q$ 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



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

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

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

$F = \{q_3\}$

$\delta(q_0, 0) =$

$\delta(q_0, 1) =$

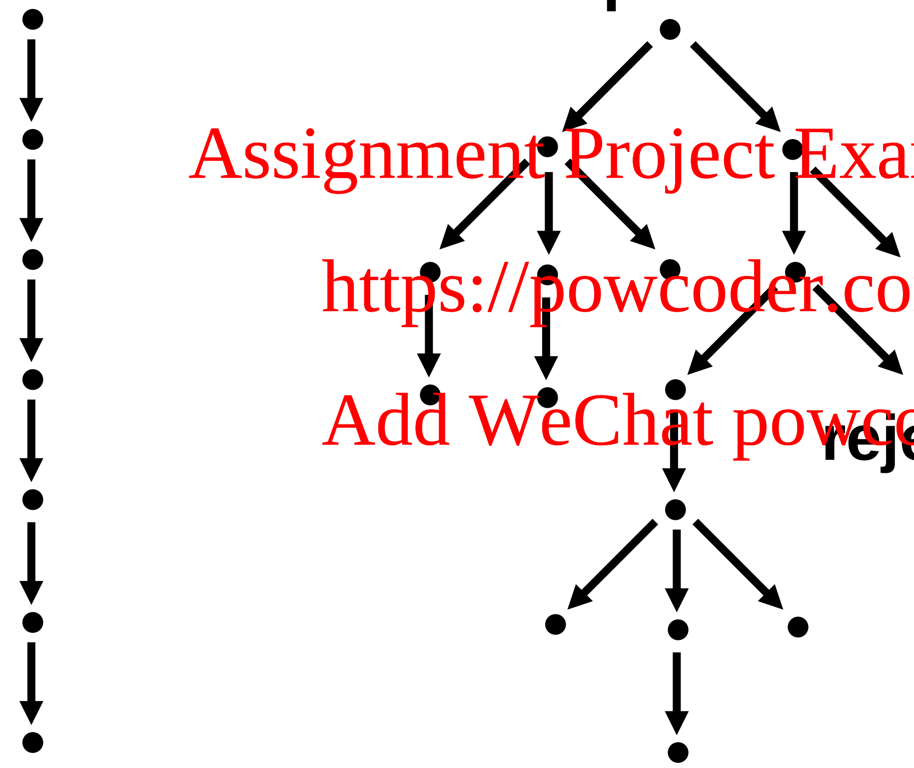
$\delta(q_1, \varepsilon) =$

$\delta(q_2, 0) =$

Nondeterminism

Deterministic Computation

Nondeterministic Computation



accept or reject

accept

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:

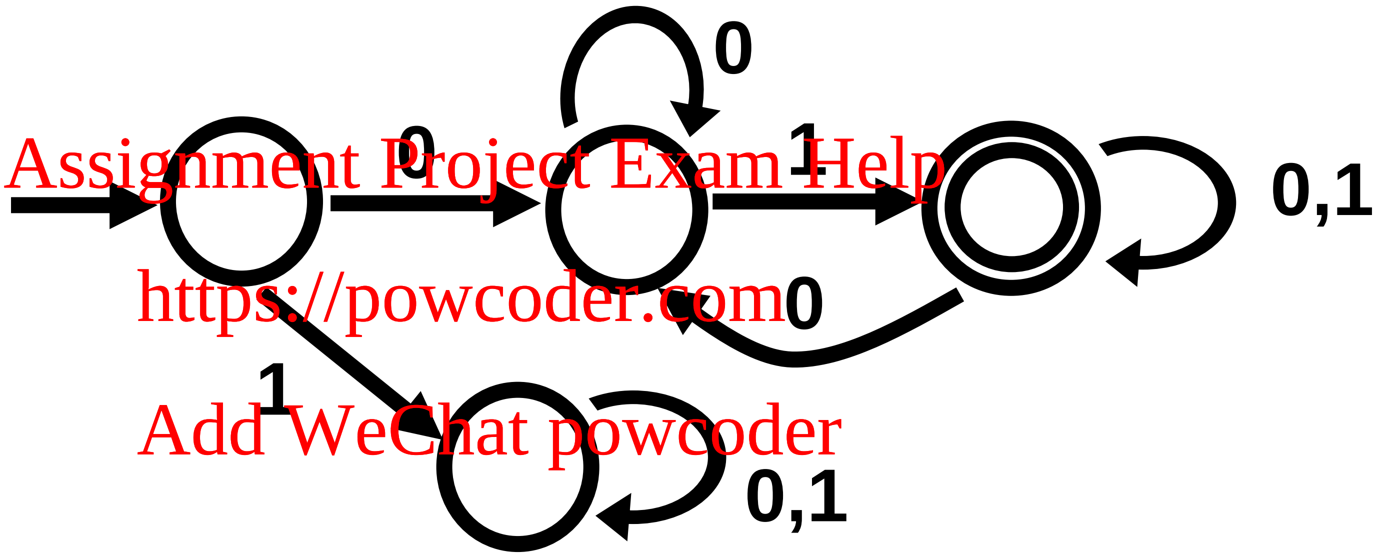
- Useful tool for understanding power of DFAs/regular languages

- NFAs can be simpler than DFAs
- Lets us study "nondeterminism" as a resource

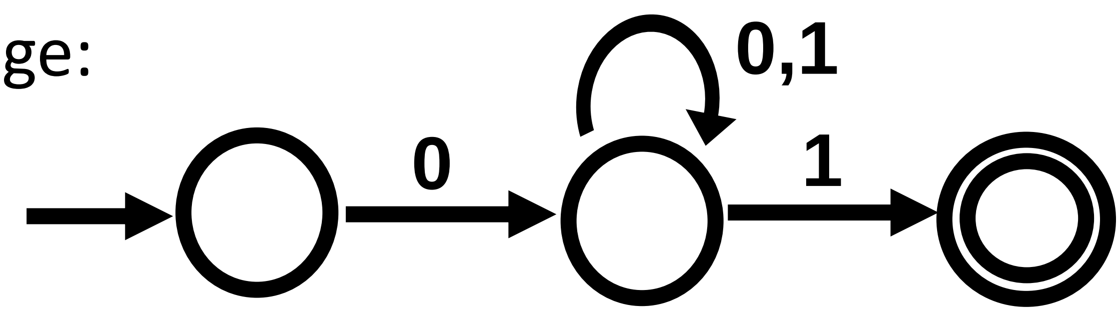
  (cf. P vs. NP)

NFAs can be simpler than DFAs

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



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

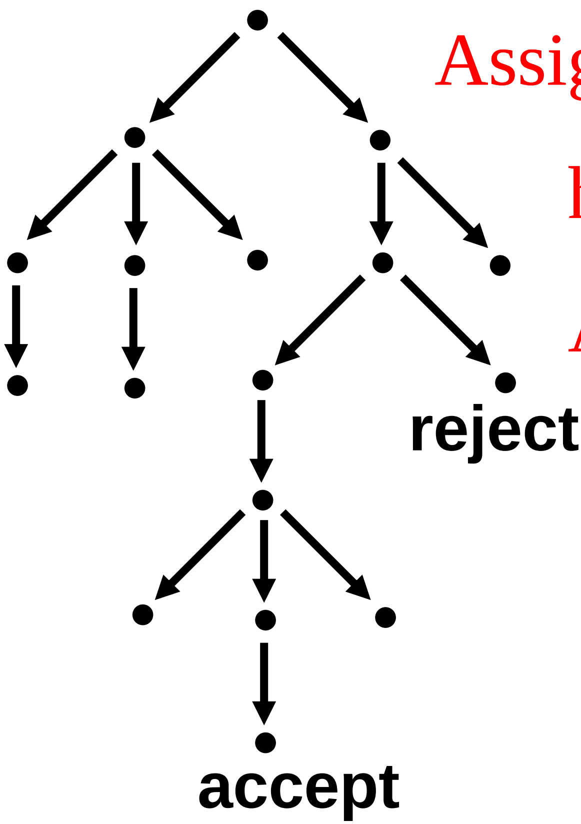
Theorem: For every NFA N, there is a DFA M such that $L(M) = L(N)$

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 $L(N)$

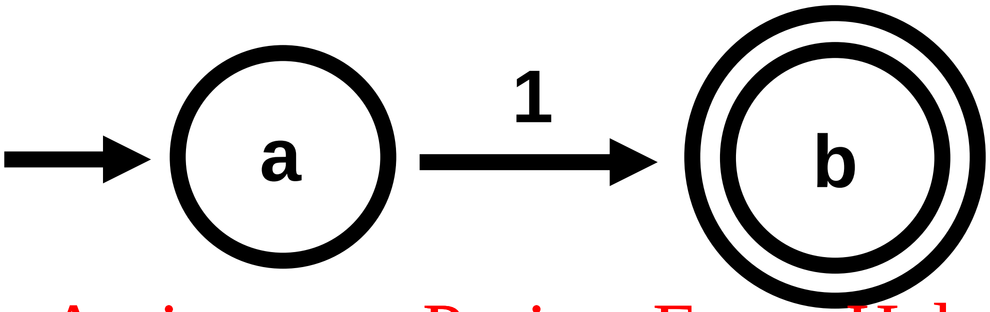


Intuition: Run all threads of N in parallel, maintaining the set of states where all threads are.

Formally: $Q' = P(Q)$

“The Subset Construction”

NFA -> DFA Example



Subset Construction (Formally, first attempt)

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

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

Q'

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

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

$q_0' =$

$F' =$

Subset Construction (Formally, for real)

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

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

$Q' = P(Q)$

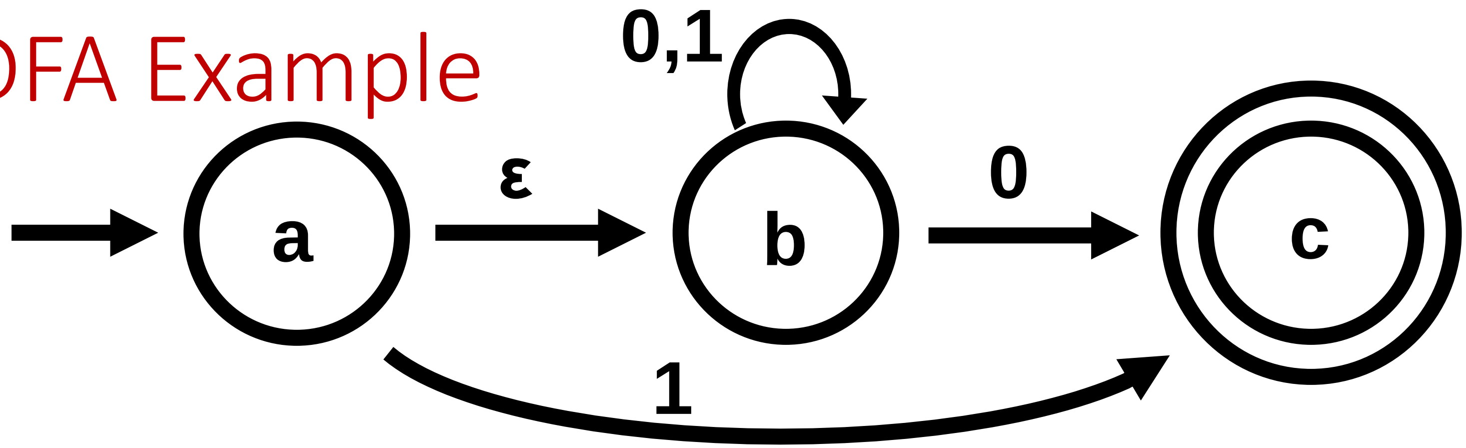
$\delta' : Q' \times \Sigma \rightarrow Q'$

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

$q_0' = \{q_0\}$

$F' = \{ R \in Q' \mid R$ 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

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

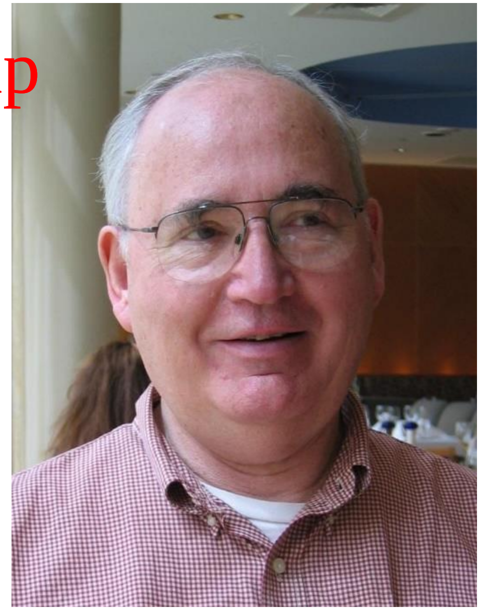
Proof idea: By induction on $|w|$

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 -> DFA: The Catch

If N is an NFA with s states, how many states does the DFA obtained using the subset construction have?

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 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).

Closure Properties

An Analogy

In algebra, we try to identify operations which are common to many different mathematical structures

Example: The integers $\mathbb{Z} = \{\ldots -2, -1, 0, 1, 2, \ldots\}$ are **closed** under

- Addition: $x + y$
- Multiplication: $x \times y$
- Negation: $-x$
- …but NOT Division: $x \mathbin{/} y$

We'd like to investigate similar closure properties of the class of regular languages

Regular operations on languages

Let $A, B \subseteq \Sigma^*$ be languages. Define

Union: $A \cup B = \{w \mid w \in A \textbf{ or } w \in B\}$

Concatenation: $A \circ B = \{xy \mid x \in A, y \in B\}$

Star: $A^* =$

Other operations

Let $A, B \subseteq \Sigma^*$ be languages. Define

Complement: $\bar{A} = \{w \mid w \notin A\}$

Intersection: $A \cap B = \{w \mid w \in A \textbf{ and } w \in B\}$

Reverse: $A^R = \{w \mid w^R \in A\}$

Closure properties of the regular languages

Theorem: The class of regular languages is closed under all three regular operations (union, concatenation, star), as well as under complement, intersection, and reverse.

i.e., if A and B are regular, applying any of these operations yields a regular language

Proving Closure Properties

Complement

Complement: $\bar{A} = \{ w \mid w \notin A \}$

Theorem: If A is regular, then $\bar{A}$ is also regular

Proof idea:

Complement, Formally

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing a language A. Which of the following represents a DFA recognizing $\bar{A}$?

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

b) $(Q, \Sigma, \delta, q_0, Q \setminus F)$, where $Q \setminus F$ is the set of states in Q that are not in F

c) $(Q, \Sigma, \delta', q_0, F)$ where $\delta'(q, s) = p$ such that $\delta(p, s) = q$

d) None of the above

Closure under Concatenation

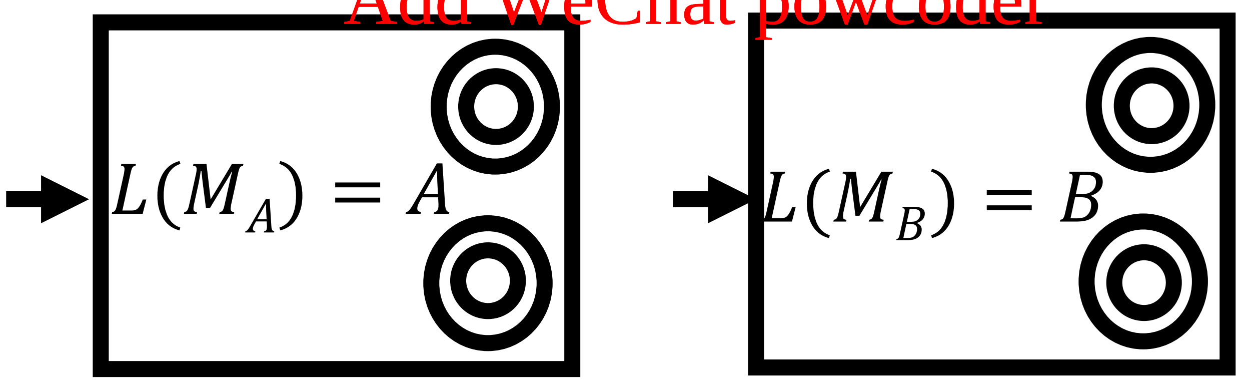
Concatenation: $A \circ B = \{ xy \mid x \in A, y \in B \}$

Theorem. If A and B are regular, $A \circ B$ is also regular.

Proof idea: Given DFAs M_A and M_B, construct NFA by

- Connecting all accept states in M_A to the start state in M_B.

- Make all states in M_A non-accepting.



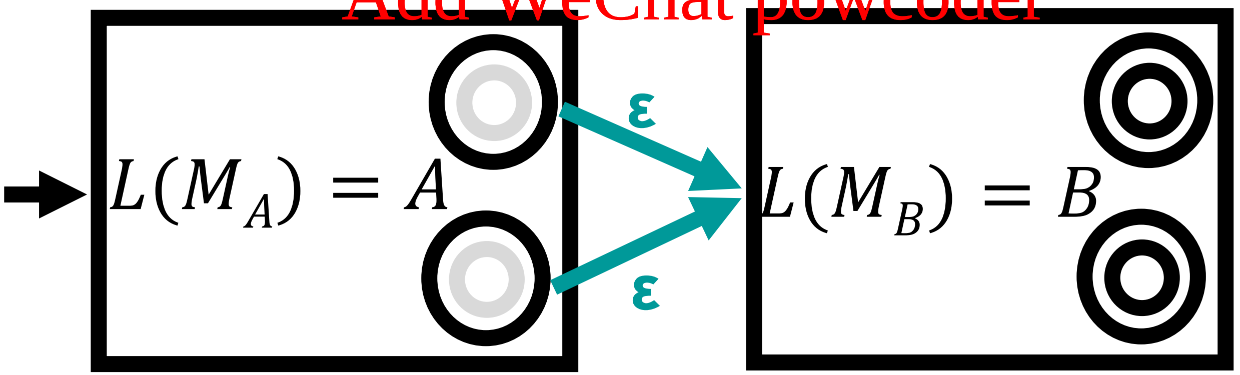
Closure under Concatenation

Concatenation: $A \circ B \; = \; \{ \, xy \; | x \in A, y \in B \, \}$

Theorem. If A and B are regular, $A \circ B$ is also regular.

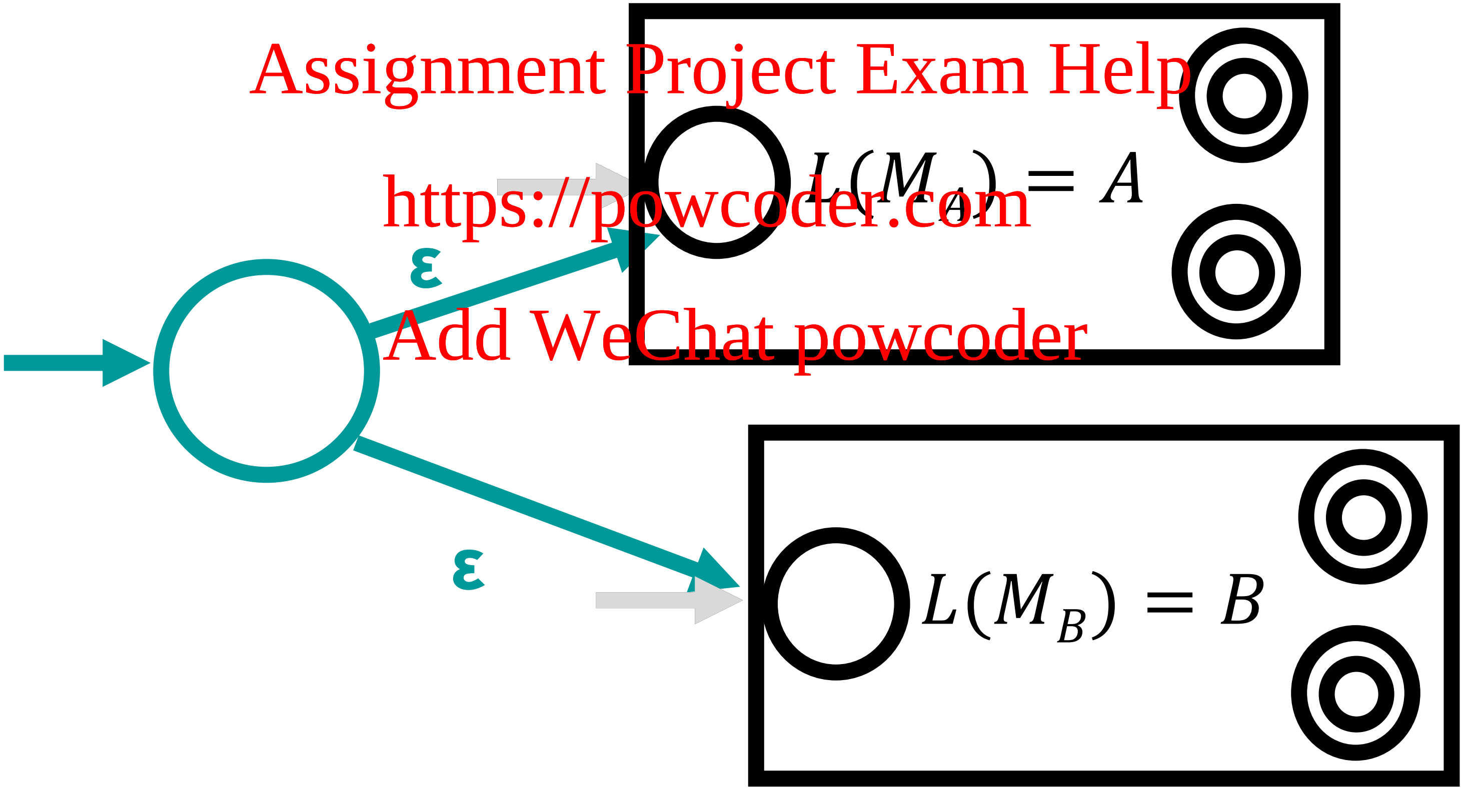
Proof idea: Given DFAs M_A and M_B, construct NFA by

- Connecting all accept states in M_A to the start state in M_B.
- Make all states in M_A non-accepting.



A Mystery Construction

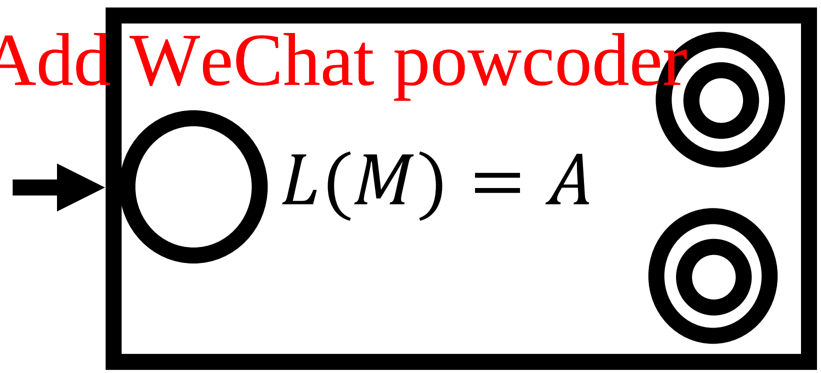
Given DFAs M_A recognizing A and M_B recognizing B, what does the following NFA recognize?



Closure under Star

Star: $A^* = \{ a_1 a_2 \ldots a_n \mid n \geq 0 \text{ and } a_i \in A \}$

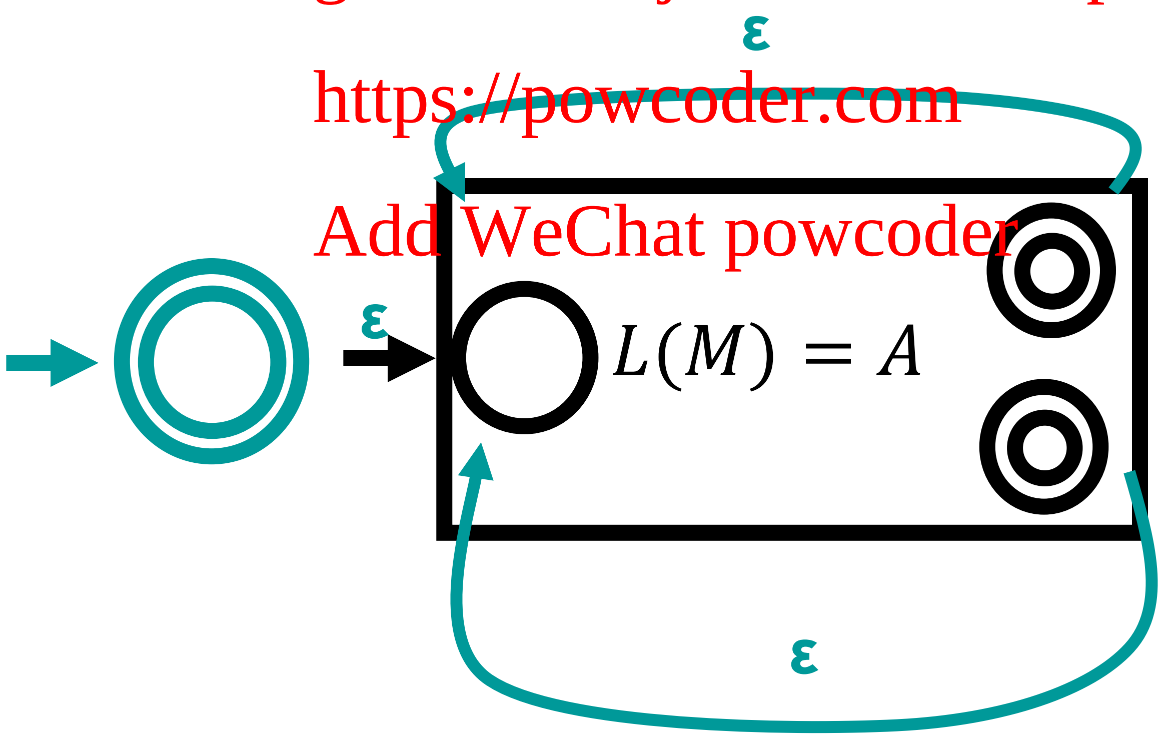
Theorem. If A is regular, A^* is also regular.



Closure under Star

Star: $A^* = \{ a_1a_2...a_n | n \geq 0 \text{ and } a_i \in A\}$

Theorem. If A is regular, A^* is also regular.



On proving your own closure properties

You'll have homework/test problems of the form "show that the regular languages are closed under operation op"

What would Sipser do?

- Give the "proof idea": Explain how to take machine(s) recognizing regular language(s) and create a new machine

- Explain in a few sentences why the construction works
- Give a formal description of the construction
- No need to formally prove the construction works