

# CS 383

## Lecture 03 – Nondeterministic Finite Automata (NFAs)

Stephen Checkoway

Fall 2023

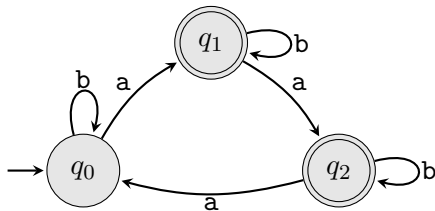
## Review from last time

DFA's are 5-tuples  $M = (Q, \Sigma, \delta, q_0, F)$

where

- $Q$  is a finite set of states
- $\Sigma$  is an alphabet (finite, nonempty set of symbols)
- $\delta : Q \times \Sigma \rightarrow Q$  is the transition function
- $q_0 \in Q$  is the start state
- $F \subseteq Q$  is the set of accepting states

A language  $A$  is regular if it is recognized by some DFA  $M$ , i.e.,  
 $A = L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$



# Operations on languages

We can define **operations on languages** which are functions that map from one or more languages to a new language

Unary operations are functions that map one language to another

- Complement:  $\overline{A} = \{w \in \Sigma^* \mid w \notin A\}$
- Reverse:  $A^{\mathcal{R}} = \{w^{\mathcal{R}} \mid w \in A\}$
- Kleene star:  $A^* = \{w_1 w_2 \cdots w_k \mid k \geq 0 \text{ and } w_i \in A \text{ for all } i\}$
- $\text{ENDSWITH}(A) = \{xw \mid x \in \Sigma^* \text{ and } w \in A\}$
- ...

# Operations on languages

We can define **operations on languages** which are functions that map from one or more languages to a new language

Unary operations are functions that map one language to another

- Complement:  $\overline{A} = \{w \in \Sigma^* \mid w \notin A\}$
- Reverse:  $A^{\mathcal{R}} = \{w^{\mathcal{R}} \mid w \in A\}$
- Kleene star:  $A^* = \{w_1 w_2 \cdots w_k \mid k \geq 0 \text{ and } w_i \in A \text{ for all } i\}$
- $\text{ENDSWITH}(A) = \{xw \mid x \in \Sigma^* \text{ and } w \in A\}$
- ...

Binary operations are functions that map a pair of languages to a new language

- Union:  $A \cup B$
- Intersection:  $A \cap B$
- Concatenation:  $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
- ...

# Complement

## Theorem

*If  $A$  is a regular language, then  $\overline{A}$  is a regular language.*

## General proof technique

- ① Start by assuming that  $A$  is a regular language
- ② Since (by assumption)  $A$  is regular, there is a DFA  $M = (Q, \Sigma, \delta, q_0, F)$  that recognizes  $A$  (i.e.,  $L(M) = A$ )
- ③ Construct a new DFA  $M' = (Q', \Sigma, \delta', q'_0, F')$  that recognizes the language we want to show is regular
- ④ Since the language is recognized by a DFA, it is regular

# Complement

## Theorem

*If  $A$  is a regular language, then  $\overline{A}$  is a regular language.*

## Proof.

- 1 Assume  $A$  is a regular language recognized by DFA  $M = (Q, \Sigma, \delta, q_0, F)$

# Complement

## Theorem

*If  $A$  is a regular language, then  $\overline{A}$  is a regular language.*

## Proof.

- ➊ Assume  $A$  is a regular language recognized by DFA  $M = (Q, \Sigma, \delta, q_0, F)$
- ➋ Construct a new DFA  $M' = (Q, \Sigma, \delta, q_0, F')$  that is identical to  $M$  except that the accepting and nonaccepting states have been swapped.  
That is,  $F' = Q \setminus F$ .

# Complement

## Theorem

*If  $A$  is a regular language, then  $\overline{A}$  is a regular language.*

## Proof.

- ① Assume  $A$  is a regular language recognized by DFA  $M = (Q, \Sigma, \delta, q_0, F)$
- ② Construct a new DFA  $M' = (Q, \Sigma, \delta, q_0, F')$  that is identical to  $M$  except that the accepting and nonaccepting states have been swapped.  
That is,  $F' = Q \setminus F$ .
- ③ If  $M$  accepts  $w$ , then when  $M$  is run on  $w$ , it ends in a state  $q \in F$ . Thus, when  $M'$  is run on  $w$ , it ends in state  $q \notin Q \setminus F = F'$  so  $M'$  rejects  $w$ .



# Complement

## Theorem

*If  $A$  is a regular language, then  $\overline{A}$  is a regular language.*

## Proof.

- 1 Assume  $A$  is a regular language recognized by DFA  $M = (Q, \Sigma, \delta, q_0, F)$
- 2 Construct a new DFA  $M' = (Q, \Sigma, \delta, q_0, F')$  that is identical to  $M$  except that the accepting and nonaccepting states have been swapped.  
That is,  $F' = Q \setminus F$ .
- 3 If  $M$  accepts  $w$ , then when  $M$  is run on  $w$ , it ends in a state  $q \in F$ . Thus, when  $M'$  is run on  $w$ , it ends in state  $q \notin Q \setminus F = F'$  so  $M'$  rejects  $w$ .
- 4 If  $M$  rejects  $w$ , then when  $M$  is run on  $w$ , it ends in state  $q \notin F$ . Thus, when  $M'$  is run on  $w$ , it ends in state  $q \in Q \setminus F = F'$  so  $M'$  accepts  $w$ .

# Complement

## Theorem

*If  $A$  is a regular language, then  $\overline{A}$  is a regular language.*

## Proof.

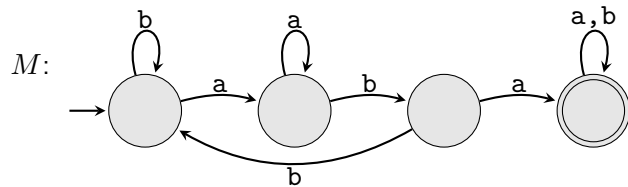
- ① Assume  $A$  is a regular language recognized by DFA  $M = (Q, \Sigma, \delta, q_0, F)$
- ② Construct a new DFA  $M' = (Q, \Sigma, \delta, q_0, F')$  that is identical to  $M$  except that the accepting and nonaccepting states have been swapped.  
That is,  $F' = Q \setminus F$ .
- ③ If  $M$  accepts  $w$ , then when  $M$  is run on  $w$ , it ends in a state  $q \in F$ . Thus, when  $M'$  is run on  $w$ , it ends in state  $q \notin Q \setminus F = F'$  so  $M'$  rejects  $w$ .
- ④ If  $M$  rejects  $w$ , then when  $M$  is run on  $w$ , it ends in state  $q \notin F$ . Thus, when  $M'$  is run on  $w$ , it ends in state  $q \in Q \setminus F = F'$  so  $M'$  accepts  $w$ .
- ⑤ Therefore,  $L(M') = \overline{A}$ . Since DFA  $M'$  recognizes  $\overline{A}$ ,  $\overline{A}$  is regular. □

## Complement example

Let  $A = \{w \mid \text{aba is a substring of } w\}$

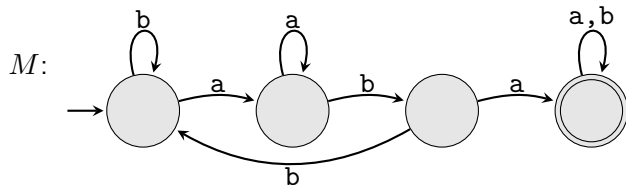
## Complement example

Let  $A = \{w \mid \text{aba is a substring of } w\}$



## Complement example

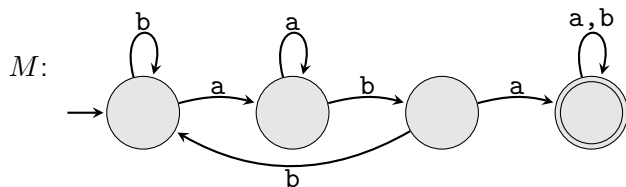
Let  $A = \{w \mid \text{aba is a substring of } w\}$



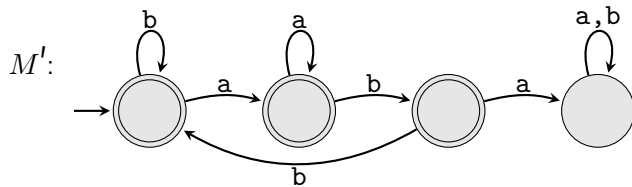
$\overline{A} = \{w \mid \text{aba is *not* a substring of } w\}$

## Complement example

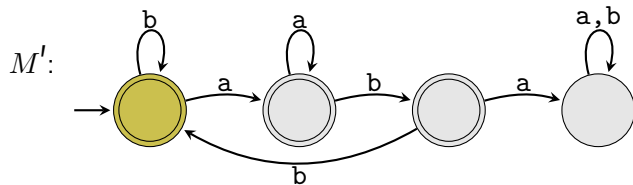
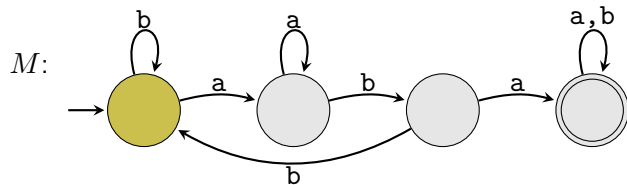
Let  $A = \{w \mid \text{aba is a substring of } w\}$



$\overline{A} = \{w \mid \text{aba is not a substring of } w\}$

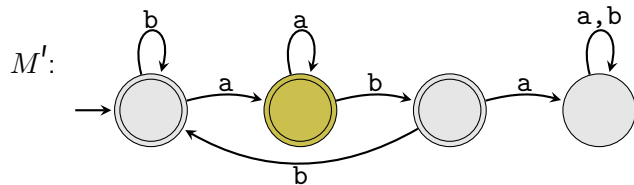
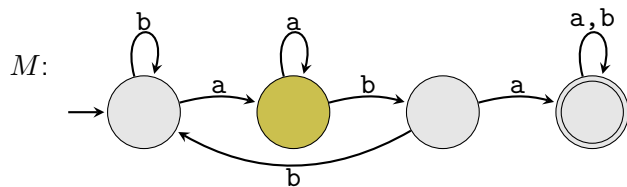


## Complement example



a b b a a b a b

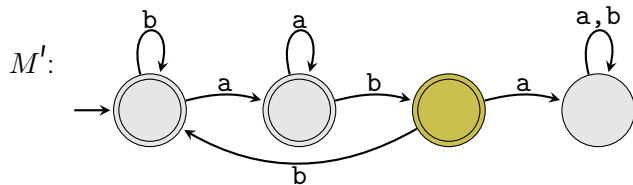
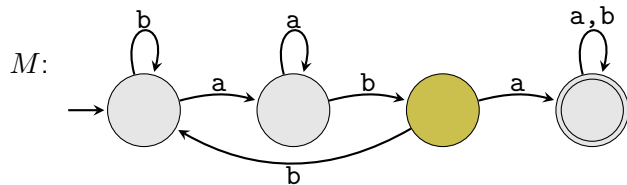
## Complement example



a**b**baabab

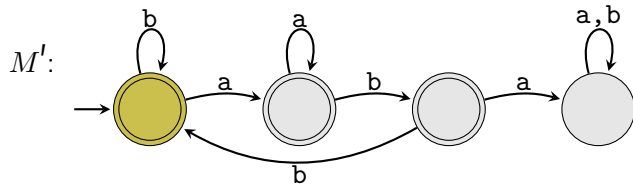
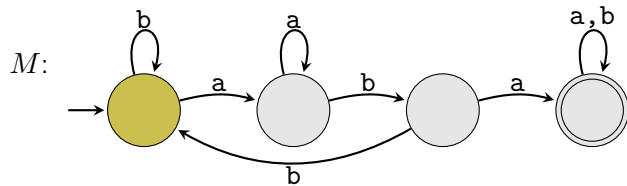


## Complement example



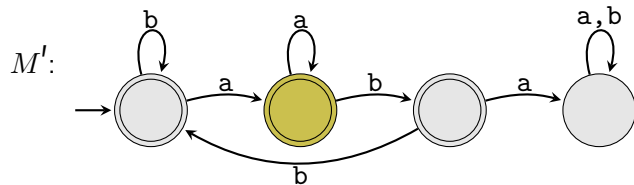
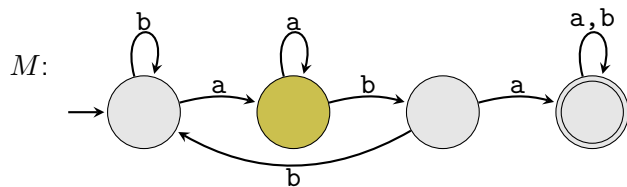
abbaabab

## Complement example



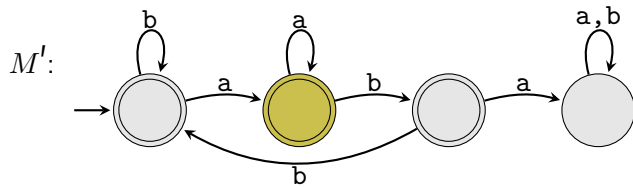
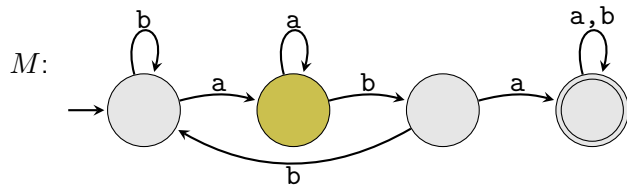
abb<sup>a</sup>abab

## Complement example



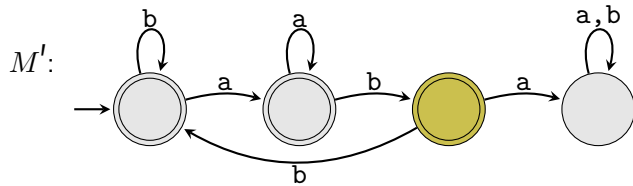
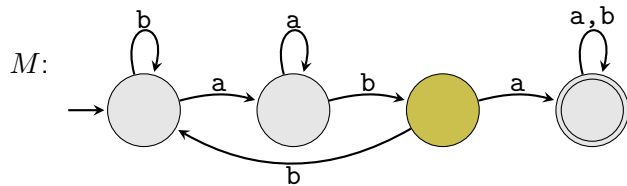
abba**a**bab

## Complement example



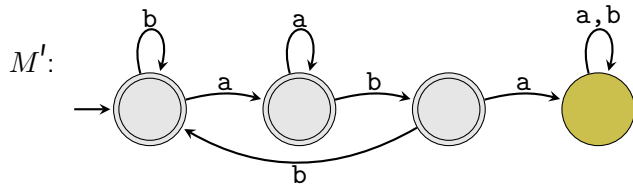
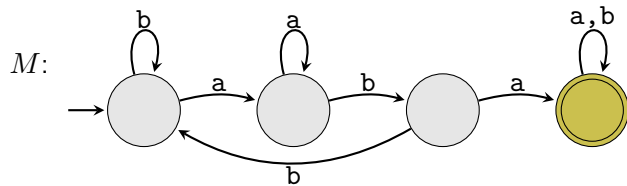
abbaa**b**ab

## Complement example



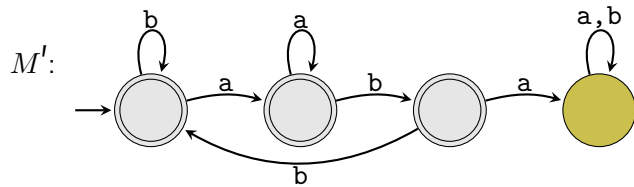
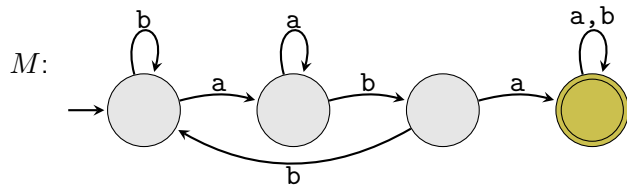
abbaab**ab**

## Complement example



abbaaba**b**

## Complement example



abbaabab

# Union

## Theorem

*If  $A$  and  $B$  are regular languages, then  $A \cup B$  is regular.*

## Proof.

- 1 Assume DFA  $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$  recognizes  $A$  and  $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$  recognizes  $B$ .



# Union

## Theorem

*If  $A$  and  $B$  are regular languages, then  $A \cup B$  is regular.*

## Proof.

- 1 Assume DFA  $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$  recognizes  $A$  and  $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$  recognizes  $B$ .
- 2 Build a new DFA  $M = (Q, \Sigma, \delta, q_0, F)$  with states consisting of pairs of states from  $M_1$  and  $M_2$ . Formally,

$$Q = Q_1 \times Q_2$$

$$q_0 = (q_1, q_2)$$

$$\delta((q, r), t) = (\delta_1(q, t), \delta_2(r, t))$$

$$F = \{(q, r) \mid q \in F_1 \text{ or } r \in F_2\}.$$

As  $M$  transitions from state  $(q, r)$  to state  $(q', r')$ , the first element changes according to  $\delta_1$  and the second according to  $\delta_2$ .

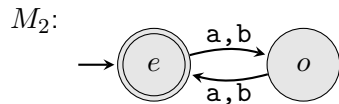
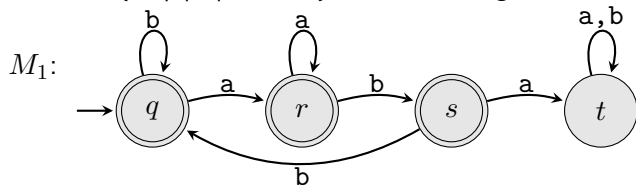
## Union

- ③ Consider running  $M_1$ ,  $M_2$ , and  $M$  on string  $w$ . The three DFAs end in states  $q$ ,  $r$ , and  $(q, r)$ , respectively. If  $w \in A$ , then  $M_1$  accepts  $w$  so  $q \in F_1$  and thus  $(q, r) \in F$  so  $M$  accepts  $w$ . Similarly, if  $w \in B$ , then  $M_2$  accepts  $w$  so  $r \in F_2$  and thus  $(q, r) \in F$ . If  $w$  is in neither  $A$  nor  $B$ , then  $q \notin F_1$  and  $r \notin F_2$  so  $(q, r) \notin F$ .
- ④ Therefore,  $L(M) = A \cup B$  so  $A \cup B$  is regular. □

## Union example

Let  $A = \{w \mid \text{aba is not a substring of } w\}$  and  $M_1$  recognize  $A$

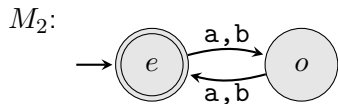
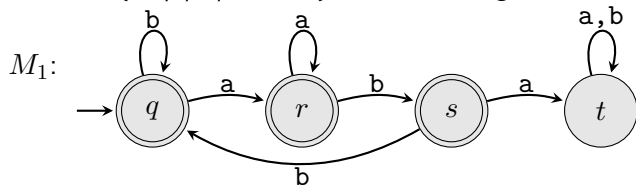
Let  $B = \{w \mid |w| \text{ is even}\}$  and  $M_2$  recognize  $B$



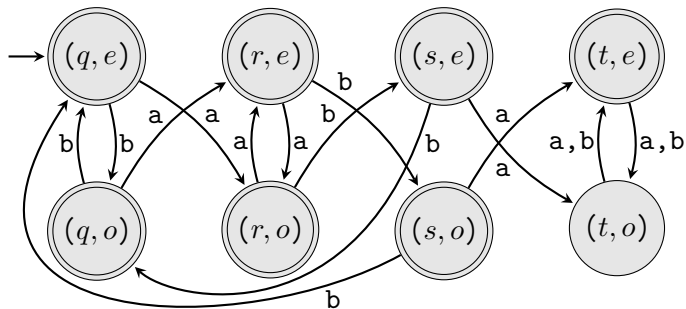
## Union example

Let  $A = \{w \mid \text{aba is not a substring of } w\}$  and  $M_1$  recognize  $A$

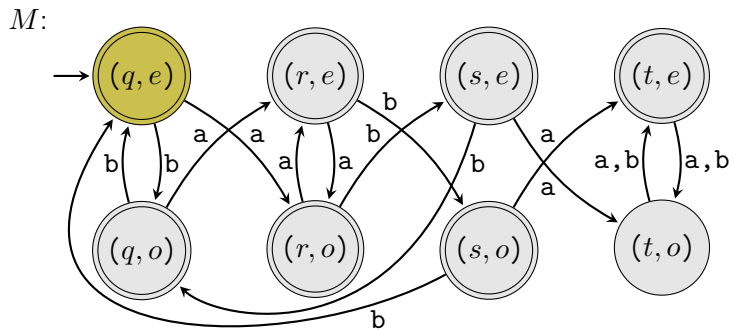
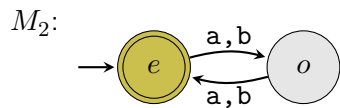
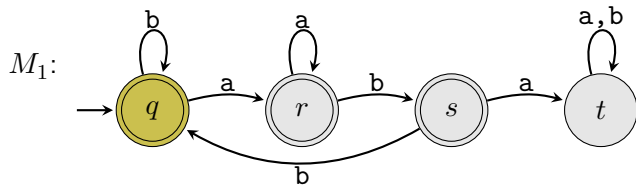
Let  $B = \{w \mid |w| \text{ is even}\}$  and  $M_2$  recognize  $B$



$M$ :

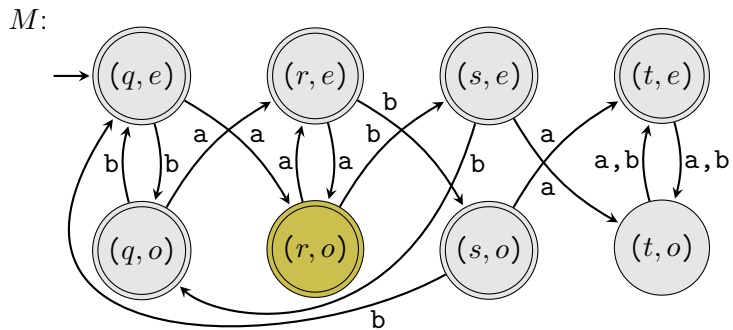
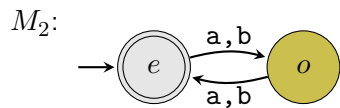
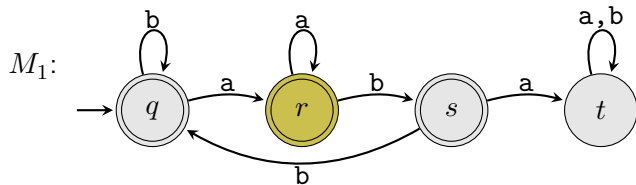


## Union example



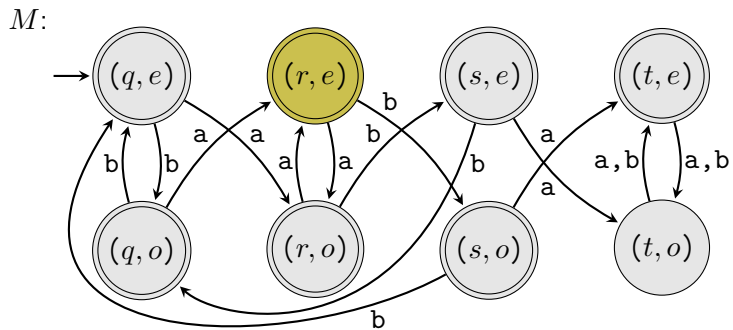
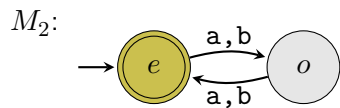
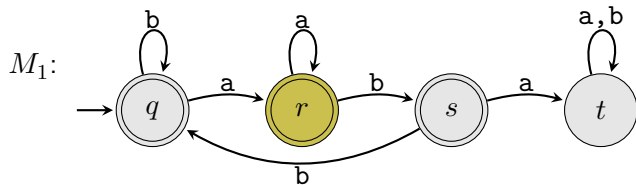
aabbaba

# Union example



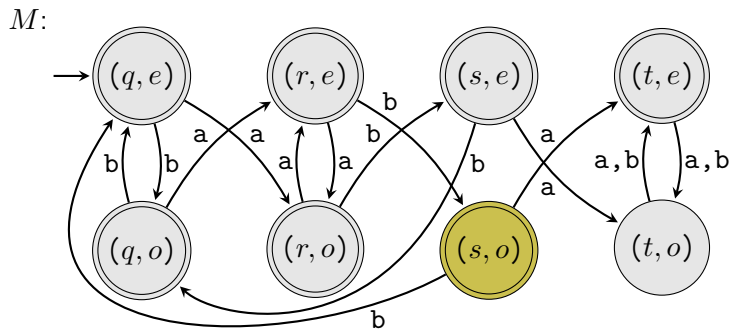
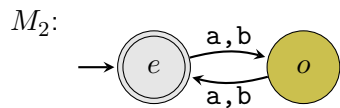
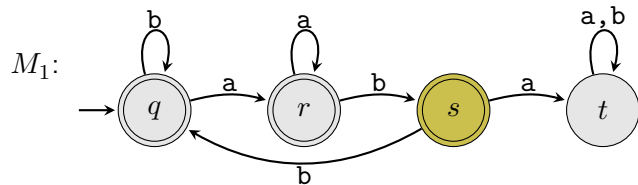
a**a**bbaba

## Union example



aa**b**ababa

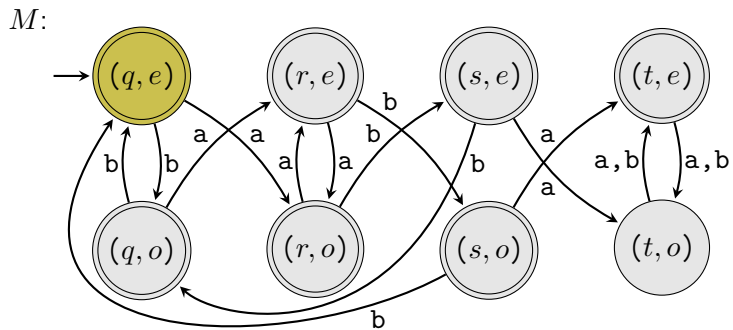
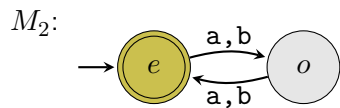
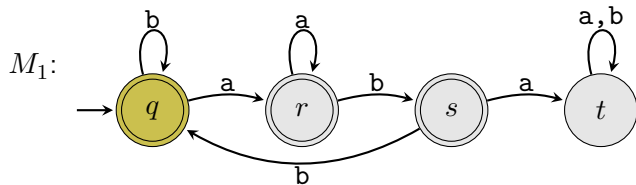
## Union example



aab**b**aba

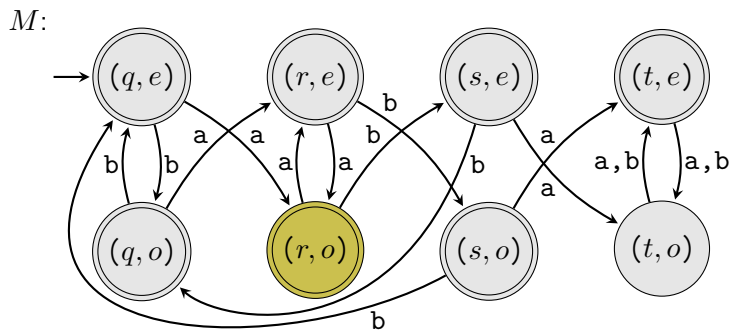
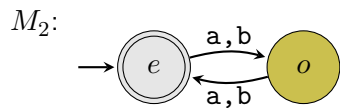
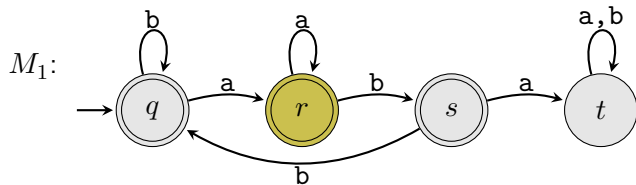


## Union example



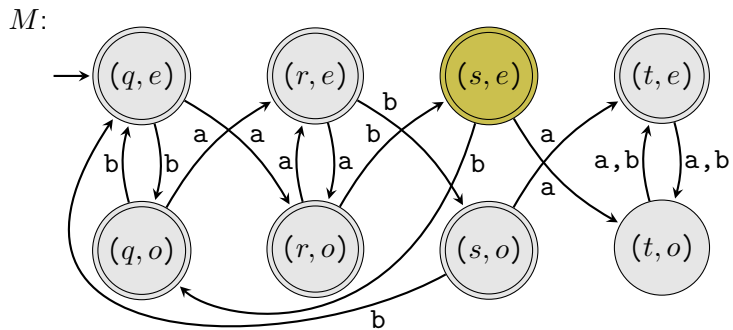
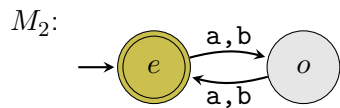
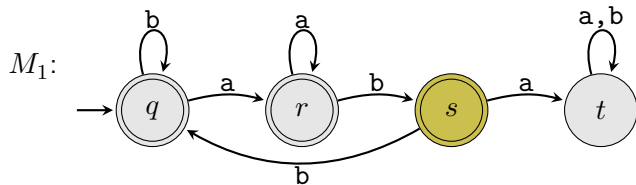
aabbaba

## Union example



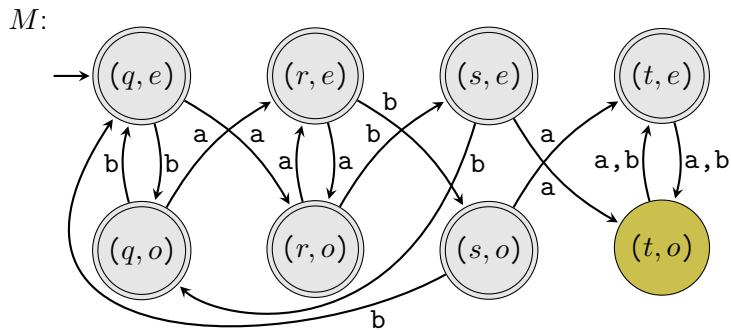
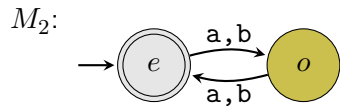
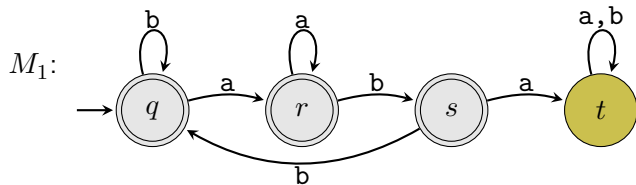
aabba**b**a

## Union example



aabbab**a**

## Union example



aabbaba ✗ Rejected

# ENDSWITH

$$\text{ENDSWITH}(A) = \{xw \mid x \in \Sigma^* \text{ and } w \in A\}$$

- $A = \{a, aab, bab\}$ ;  $\text{ENDSWITH}(A) = \{w \mid w \text{ ends with } a, aab, \text{ or } bab\}$
- $B = \{b^k \mid k > 0\}$ ;  $\text{ENDSWITH}(B) = \{w \mid w \text{ ends with 1 or more } b\}$
- $C = \{a^k b^k \mid k \geq 0\}$ ;  
 $\text{ENDSWITH}(C) = \{w \mid w \text{ ends with } a^k b^k \text{ for some } k \geq 0\} = \Sigma^* \text{ [Why?]}$

## A simple theorem

### Theorem

*If  $A$  is regular, then  $\text{ENDSWITH}(A) = \{xw \mid x \in \Sigma^* \text{ and } w \in A\}$  is regular.*

### Proof technique

Start by assuming that  $A$  is regular and thus there exists a DFA  $M$  such that  $L(M) = A$

Now construct a new DFA  $M'$  such that  $L(M') = \text{ENDSWITH}(A)$ .

Ideally, this new DFA would have two parts:

- ① some states that read symbols from  $\Sigma^*$  (i.e., matching the symbols of  $x$ )
- ② a copy of  $M$  to accept the last part of the string which should be in  $A$

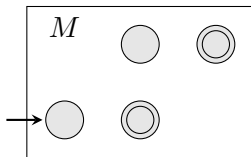
# A simple theorem proof difficulty

The two parts are individually easy

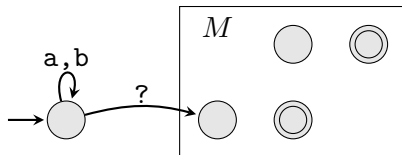
- 1 Match symbols from  $\Sigma^*$  (assume  $\Sigma = \{a, b\}$ , easy to generalize)



- 2 A copy of  $M$



But how can we combine them?



# Determinism

DFAs are **deterministic** because at every step, the DFA has exactly one thing it can do

When  $M$  is in some state  $q \in Q$  and the next input symbol is  $t \in \Sigma$ , the only thing it can do is move to state  $\delta(q, t)$

Graphically, we don't allow any state to have multiple edges (transitions) labeled with the same symbol going to different states

Similarly, we don't allow a state to not have a transition labeled with a symbol of  $\Sigma$

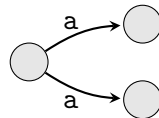


# Nondeterminism

Let's build a new type of machine, a **nondeterministic** finite automaton (NFA), where at each step, it has zero or more things it can do

Three new options

- 1 Multiple transitions from a state on the same symbol

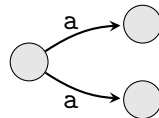


# Nondeterminism

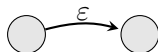
Let's build a new type of machine, a **nondeterministic** finite automaton (NFA), where at each step, it has zero or more things it can do

Three new options

- 1 Multiple transitions from a state on the same symbol



- 2 Transitions on no input

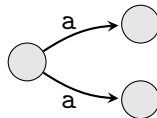


# Nondeterminism

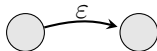
Let's build a new type of machine, a **nondeterministic** finite automaton (NFA), where at each step, it has zero or more things it can do

Three new options

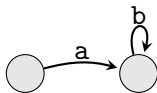
- ① Multiple transitions from a state on the same symbol



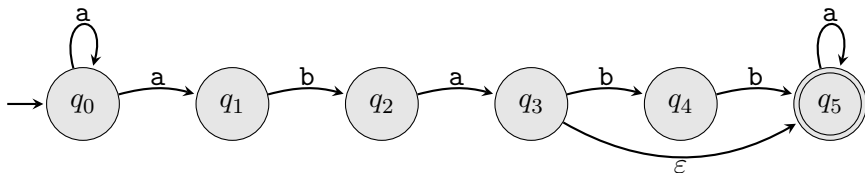
- ② Transitions on no input



- ③ States without transitions on some (or all) symbols

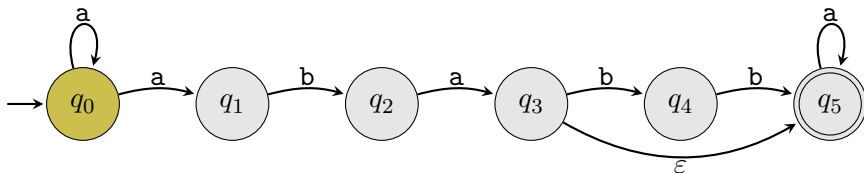


## Example



Let's run this on input ababb

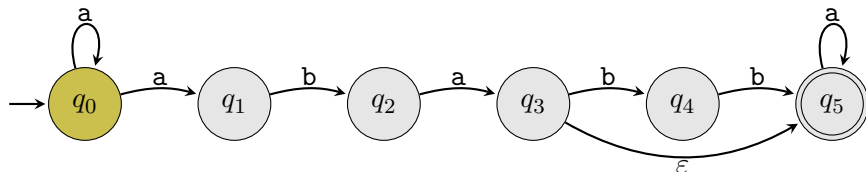
## Example



Let's run this on input ababb

- 1 Start in  $q_0$ , first symbol is a, two choices, let's stay in  $q_0$

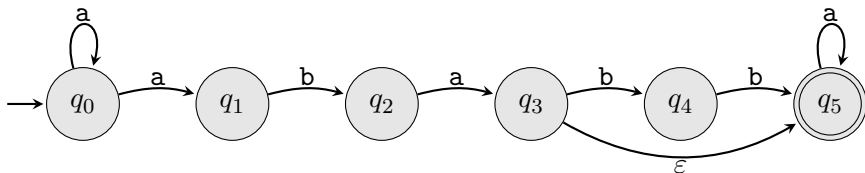
## Example



Let's run this on input ababb

- 1 Start in  $q_0$ , first symbol is a, two choices, let's stay in  $q_0$
- 2 Next symbol is b, but there are no transitions labeled b

## Example

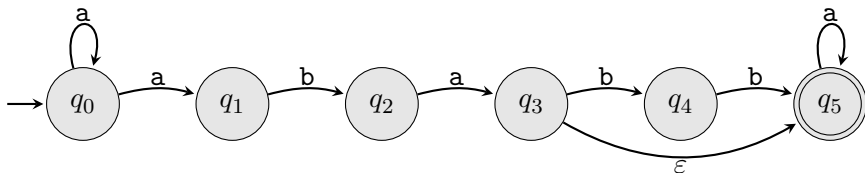


Let's run this on input ababb

- 1 Start in  $q_0$ , first symbol is a, two choices, let's stay in  $q_0$
- 2 Next symbol is b, but there are no transitions labeled b
- 3 Now the machine is dead because there's no active state

Since the machine didn't end in an accepting state. Is ababb **✗Rejected**?

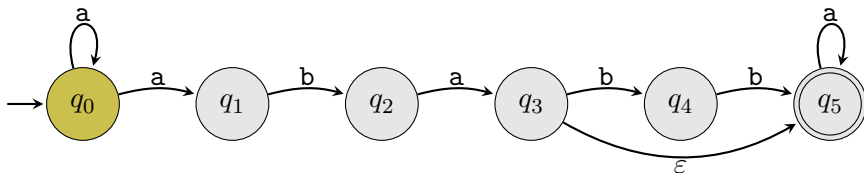
## Example again



Let's run this on input ababb again



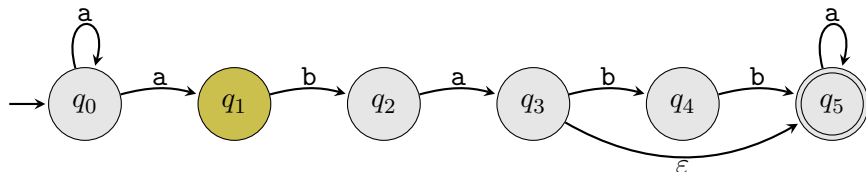
## Example again



Let's run this on input ababb again

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$

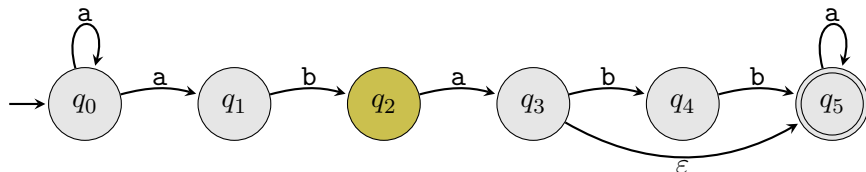
## Example again



Let's run this on input ababb again

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$
- 2 Next symbol is b, go to  $q_2$

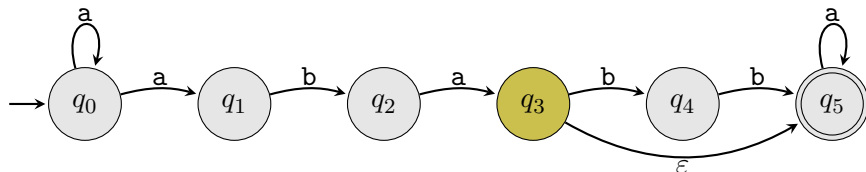
## Example again



Let's run this on input ababb again

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$
- 2 Next symbol is b, go to  $q_2$
- 3 Next symbol is a, go to  $q_3$

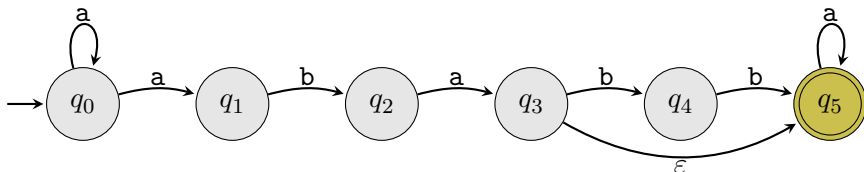
## Example again



Let's run this on input ababb again

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$
- 2 Next symbol is b, go to  $q_2$
- 3 Next symbol is a, go to  $q_3$
- 4 We have two choices: follow the  $\varepsilon$  transition or not, let's follow it

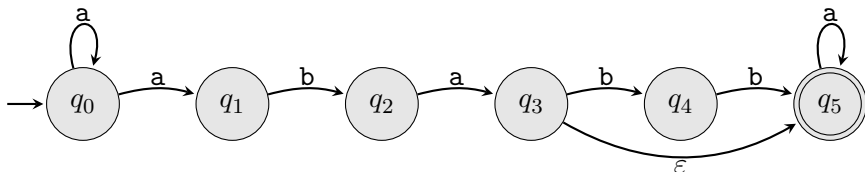
## Example again



Let's run this on input ababb again

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$
- 2 Next symbol is b, go to  $q_2$
- 3 Next symbol is a, go to  $q_3$
- 4 We have two choices: follow the  $\varepsilon$  transition or not, let's follow it
- 5 Next symbol is b, but there are no transitions labeled b

## Example again

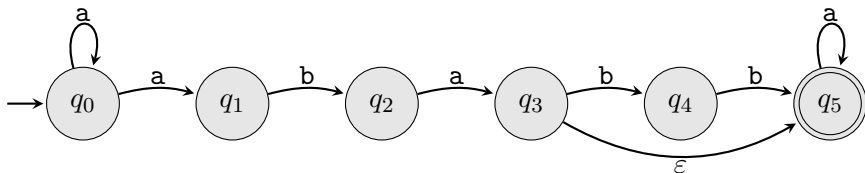


Let's run this on input ababb again

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$
- 2 Next symbol is b, go to  $q_2$
- 3 Next symbol is a, go to  $q_3$
- 4 We have two choices: follow the  $\varepsilon$  transition or not, let's follow it
- 5 Next symbol is b, but there are no transitions labeled b
- 6 Now the machine is dead because there's no active state

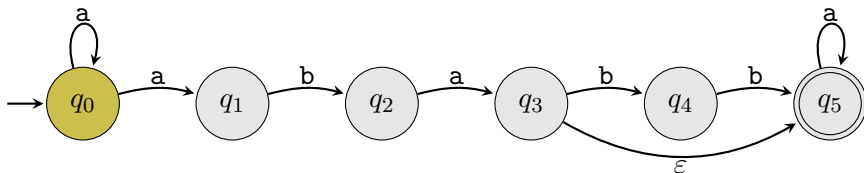
Once again, it didn't end in an accepting state.

## Example yet again



Let's run this on input ababb a third time

## Example yet again

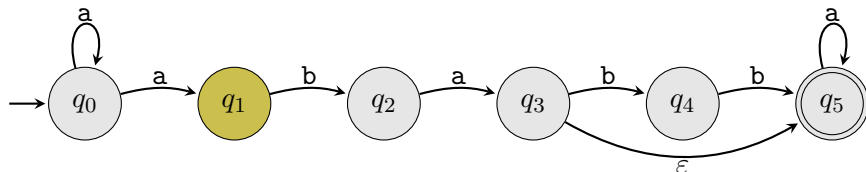


Let's run this on input ababb a third time

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$



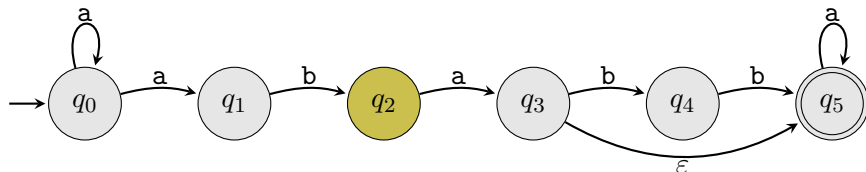
## Example yet again



Let's run this on input ababb a third time

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$
- 2 Next symbol is b, go to  $q_2$

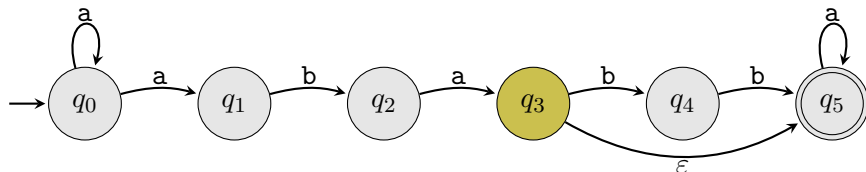
## Example yet again



Let's run this on input ababb a third time

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$
- 2 Next symbol is b, go to  $q_2$
- 3 Next symbol is a, go to  $q_3$

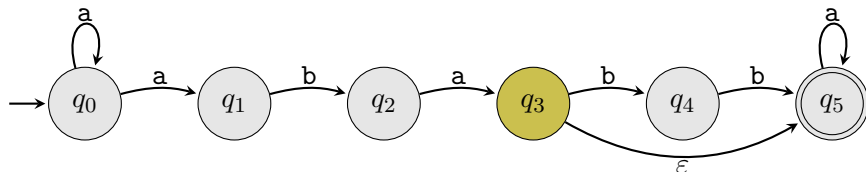
## Example yet again



Let's run this on input ababb a third time

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$
- 2 Next symbol is b, go to  $q_2$
- 3 Next symbol is a, go to  $q_3$
- 4 We have two choices: follow the  $\varepsilon$  transition or not, let's *not* follow it

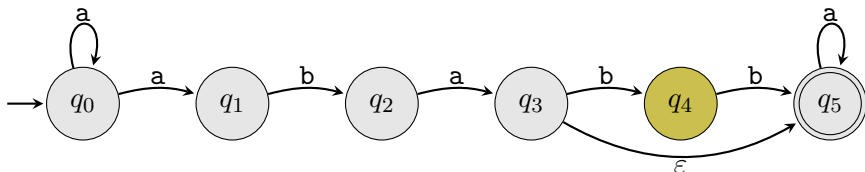
## Example yet again



Let's run this on input ababb a third time

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$
- 2 Next symbol is b, go to  $q_2$
- 3 Next symbol is a, go to  $q_3$
- 4 We have two choices: follow the  $\varepsilon$  transition or not, let's *not* follow it
- 5 Next symbol is b, go to  $q_4$

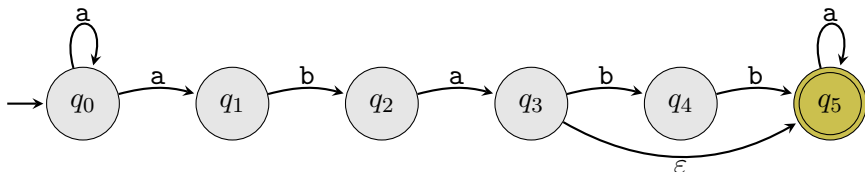
## Example yet again



Let's run this on input ababb a third time

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$
- 2 Next symbol is b, go to  $q_2$
- 3 Next symbol is a, go to  $q_3$
- 4 We have two choices: follow the  $\epsilon$  transition or not, let's *not* follow it
- 5 Next symbol is b, go to  $q_4$
- 6 Next symbol is b, go to  $q_5$

## Example yet again



Let's run this on input ababb a third time

- 1 Start in  $q_0$ , first symbol is a, two choices, let's go to  $q_1$
- 2 Next symbol is b, go to  $q_2$
- 3 Next symbol is a, go to  $q_3$
- 4 We have two choices: follow the  $\epsilon$  transition or not, let's *not* follow it
- 5 Next symbol is b, go to  $q_4$
- 6 Next symbol is b, go to  $q_5$
- 7 There's no more input and the machine ended in an accepting state so ababb is  
✓ Accepted

## Was ababb accepted or rejected?

Two choices we made led to the machine dying because it couldn't follow a transition

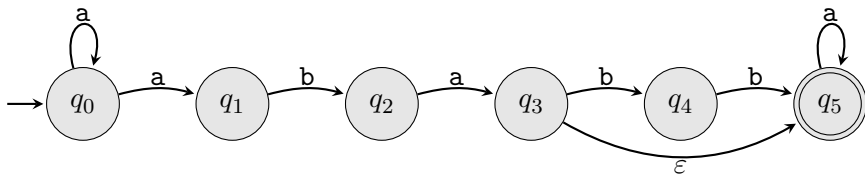
The third choice we made ended in an accepting state

Let's say an **NFA accepts a string** if *any* path through the NFA ends in an accepting state

So ababb was  **Accepted**

# Language of the NFA

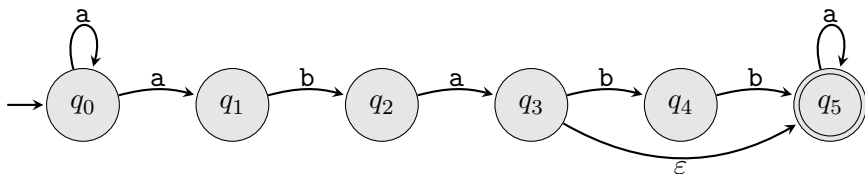
What strings are accepted by this NFA?





# Language of the NFA

What strings are accepted by this NFA?



Strings starting with at least 1 a, followed by ba, optionally followed by bb, followed by any number of as:  $\{a^m b a w a^n \mid m \geq 1 \text{ and } n \geq 0 \text{ and } w \in \{\epsilon, bb\}\}$

## Running NFAs

It was a pain to run the NFA multiple times on the same input, making different choices

Let's instead keep track of all possible states the NFA  $N$  can be in at each point in its computation

Rather than having a single current state, let's have a set of current states, call it  $C$

At each step, we're going to update  $C$

# Procedure for running NFAs

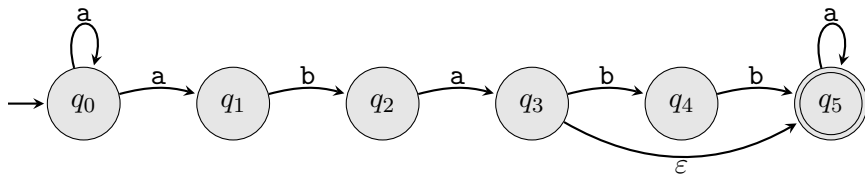
## Procedure

- ➊ Set  $C = \{q_0\}$ , the set containing only the start state
- ➋ Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- ➌ For each successive symbol  $t$  in the input  $w$ ,
- ➍ Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- ➎ Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- ➏ If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

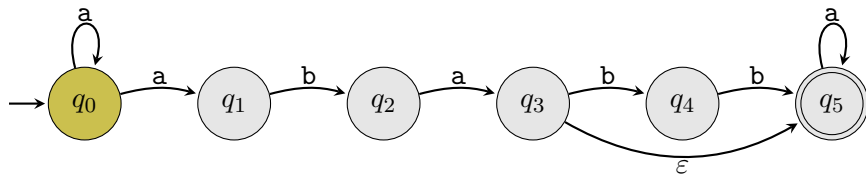


ababb

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

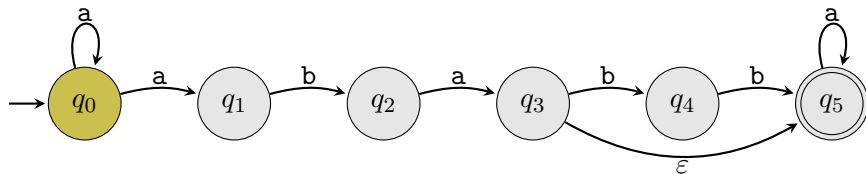


ababb

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

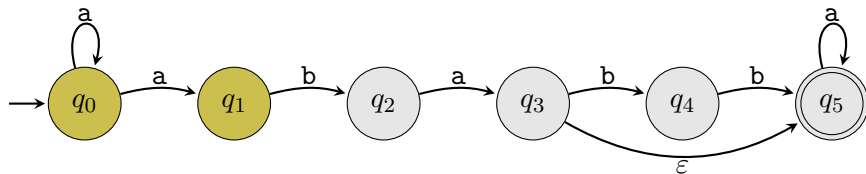


ababb

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

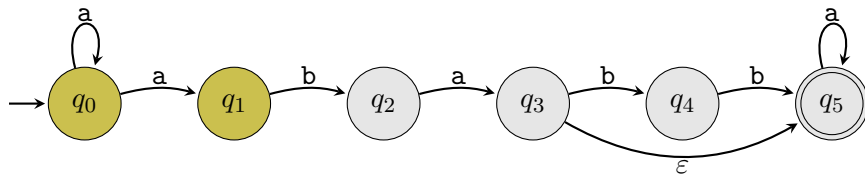


a**b**abb

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$



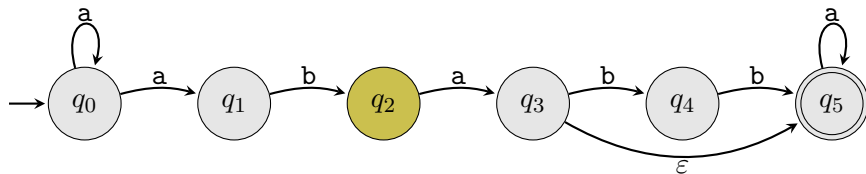
a**b**abb



# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

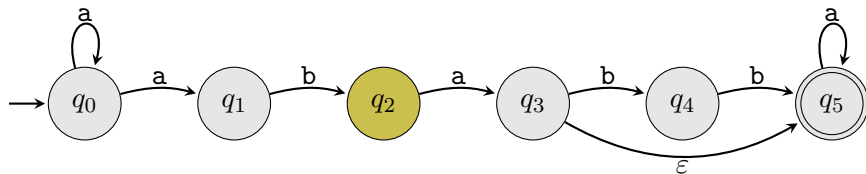


ababb

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

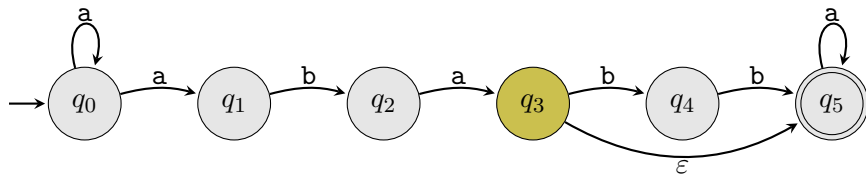


ababb

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

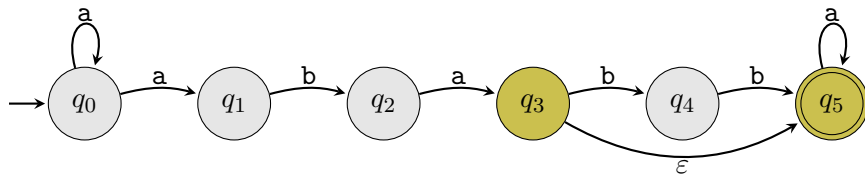


aba**bb**

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

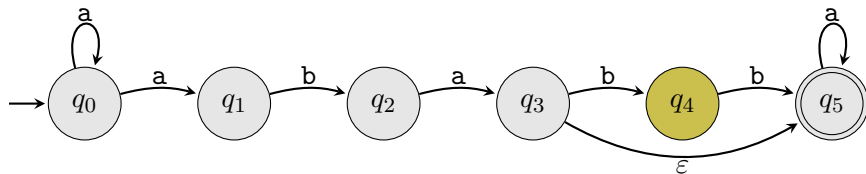


aba**b**

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

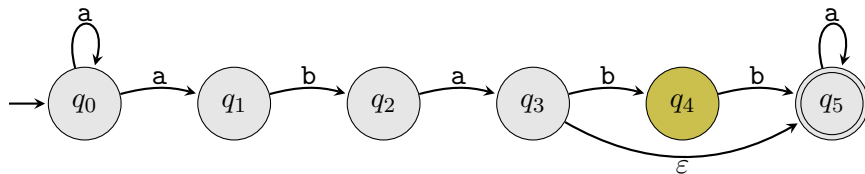


abab**b**

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

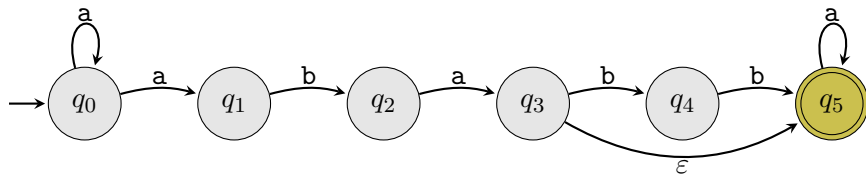


abab**b**

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

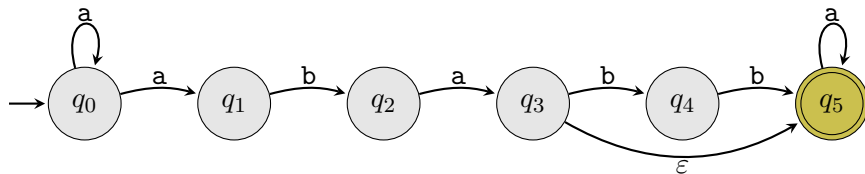


ababb

# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$



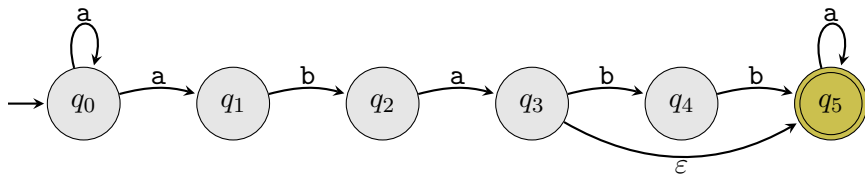
ababb



# Running our NFAs

## Procedure

- 1 Set  $C = \{q_0\}$ , the set containing only the start state
- 2 Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 3 For each successive symbol  $t$  in the input  $w$ ,
- 4   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- 5   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- 6 If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$



ababb ✓ Accepted

# Nondeterministic finite automaton (NFA)

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

- $Q$  is a finite set of **states**
- $\Sigma$  is an **alphabet**
- $\delta : Q \times \Sigma_\epsilon \rightarrow P(Q)$  is the **transition function**
- $q_0 \in Q$  is the **start state**
- $F \subseteq Q$  is the **set of accepting (or final) states**

$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$  is the alphabet  $\Sigma$  augmented with an additional symbol  $\epsilon$  which we use to denote transitions on no input

$P(Q)$  is the power set of  $Q$  so  $\delta$  returns a **set** of next states

## Transition functions

DFAs have transitions of the form  $\delta : Q \times \Sigma \rightarrow Q$

For each (state, symbol) pair,  $\delta$  returns a single state

NFAs have transitions of the form  $\delta : Q \times \Sigma_\epsilon \rightarrow P(Q)$

For each (state, symbol) pair,  $\delta$  returns 0 or more states

For each (state,  $\epsilon$ ),  $\delta$  returns 0 or more states

## Formalizing NFA computation

Let  $N = (Q, \Sigma, \delta, q_0, F)$  be an NFA and let  $w = w_1w_2\cdots w_n$  be a string where  $w_i \in \Sigma_\varepsilon$

$N$  accepts  $w$  if there exist states  $r_0, r_1, \dots, r_n \in Q$  such that

- ❶  $r_0 = q_0$   
[The NFA starts in the start state]
- ❷  $r_i \in \delta(r_{i-1}, w_i)$  for  $i \in \{1, 2, \dots, n\}$   
[The NFA moves from state  $r_{i-1}$  to one of the possible next states according to  $\delta$ ]
- ❸  $r_n \in F$   
[The NFA ends in an accepting state]

## Formalizing NFA computation

Let  $N = (Q, \Sigma, \delta, q_0, F)$  be an NFA and let  $w = w_1w_2\cdots w_n$  be a string where  $w_i \in \Sigma_\varepsilon$

$N$  accepts  $w$  if there exist states  $r_0, r_1, \dots, r_n \in Q$  such that

- ①  $r_0 = q_0$   
[The NFA starts in the start state]
- ②  $r_i \in \delta(r_{i-1}, w_i)$  for  $i \in \{1, 2, \dots, n\}$   
[The NFA moves from state  $r_{i-1}$  to one of the possible next states according to  $\delta$ ]
- ③  $r_n \in F$   
[The NFA ends in an accepting state]

Two key differences from DFAs

- ①  $w_i$  is either an alphabet symbol or  $\varepsilon$   
E.g., if  $w = \text{abaa}$ , then we can write  $w = \varepsilon \text{ab} \varepsilon \varepsilon \varepsilon \text{a} \varepsilon \text{a}$
- ②  $r_i \in \delta(r_{i-1}, w_i)$  since  $\delta$  returns a set of next possible states

The sequence of  $n + 1$  states  $r_0, r_1, \dots, r_n$  is one of the possible sequences of states that the NFA moves through on input  $w$

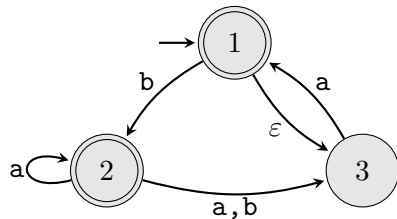
## Language of an NFA

The **language** of an NFA  $N$  is  $L(N) = \{w \mid N \text{ accepts } w\}$

We say  $N$  **recognizes** a language  $A$  to mean  $L(N) = A$

[This is analogous to DFAs]

## Example



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

$$Q = \{1, 2, 3\}$$

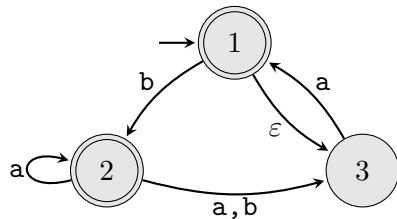
$$\Sigma = \{a, b\}$$

$$q_0 = 1$$

$$F = \{1, 2\}$$

$\delta :$	a	b	$\varepsilon$
1	$\emptyset$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\emptyset$
3	$\{1\}$	$\emptyset$	$\emptyset$

## Example



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

$$Q = \{1, 2, 3\}$$

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

$$q_0 = 1$$

$$F = \{1, 2\}$$

$\delta :$	a	b	$\varepsilon$
1	$\emptyset$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\emptyset$
3	$\{1\}$	$\emptyset$	$\emptyset$

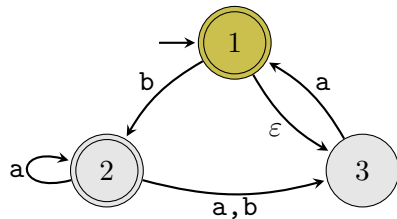
Consider string  $w = abaa$

Write  $w$  as  $\varepsilon abaa$  then one of the possible sequences of states  $N$  moves through is

$r_0 \quad r_1 \quad r_2 \quad r_3 \quad r_4 \quad r_5$



## Example



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

$$Q = \{1, 2, 3\}$$

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

$$q_0 = 1$$

$$F = \{1, 2\}$$

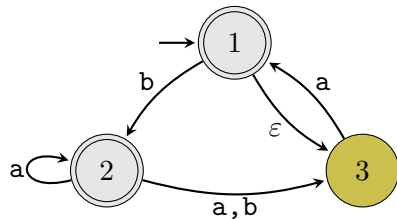
$\delta :$	a	b	$\varepsilon$
1	$\emptyset$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\emptyset$
3	$\{1\}$	$\emptyset$	$\emptyset$

Consider string  $w = abaa$

Write  $w$  as  $\varepsilon abaa$  then one of the possible sequences of states  $N$  moves through is

$$\begin{array}{cccccc} r_0 & r_1 & r_2 & r_3 & r_4 & r_5 \\ 1 & & & & & \end{array}$$

## Example



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

$$Q = \{1, 2, 3\}$$

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

$$q_0 = 1$$

$$F = \{1, 2\}$$

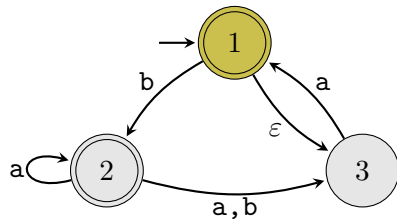
$\delta :$	a	b	$\epsilon$
1	$\emptyset$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\emptyset$
3	$\{1\}$	$\emptyset$	$\emptyset$

Consider string  $w = abaa$

Write  $w$  as  $\epsilon abaa$  then one of the possible sequences of states  $N$  moves through is

$r_0$	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$
1	3				

## Example



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

$$Q = \{1, 2, 3\}$$

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

$$q_0 = 1$$

$$F = \{1, 2\}$$

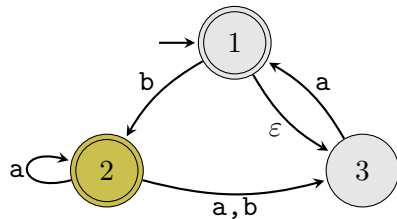
$\delta :$	a	b	$\varepsilon$
1	$\emptyset$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\emptyset$
3	$\{1\}$	$\emptyset$	$\emptyset$

Consider string  $w = abaa$

Write  $w$  as  $\varepsilon abaa$  then one of the possible sequences of states  $N$  moves through is

$r_0$	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$
1	3	1			

## Example



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

$$Q = \{1, 2, 3\}$$

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

$$q_0 = 1$$

$$F = \{1, 2\}$$

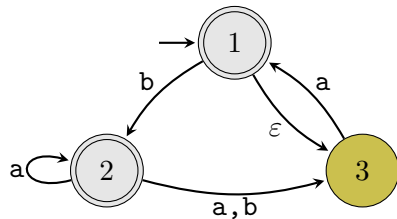
$\delta :$	a	b	$\epsilon$
1	$\emptyset$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\emptyset$
3	$\{1\}$	$\emptyset$	$\emptyset$

Consider string  $w = abaa$

Write  $w$  as  $\epsilon abaa$  then one of the possible sequences of states  $N$  moves through is

$r_0$	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$
1	3	1	2		

## Example



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

$$Q = \{1, 2, 3\}$$

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

$$q_0 = 1$$

$$F = \{1, 2\}$$

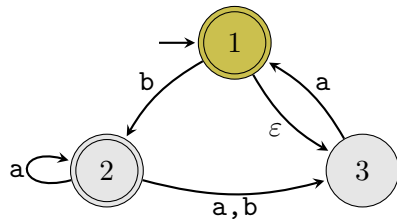
$\delta :$	a	b	$\varepsilon$
1	$\emptyset$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\emptyset$
3	$\{1\}$	$\emptyset$	$\emptyset$

Consider string  $w = abaa$

Write  $w$  as  $\varepsilon abaa$  then one of the possible sequences of states  $N$  moves through is

$r_0$	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$
1	3	1	2	3	

## Example



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

$$Q = \{1, 2, 3\}$$

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

$$q_0 = 1$$

$$F = \{1, 2\}$$

$\delta :$	a	b	$\varepsilon$
1	$\emptyset$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\emptyset$
3	$\{1\}$	$\emptyset$	$\emptyset$

Consider string  $w = abaa$

Write  $w$  as  $\varepsilon abaa$  then one of the possible sequences of states  $N$  moves through is

$r_0$	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$
1	3	1	2	3	1

All three conditions for acceptance hold

- ①  $r_0 = q_0$
- ②  $r_i \in \delta(r_{i-1}, w_i)$  for  $i \in \{1, 2, \dots, n\}$
- ③  $r_n \in F$

# Converting NFAs to DFAs

## Theorem

*For every NFA  $N$ , there exists a DFA  $M$  such that  $L(M) = L(N)$ .*

We can prove this by following our procedure for running NFAs

## Procedure

- ① Set  $C = \{q_0\}$ , the set containing only the start state
- ② Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- ③ For each successive symbol  $t$  in the input  $w$ ,
- ④   Set  $C = \{q \mid \text{there is a transition to } q \text{ on symbol } t \text{ from some state in } C\}$
- ⑤   Set  $C = \{q \mid q \text{ is reachable from } C \text{ by following 0 or more } \varepsilon\text{-transitions}\}$
- ⑥ If  $C$  contains any accepting states,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

## Some helpful notation

Given an NFA  $N = (Q, \Sigma, \delta, q_0, F)$ , define a new function  $E$  that takes a set of states  $S \subseteq Q$  as input and returns the set of states reachable by following 0 or more  $\varepsilon$ -transitions from states in  $S$

Formally,  $E : P(Q) \rightarrow P(Q)$  given by

$E(S) = \{q \mid q \text{ is reachable from some } r \in S \text{ by following 0 or more } \varepsilon\text{-transitions}\}$

$E(S)$  is called the  $\varepsilon$ -closure of  $S$



## Some helpful notation

Given an NFA  $N = (Q, \Sigma, \delta, q_0, F)$ , define a new function  $E$  that takes a set of states  $S \subseteq Q$  as input and returns the set of states reachable by following 0 or more  $\varepsilon$ -transitions from states in  $S$

Formally,  $E : P(Q) \rightarrow P(Q)$  given by

$E(S) = \{q \mid q \text{ is reachable from some } r \in S \text{ by following 0 or more } \varepsilon\text{-transitions}\}$

$E(S)$  is called the  $\varepsilon$ -closure of  $S$

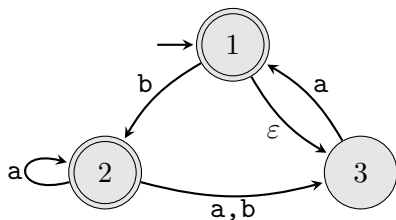
### Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3   Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

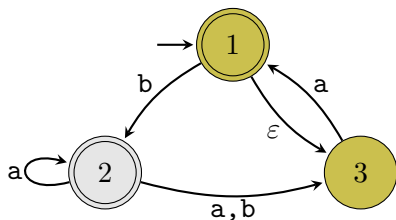


• abaabba

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

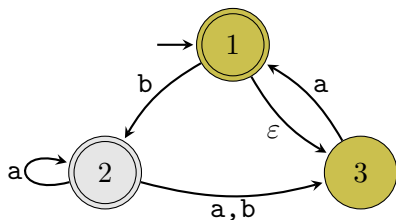


• abaabba

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

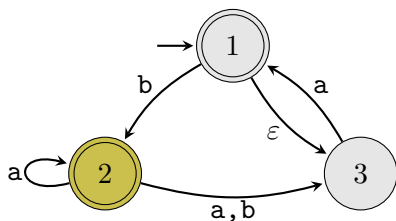


• a**b**aabba

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

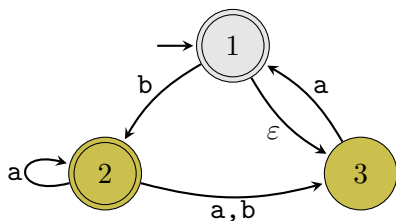


• ab**a**abba

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

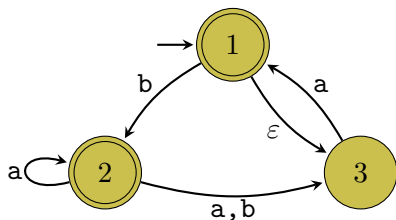


• aba**a**bba

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

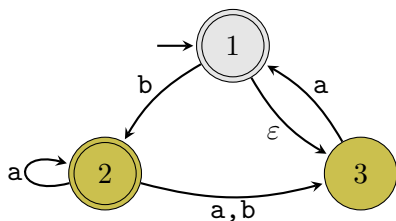


• abaa**b**ba

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$



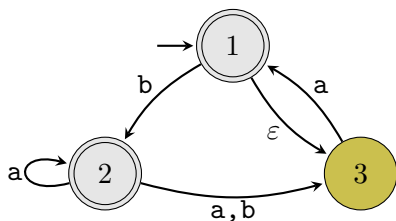
• abaab**b**a



# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

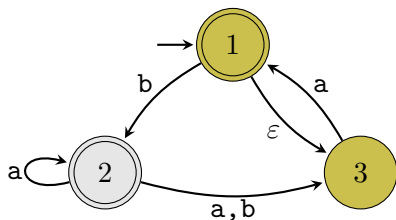


• abaabba

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

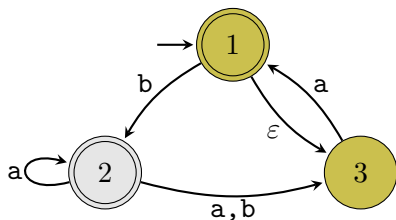


• abaabba

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

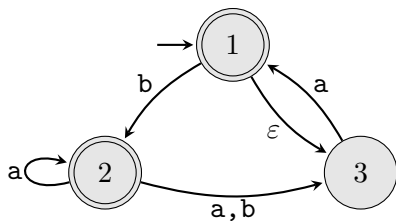


• abaabba ✓ Accepted

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$



- abaabba

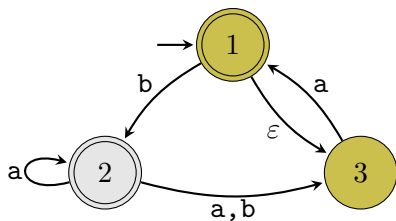
✓ Accepted

- bbbab

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$



• abaabba

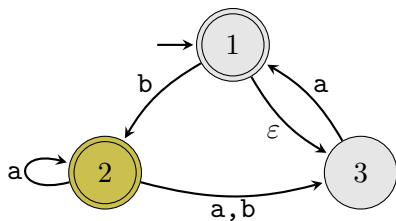
✓ Accepted

• bbbab

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$



- abaabba

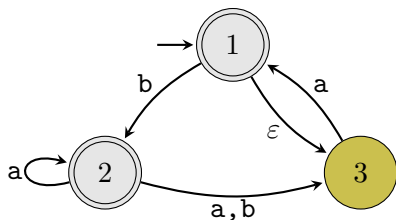
✓ Accepted

- b**b**bab

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

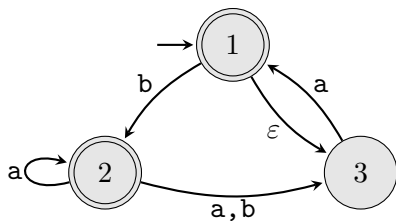


- abaabba ✓ Accepted
- bbab

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$



• abaabba

✓ Accepted

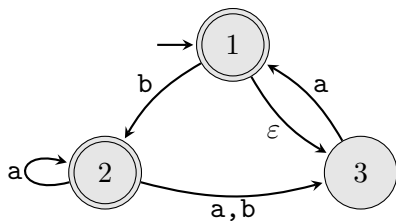
• bbbab



# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

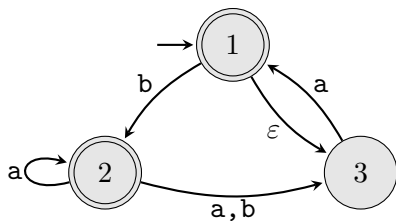


- abaabba ✓ Accepted
- bbbab

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

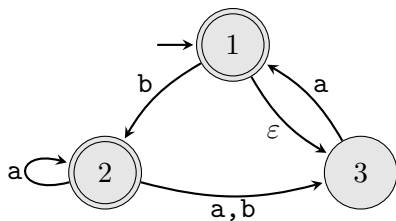


- abaabba ✓ Accepted
- bbbab

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$



• abaabba

✓ Accepted

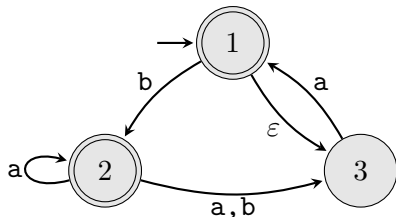
• bbbab

✗ Rejected

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$



- abaabba

✓ Accepted

- bbbab

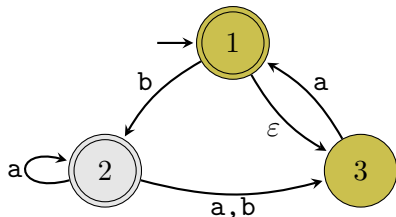
✗ Rejected

- bb

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

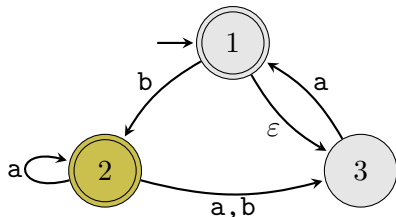


- abaabba ✓ Accepted
- bbbab ✗ Rejected
- **bb**

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

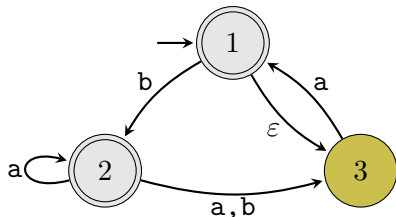


- abaabba ✓ Accepted
- bbbab ✗ Rejected
- bb

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

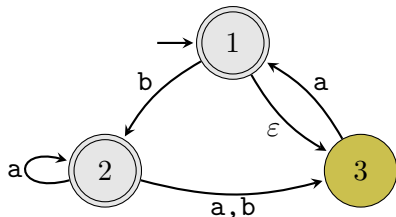


- abaabba ✓ Accepted
- bbbab ✗ Rejected
- bb

# Running the procedure again

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$



- abaabba ✓ Accepted
- bbbab ✗ Rejected
- bb ✗ Rejected



# Converting an NFA to a DFA

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3     Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

Given an NFA  $N = (Q, \Sigma, \delta, q_0, F)$ , we can convert our procedure into a DFA  $M = (Q', \Sigma, \delta', q'_0, F')$

# Converting an NFA to a DFA

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3     Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

Given an NFA  $N = (Q, \Sigma, \delta, q_0, F)$ , we can convert our procedure into a DFA  $M = (Q', \Sigma, \delta', q'_0, F')$

- States in  $M$  are sets of states in  $N$ :  $Q' = P(Q)$

# Converting an NFA to a DFA

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3     Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

Given an NFA  $N = (Q, \Sigma, \delta, q_0, F)$ , we can convert our procedure into a DFA  $M = (Q', \Sigma, \delta', q'_0, F')$

- States in  $M$  are sets of states in  $N$ :  $Q' = P(Q)$
- $M$ 's start state is  $q'_0 = E(\{q_0\})$

# Converting an NFA to a DFA

## Procedure (ver. 2)

- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3   Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

Given an NFA  $N = (Q, \Sigma, \delta, q_0, F)$ , we can convert our procedure into a DFA  $M = (Q', \Sigma, \delta', q'_0, F')$

- States in  $M$  are sets of states in  $N$ :  $Q' = P(Q)$
- $M$ 's start state is  $q'_0 = E(\{q_0\})$
- $M$ 's transition function  $\delta' : P(Q) \times \Sigma \rightarrow P(Q)$  is  
 $\delta'(C, t) = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$

# Converting an NFA to a DFA

## Procedure (ver. 2)

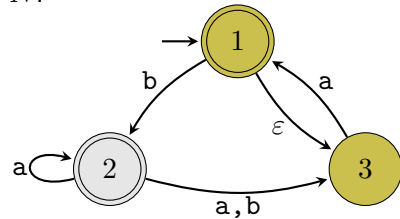
- 1 Set  $C = E(\{q_0\})$
- 2 For each successive symbol  $t$  in the input  $w$ ,
- 3 Set  $C = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- 4 If  $C \cap F \neq \emptyset$ ,  $N$  accepts  $w$ , otherwise  $N$  rejects  $w$

Given an NFA  $N = (Q, \Sigma, \delta, q_0, F)$ , we can convert our procedure into a DFA  $M = (Q', \Sigma, \delta', q'_0, F')$

- States in  $M$  are sets of states in  $N$ :  $Q' = P(Q)$
- $M$ 's start state is  $q'_0 = E(\{q_0\})$
- $M$ 's transition function  $\delta' : P(Q) \times \Sigma \rightarrow P(Q)$  is  
 $\delta'(C, t) = \{q \mid q \in E(\delta(r, t)) \text{ for some } r \in C\}$
- $M$ 's accepting states are every subset of  $Q$  that contains at least one of  $N$ 's accepting states:  $F' = \{S \mid S \subseteq Q \text{ and } S \cap F \neq \emptyset\}$

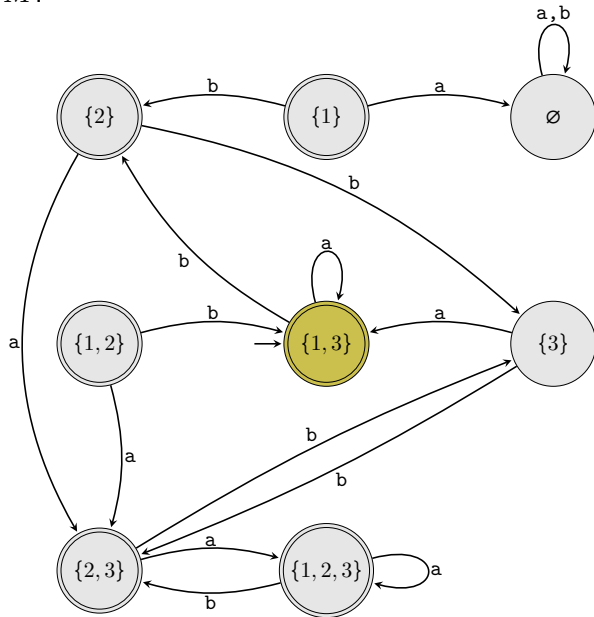
# Converting our example to a DFA

$N$ :



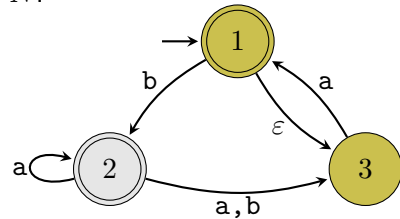
abaababb

$M$ :



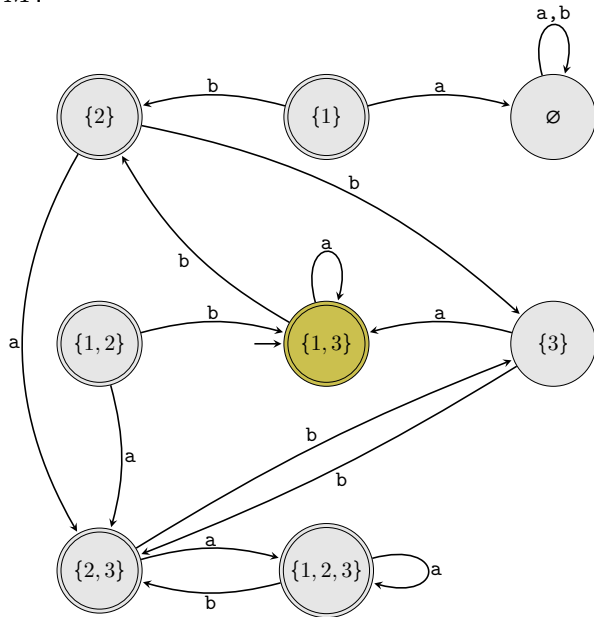
# Converting our example to a DFA

$N$ :



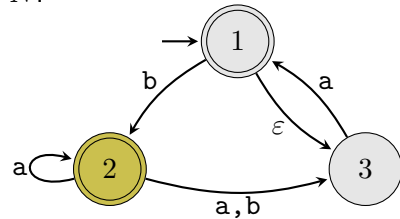
a**b**aababb

$M$ :



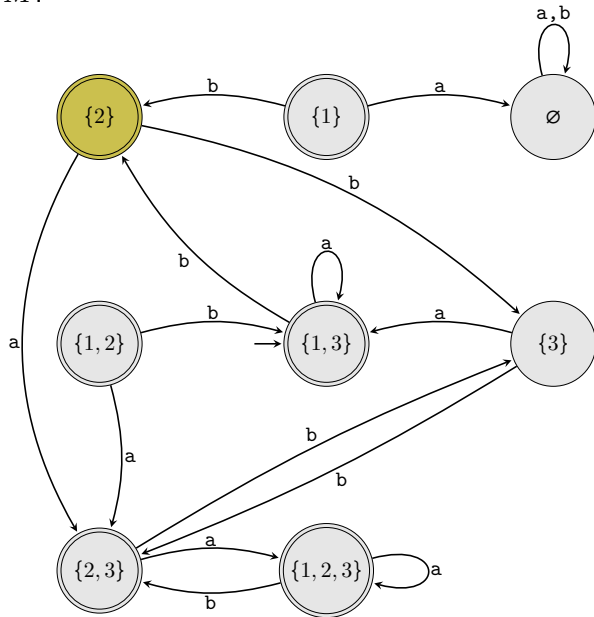
# Converting our example to a DFA

$N$ :



abababb

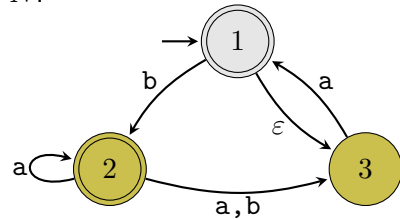
$M$ :





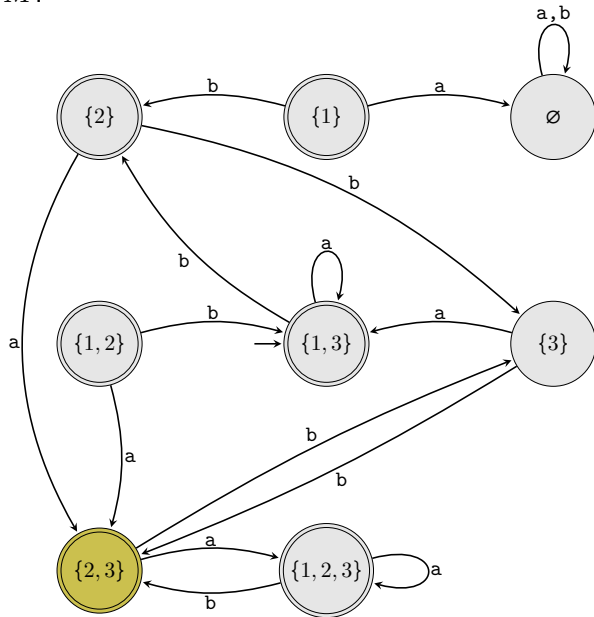
# Converting our example to a DFA

$N$ :



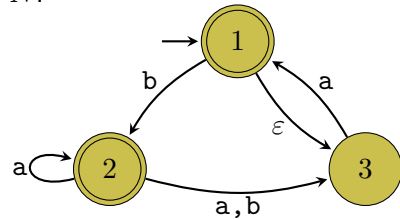
aba**a**babb

$M$ :



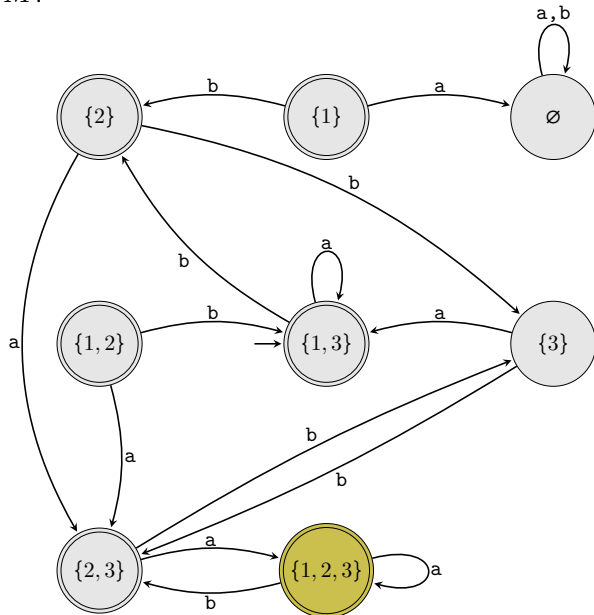
# Converting our example to a DFA

$N$ :



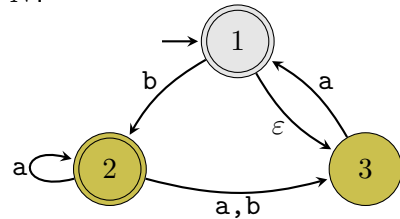
abaababb

$M$ :



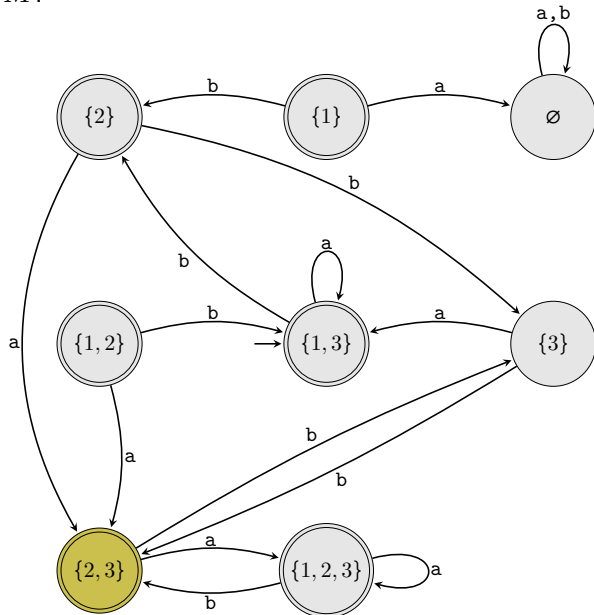
# Converting our example to a DFA

$N$ :



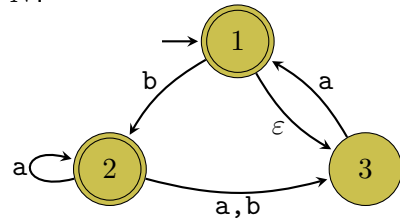
abaababb

$M$ :



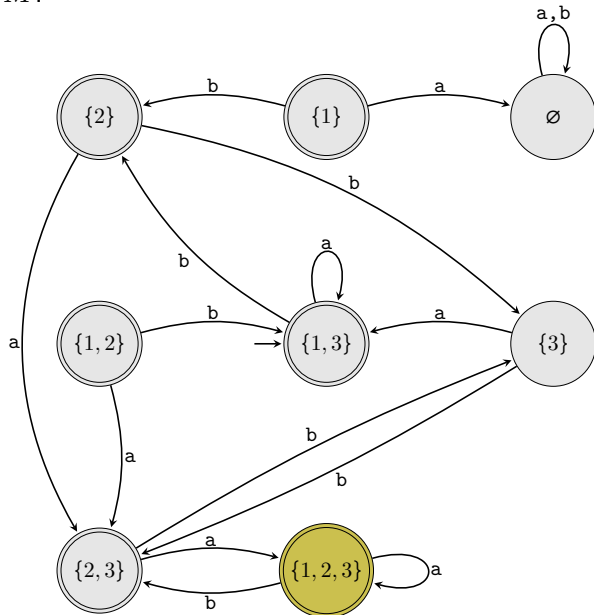
# Converting our example to a DFA

$N$ :



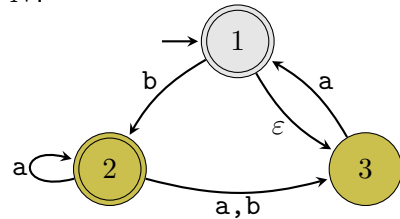
abaaba**b**b

$M$ :



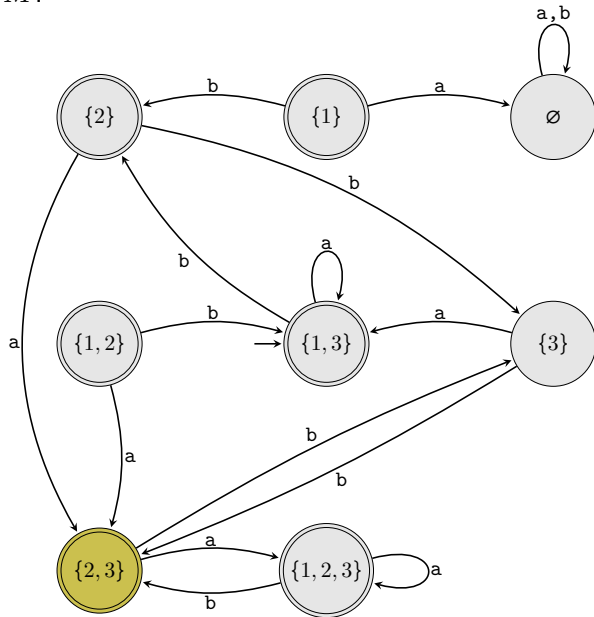
# Converting our example to a DFA

$N$ :



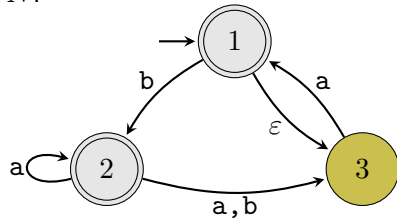
abaabab**b**

$M$ :



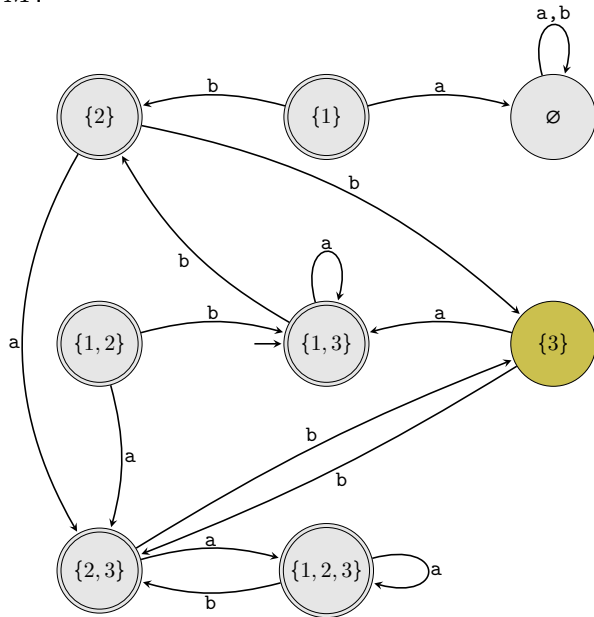
# Converting our example to a DFA

$N$ :



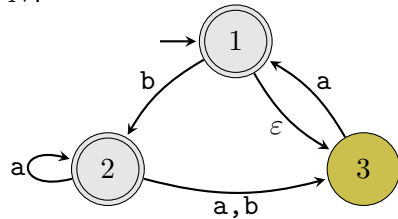
abaababb

$M$ :



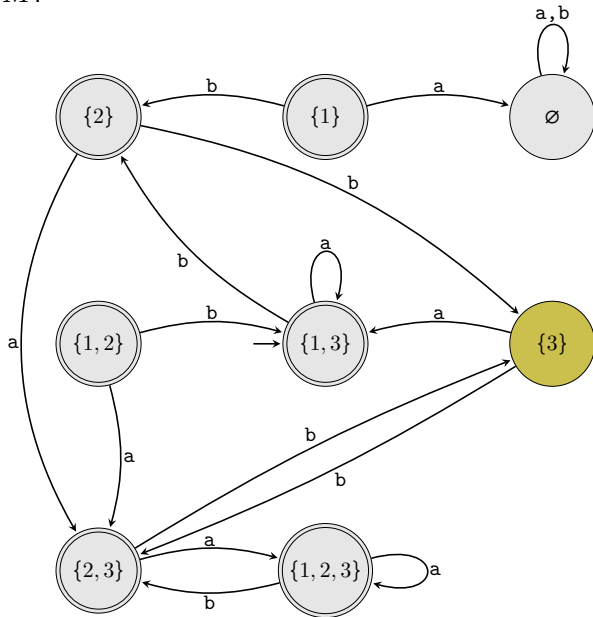
# Converting our example to a DFA

$N$ :



abaababb ✗ Rejected

$M$ :



# Regular languages

## Theorem

*A language  $A$  is regular if and only if it is recognized by some NFA  $N$ .*

## Proof.

$\implies$

If  $A$  is regular, then it is recognized by a DFA  $M$ . DFAs are NFAs where each state has exactly one next state for each alphabet symbol so  $M$  is an NFA.

$\impliedby$

If NFA  $N$  recognizes  $A$ , then using the NFA to DFA construction, we can build an DFA  $M$  such that  $L(M) = A$ . Therefore,  $A$  is regular. □



# Regular languages closed under operations

Let  $f$  be an operation on languages

[Recall that means  $f$  takes some languages as input and produces a new language as output]

We say **regular languages are closed under  $f$**  to mean

**Unary** If  $A$  is regular, then  $f(A)$  is regular

**Binary** If  $A$  and  $B$  are regular, then  $f(A, B)$  is regular

**$n$ -ary** If  $A_1, A_2, \dots, A_n$  are regular, then  $f(A_1, A_2, \dots, A_n)$  is regular

# Regular languages are closed under regular operations

## Regular operations

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

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

Kleene star  $A^* = \{w_1 w_2 \cdots w_k \mid k \geq 0 \text{ and } w_i \in A \text{ for all } i\}$

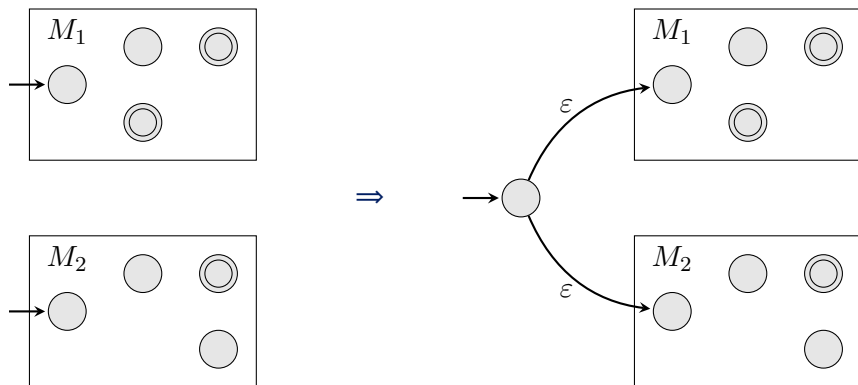
## Theorem

*Regular languages are closed under union, concatenation, and Kleene star.*

In other words, if  $A$  and  $B$  are regular languages, then  $A \cup B$ ,  $A \circ B$ , and  $A^*$  are regular.

# Union

Let  $A$  and  $B$  be regular languages recognized by DFAs  $M_1$  and  $M_2$



# Regular languages are closed under union

Proof.

Let  $A$  and  $B$  be regular languages  
recognized by DFAs

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

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

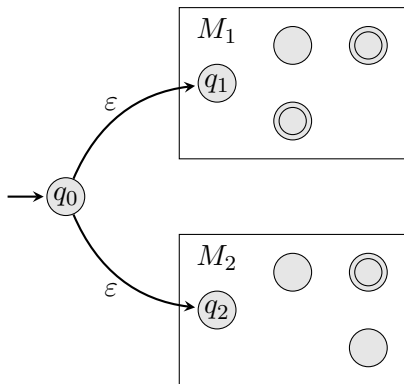
Build NFA  $N = (Q, \Sigma, \delta, q_0, F)$  where

$$Q = Q_1 \cup Q_2 \cup \{q_0\}$$

$$F = F_1 \cup F_2$$

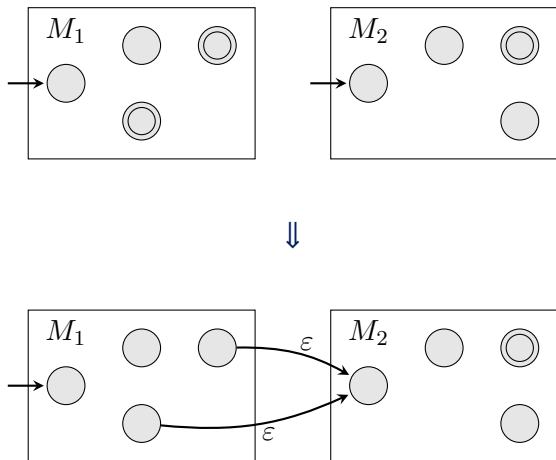
$$\delta(q, \varepsilon) = \begin{cases} \{q_1, q_2\} & \text{if } q = q_0 \\ \emptyset & \text{otherwise} \end{cases}$$

$$\delta(q, t) = \begin{cases} \emptyset & \text{if } q = q_0 \\ \{\delta_1(q, t)\} & \text{for } q \in Q_1 \\ \{\delta_2(q, t)\} & \text{for } q \in Q_2 \end{cases} \quad \square$$



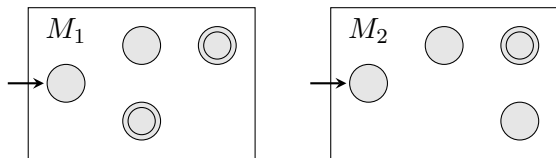
# Concatenation

Let  $A$  and  $B$  be regular languages recognized by DFAs  $M_1$  and  $M_2$

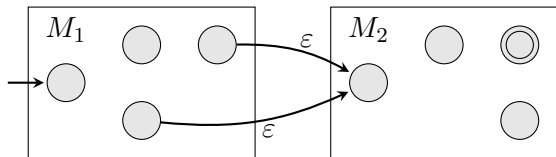


# Concatenation

Let  $A$  and  $B$  be regular languages recognized by DFAs  $M_1$  and  $M_2$



⇓



Let

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

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

Build NFA  $N = (Q, \Sigma, \delta, q_1, F_2)$  where

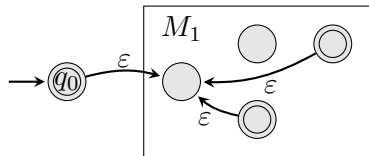
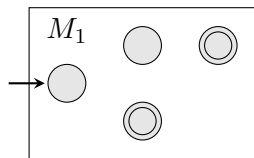
$$Q = Q_1 \cup Q_2$$

$$\delta(q, \varepsilon) = \begin{cases} \{q_2\} & \text{if } q \in F_1 \\ \emptyset & \text{otherwise} \end{cases}$$

$$\delta(q, t) = \begin{cases} \{\delta_1(q, t)\} & \text{for } q \in Q_1 \\ \{\delta_2(q, t)\} & \text{for } q \in Q_2. \end{cases}$$

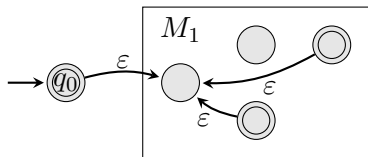
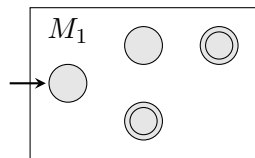
# Kleene Star

Let  $A$  be a regular language recognized by DFA  $M_1$



# Kleene Star

Let  $A$  be a regular language recognized by DFA  $M_1$



Let  $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ .

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

$$Q = Q_1 \cup \{q_0\}$$

$$F = F_1 \cup \{q_0\}$$

$$\delta(q, \epsilon) = \begin{cases} \{q_1\} & \text{if } q \in F \\ \emptyset & \text{otherwise} \end{cases}$$

$$\delta(q, t) = \begin{cases} \emptyset & \text{if } q = q_0 \\ \{\delta_1(q, t)\} & \text{for } q \in Q_1 \end{cases}$$



## Let's build some NFAs!

- $A = \{w \mid w \text{ starts with a and ends with b}\}$
- $B = \emptyset$
- $C = \{\varepsilon\}$
- $D = \{w \mid w \text{ has an even number of a's or exactly 2 b's}\}$
- $E = \{aa, aba, bab, bbb\}$