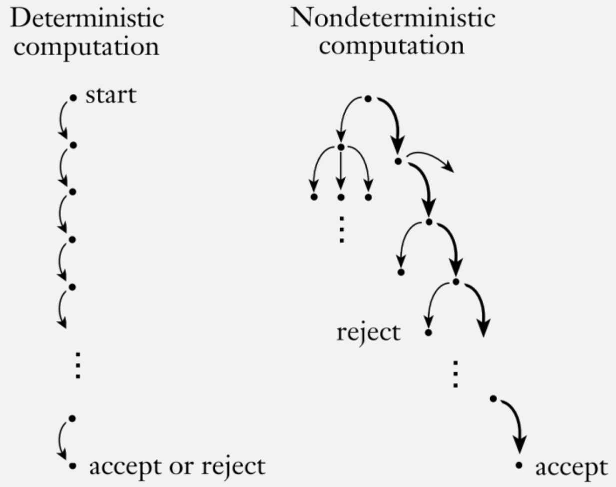
CS 622
Nondeterminism

Wednesday, February 16, 2024

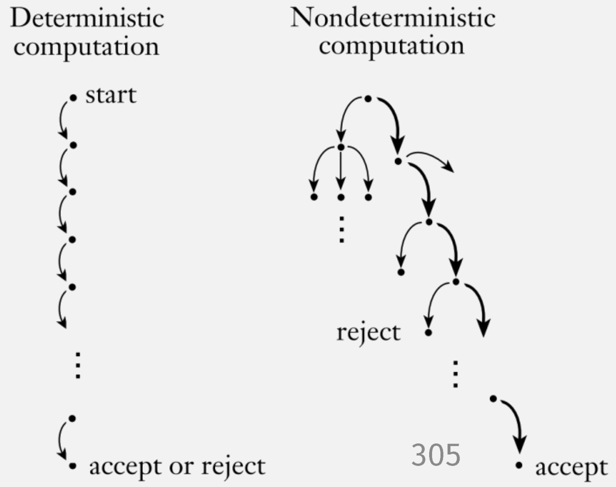
UMass Boston Computer Science



Announcements

- HW 2 out
  - ~~Due Mon 2/19 12pm EST (noon)~~
  - Due Wed 2/21 12pm EST (noon)

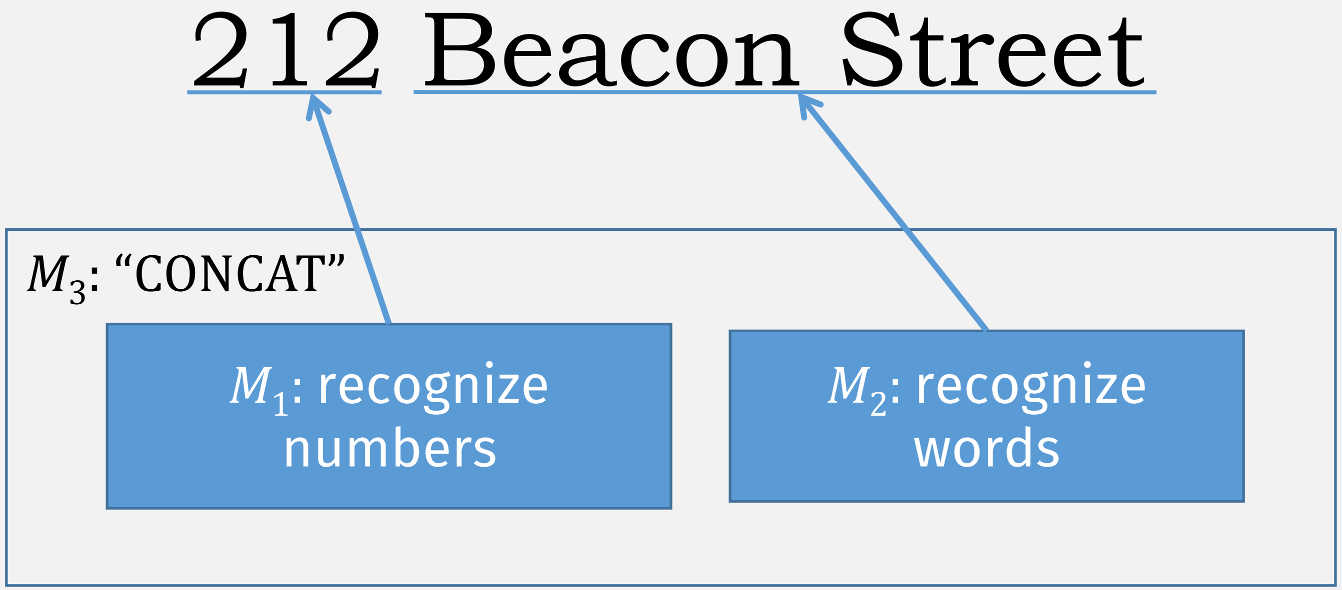
- No class Mon (2/19)



Previously

Another operation: Concatenation

Example: **Recognizing street addresses**



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

Concatenation of Languages

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

If $A = \{\texttt{fort}, \texttt{south}\}$ $B = \{\texttt{point}, \texttt{boston}\}$

$$A \circ B = \{\texttt{fortpoint}, \texttt{fortboston}, \texttt{southpoint}, \texttt{southboston}\}$$

Is Concatenation Closed?

THEOREM

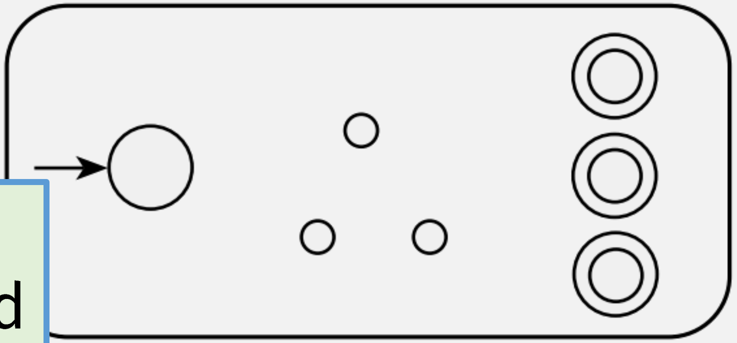
The class of regular languages is closed under the concatenation operation.

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

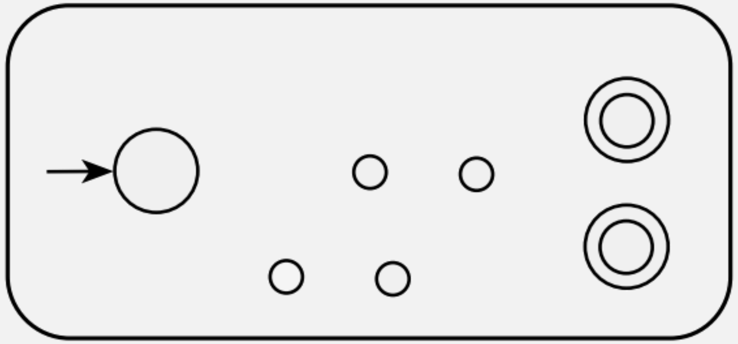
- Construct: ***new* machine M recognizing $A_1 \circ A_2$?** (like union)
  - Using: **DFA M_1** (which recognizes A_1),
  - and **DFA M_2** (which recognizes A_2)

Concatentation

M_1



M_2



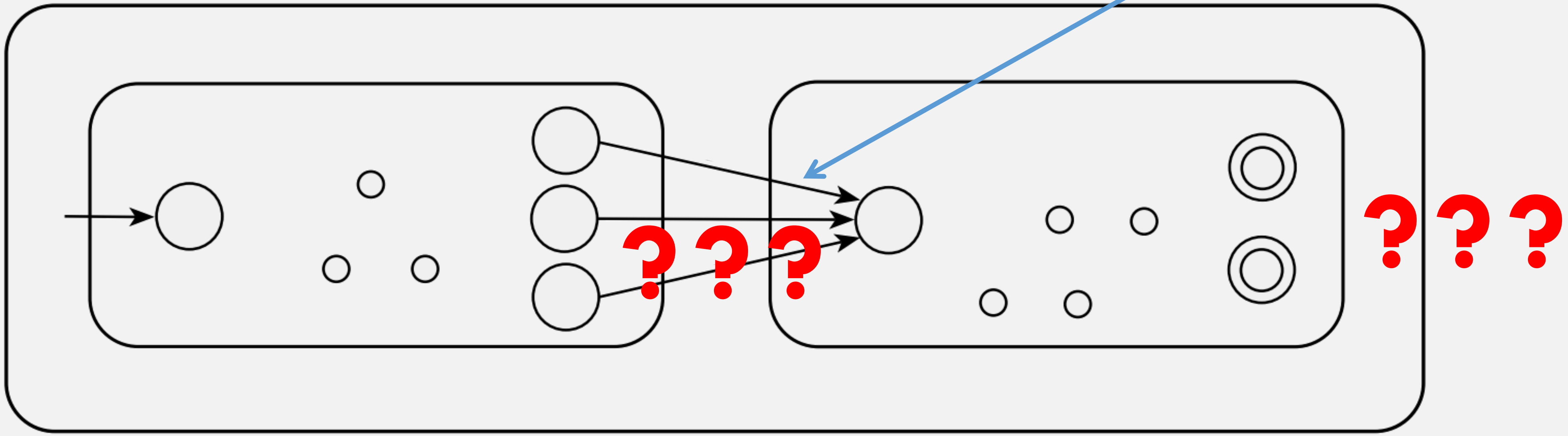
PROBLEM:
Can only read input once!
(can't backtrack)

Let M_1 recognize A_1, and M_2 recognize A_2.

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

Need to switch machines at some point, but when?

M



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

Overlapping Concatenation Example

- Let M_1 recognize language A = { jen, jens }
- and M_2 recognize language B = { smith }
- Want: Construct M to recognize $A \circ B$ = { jensmith, jenssmith }

- If M sees jen…
- M must decide to either:

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

Overlapping Concatenation Example

- Let M_1 recognize language A = { `jen`, `jens` }
- and M_2 recognize language B = { `smith` }
- Want: Construct M to recognize $A \circ B$ = { `jensmith`, `jenssmith` }

- If M sees `jen`...
- M must decide to either:
  - stay in M_1 (correct, if full input is `jenssmith`)

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

Overlapping Concatenation Example

- Let M_1 recognize language A = { `jen`, `jens` }
- and M_2 recognize language B = { `smith` }
- Want: **Construct M to recognize $A \circ B$** = { `jensmith`, `jenssmith` }

- If M sees `jen`...
- M must decide to <u>either</u>:
  - <u>stay</u> in M_1 (correct, if full input is `jenssmith`)
  - or <u>switch</u> to M_2 (correct, if full input is `jensmith`)
- **To recognize $A \circ B$, it needs to handle <u>both cases</u>!!**
  - (Without backtracking)

A **DFA** can't (easily) **do this!**

Is Concatenation Closed?

FALSE?

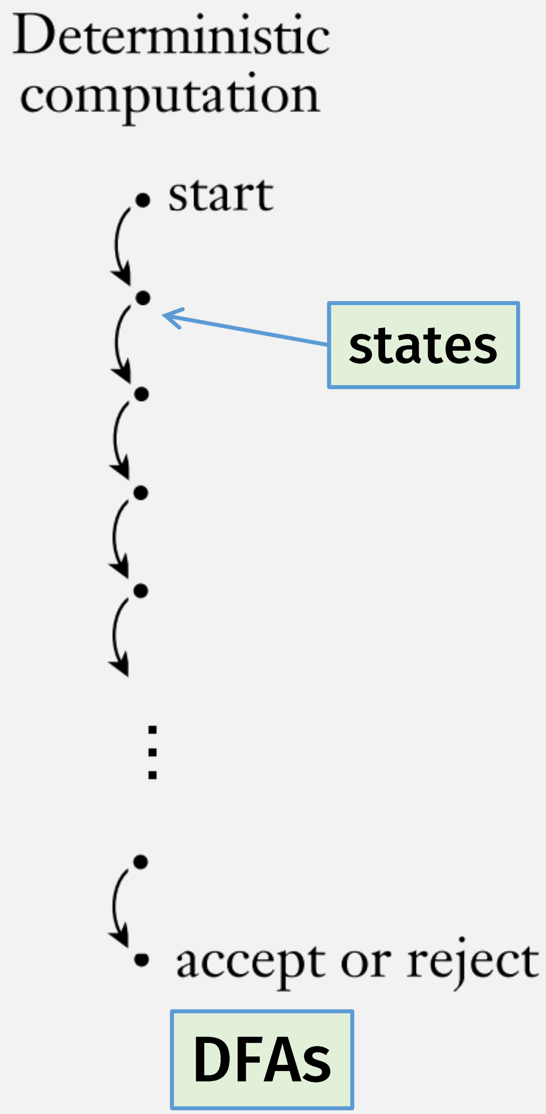
THEOREM

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

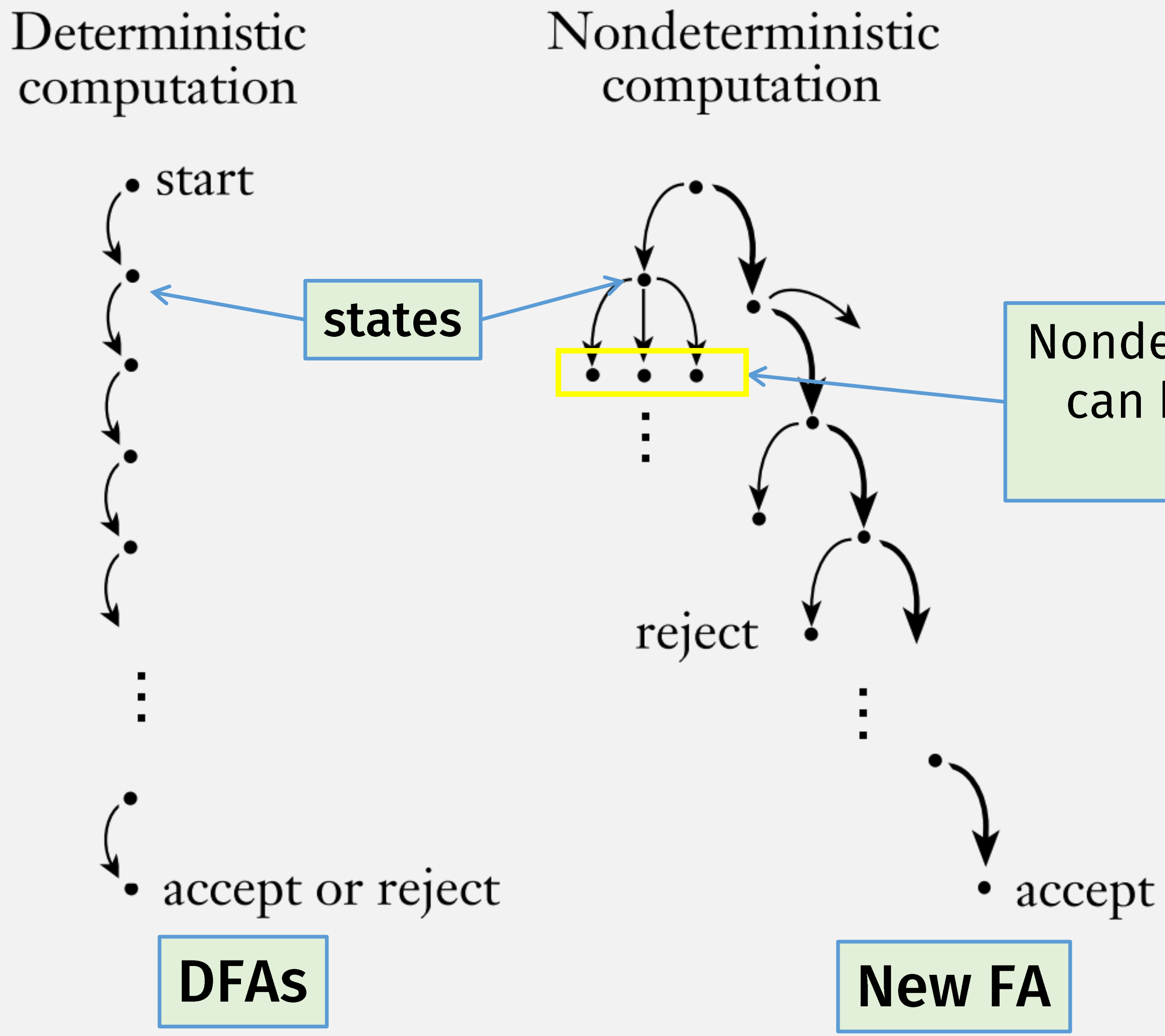
In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

- Cannot **combine A_1 and A_2's machine because:**
  - **Need to switch from A_1 to A_2 at some point …**
  - **… but we don't know when!** (we can only read input once)
- What if: **we create a new kind of machine!**
- But **does this mean concatenation is not closed for regular langs?**

Deterministic vs Nondeterministic



Deterministic vs Nondeterministic



Nondeterministic computation can be in multiple states at the same time

Previously

DFAs: The Formal Definition

DEFINITION

deterministic

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. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

Deterministic Finite Automata (DFA)

Nondeterministic Finite Automata (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. $\delta \colon Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

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. $\delta \colon Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

Difference

Power set, i.e. a transition results in set of states

Power Sets

- A **power set** is the set of all subsets of a set
- Example: S = {**a**, **b**, **c**}
- Power set of S =
  - { { }, {**a**}, {**b**}, {**c**}, {**a**, **b**}, {**a**, **c**}, {**b**, **c**}, {**a**, **b**, **c**} }
  - Note: includes the empty set!

Nondeterministic Finite Automata (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. $\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

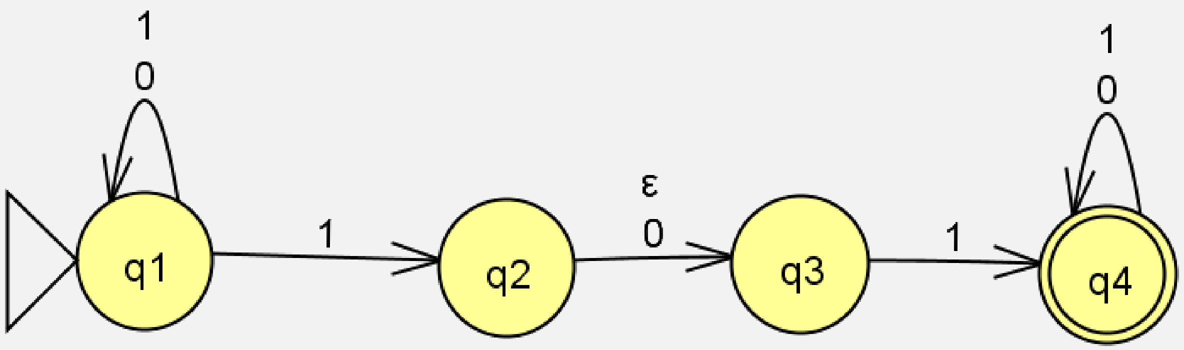
Transition label can be "empty", i.e., machine can transition without reading input

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

CAREFUL:
ε symbol is reused here, as a transition label.
- It's not the empty string!
- And it's (still) not a character in the alphabet Σ!

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

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

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

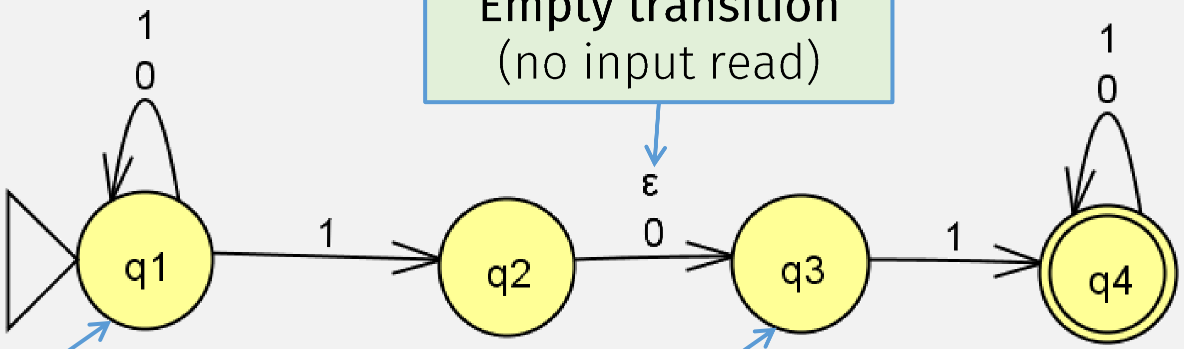
Empty transition (no input read)

	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

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

Empty transition (no input read)

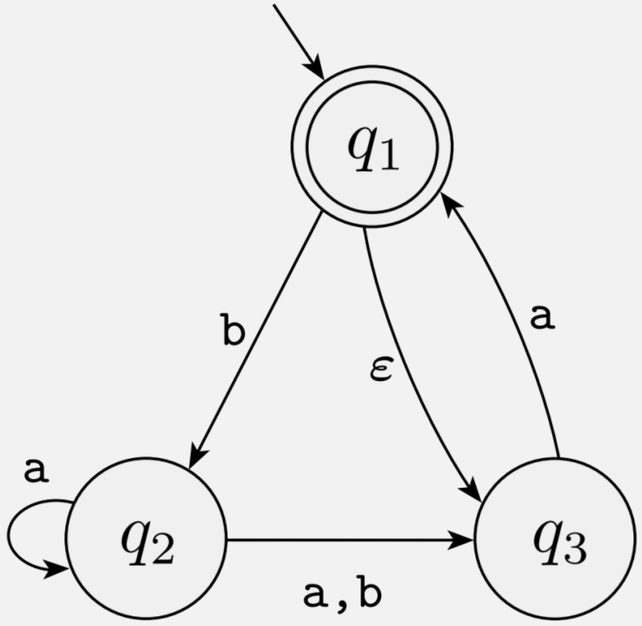


Multiple 1 transitions

No 0 transition

In-class Exercise

- Come up with a formal description for the following NFA
  - $\Sigma = \{ \texttt{a}, \texttt{b} \}$



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. $\delta \colon Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

In-class Exercise Solution

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

- $Q = \{ q_1, q_2, q_3 \}$
- $\Sigma = \{ \texttt{a}, \texttt{b} \}$
- δ ...
- $q_0 = q_1$
- $F = \{ q_1 \}$

$\delta(q_1, \texttt{a}) = \{ \}$

$\delta(q_1, \texttt{b}) = \{ q_2 \}$

$\delta(q_1, \varepsilon) = \{ q_3 \}$

$\delta(q_2, \texttt{a}) = \{ q_2, q_3 \}$

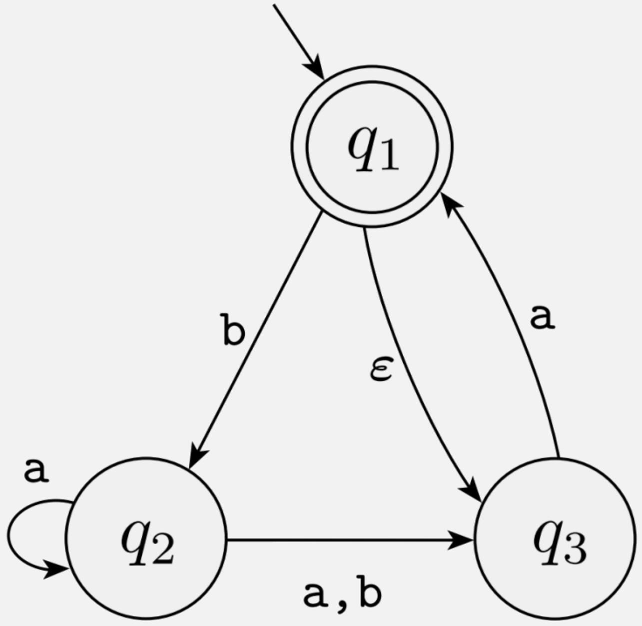
$\delta(q_2, \texttt{b}) = \{ q_3 \}$

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

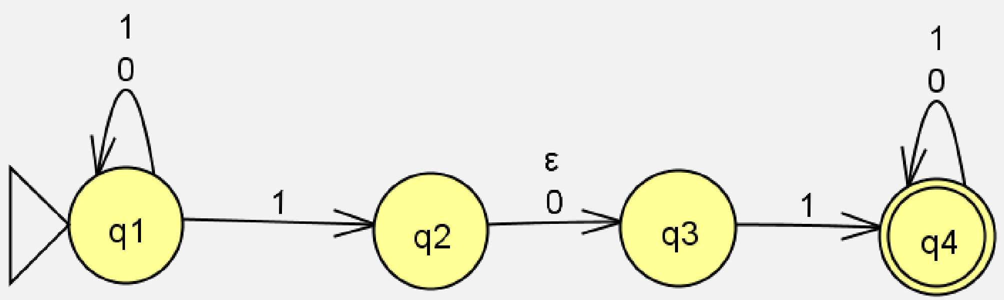
$\delta(q_3, \texttt{a}) = \{ q_1 \}$

$\delta(q_3, \texttt{b}) = \{ \}$

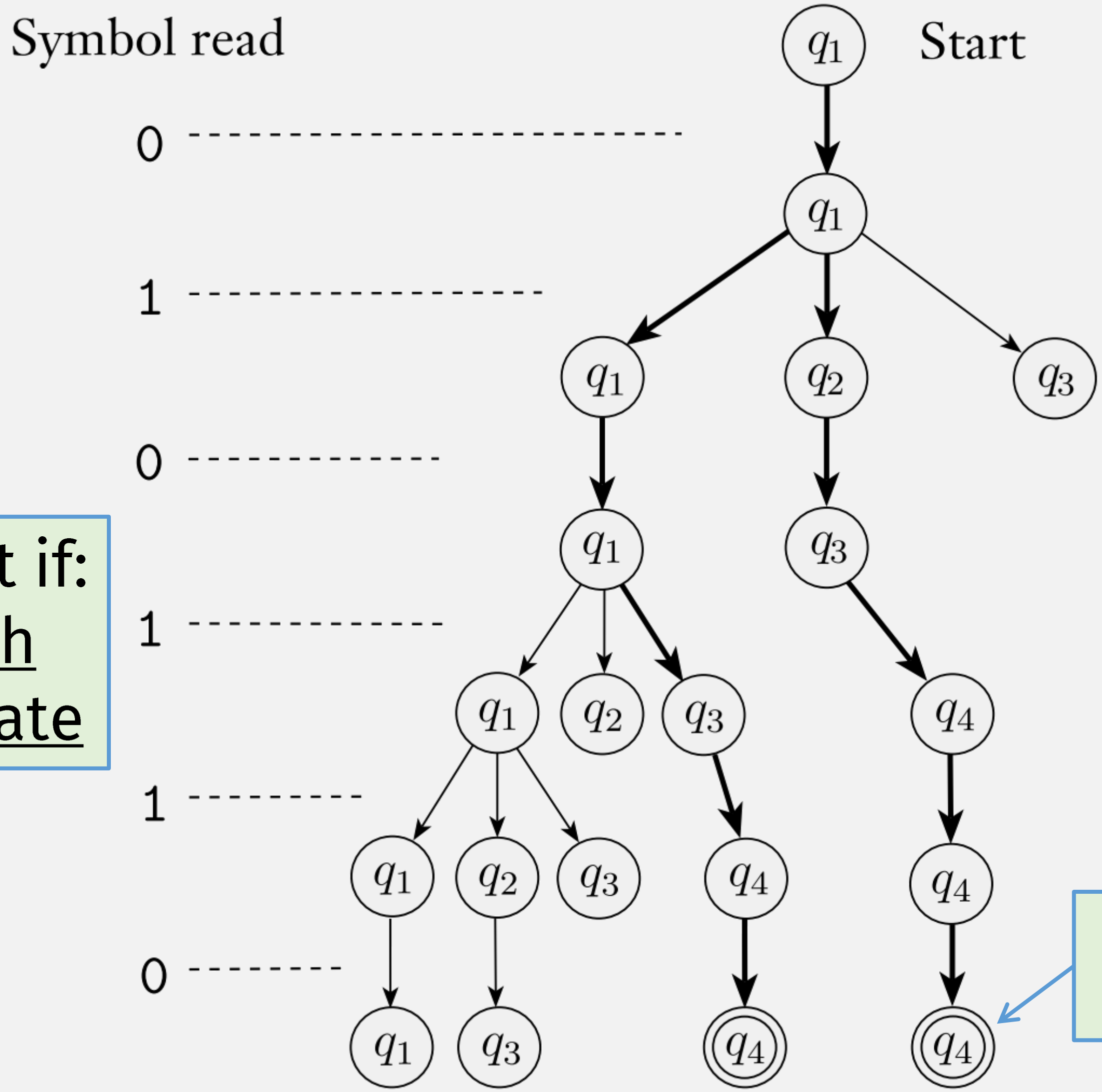
$\delta(q_3, \varepsilon) = \{ \}$



NFA Computation (JFLAP demo): `010110`



NFA Computation Sequence



NFA **accepts** input if: at least <u>one path</u> <u>ends in accept state</u>

Each step can branch into <u>multiple states at the same time</u>!

So this is an **accepting computation**

Submit in-class work 2/16

On gradescope