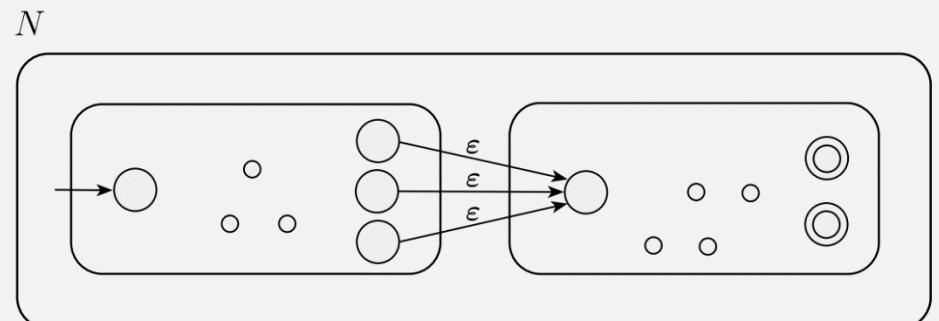


CS420

Combining Automata & Closed Operations

Monday, February 6, 2023

UMass Boston Computer Science



Announcements

- HW 1
 - Due Tue 2/7 11:59pm EST

Quiz Preview

- To prove the statement:
 - “The class of regular languages is **closed** under the union operation”
 - What is the equivalent IF-THEN statement to prove?

Last Time: Proving a Language is Regular

Statements

1. If an FSM recognizes L , then L is a regular language

2. $M = \xrightarrow{\quad} q_{\text{even}} \xrightarrow{1} q_{\text{odd}} \xrightarrow{0} q_{\text{even}}$ is an FSM

3. M recognizes L

Key is creating
this FSM!

4. $L = \{ w \mid w \text{ is string with odd } \# \text{ of } 1s \}$ is a regular language

Justifications

1. Def. of a Regular Language
2. Definition of an FSM
3. See examples. This isn't a proof, but good enough for programmers(?), and CS 420

Last Time: Tips on Designing Finite Automata

Analogy

Finite Automata ~ “Programs” ::

Designing Finite Automata ~ “Programming”!

1. Confirm understanding of the problem
 - Create tests: example inputs vs expected results (accept / reject)
2. Decide information that machine “remembers”
 - These are the machine states: some are accept states; one is start state
3. Determine transitions between states
4. Test machine behaves as expected
 - Use initial examples; and create additional tests if needed

Last Time: Combining DFAs?

Password Requirements

- » Passwords must have a minimum length of ten (10) characters - but more is better!
- » Passwords **must include at least 3** different types of characters:
 - » upper-case letters (A-Z) ← DFA
 - » lower-case letters (a-z)
 - » symbols or special characters (%,&,*,\$,etc.) ← DFA
 - » numbers (0-9) ← DFA
- » Passwords cannot contain all or part of your email address ← DFA
- » Passwords cannot be re-used ← DFA

To match all requirements, combine smaller DFAs into one big DFA?

<https://www.umb.edu/it/password>

(We do this with programs all the time)

Password Checker DFAs

What if this
is not a DFA?

M_5 : AND

M_3 : OR

M_1 : Check special chars

M_2 : Check uppercase

M_4 : Check length

Want to be able to
easily combine DFAs,
i.e., composability

We want these operations:

OR : DFA \times DFA \rightarrow DFA

AND : DFA \times DFA \rightarrow DFA

To combine more than once,
operations must be **closed!**

“Closed” Operations

- Set of Natural numbers = {0, 1, 2, ...}
 - Closed under addition:
 - if x and y are Natural numbers,
 - then $z = x + y$ is a Natural number
 - Closed under multiplication?
 - yes
 - Closed under subtraction?
 - no
- Integers = {..., -2, -1, 0, 1, 2, ...}
 - Closed under addition and multiplication
 - Closed under subtraction?
 - yes
 - Closed under division?
 - no
- Rational numbers = { $x \mid x = y/z$, y and z are Integers}
 - Closed under division?
 - No?
 - Yes if $z \neq 0$

A set is closed under an operation if:
result of the operation is in the same set
as inputs to the operation

We Want “Closed” Ops For Regular Langs!

- Set of Regular Languages = $\{L_1, L_2, \dots\}$
 - Closed under ...?
 - OR (union)
 - AND (intersection)
 - ...

A set is closed under an operation if:
result of the operation is in the same set
as inputs to the operation

Why Care About Closed Ops on Reg Langs?

- Closed operations for regulars langs preserve “regularness”
- I.e., it preserves the same computation model!
- This allows “combining” smaller computation to get bigger ones:

For Example:

OR: $\text{DFA} \times \text{DFA} \rightarrow \text{DFA}$

- So this semester, we will look for operations that are **closed**!

Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

Union 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

$$A \cup B = \{\text{good}, \text{bad}, \text{boy}, \text{girl}\}$$

Is Union Closed For Regular Langs?

In this course, we are interested in closed operations for a set of languages (here the set of regular languages)

(In general, a **set** is **closed** under an operation if applying the operation to members of the set produces a result in the same set)

The **class of regular languages** is **closed** under the union operation.

Want to prove this statement

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

Or this (same) statement

Is Union Closed For Regular Langs?

THEOREM

(In general, a set is **closed** under an operation if applying the **operation** to **members of the set** produces a **result in the same set**)

The class of regular languages is **closed** under the **union operation**.

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

Or this (same) statement

A member of the set of regular languages is ...

... a regular language, which itself is a set (of strings) ...

... so the operations we're interested in are **set operations**

Want to prove this statement

Is Union Closed For Regular Langs?

THEOREM

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

Want to prove this statement

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

Or this (same) statement

Flashback: Mathematical Statements: IF-THEN

Using:

- If we know: $P \rightarrow Q$ is TRUE,
what do we know about P and Q individually?
 - Either P is FALSE (not too useful, can't prove anything about Q), or
 - If P is TRUE, then Q is TRUE (**modus ponens**)

Proving:

p	q	$p \rightarrow q$
True	True	True
True	False	False
False	True	True
False	False	True



173

Flashback: Mathematical Statements: IF-THEN

L THEOREM -----

- The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$. t Q), or
• If P is TRUE, then Q is TRUE (modus ponens)

Would have to prove there are no
regular languages (impossible)

Proving:

- To prove: $P \rightarrow Q$ is TRUE:
 - Prove P is FALSE (usually hard or impossible)
 - Assume P is TRUE, then prove Q is TRUE

p	q	$p \rightarrow q$
True	True	True
True	False	False
False	True	True
False	False	True

Is Union Closed For Regular Langs?

Statements

Do we know anything about A_1 and A_2 ?

1. A_1 and A_2 are regular languages
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2
4. • Construct DFA $M = (Q, \Sigma, \delta, q_0, F)$ (todo)
???
5. M recognizes $A_1 \cup A_2$
How to create this? Don't know what A_1 and A_2 are!
6. $A_1 \cup A_2$ is a regular language
7. The class of regular languages is closed under the union operation.

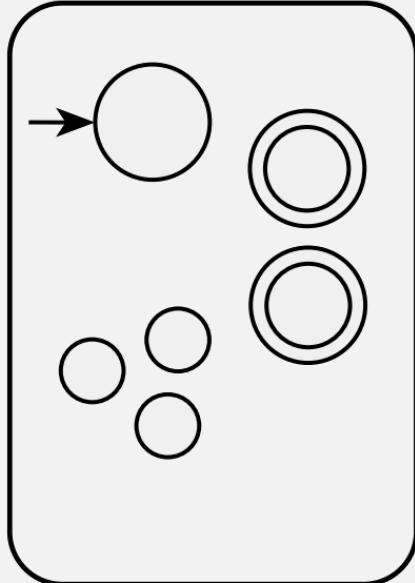
In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Justifications

1. Assumption
2. Def of Regular Language
3. Def of Regular Language
4. Def of DFA
5. See examples
6. Def of Regular Language
7. From stmt #1 and #6

M_1

recognizes A_1



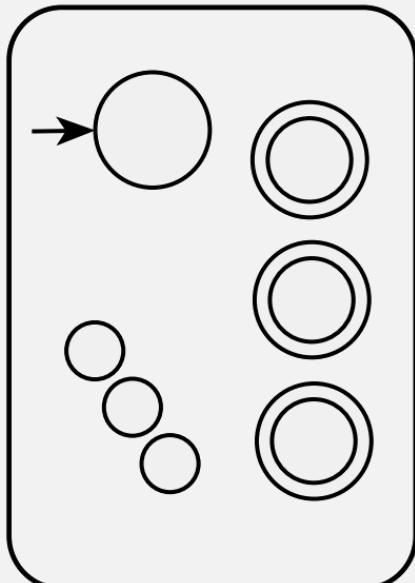
Regular language A_1
Regular language A_2

If we don't know what exactly these languages are, we still know these facts...

A language is called a *regular language* if some finite automaton recognizes it.

M_2

recognizes A_2



DEFINITION

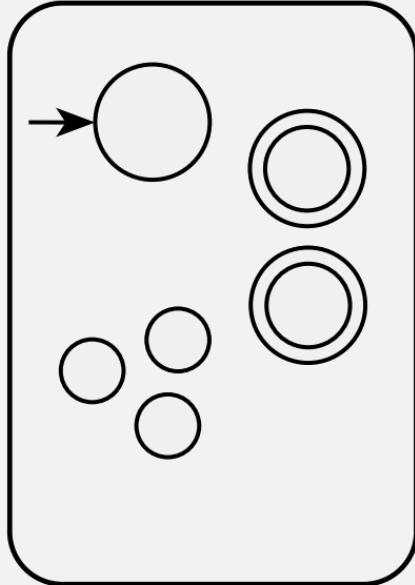
A *finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

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*.

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1 ,
 $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2 ,

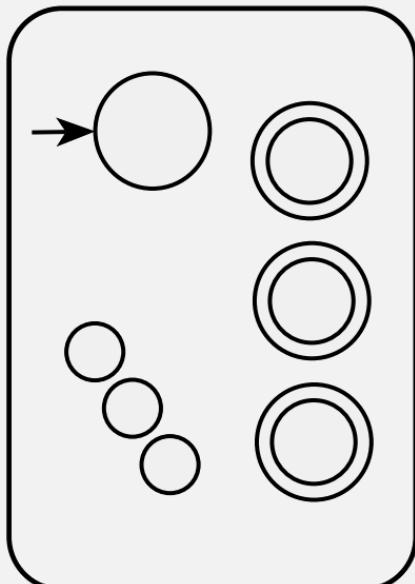
M_1

recognizes A_1



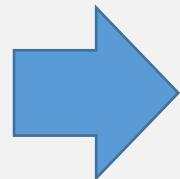
M_2

recognizes A_2



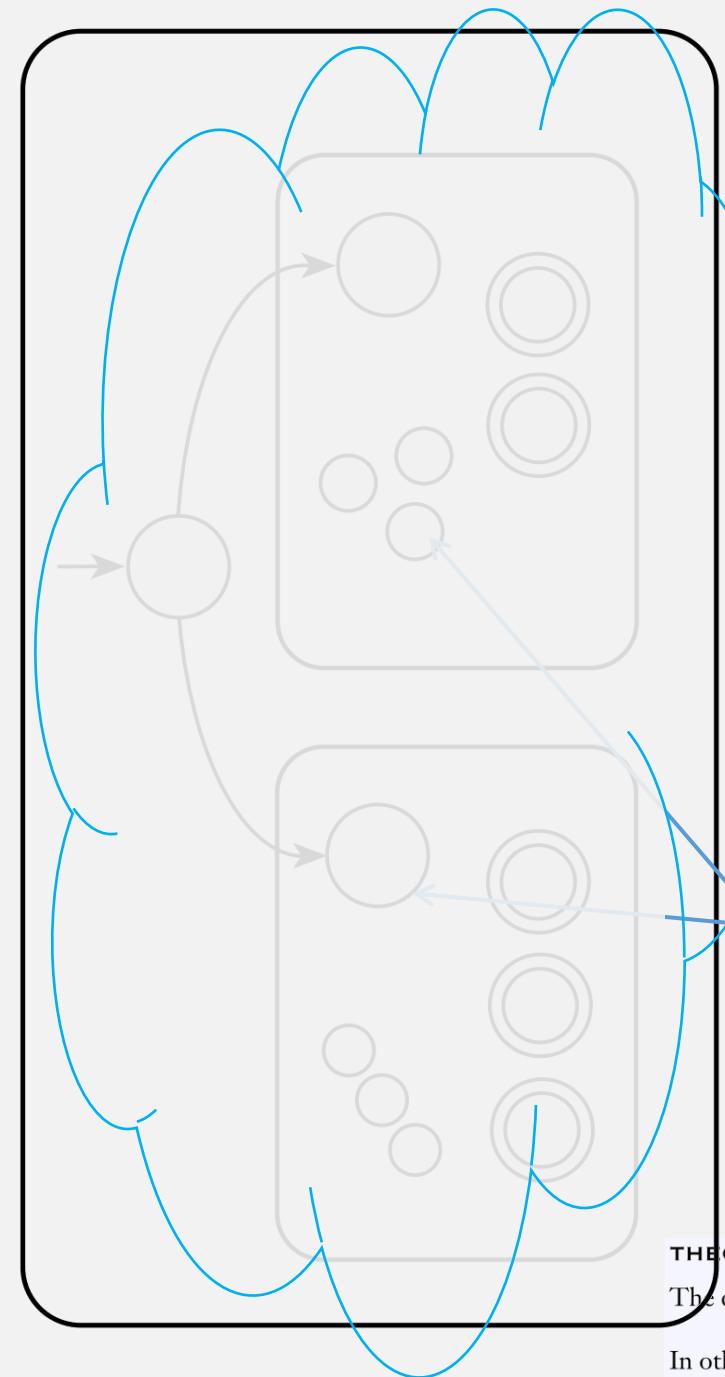
Want: M

Recognizes
 $A_1 \cup A_2$



Union

Rough sketch Idea:
 M is a combination of M_1 and M_2 that checks whether its input is accepted by either M_1 and M_2



But, a DFA can only read its input once!

Need to somehow simulate “being in” both an M_1 and M_2 state simultaneously

THEOREM

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

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

Union is Closed For Regular Languages

Proof

- Given: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1 ,
 $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2 ,
Want: M that can simultaneously
be in both an M_1 and M_2 state
- Construct: $M = (Q, \Sigma, \delta, q_0, F)$, using M_1 and M_2 , that recognizes $A_1 \cup A_2$
- states of M :
$$Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2$$

This set is the **Cartesian product** of sets Q_1 and Q_2

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**,¹
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states**.

A state of M is a **pair**:

- the **first** part is a state of M_1 and
- the **second** part is a state of M_2

So the states of M is **all possible**
combinations of the states of M_1 and M_2

Union is Closed For Regular Languages

Proof

- Given: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1 ,
 $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2 ,
- Construct: $M = (Q, \Sigma, \delta, q_0, F)$, using M_1 and M_2 , that recognizes $A_1 \cup A_2$
- states of M : $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2$
This set is the **Cartesian product** of sets Q_1 and Q_2

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

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

A step in M is includes both:

- a step in M_1 , and
- a step in M_2

Union is Closed For Regular Languages

Proof

- Given: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1 ,
 $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2 ,
- Construct: $M = (Q, \Sigma, \delta, q_0, F)$, using M_1 and M_2 , that recognizes $A_1 \cup A_2$
- states of M :
$$Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2$$
This set is the **Cartesian product** of sets Q_1 and Q_2
- M transition fn: $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
- M start state: (q_1, q_2)

Start state of M is both
start states of M_1 and M_2

Union is Closed For Regular Languages

Proof

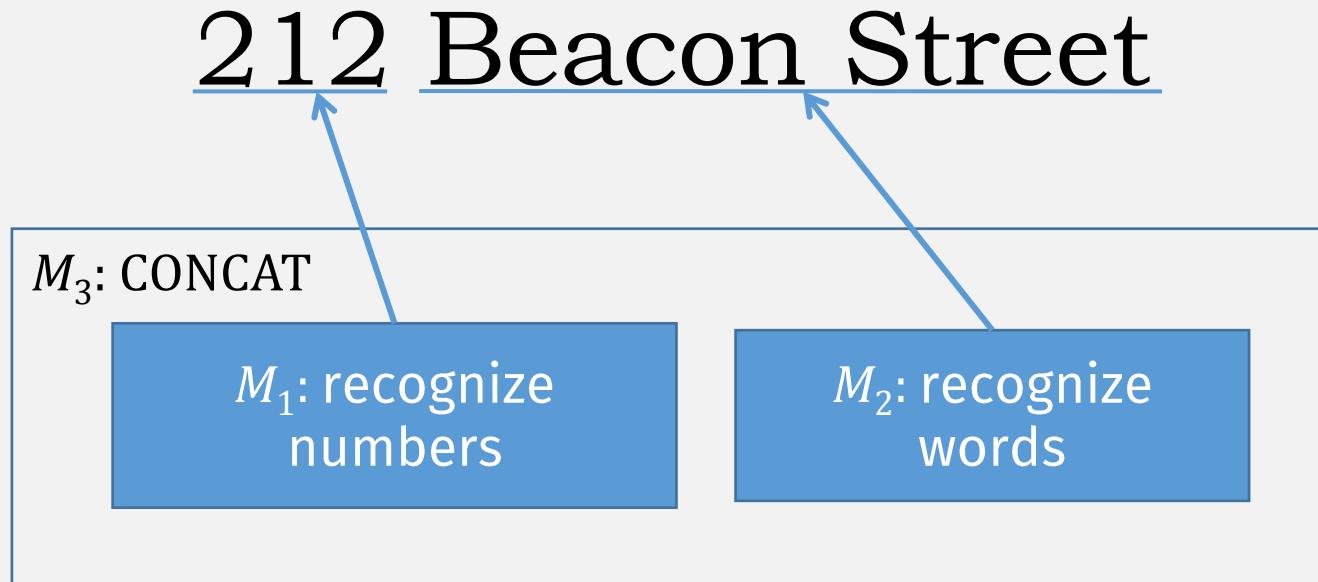
- Given: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1 ,
 $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2 ,
- Construct: $M = (Q, \Sigma, \delta, q_0, F)$, using M_1 and M_2 , that recognizes $A_1 \cup A_2$
- states of M : $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2$
This set is the **Cartesian product** of sets Q_1 and Q_2
- M transition fn: $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
- M start state: (q_1, q_2)
 - Accept if either M_1 or M_2 accept
- M accept states: $F = \{(r_1, r_2) \mid r_1 \in F_1 \boxed{\text{or}} r_2 \in F_2\}$

Remember:
Accept states must
be subset of Q

(Q.E.D.) ■

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 $\{a, b, \dots, z\}$.

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

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

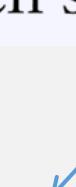
Is Concatenation Closed?

THEOREM

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

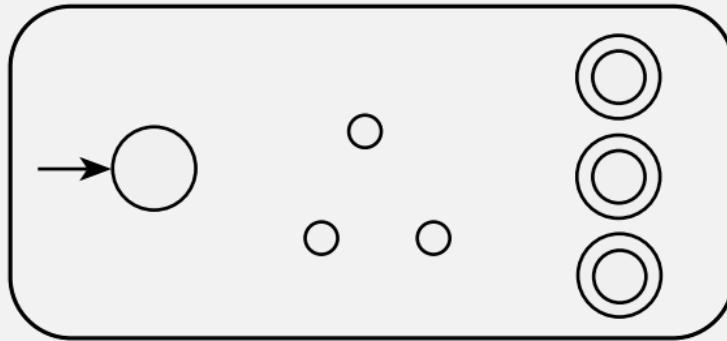
In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

- Construct a 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)

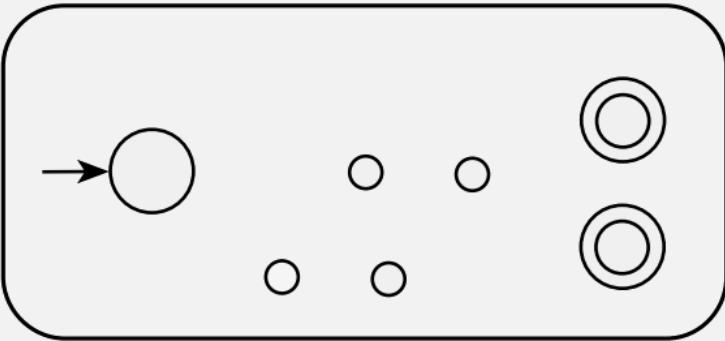


Concatenation

M_1



M_2

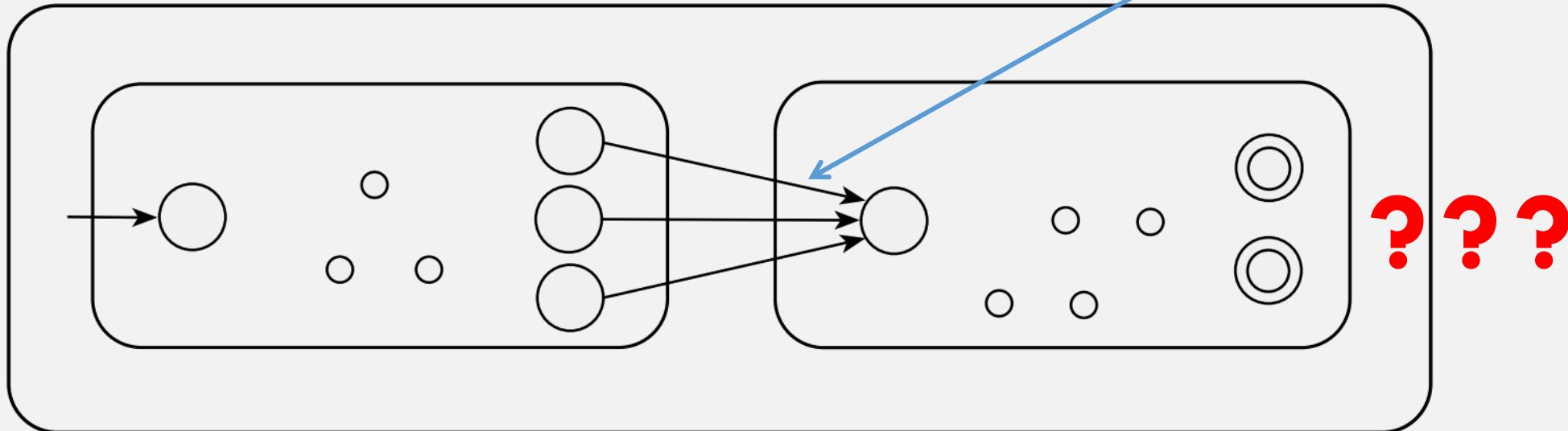


PROBLEM:
Can only
read input
once, can't
backtrack

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?



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

Overlapping Concatenation Example

- Let M_1 recognize language $A = \{ \text{jen}, \text{jens} \}$
- and M_2 recognize language $B = \{ \text{smith} \}$
- Want: Construct M to recognize $A \circ B = \{ \boxed{\text{jen}}\text{smith}, \boxed{\text{jens}}\text{smith} \}$
- If M sees **jen** ...
- M must decide to either:

Overlapping Concatenation Example

- Let M_1 recognize language $A = \{ \text{jen}, \boxed{\text{jens}} \}$
- and M_2 recognize language $B = \{ \boxed{\text{smith}} \}$
- Want: Construct M to recognize $A \circ B = \{ \text{jensmith}, \text{jenssmith} \}$
- If M sees **jen** ...
- M must decide to either:
 - stay in M_1 (correct, if full input is **jenssmith**)

Overlapping Concatenation Example

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

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

A DFA can't do this!

Is Concatenation Closed?

FALSE?

THEOREM

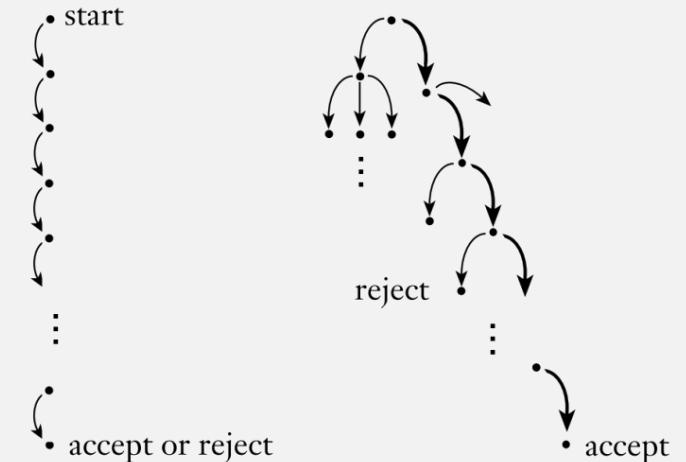
The class of regular languages is closed under the concatenation operation.

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

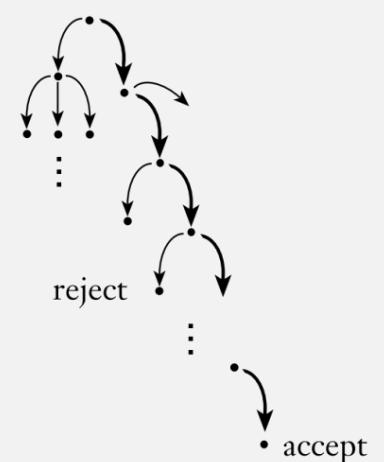
- Cannot combine A_1 and A_2 's machine because:
 - Need to switch from A_1 to A_2 at some point ...
 - ... but we don't know when! (we can only read input once)
- This requires a new kind of machine!
- But does this mean concatenation is not closed for regular langs?

Nondeterminism

Deterministic
computation

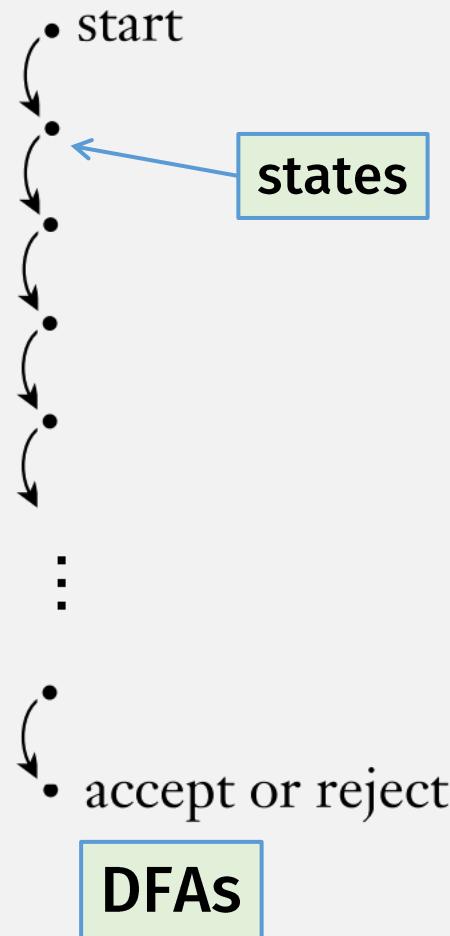


Nondeterministic
computation



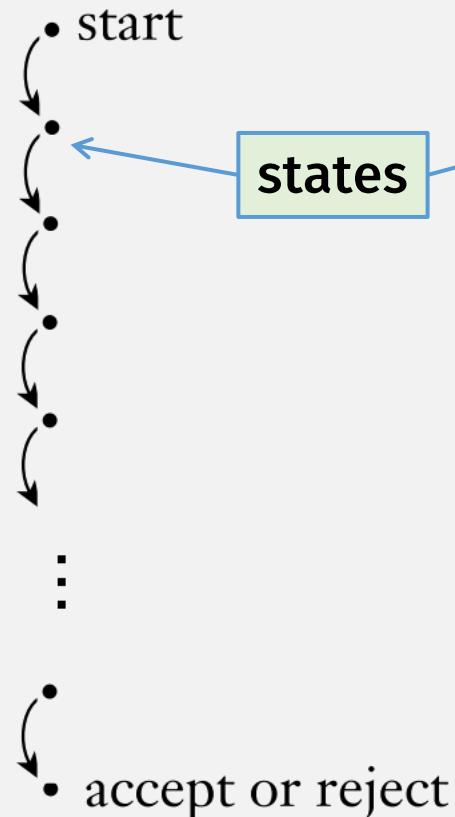
Deterministic vs Nondeterministic

Deterministic
computation

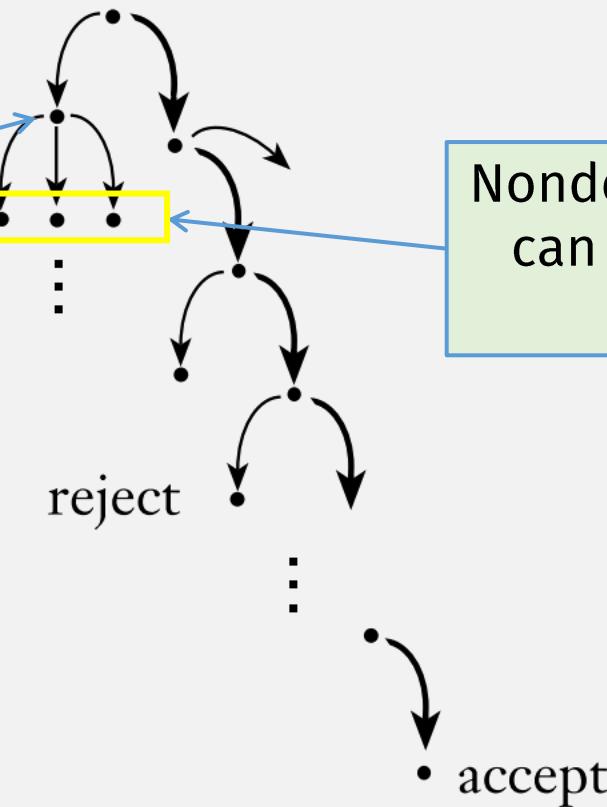


Deterministic vs Nondeterministic

Deterministic
computation



Nondeterministic
computation



Nondeterministic computation
can be in multiple states at
the same time

DFAs

New FA

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

Also called a **Deterministic Finite Automata (DFA)**

Precise Terminology is Important

- A **finite automata** or **finite state machine (FSM)** defines ...
... computation with a finite number of states
- There are many kinds of FSMs
- We've learned one kind, the **Deterministic Finite Automata (DFA)**
 - (So currently, the terms **DFA** and **FSM** refer to the same definition)
- We will learn other kinds, e.g., **Nondeterministic Finite Automata (NFA)**
- Be careful with terminology!

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

Difference

Power set, i.e. a transition results in 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**.

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

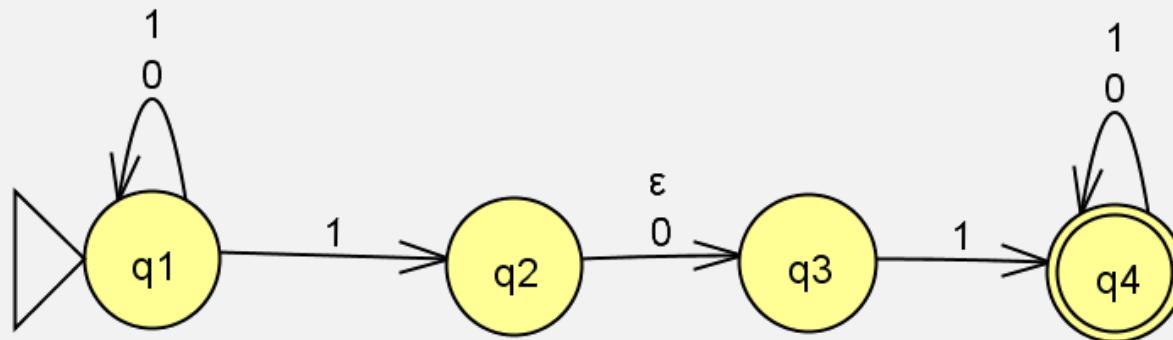
1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

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

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

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)

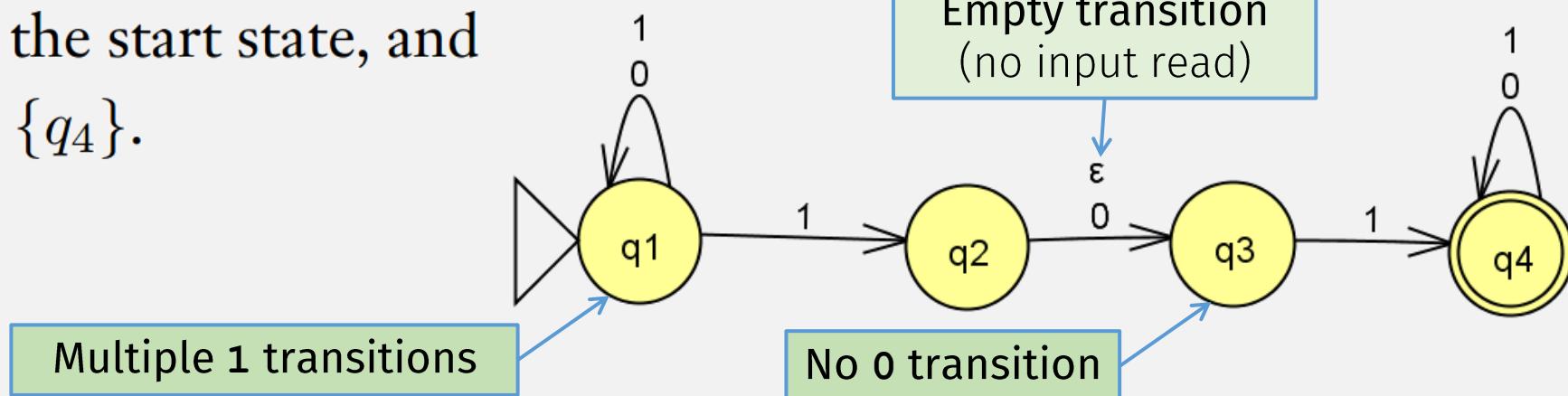
$$\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$$

	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. $F = \{q_4\}$.



In-class Exercise

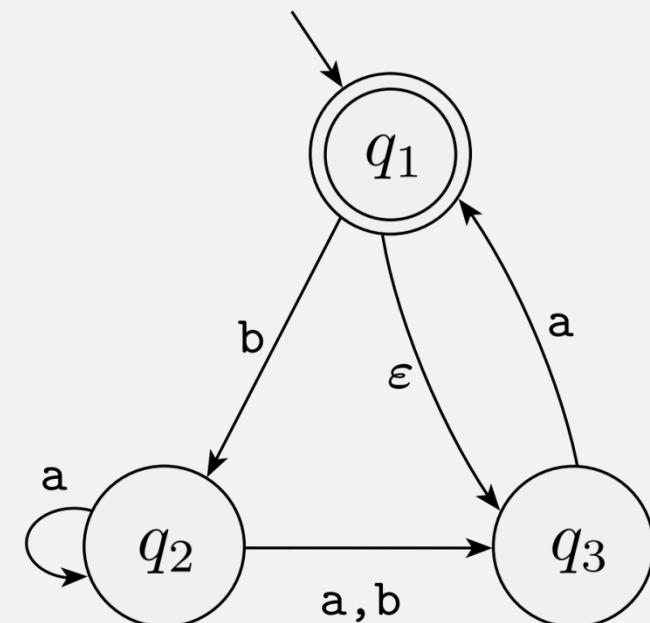
- Come up with a formal description for the following NFA
 - $\Sigma = \{ a, b \}$

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



In-class Exercise Solution

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

- $Q = \{ q_1, q_2, q_3 \}$
- $\Sigma = \{ a, b \}$

• $\delta \dots \xrightarrow{\hspace{1cm}}$

- $q_0 = q_1$
- $F = \{ q_1 \}$

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

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

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

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

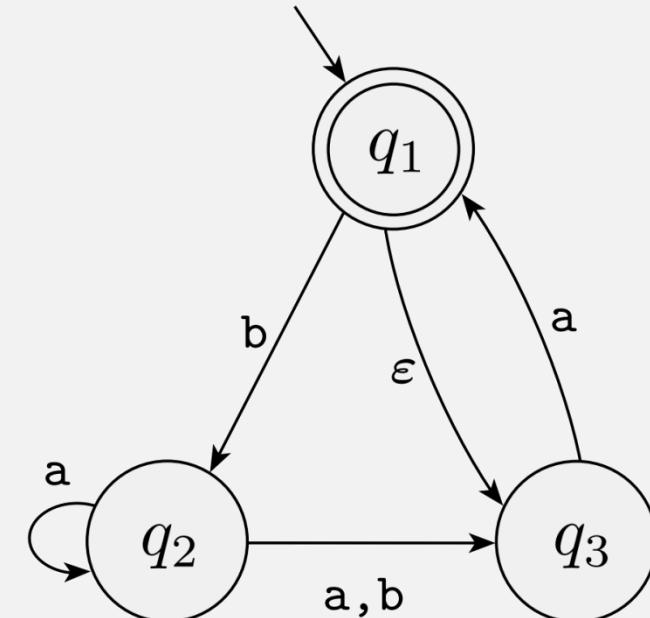
$$\delta(q_2, b) = \{ q_3 \}$$

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

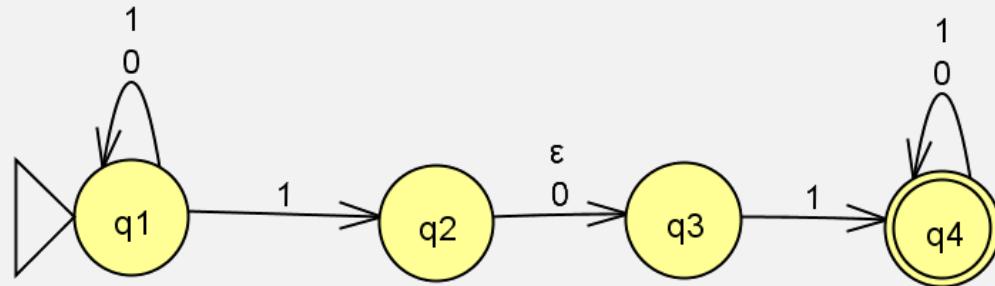
$$\delta(q_3, a) = \{ q_1 \}$$

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

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



Next Time: Running Programs, NFAs (JFLAP demo): 010110



Check-in Quiz 2/6

On gradescope