

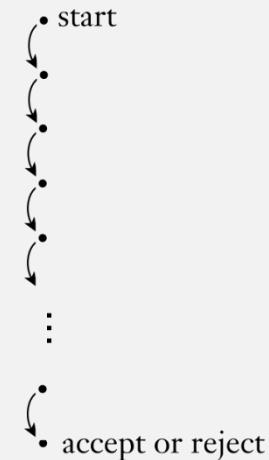
CS420

Nondeterminism and NFAs

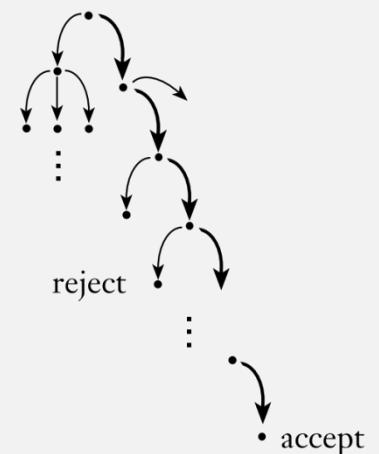
Wednesday, February 2, 2022

UMass Boston Computer Science

Deterministic
computation



Nondeterministic
computation



Announcements

- HW 1
 - due Sunday 2/6 11:59pm EST

Last Time: Formal Definition of NFAs

DEFINITION

A **nondeterministic finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,

2. Σ is a finite alphabet,

3. $\delta: Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$ is the transition function,

4. $q_0 \in Q$ is the start state, and

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

NFA transition not required to read input, $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

Transition results in a set of states

Compare with DFA:

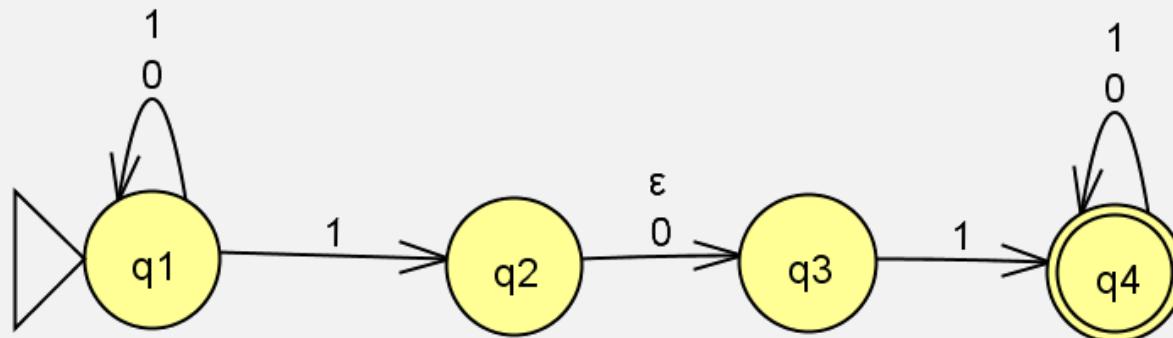
A **finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta: Q \times \Sigma \rightarrow Q$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states**.

Different transition fn

Last Time: NFA Example

- Come up with a formal description of the following NFA:



DEFINITION

A *nondeterministic finite automaton*

is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta: Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

The formal description of N_1 is $(Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3, q_4\}$,
2. $\Sigma = \{0, 1\}$,
3. δ is given as

Empty transition
(no input read)

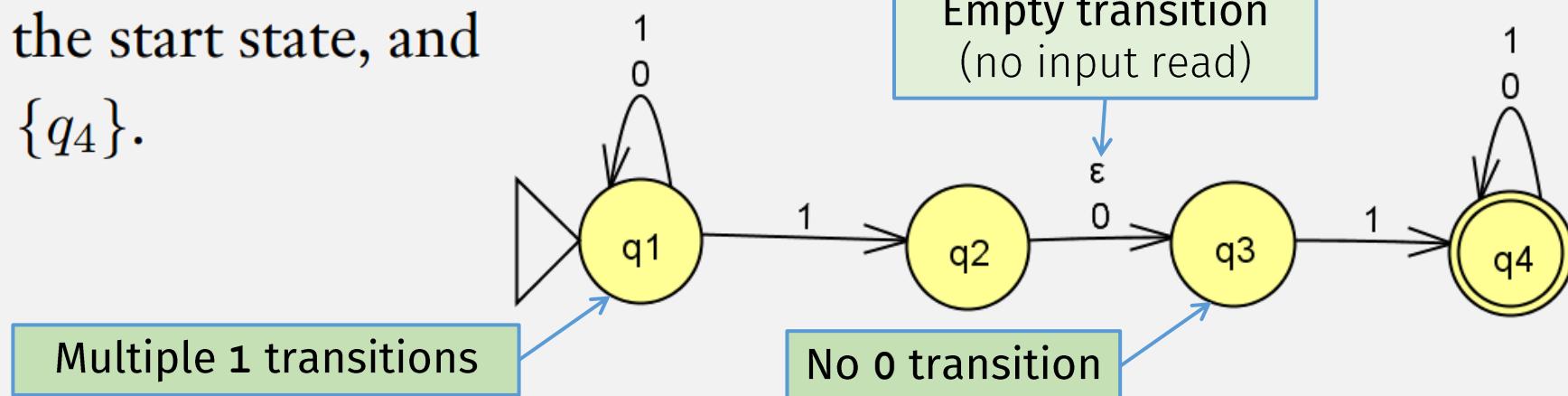
$$\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$$

| | 0 | 1 | ε |
|-------|-------------|----------------|---------------|
| q_1 | $\{q_1\}$ | $\{q_1, q_2\}$ | \emptyset |
| q_2 | $\{q_3\}$ | \emptyset | $\{q_3\}$ |
| q_3 | \emptyset | $\{q_4\}$ | \emptyset |
| q_4 | $\{q_4\}$ | $\{q_4\}$ | \emptyset |

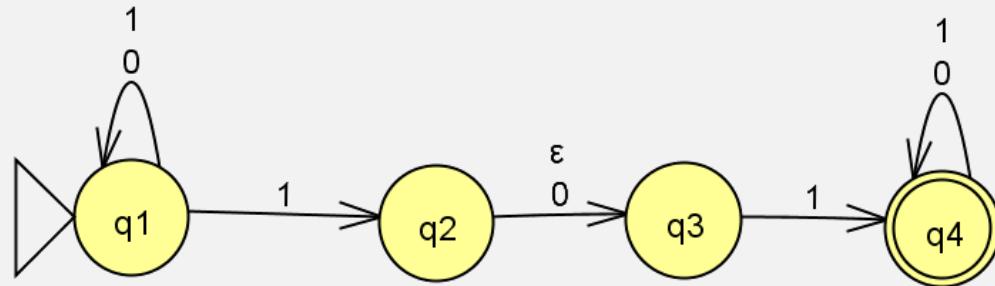
Result of transition
is a set

Empty transition
(no input read)

4. q_1 is the start state, and
5. $F = \{q_4\}$.



Running Programs, NFAs (JFLAP demo): 010110



NFA Computation Sequence

Symbol read

0

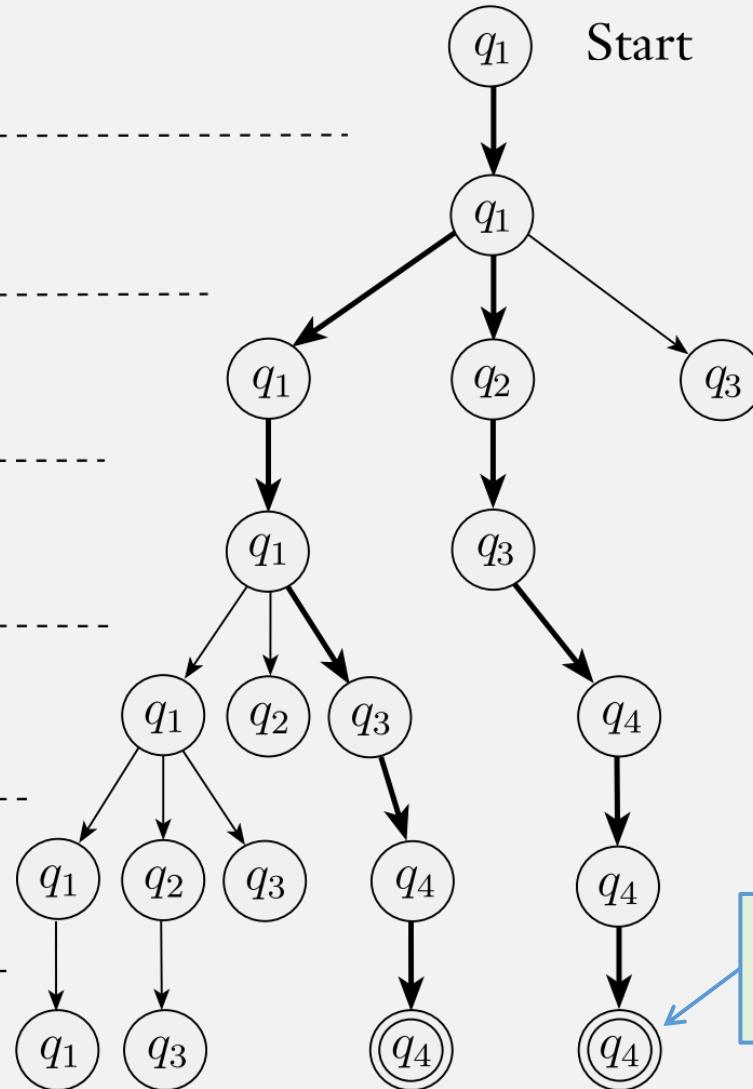
1

0

1

1

0



An NFA accepts its input if at least one path ends in an accept state

Each step can branch into multiple states at the same time!

So this is an accepting computation

Flashback: DFA Computation Model

Informally

- Machine = a DFA
- Input = string of chars

Machine accepts input if:

- Start in “start state”
- Repeat:
 - Read 1 char, change state according to transition fn
- Result =
 - “Accept” if last state is “Accept” state
 - “Reject” otherwise

Formally

- $M = (Q, \Sigma, \delta, q_0, F)$
 - $w = w_1 w_2 \cdots w_n$
- M **accepts** w if sequence of states r_0, r_1, \dots, r_n in Q exists with
- $r_0 = q_0$
 - $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \dots, n - 1$
 - $r_n \in F$

NFA

~~Flashback:~~ ~~DFA~~ Computation Model

Informally

- Machine = a ~~DFA~~ NFA
- Input = string of chars

Machine accepts input if:

- Start in “start state”
- Repeat:
 - Read 1 char, change state according to transition fn
- Result =
 - “Accept” if last state is “Accept” state
 - “Reject” otherwise

Formally

- $M = (Q, \Sigma, \delta, q_0, F)$
 - $w = w_1 w_2 \cdots w_n$
- M **accepts** w if sequence of states r_0, r_1, \dots, r_n in Q exists with
- $r_0 = q_0$
 - $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \dots, n - 1$
 - $r_n \in F$
- $r_{i+1} \in \delta(r_i, w_{i+1})$ This is now a set
- DEFINITION**

A *nondeterministic finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

 1. Q is a finite set of states,
 2. Σ is a finite alphabet
 3. $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition function,
 4. $q_0 \in Q$ is the start state, and
 5. $F \subseteq Q$ is the set of accept states.

Flashback: DFA Extended Transition Fn

Define the extended transition function: $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$

- Inputs:
 - Beginning state $q \in Q$
 - Input string $w = w_1 w_2 \cdots w_n$
- Output:
 - Ending state

(Defined recursively)

- Base case: $\hat{\delta}(q, \varepsilon) = q$
- Recursive case: $\hat{\delta}(q, w) = \hat{\delta}(\delta(q, \text{First char}), \text{Remaining chars})$
 - Empty string
 - First char
 - Remaining chars
 - Recursive call
 - Single transition step

NFA

~~Flashback:~~ DFA - Extended Transition Fn

Define the extended transition function: $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$

- Inputs:
 - Beginning state $q \in Q$
 - Input string $w = w_1 w_2 \cdots w_n$
- Output:
 - Ending state Set of ending states

$$\hat{\delta} : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$$

Result is set of states

(Defined recursively)

• Base case: $\hat{\delta}(q, \epsilon) = q$ $\hat{\delta}(q, \epsilon) = \{q\}$

First char

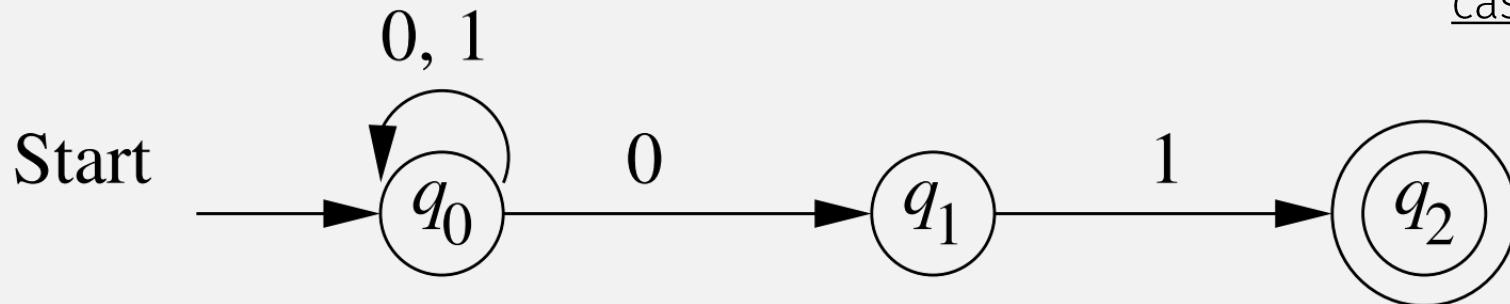
$$\delta(q, w_1) = \{q_1, \dots, q_k\}$$

• Recursive case: $\hat{\delta}(q, w) = \hat{\delta}(\delta(q, w_1), u)$

$$\hat{\delta}(q, w) = \bigcup_{i=1}^k \hat{\delta}(q_i, w_2 \cdots w_n)$$

Combine results of recursive calls for each q_i

NFA Extended δ Example



- $\hat{\delta}(q_0, \epsilon) = \{q_0\}$ Stay in start state
- $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$ Same as single step δ
- $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$ Combine results of recursive calls with “rest of input”
- $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

Base case: $\hat{\delta}(q, \epsilon) = \{q\}$

Recursive case: $\hat{\delta}(q, w) = \bigcup_{i=1}^k \hat{\delta}(q_i, w_2 \cdots w_n)$

where: $\delta(q, w_1) = \{q_1, \dots, q_k\}$

We haven't considered empty transitions!

Adding Empty Transitions

- Define the set $\varepsilon\text{-REACHABLE}(q)$
 - ... to be all states reachable from q via zero or more empty transitions

(Defined recursively)

- **Base case:** $q \in \varepsilon\text{-REACHABLE}(q)$

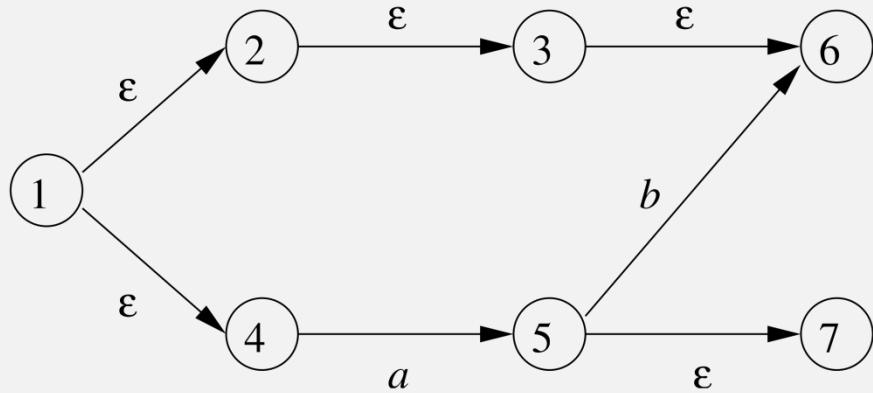
- **Inductive case:**

A state is in the reachable set if ...

$$\varepsilon\text{-REACHABLE}(q) = \{r \mid p \in \varepsilon\text{-REACHABLE}(q) \text{ and } r \in \delta(p, \varepsilon)\}$$

... there is an empty transition to it from another state in the reachable set

ε -REACHABLE Example



$$\varepsilon\text{-REACHABLE}(1) = \{1, 2, 3, 4, 6\}$$

NFA Extended Transition Function

Define the extended transition function: $\hat{\delta} : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$

- *Inputs:*
 - Some beginning state q
 - Input string $w = w_1 w_2 \cdots w_n$
- *Output:*
 - Set of ending states

(Defined recursively)

- Base case: $\hat{\delta}(q, \epsilon) = \{q\}$
- Recursive case:
 - If: $\delta(q, w_1) = \{q_1, \dots, q_k\}$

- Then:
$$\hat{\delta}(q, w) = \bigcup_{i=1}^k \hat{\delta}(q_i, w_2 \cdots w_n)$$

NFA Extended Transition Function

Define the extended transition function: $\hat{\delta} : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$

- *Inputs:*
 - Some beginning state q
 - Input string $w = w_1 w_2 \cdots w_n$
- *Output:*
 - Set of ending states

(Defined recursively)

- Base case: $\hat{\delta}(q, \epsilon) = \{q\}$ $\varepsilon\text{-REACHABLE}(q)$

- Recursive case:

- If: $\delta(q, w_1) = \{q_1, \dots, q_k\} \xrightarrow{\quad} \bigcup_{i=1}^k \varepsilon\text{-REACHABLE}(q_i) = \{r_1, \dots, r_m\}$

- Then: $\hat{\delta}(q, w) = \bigcup_{i=1}^k \hat{\delta}(q_i, w_2 \cdots w_n)$

Summary: NFAs vs DFAs

DFAs

- Can only be in one state
- Transition:
 - Must read 1 char
- Acceptance:
 - If final state is accept state

NFAs

- Can be in multiple states
- Transition
 - Can read no chars
 - i.e., empty transition
- Acceptance:
 - If one of final states is accept state

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

Last Time: Concatenation of Languages

Let the alphabet Σ be the standard 26 letters $\{a, b, \dots, z\}$.

If $A = \{\text{good}, \text{bad}\}$ and $B = \{\text{boy}, \text{girl}\}$, then

$$A \circ B = \{\text{goodboy}, \text{goodgirl}, \text{badboy}, \text{badgirl}\}$$

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

Last Time: Concatenation is Closed?

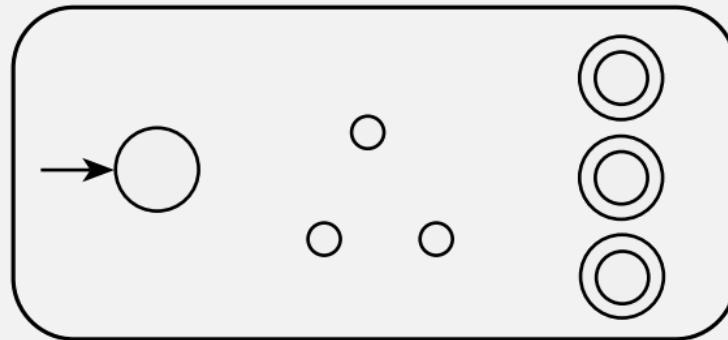
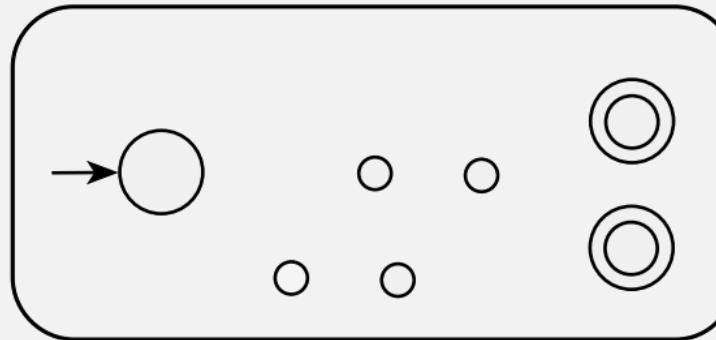
THEOREM

The class of regular languages is closed under the concatenation operation.

In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

Proof: Construct a new machine

- How does it know when to switch machines?
 - Can only read input once

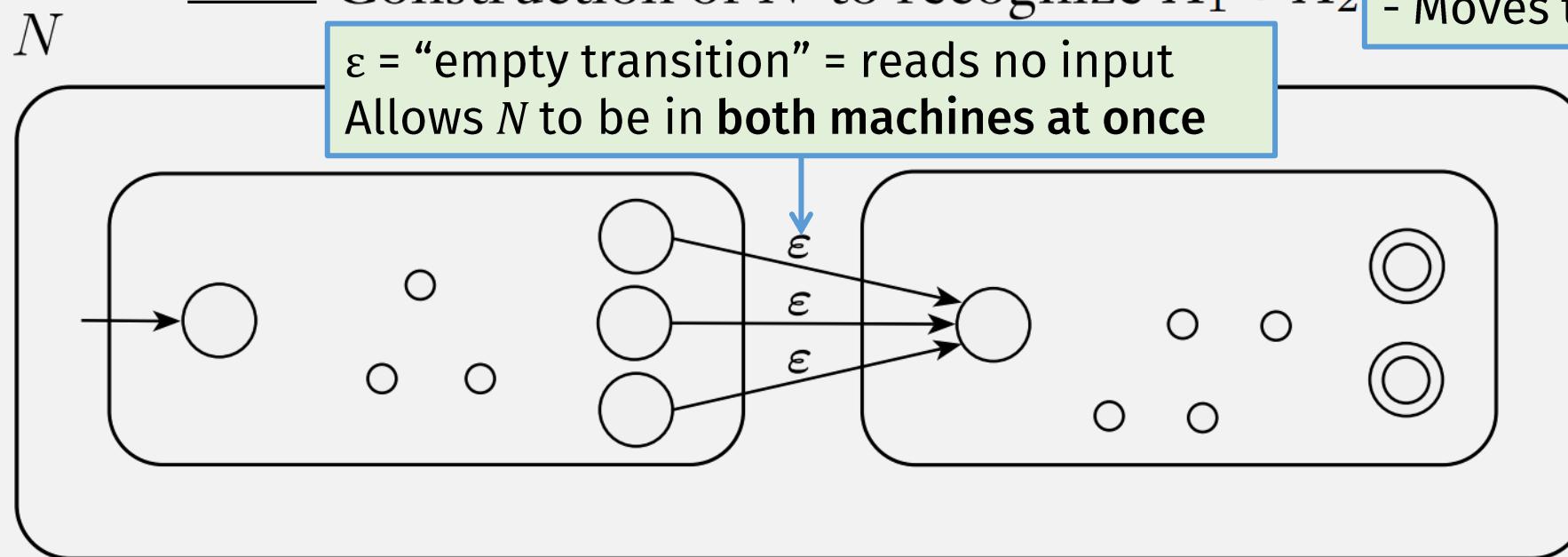
N_1  N_2 

Let N_1 recognize A_1 , and N_2 recognize A_2 .

Want: Construction of N to recognize $A_1 \circ A_2$

N is an NFA! It simultaneously:

- Keeps checking 1st part with N_1 **and**
- Moves to N_2 to check 2nd part



Concatenation is Closed for Regular Langs

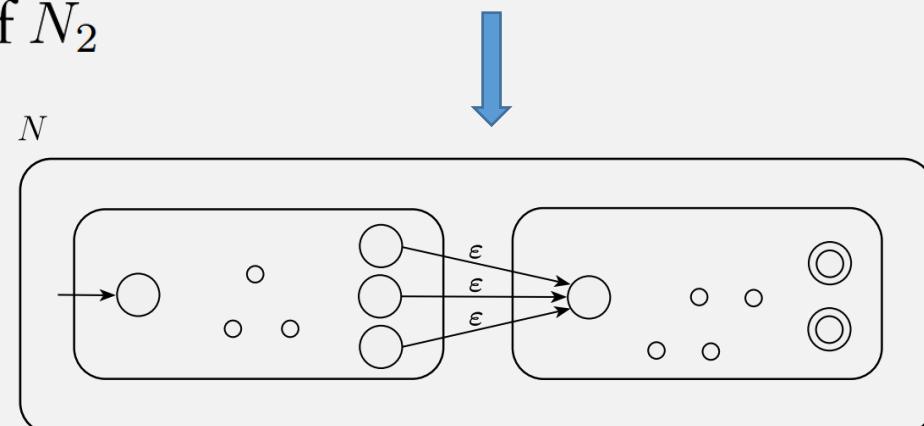
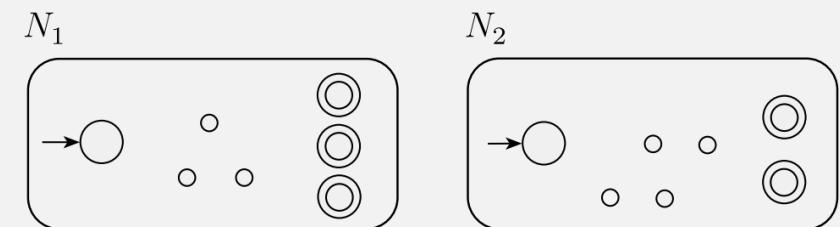
PROOF

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1 , and

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2 .

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$

1. $Q = Q_1 \cup Q_2$
2. The state q_1 is the same as the start state of N_1
3. The accept states F_2 are the same as the accept states of N_2
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$,



Concatenation is Closed for Regular Langs

PROOF

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1 , and

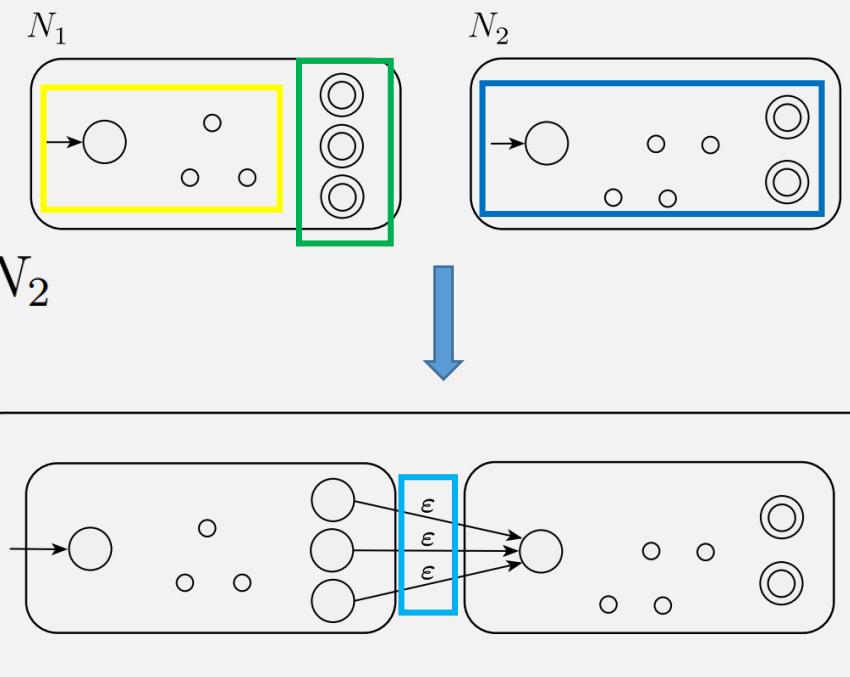
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2 .

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$

1. $Q = Q_1 \cup Q_2$
2. The state q_1 is the same as the start state of N_1
3. The accept states F_2 are the same as the accept states of N_2
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_2\} & q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q, a) & q \in Q_2. \end{cases}$$

Wait, is this true?



???

Flashback: A DFA's Language

- For DFA $M = (Q, \Sigma, \delta, q_0, F)$
- M **accepts** w if $\hat{\delta}(q_0, w) \in F$
- M **recognizes language** A if $A = \{w \mid M \text{ accepts } w\}$
- A language is a **regular language** if a DFA recognizes it

An NFA's Language

- For NFA $N = (Q, \Sigma, \delta, q_0, F)$
- N *accepts* w if $\hat{\delta}(q_0, w) \cap F \neq \emptyset$
 - i.e., if the final states have at least one accept state
- Language of $N = L(N) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$

Q: How does an NFA's language relate to regular languages

- Definition: A language is regular if a DFA recognizes it

Is Concatenation Closed for Reg Langs?

- Concatenation of DFAs produces an NFA
- To finish the proof, we must prove that NFAs *also* recognize regular languages.
- Specifically, we must prove:
 - NFAs \Leftrightarrow regular languages

How to Prove a Statement: $X \Leftrightarrow Y$

- $X \Leftrightarrow Y$ = “ X if and only if Y ” = X iff Y = $X \Leftrightarrow Y$
- Proof at minimum has 2 parts:
 1. \Rightarrow if X , then Y
 - “forward” direction
 - assume X , then use it to prove Y
 2. \Leftarrow if Y , then X
 - “reverse” direction
 - assume Y , then use it to prove X

Proving NFAs Recognize Regular Langs

Theorem:

A language L is regular if and only if some NFA N recognizes L .

Proof:

⇒ If L is regular, then some NFA N recognizes it.

- *Easier*
- We know: if L is regular, then a DFA exists that recognizes it.
- So to prove: Convert that DFA to an equivalent NFA! (see HW 2)

⇐ If an NFA N recognizes L , then L is regular.

- *Harder*
- We know: for L to be regular, there must be a DFA recognizing it
- Proof Idea: Convert given NFA to DFA

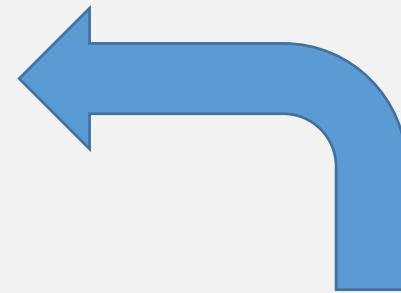
How to convert NFA→DFA?

A **finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta: Q \times \Sigma \rightarrow Q$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states**.

Proof idea:

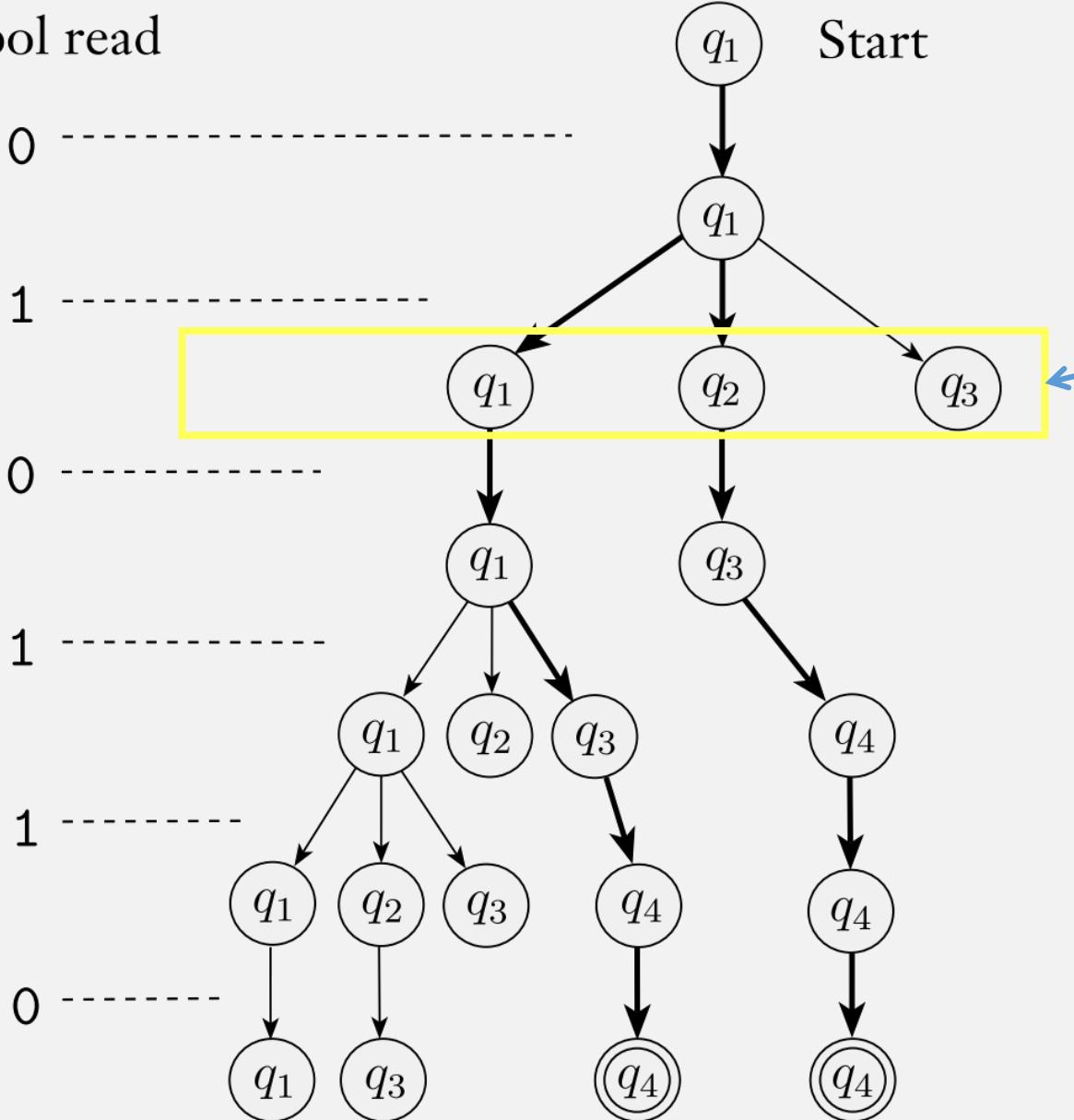
Let each “state” of the DFA be a set of states in the NFA



A **nondeterministic finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

Symbol read



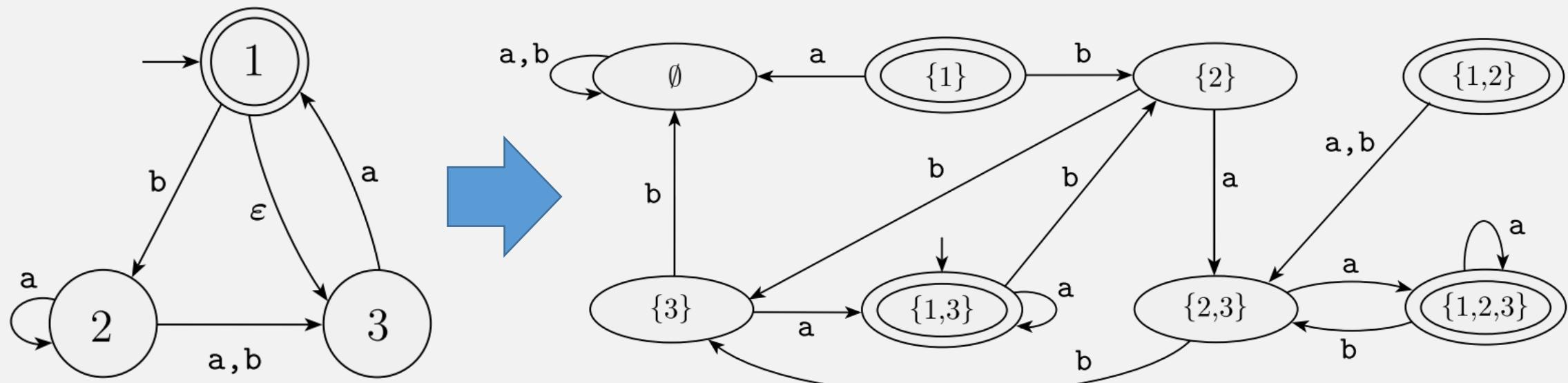
In a DFA, all these states at each step must be only **one** state

So design a state in
the DFA to be a
set of NFA states!

Convert NFA→DFA, Formally

- Let NFA $N = (Q, \Sigma, \delta, q_0, F)$
- An equivalent DFA M has states $Q' = \mathcal{P}(Q)$ (power set of Q)

Example:



The NFA N_4

A DFA D that is equivalent to the NFA N_4

NFA→DFA

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

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

1. $Q' = \mathcal{P}(Q)$ A state for M is a set of states in N

2. For $R \in Q'$ and $a \in \Sigma$, $R = \text{a state in } M = \text{a set of states in } N$

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$

Next state for DFA state R = next states of each NFA state r in R

3. $q_0' = \{q_0\}$

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

Flashback: Adding Empty Transitions

- Define the set $\varepsilon\text{-REACHABLE}(q)$
 - ... to be all states reachable from q via zero or more empty transitions

(Defined recursively)

- **Base case:** $q \in \varepsilon\text{-REACHABLE}(q)$

- **Inductive case:**

A state is in the reachable set if ...

$$\varepsilon\text{-REACHABLE}(q) = \{r \mid p \in \varepsilon\text{-REACHABLE}(q) \text{ and } r \in \delta(p, \varepsilon)\}$$

... there is an empty transition to it from another state in the reachable set

NFA \rightarrow DFA

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

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

1. $Q' = \mathcal{P}(Q)$

2. For $R \in Q'$ and $a \in \Sigma$,

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$

~~$\delta(r, a)$~~ $\varepsilon\text{-REACHABLE}(\delta(r, a))$

3. $q_0' = \underline{\{q_0\}}$ $\varepsilon\text{-REACHABLE}(q_0)$

Almost the same, except ...

Requires extending the fn
to sets of states (see HW 2)

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

Proving NFAs Recognize Regular Langs

Theorem:

A language L is regular if and only if some NFA N recognizes L .

Proof:

⇒ If L is regular, then some NFA N recognizes it.

- We know: If L is regular, then a DFA recognizes it.
- We show: How to convert a DFA to an equivalent NFA

⇐ If an NFA N recognizes L , then L is regular.

- We know: For L to be regular, there must be a DFA recognizing it
- We show: How to convert NFA N to an equivalent DFA ...
 - ... using the NFA→DFA algorithm we just defined!



Check-in Quiz 2/2

On gradescope