

CS 420

Nondeterminism

Wednesday, February 14, 2024

UMass Boston Computer Science



Announcements

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

Previously

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

Previously

Is Union Closed For Regular Langs?

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 $A_1 \cup A_2$.

(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)

Want to prove this statement

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

Previously

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

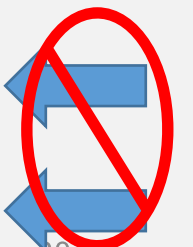
Using:

- 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:

- 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 $A_1 \cup A_2$
6. $A_1 \cup A_2$ is a regular language
7. The class of regular languages is closed under the union operation.

How to create this? Don't know what A_1 and A_2 are!

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

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

To prove $P \rightarrow Q$ is TRUE: Assume P is TRUE, then prove Q is TRUE

Wait! If A Then B \neq If B Then A

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

1. Assumption

2. Def of Regular Language

3. Def of Regular Language

If a **DFA** recognizes a language L ,
then L is a **regular language**

==

If L is a **regular language**,
then a **DFA** recognizes L ???

Equivalence of Conditional Statements

- Yes or No? “If X then Y ” is equivalent to:
 - “If Y then X ” (**converse**)
 - No!

If Regular, Then DFA?

If a **DFA** recognizes a language L , then L is a **regular language**

• Prove: If L is a **regular language**, then a **DFA** recognizes L

• Proof (Sketch)

Case analysis:

• Look at all If-then statements of the form:

• “If ... language L , then L is a **regular language**”

• (At least one is true, because we know “ L is a **regular language**”!)

• Figure out which one(s) led to conclusion:

• “ L is a **regular language**”

• (There’s only 1!) 

• So it must be that:

“Corollary”

If L is a **regular language**, then a **DFA** recognizes L

Is Union Closed For Regular Langs?

Statements

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 $A_1 \cup A_2$
6. $A_1 \cup A_2$ is a regular language
7. The class of regular languages is closed under the union operation.

How to create this? Don't know what A_1 and A_2 are!

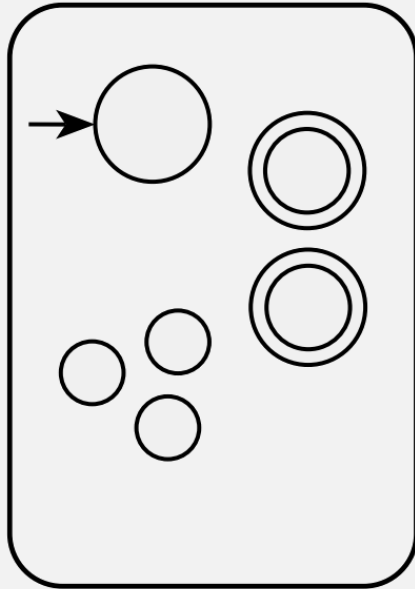
Justifications

1. Assumption "Corollary"
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

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

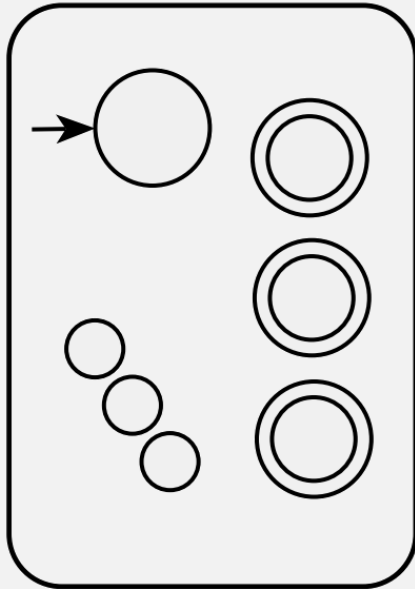
M_1

recognizes A_1



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*.

Regular language A_1
Regular language A_2

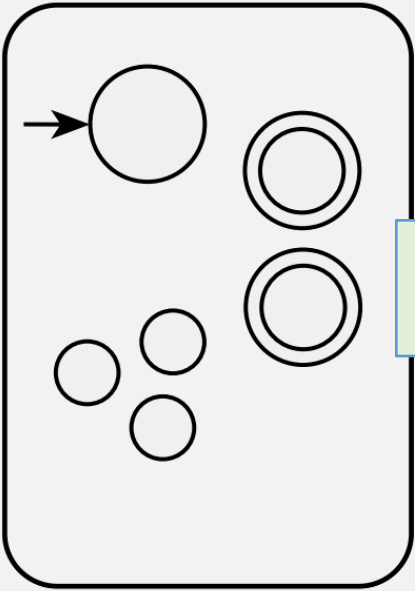
Even if we don't know what these languages are, we still know...

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

If L is a **regular language**, then a **DFA** recognizes L

Union

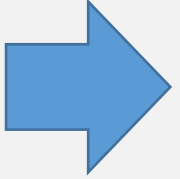
M_1
recognizes A_1



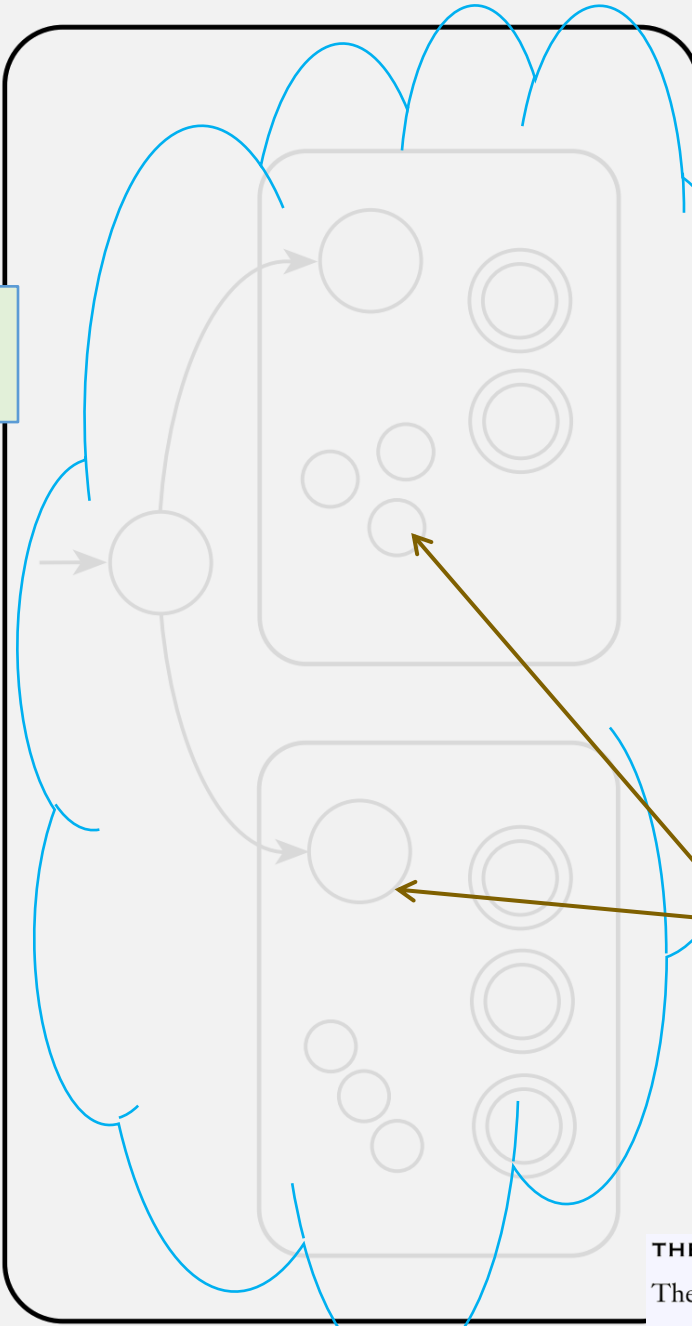
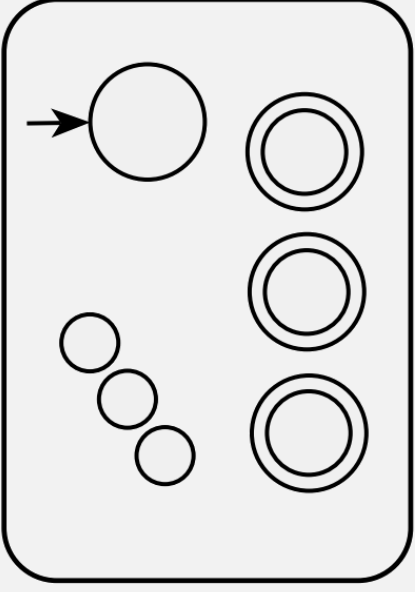
Want: M

Recognizes
 $A_1 \cup A_2$

(to prove $A_1 \cup A_2$
is regular)



M_2
recognizes A_2



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

- 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:
- first part: state of M_1
- second part: state of M_2

states of M : all possible pair combinations of states of M_1 and M_2

Union is Closed For Regular Languages

Proof (continuation)

- 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

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*.

A step in M is both:
- a step in M_1 , and
- a step in M_2

Union is Closed For Regular Languages

Proof (continuation)

- 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 (continuation)

- 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)
- M accept states: $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$.

Accept if either M_1 or M_2 accept

Remember:
Accept states must
be subset of Q

Q.E.D.?

Is Union Closed For Regular Langs?

Statements

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 $M = (Q, \Sigma, \delta, q_0, F)$**
5. M recognizes $A_1 \cup A_2$
6. $A_1 \cup A_2$ is a regular language
7. The class of regular languages is closed under the union operation.

How to create this? Don't know what A_1 and A_2 are!

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

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

“Prove” that DFA recognizes a language

Let $s_1 \in A_1$ and $s_2 \in A_2$

Let $s_3 \notin A_1$ and $s_4 \notin A_2$

Be careful when choosing examples!

In this class, a table like this is sufficient to “prove” that a DFA recognizes a language

String	In lang $A_1 \cup A_2$?	Accepted by M ?
s_1	Yes	
s_2	Yes	
s_3	???	
s_4	???	

Don't know A_1 and A_2 exactly ...

... but we know ...

... they are **sets of strings!**

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1 ,

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2 ,

constructed $M = (Q, \Sigma, \delta, q_0, F)$ recognizes $A_1 \cup A_2$?

“Prove” that DFA recognizes a language

Let $s_1 \in A_1$ and $s_2 \in A_2$

~~Let $s_3 \notin A_1$ and $s_4 \notin A_2$~~

Let $s_5 \notin A_1$ and $\notin A_2$

String	In lang $A_1 \cup A_2$?	Accepted by M ?
s_1	Yes	
s_2	Yes	
s_3	???	
s_4	???	
s_5	No	

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1 ,

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2 ,

constructed $M = (Q, \Sigma, \delta, q_0, F)$ recognizes $A_1 \cup A_2$?

Union is Closed For Regular Languages

Proof (continuation)

- 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)
- M accept states: $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$.

Accept if either M_1 or M_2 accept

“Prove” that DFA recognizes a language

Let $s_1 \in A_1$ and $s_2 \in A_2$

Let $s_5 \notin A_1$ and $\notin A_2$

String	In lang $A_1 \cup A_2$?	Accepted by M ?
s_1	Yes	Accept
s_2	Yes	Accept
s_3	???	???
s_4	???	???
s_5	No	Reject

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize A_1 ,

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize A_2 ,

constructed $M = (Q, \Sigma, \delta, q_0, F)$ Accept if either M_1 or M_2 accept

Is Union Closed For Regular Langs?

Statements

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 $M = (Q, \Sigma, \delta, q_0, F)$
5. M recognizes $A_1 \cup A_2$
6. $A_1 \cup A_2$ is a regular language
7. The class of regular languages is closed under the union operation.

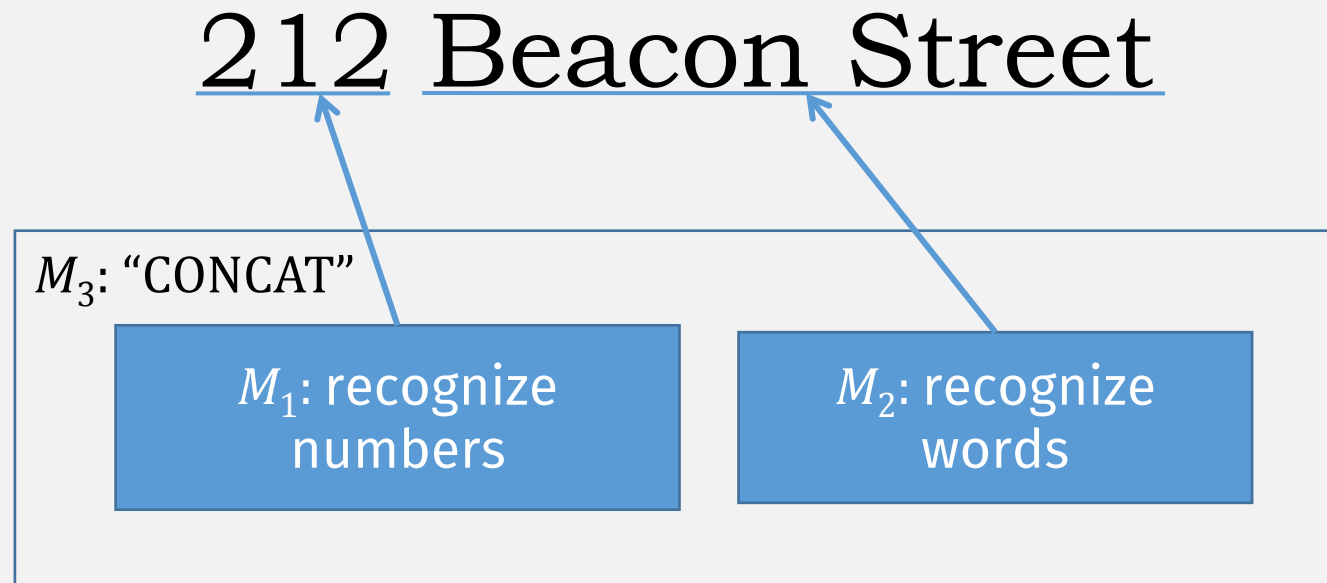
In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

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

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

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

Is Concatenation Closed?

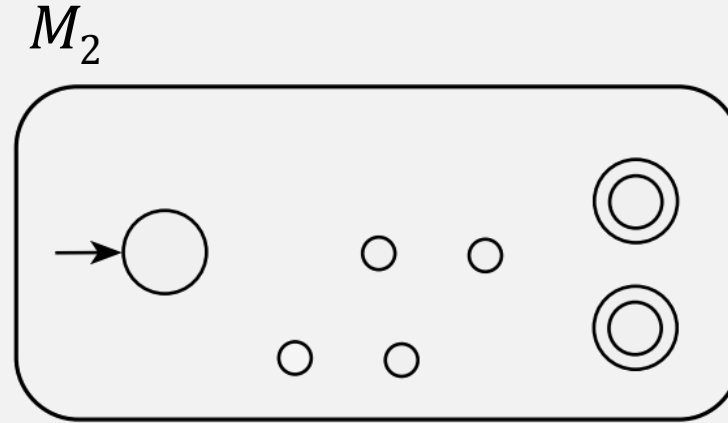
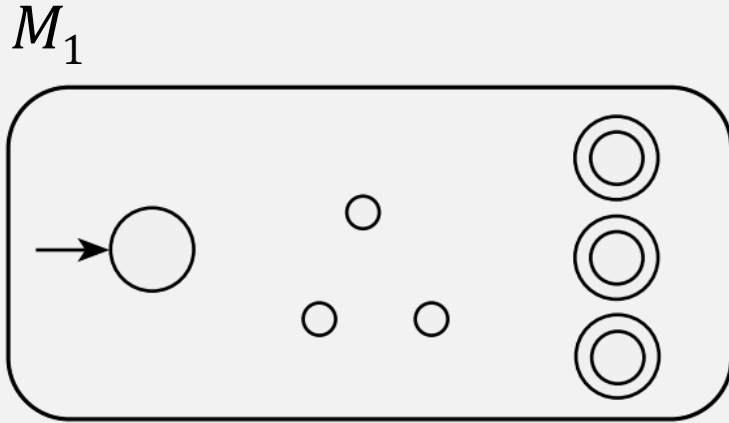
THEOREM

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 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

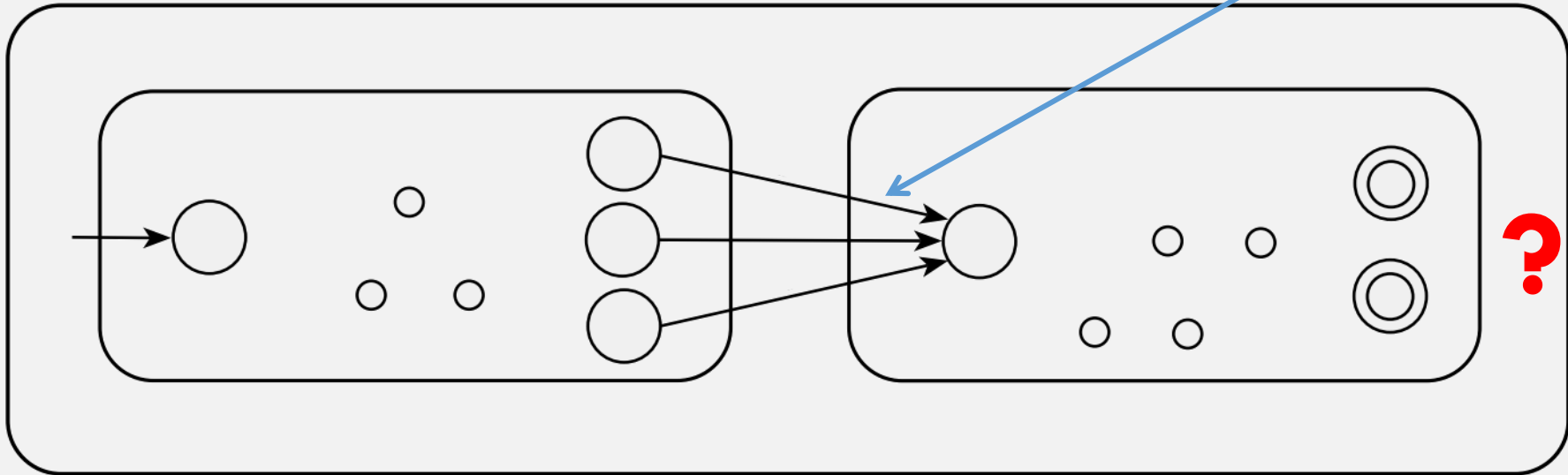


PROBLEM:
Can only read input once, can't backtrack

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 $A = \{ \text{j en, j ens} \}$
- and M_2 recognize language $B = \{ \text{smith} \}$
- Want: Construct M to recognize $A \circ B = \{ \text{j ensmith, j enssmith} \}$

- If M sees **j en** ...
- M must decide to either:

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

Overlapping Concatenation Example

- Let M_1 recognize language $A = \{ \text{j en, j ens} \}$
- and M_2 recognize language $B = \{ \text{smith} \}$
- Want: Construct M to recognize $A \circ B = \{ \text{j ensmith, j enssmith} \}$
- If M sees **j en** ...
- M must decide to either:
 - stay in M_1 (correct, if full input is **j en**←**smith**)

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

Overlapping Concatenation Example

- Let M_1 recognize language $A = \{ \text{j en, j ens} \}$
- and M_2 recognize language $B = \{ \text{smith} \}$
- Want: Construct M to recognize $A \circ B = \{ \text{j ensmith, j enssmith} \}$

- 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 **jenssmith**)

- 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.

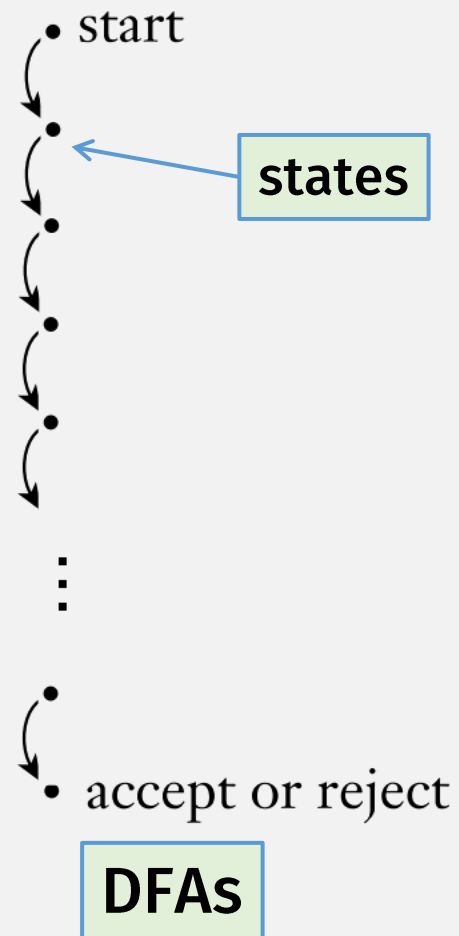
In other words, if A_1 and A_2 are regular languages then so is $A_1 \circ A_2$.

- 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 vs Nondeterministic

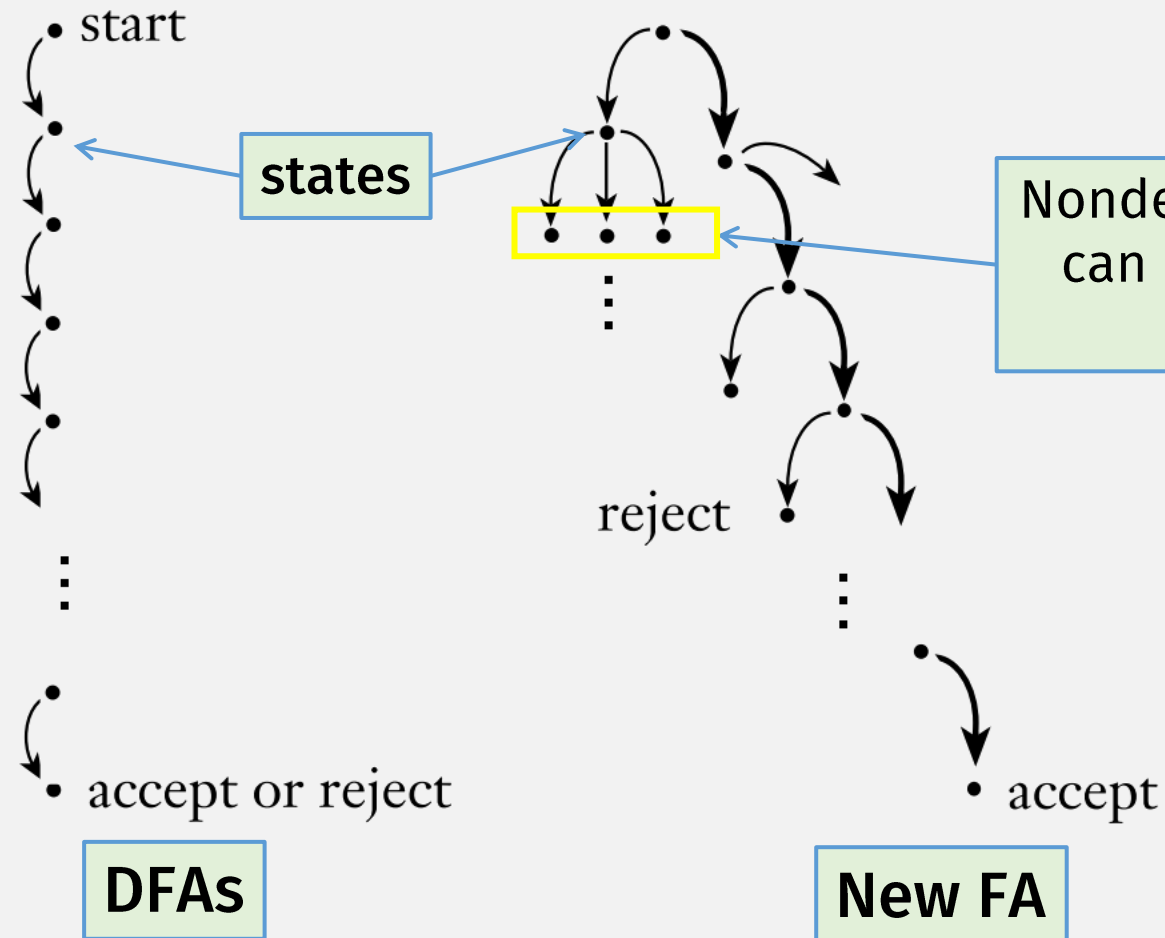
Deterministic
computation



Deterministic vs Nondeterministic

Deterministic
computation

Nondeterministic
computation



DFA: 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*.

Deterministic Finite Automata (DFA)

Nondeterministic Finite Automata (NFA)

DEFINITION

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*.

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.

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

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

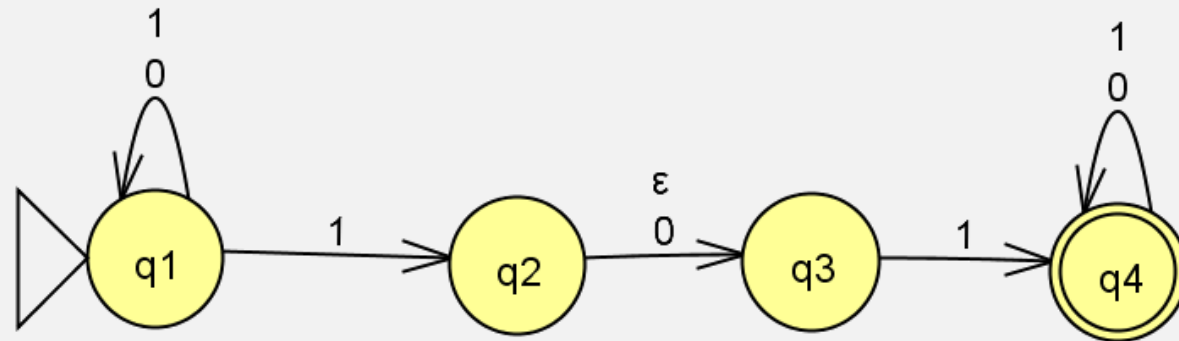
CAREFUL:

- 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 !

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

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

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

Result of transition is a set

	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

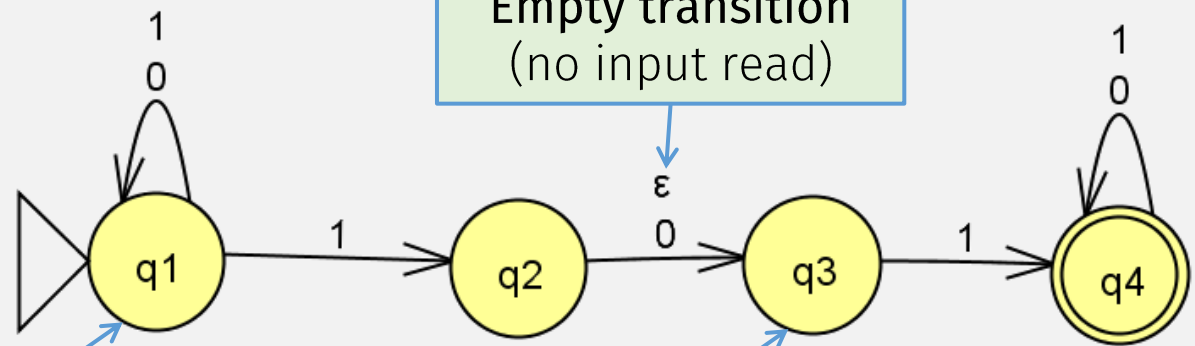
Empty transition (no input read)

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

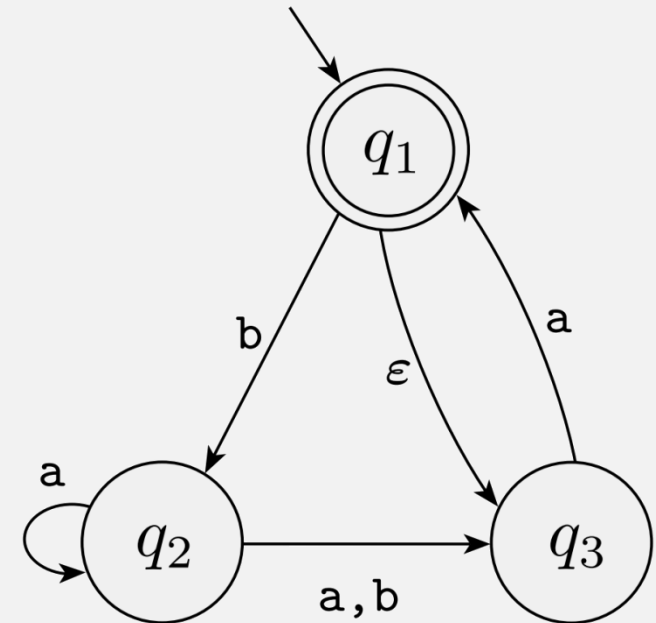
- Come up with a formal description for the following NFA
 - $\Sigma = \{ a, b \}$

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



In-class Exercise Solution

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

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

- $\Sigma = \{ a, b \}$

- $\delta \dots \longrightarrow$

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

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

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

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

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

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

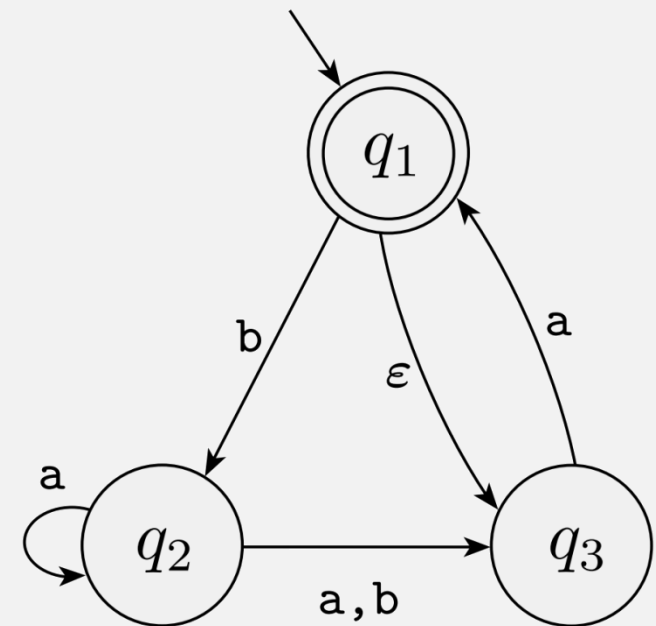
$$\delta(q_3, a) = \{ q_1 \}$$

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

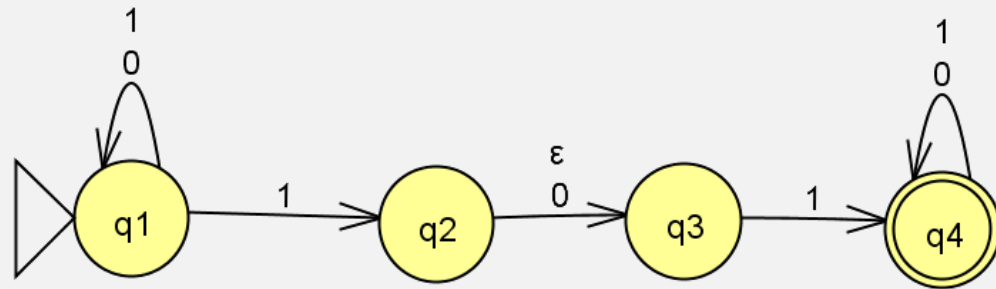
$$\delta(q_3, \varepsilon) = \{ \}$$

- $q_0 = q_1$

- $F = \{ q_1 \}$

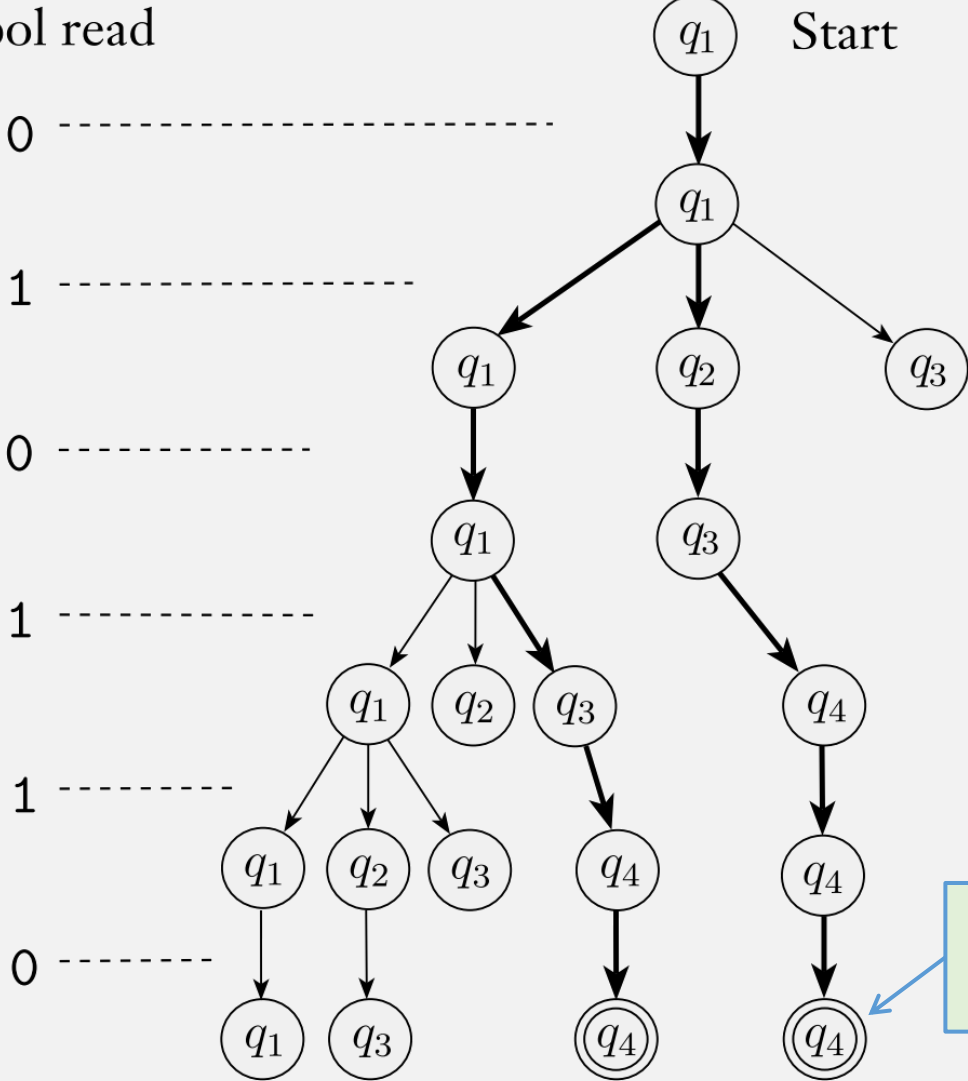


NFA Computation (JFLAP demo): **010110**



NFA Computation Sequence

Symbol read



NFA **accepts** input if at least one path ends in accept state

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

So this is an **accepting computation**

Submit in-class work 2/14

On gradescope