CS 383

Lecture 03 – Nondeterministic Finite Automata (NFAs)

Stephen Checkoway

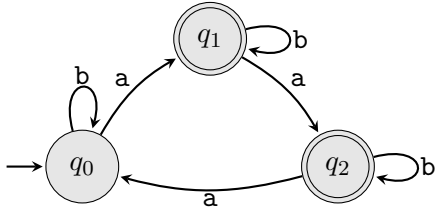
Fall 2023

Review from last time

DFAs are 5-tuples $M = (Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is an alphabet (finite, nonempty set of symbols)
- $\delta : Q \times \Sigma \to Q$ 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: $A^* = \{w_1 w_2 \cdots w_k \mid k \geq 0 \text{ and } w_i \in A \text{ for all } i\}$
- $\textsc{EndsWith}(A) = \{xw \mid x \in \Sigma^* \text{ and } w \in A\}$
- ...

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: $A^* = \{w_1 w_2 \cdots w_k \mid k \geq 0 \text{ and } w_i \in A \text{ for all } i\}$
- $\textsc{EndsWith}(A) = \{xw \mid x \in \Sigma^* \text{ and } w \in A\}$
- ...

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.

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, $F' = Q \setminus F$.

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. □

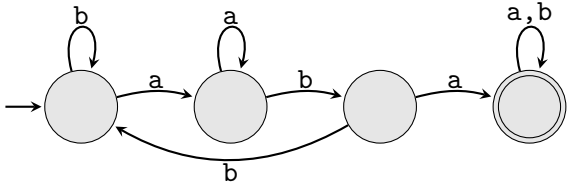
Complement example

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

Complement example

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

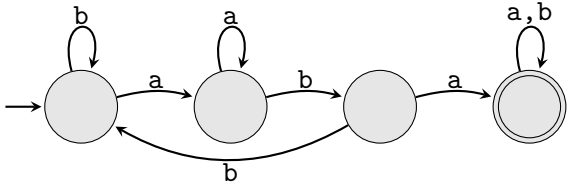
M:



Complement example

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

M:

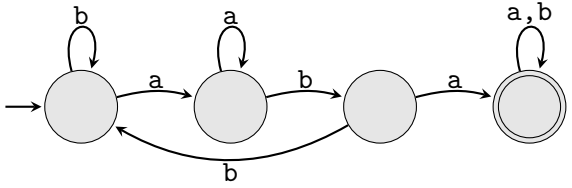


$\overline{A}$ = $\{w \mid$ aba is *not* a substring of $w\}$

Complement example

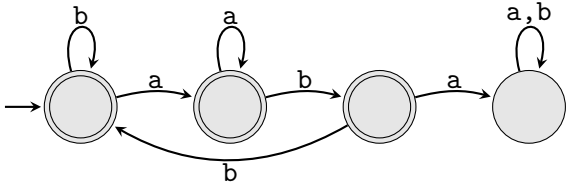
Let A = $\{w \mid$ aba is a substring of $w\}$

M:



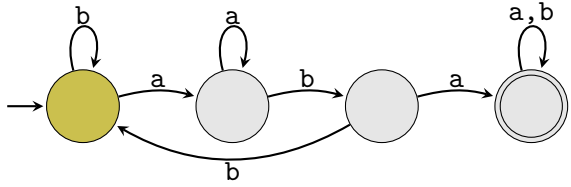
$\overline{A}$ = $\{w \mid$ aba is *not* a substring of $w\}$

M':

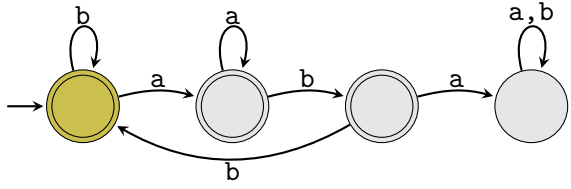


Complement example

M:



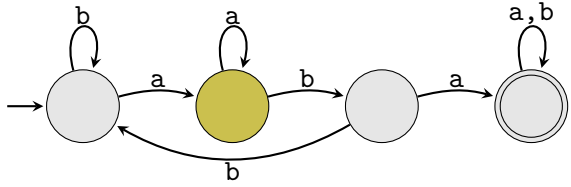
M':



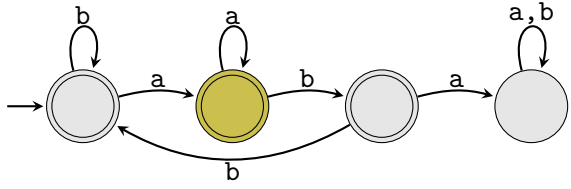
abbaabab

Complement example

M:



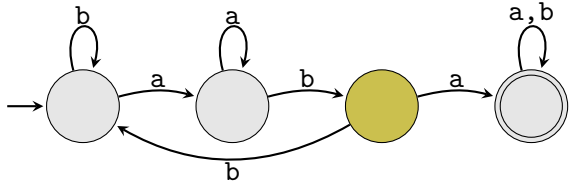
M':



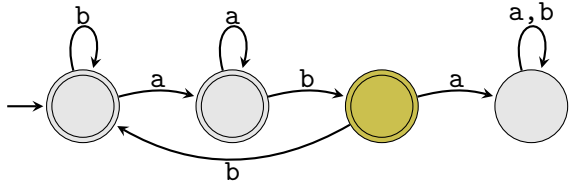
abbaabab

Complement example

M:



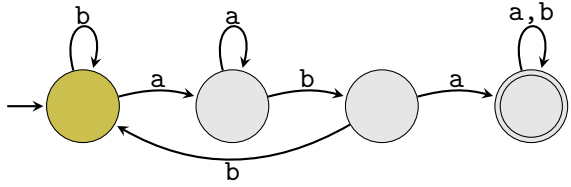
M':



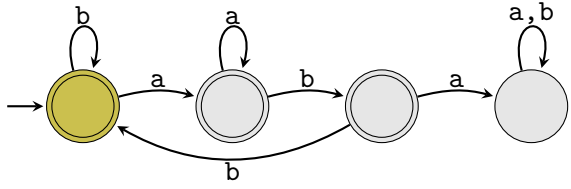
abbaabab

Complement example

M:



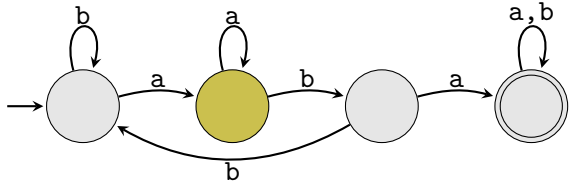
M':



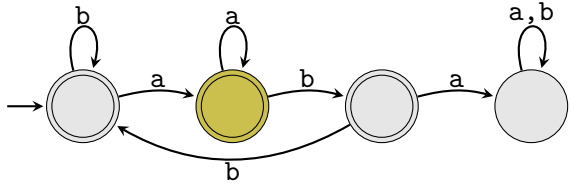
abbaabab

Complement example

M:



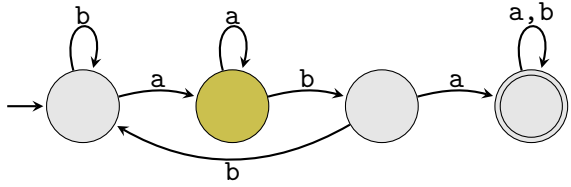
M':



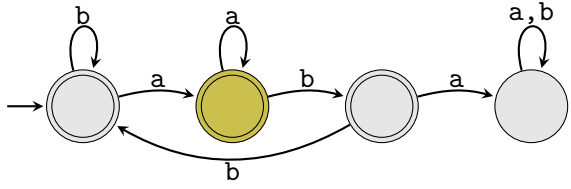
abbaabab

Complement example

M:



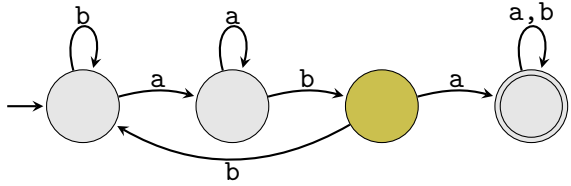
M':



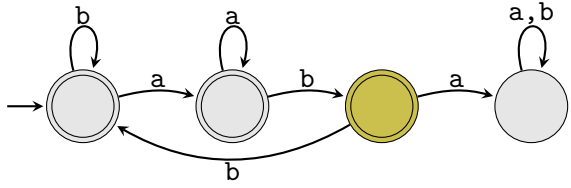
abbaabab

Complement example

M:



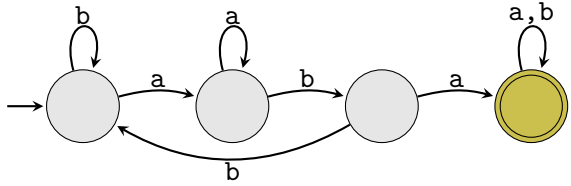
M':



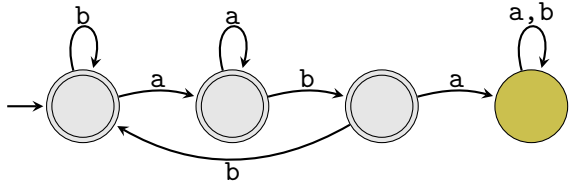
abbaabab

Complement example

M:



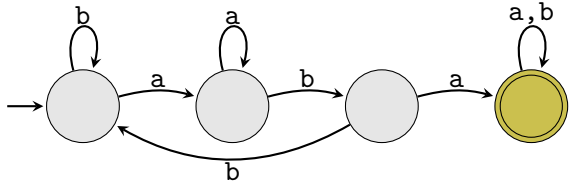
M':



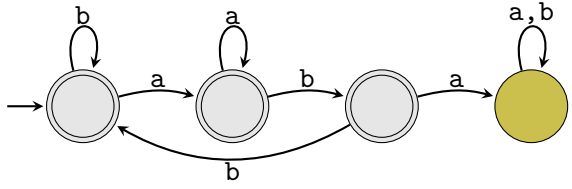
abbaabab

Complement example

M:



M':



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,

$$
\begin{aligned}
Q &= Q_1 \times Q_2 \\
q_0 &= (q_1, q_2) \\
\delta\big((q, r), t\big) &= \big(\delta_1(q, t),\ \delta_2(r, t)\big) \\
F &= \{(q, r) \mid q \in F_1 \text{ or } r \in F_2\}.
\end{aligned}
$$

As M transitions from state (q, r) to state (q', r'), the first element changes according to δ_1 and the second according to δ_2.

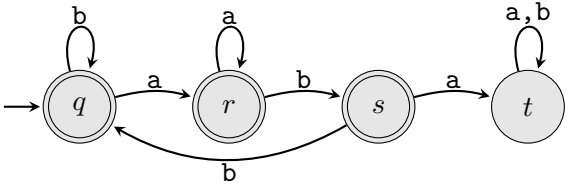
Union

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. □

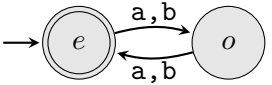
Union example

Let A = $\{w \mid$ aba is not a substring of $w\}$ and M_1 recognize A
Let B = $\{w \mid |w|$ is even$\}$ and M_2 recognize B

M_1:



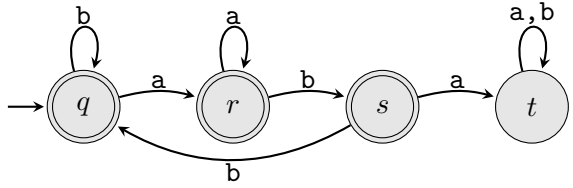
M_2:



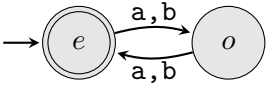
Union example

Let A = {w | aba is not a substring of w} and M_1 recognize A
Let B = {w | $|w|$ is even} and M_2 recognize B

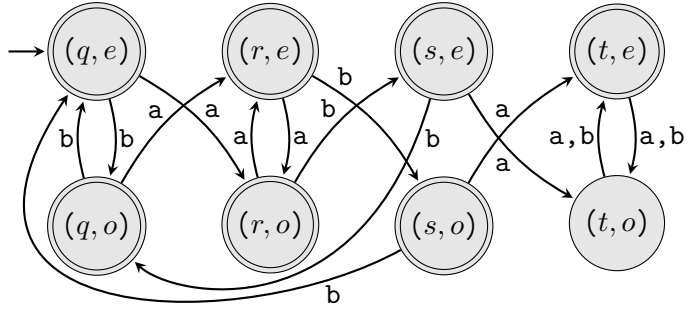
M_1:



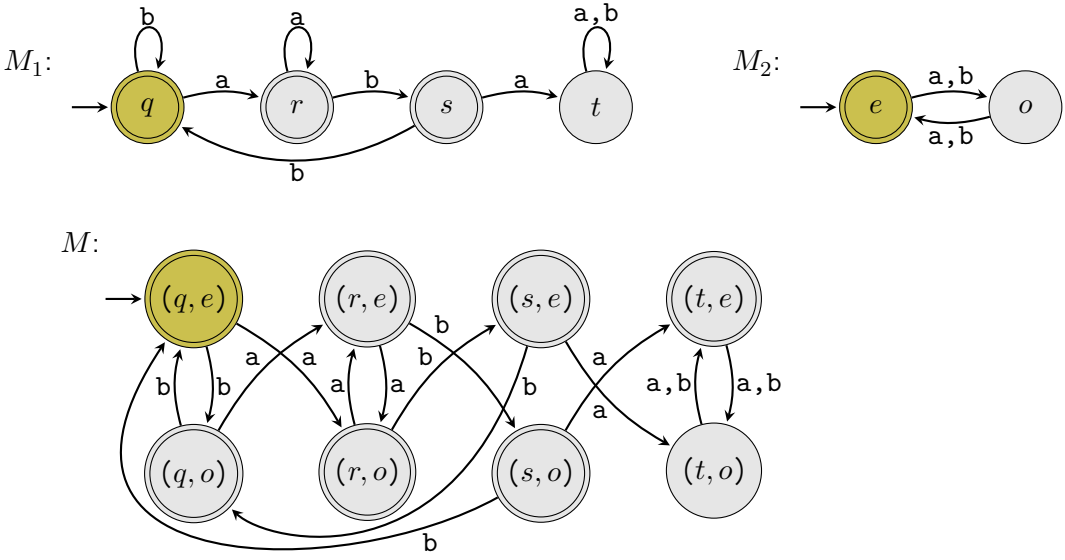
M_2:



M:

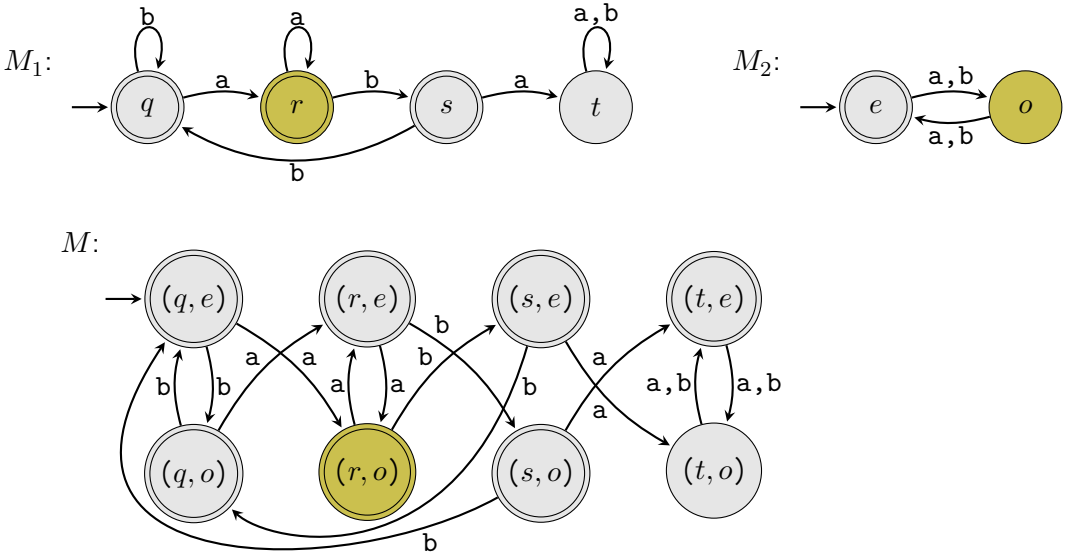


Union example



aabbaba

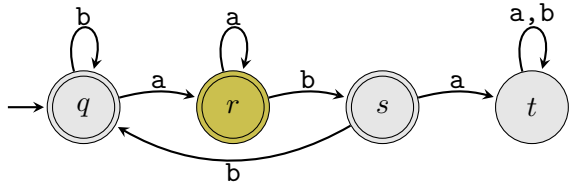
Union example



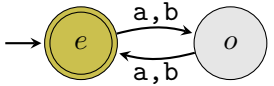
aabbaba

Union example

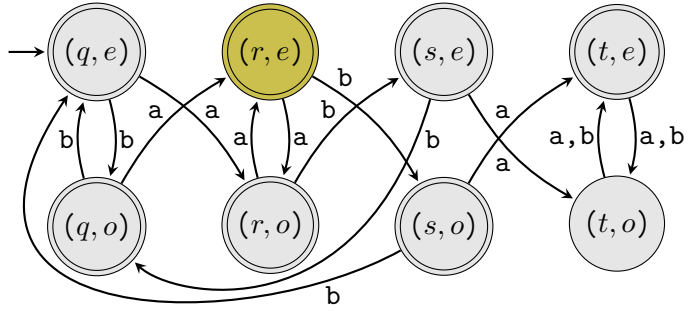
M_1:



M_2:

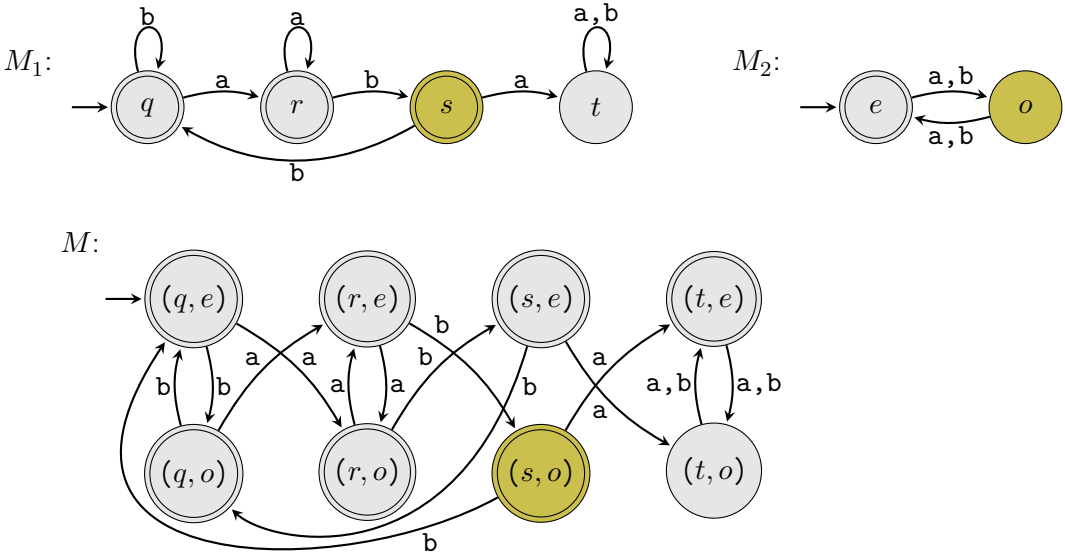


M:



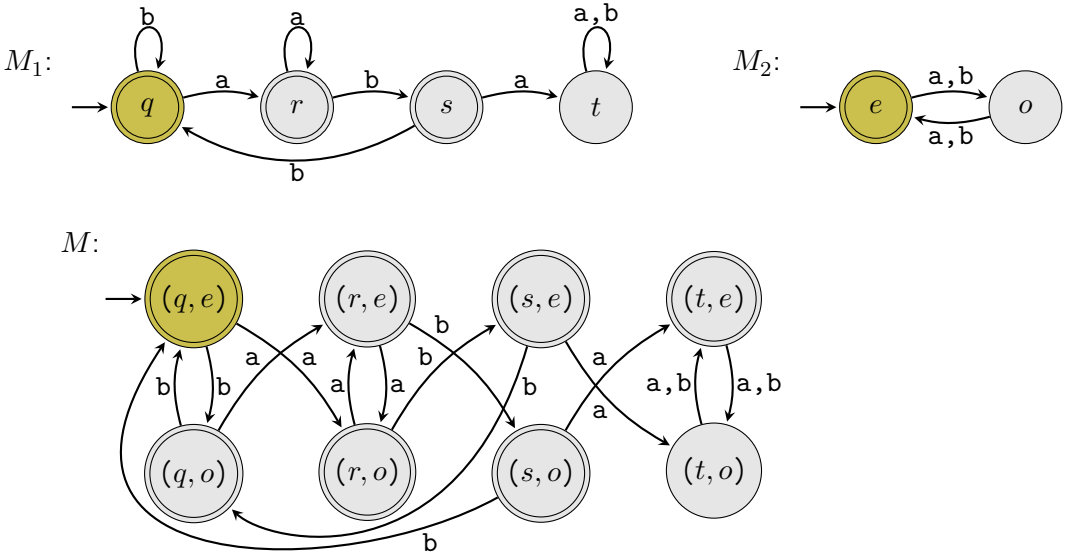
aabbbaba

Union example



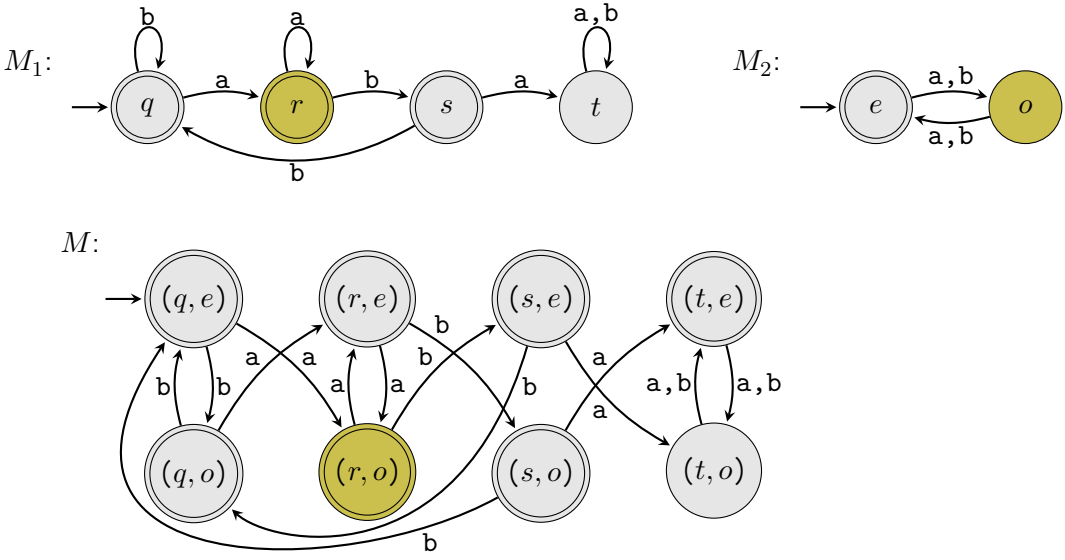
aabbaba

Union example



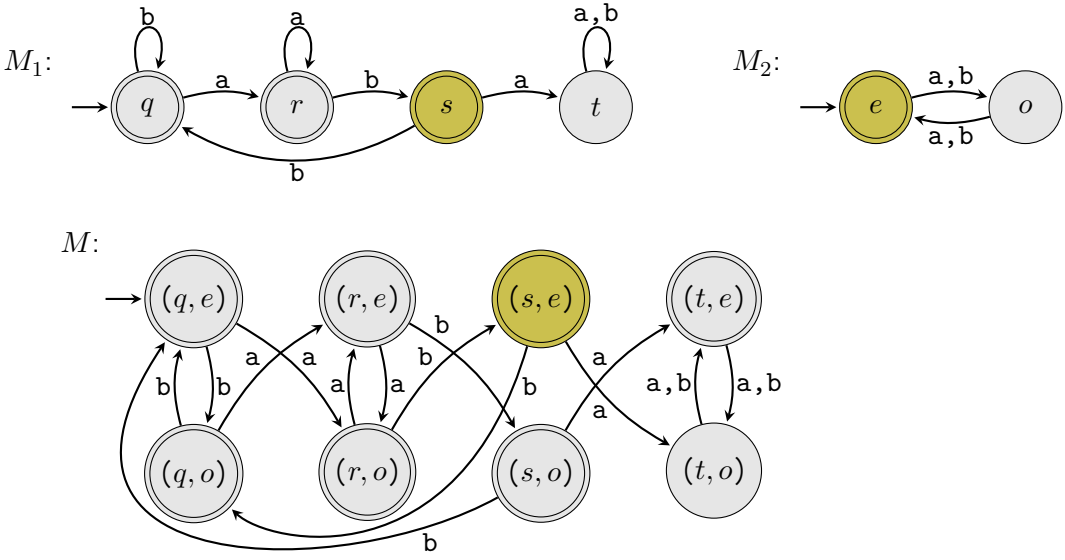
aabbaba

Union example



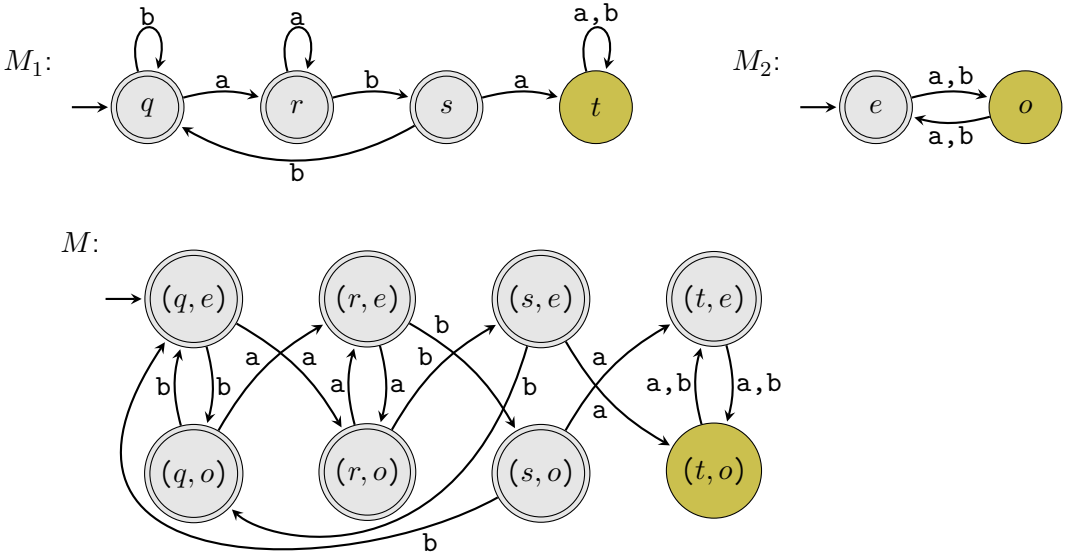
aabbabba

Union example



aabbaba

Union example



aabbaba ✖Rejected

EndsWith

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

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

A simple theorem

Theorem

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

Ideally, this new DFA would have two parts:

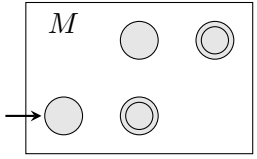
1. some states that read symbols from Σ^* (i.e., matching the symbols of x)
2. a copy of M to accept the last part of the string which should be in A

A simple theorem proof difficulty

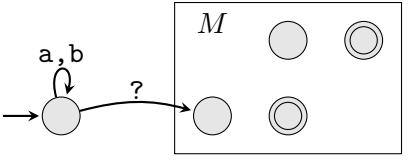
The two parts are individually easy

1. Match symbols from Σ^* (assume $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, 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

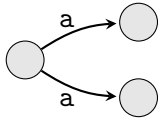
Similarly, we don't allow a state to not have a transition labeled with a symbol of Σ

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

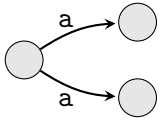
1. Multiple transitions from a state on the same symbol

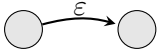


Nondeterminism

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

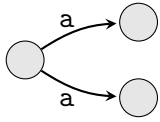
1. Multiple transitions from a state on the same symbol 

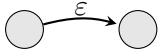
2. Transitions on no input 

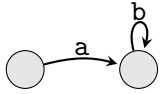
Nondeterminism

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

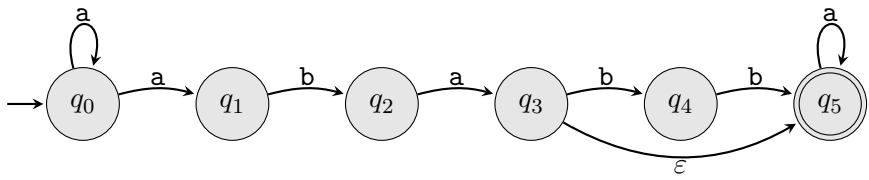
Three new options

1. Multiple transitions from a state on the same symbol 

2. Transitions on no input 

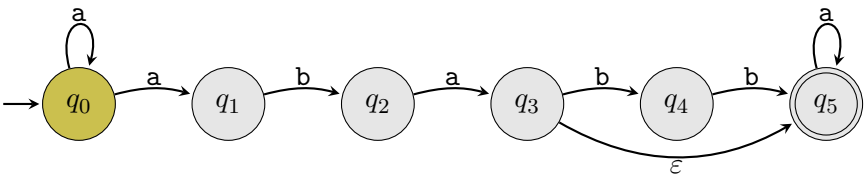
3. States without transitions on some (or all) symbols 

Example



Let's run this on input ababb

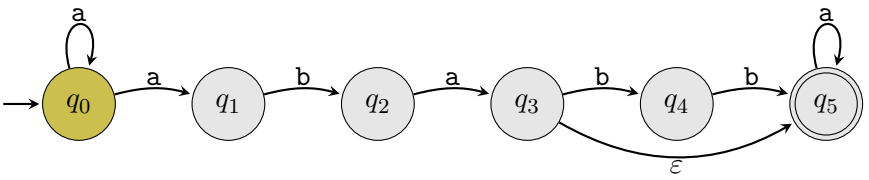
Example



Let's run this on input ababb

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

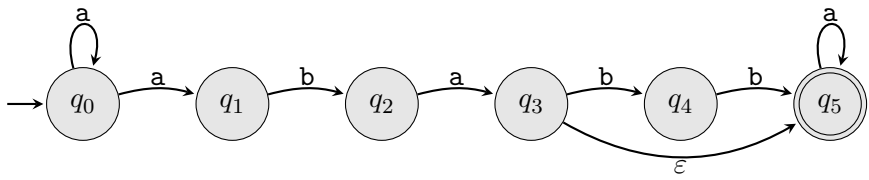
Example



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

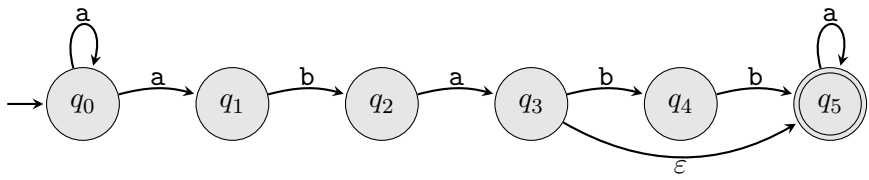


Let's run this on input ababb

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

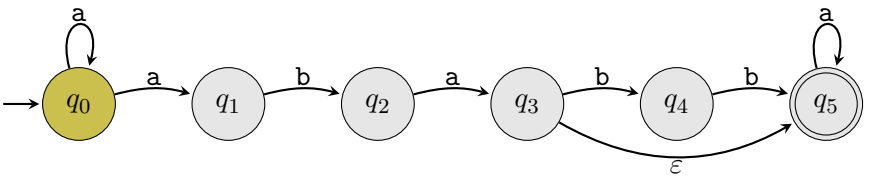
Since the machine didn't end in an accepting state. Is ababb ✖Rejected?

Example again



Let's run this on input ababb again

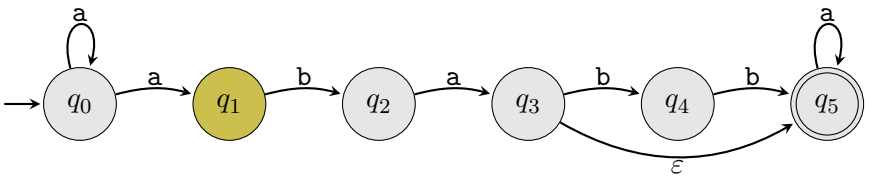
Example again



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

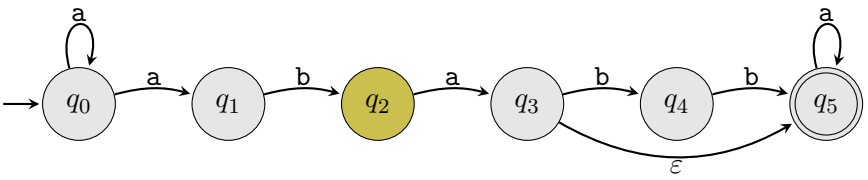
Example again



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

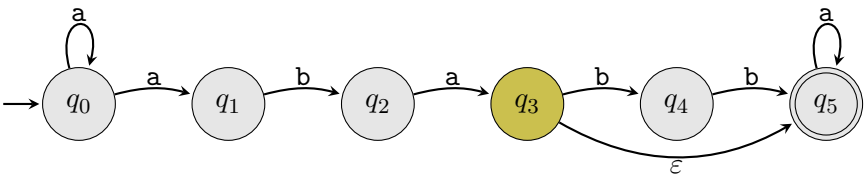
Example again



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

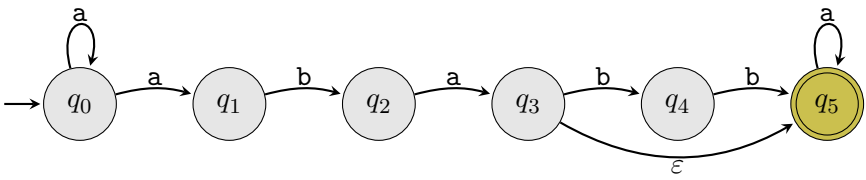
Example again



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 ε transition or not, let's follow it

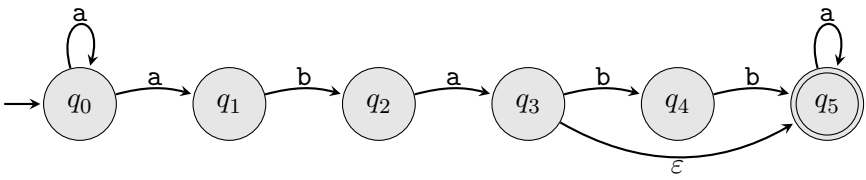
Example again



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 ε transition or not, let's follow it
5. Next symbol is b, but there are no transitions labeled b

Example again

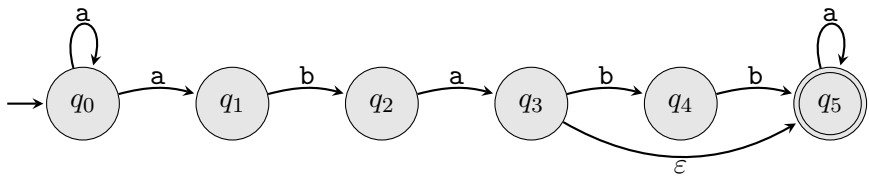


Let's run this on input ababb again

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 ε 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

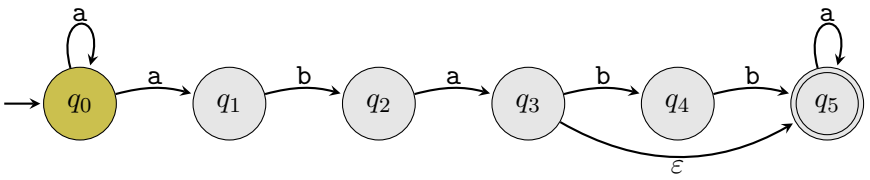
Once again, it didn't end in an accepting state.

Example yet again



Let's run this on input ababb a third time

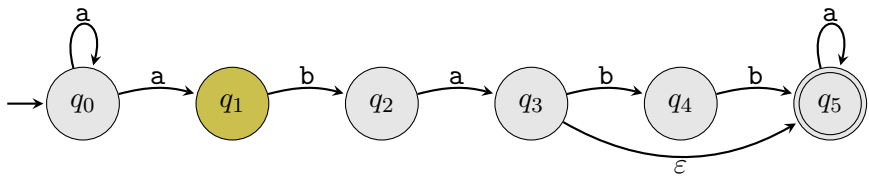
Example yet again



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

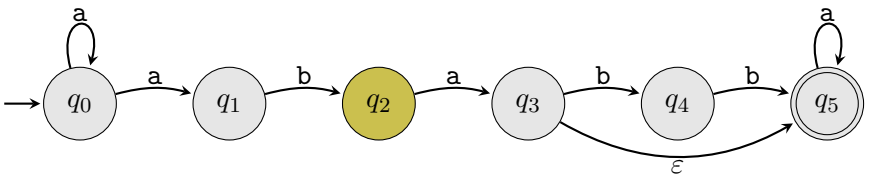
Example yet again



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

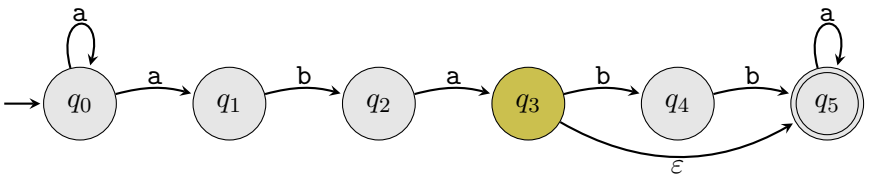
Example yet again



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

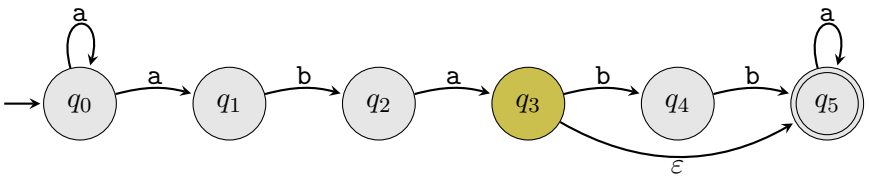
Example yet again



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 ε transition or not, let's *not* follow it

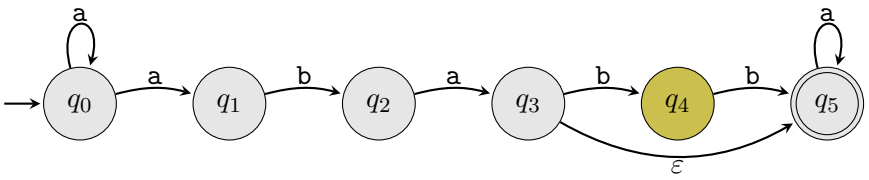
Example yet again



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 ε transition or not, let's *not* follow it
5. Next symbol is b, go to q_4

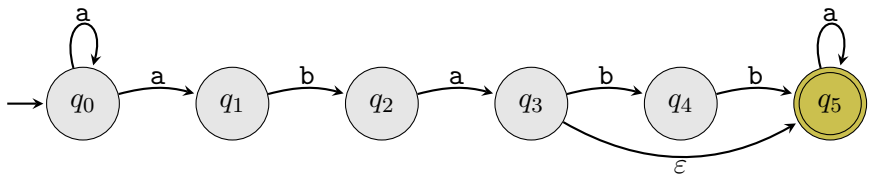
Example yet again



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 ε 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 ε 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

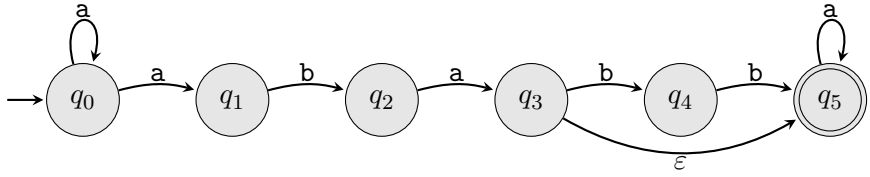
The third choice we made ended in an accepting state

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

So ababb was ✔Accepted

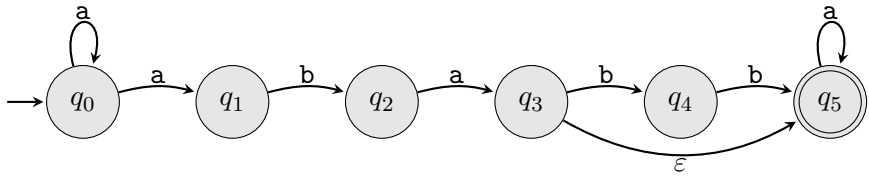
Language of the NFA

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: $\{\mathtt{a}^m\mathtt{ba}w a^n \mid m \geq 1 \text{ and } n \geq 0 \text{ and } w \in \{\varepsilon, \mathtt{bb}\}\}$

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

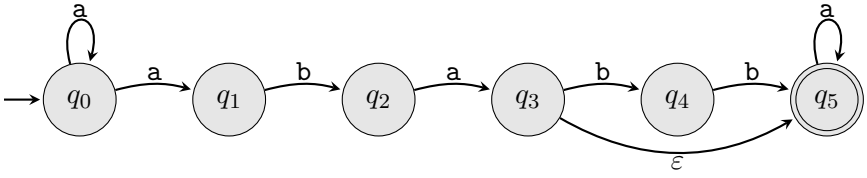
Procedure

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

Running our NFAs

Procedure

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

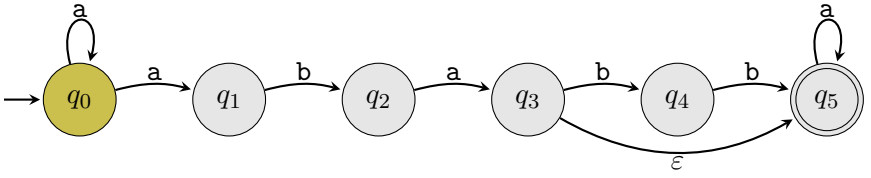


ababb

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

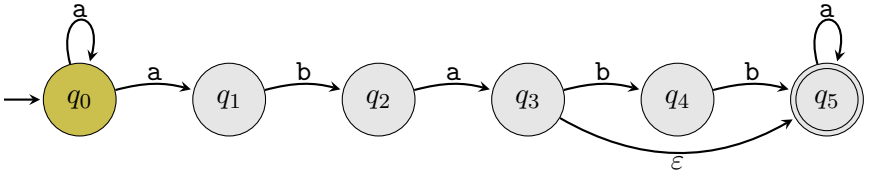


ababb

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

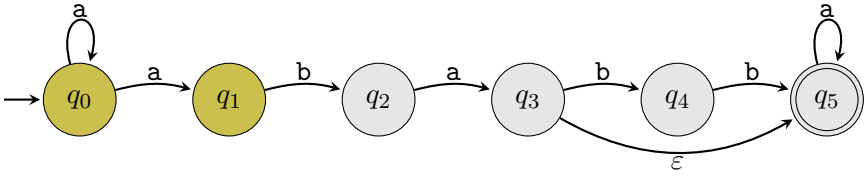


ababb

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

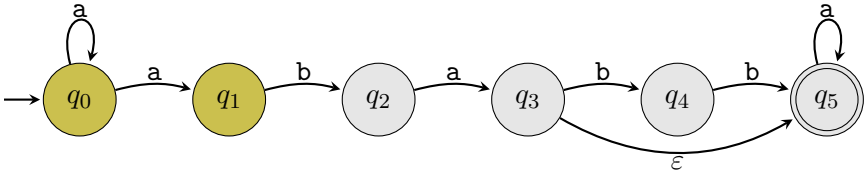


ababb

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

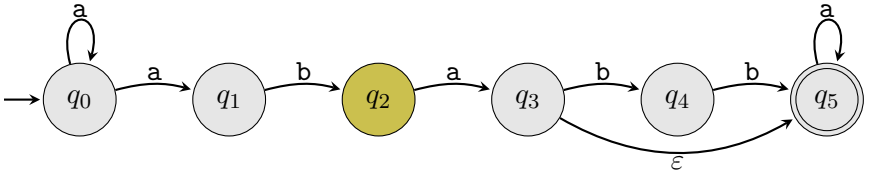


ababb

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

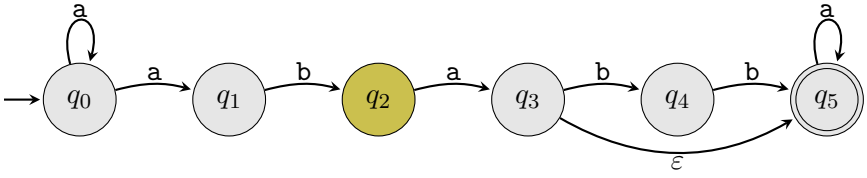


ababb

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

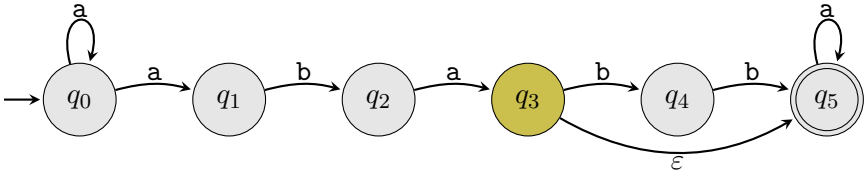


ababb

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

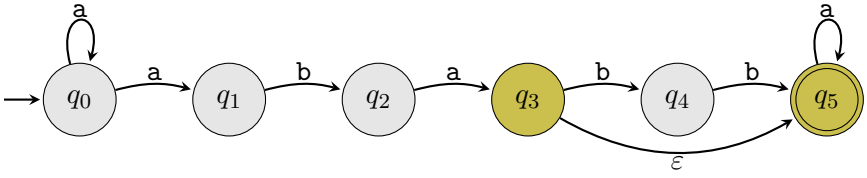


ababb

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

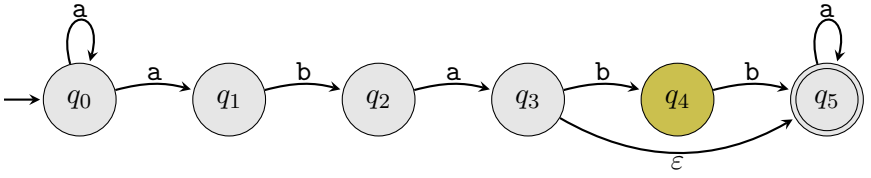


ababb

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

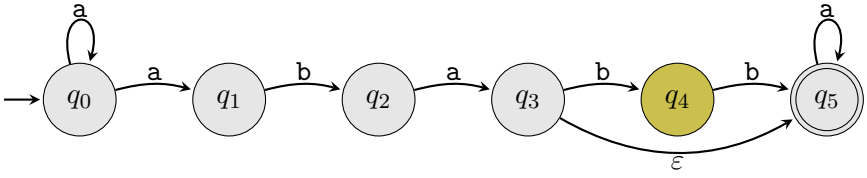


ababb

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

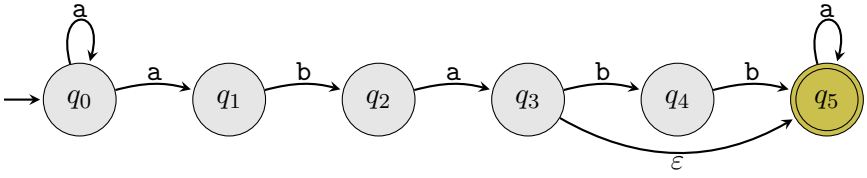


ababb

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

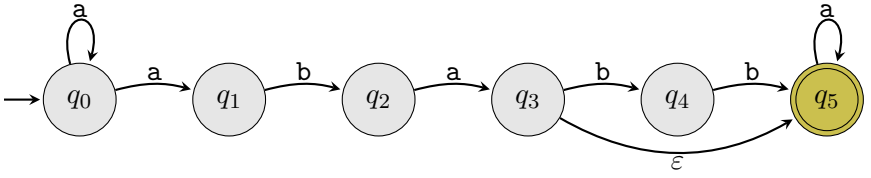


ababb

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

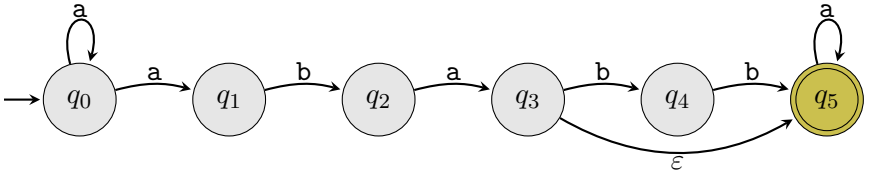


ababb

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
- Σ is an alphabet
- $\delta : Q \times \Sigma_\varepsilon \to P(Q)$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accepting (or final) states

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

$P(Q)$ is the power set of Q so δ returns a set of next states

Transition functions

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

For each (state, symbol) pair, δ returns a single state

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

For each (state, symbol) pair, δ returns 0 or more states

For each (state, ε), δ returns 0 or more states

Formalizing NFA computation

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and let $w = w_1 w_2 \cdots w_n$ be a string where $w_i \in \Sigma_\varepsilon$

N accepts w if there exist states $r_0, r_1, \ldots, r_n \in Q$ 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 $i \in \{1, 2, \ldots, n\}$
   [The NFA moves from state r_{i-1} to one of the possible next states according to δ]
3. $r_n \in F$
   [The NFA ends in an accepting state]

Formalizing NFA computation

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and let $w = w_1 w_2 \cdots w_n$ be a string where $w_i \in \Sigma_\varepsilon$

N accepts w if there exist states $r_0, r_1, \ldots, r_n \in Q$ 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 $i \in \{1, 2, \ldots, n\}$
   [The NFA moves from state r_{i-1} to one of the possible next states according to δ]
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 ε
   E.g., if w = abaa, then we can write $w = \varepsilon\mathtt{ab}\varepsilon\varepsilon\varepsilon\mathtt{a}\varepsilon\mathtt{a}$
2. $r_i \in \delta(r_{i-1}, w_i)$ since δ returns a set of next possible states

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

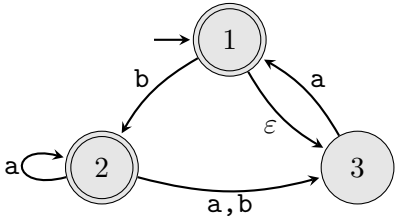
Language of an NFA

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

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

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

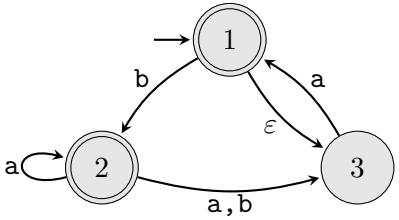
$q_0 = 1$

$F = \{1, 2\}$

δ :

	a	b	ε
1	$\varnothing$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\varnothing$
3	$\{1\}$	$\varnothing$	$\varnothing$

Example



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

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

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

$q_0 = 1$

$F = \{1, 2\}$

δ :

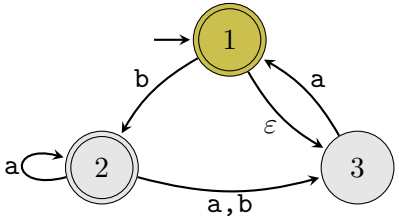
	a	b	ε
1	$\varnothing$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\varnothing$
3	$\{1\}$	$\varnothing$	$\varnothing$

Consider string w = abaa

Write w as εabaa 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

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

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

$q_0 = 1$

$F = \{1, 2\}$

δ :

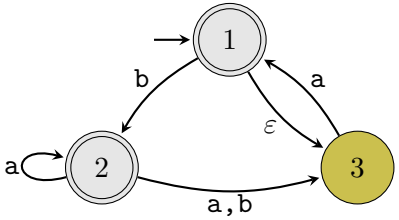
	a	b	ε
1	$\varnothing$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\varnothing$
3	$\{1\}$	$\varnothing$	$\varnothing$

Consider string w = abaa

Write w as εabaa 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					

Example



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

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

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

$q_0 = 1$

$F = \{1, 2\}$

δ :

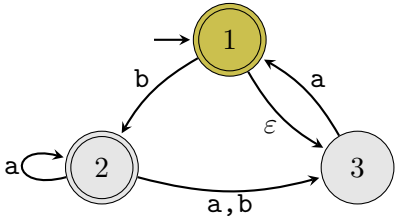
	a	b	ε
1	$\varnothing$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\varnothing$
3	$\{1\}$	$\varnothing$	$\varnothing$

Consider string w = abaa

Write w as εabaa 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

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

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

$q_0 = 1$

$F = \{1, 2\}$

δ :

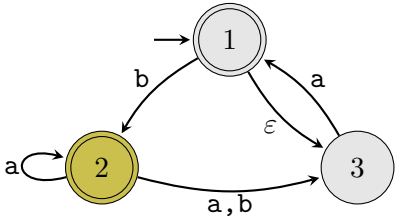
	a	b	ε
1	$\varnothing$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\varnothing$
3	$\{1\}$	$\varnothing$	$\varnothing$

Consider string w = abaa

Write w as εabaa 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

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

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

$q_0 = 1$

$F = \{1, 2\}$

δ :

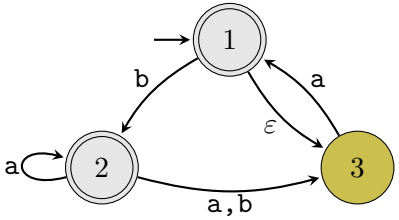
	a	b	ε
1	$\varnothing$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\varnothing$
3	$\{1\}$	$\varnothing$	$\varnothing$

Consider string w = abaa

Write w as εabaa 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

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

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

$q_0 = 1$

$F = \{1, 2\}$

δ :

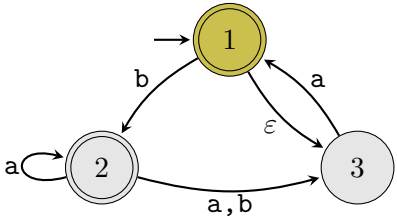
	a	b	ε
1	$\varnothing$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\varnothing$
3	$\{1\}$	$\varnothing$	$\varnothing$

Consider string w = abaa

Write w as εabaa 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

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

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

$q_0 = 1$

$F = \{1, 2\}$

δ :

	a	b	ε
1	$\varnothing$	$\{2\}$	$\{3\}$
2	$\{2, 3\}$	$\{3\}$	$\varnothing$
3	$\{1\}$	$\varnothing$	$\varnothing$

Consider string w = abaa

Write w as εabaa 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 $i \in \{1, 2, \ldots, n\}$
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 ε-transitions from states in S

Formally, $E : P(Q) \to P(Q)$ given by
$E(S) = \{q \mid q$ is reachable from some $r \in S$ by following 0 or more ε-transitions$\}$

$E(S)$ is called the ε-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 ε-transitions from states in S

Formally, $E : P(Q) \to P(Q)$ given by
$E(S) = \{q \mid q$ is reachable from some $r \in S$ by following 0 or more ε-transitions$\}$

$E(S)$ is called the ε-closure of S

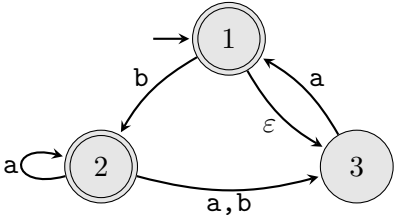
Procedure (ver. 2)

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

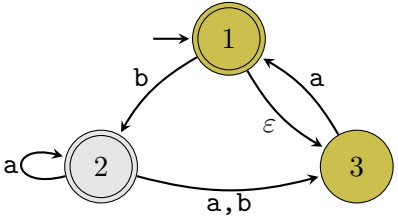


- `abaabba`

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

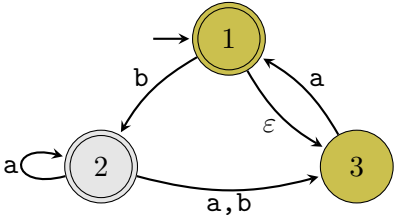


- abaabba

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

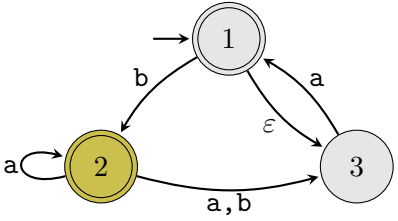


- abaabba

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

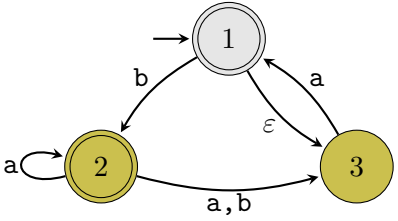


- abaabba

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

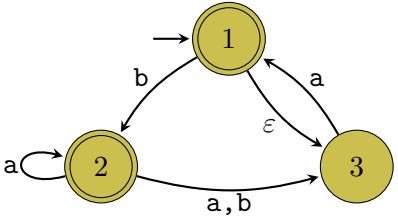


- abaabba

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

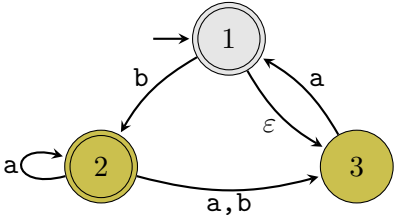


- abaabba

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

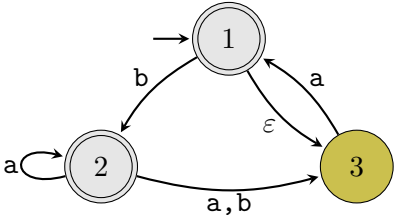


- abaabba

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

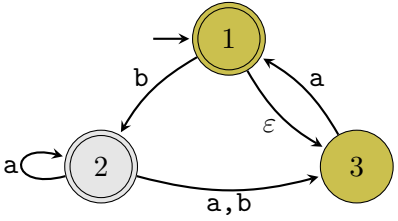


- abaabba

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 $C = \{q \mid q \in E(\delta(r, t))$ for some $r \in C\}$
4. If $C \cap F \neq \varnothing$, N accepts w, otherwise N rejects w

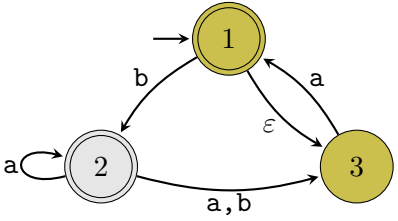


- abaabba

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

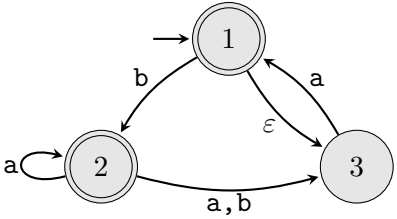


- abaabba ✔Accepted

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

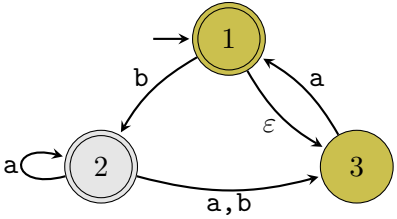


- `abaabba` ✔Accepted
- `bbbab`

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

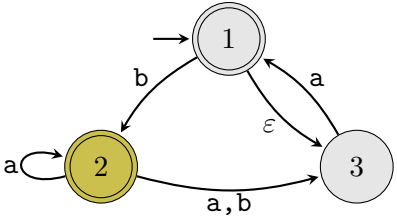


- abaabba ✔Accepted
- bbbab

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

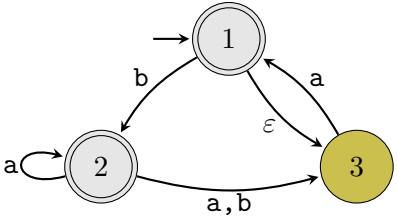


- `abaabba` ✔Accepted
- `bbbab`

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

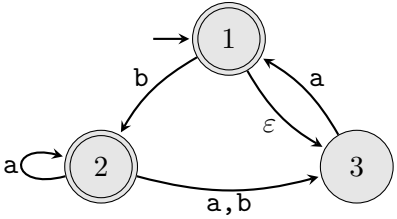


- abaabba ✔Accepted
- bbbab

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

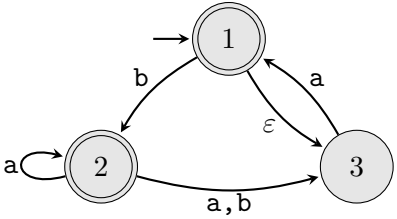


- abaabba ✔Accepted
- bbbab

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

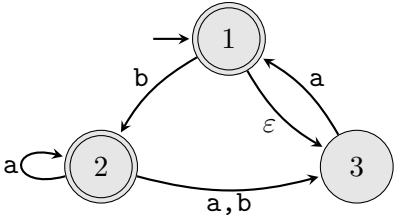


- abaabba ✔Accepted
- bbbab

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 $C \cap F \neq \varnothing$, N accepts w, otherwise N rejects w

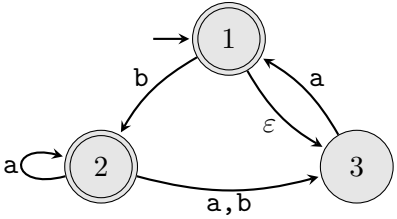


- `abaabba` ✔Accepted
- `bbbab`

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

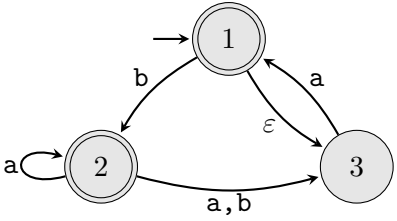


- `abaabba` ✔Accepted
- `bbbab` ✖Rejected

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 $C \cap F \neq \varnothing$, N accepts w, otherwise N rejects w

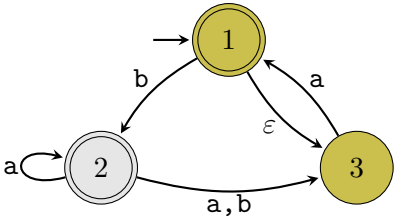


- `abaabba` ✔Accepted
- `bbbab` ✖Rejected
- `bb`

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 $C \cap F \neq \varnothing$, N accepts w, otherwise N rejects w

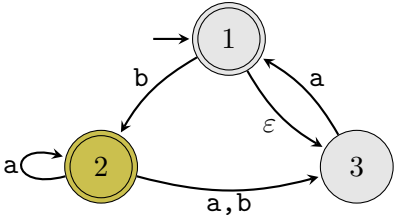


- `abaabba` ✔Accepted
- `bbbab` ✖Rejected
- `bb`

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 $C \cap F \neq \varnothing$, N accepts w, otherwise N rejects w

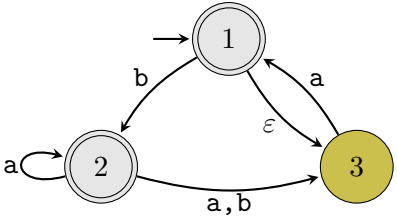


- abaabba ✔Accepted
- bbbab ✖Rejected
- bb

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 $C \cap F \neq \varnothing$, N accepts w, otherwise N rejects w

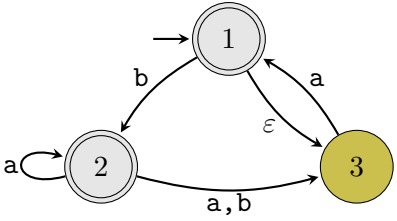


- `abaabba` ✔Accepted
- `bbbab` ✖Rejected
- `bb`

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 $C \cap F \neq \varnothing$, 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 $C \cap F \neq \varnothing$, 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 $C \cap F \neq \varnothing$, 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 $C \cap F \neq \varnothing$, 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 $C \cap F \neq \varnothing$, 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 $\delta' : P(Q) \times \Sigma \to P(Q)$ 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)

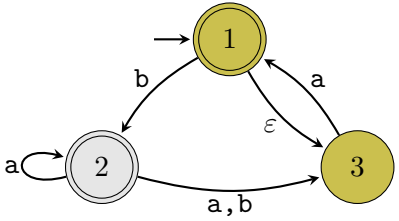
1. Set $C = E(\{q_0\})$
2. For each successive symbol t in the input w,
3. Set $C = \{q \mid q \in E(\delta(r,t))$ for some $r \in C\}$
4. If $C \cap F \neq \varnothing$, N accepts w, otherwise N rejects w

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 $\delta' : P(Q) \times \Sigma \to P(Q)$ is $\delta'(C,t) = \{q \mid q \in E(\delta(r,t))$ for some $r \in C\}$
- M's accepting states are every subset of Q that contains at least one of N's accepting states: $F' = \{S \mid S \subseteq Q$ and $S \cap F \neq \varnothing\}$

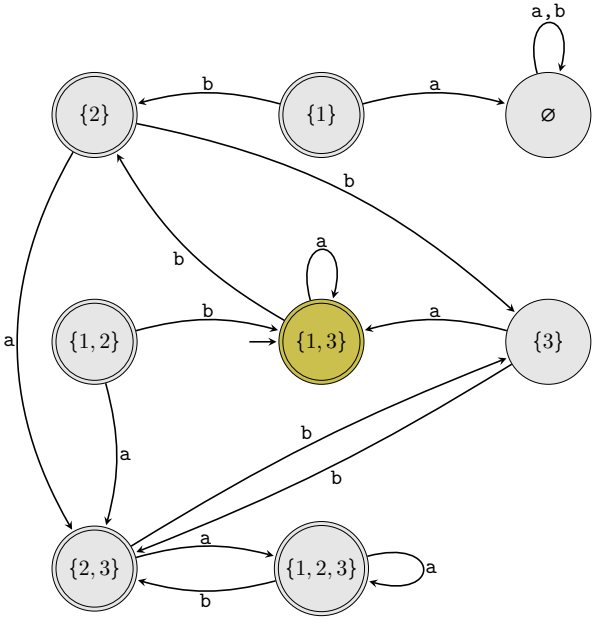
Converting our example to a DFA

N:



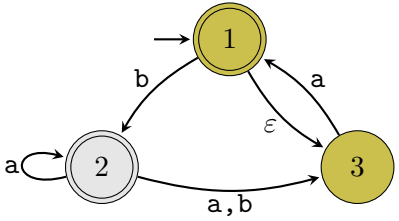
abaababb

M:



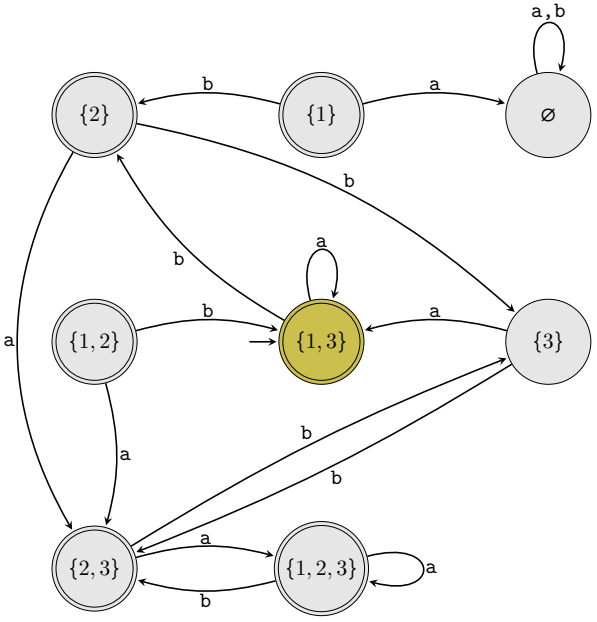
Converting our example to a DFA

N:



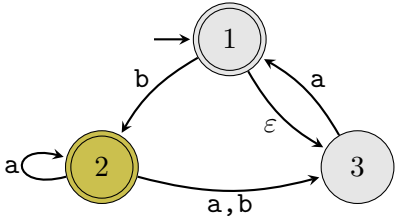
abaababb

M:



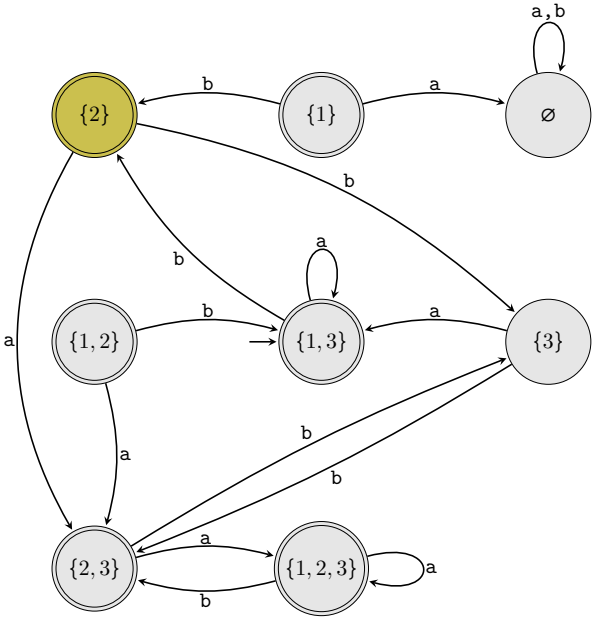
Converting our example to a DFA

N:



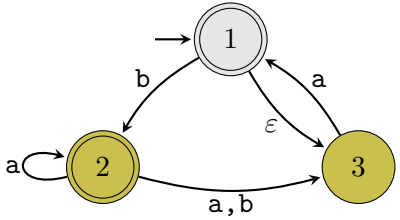
abaababb

M:



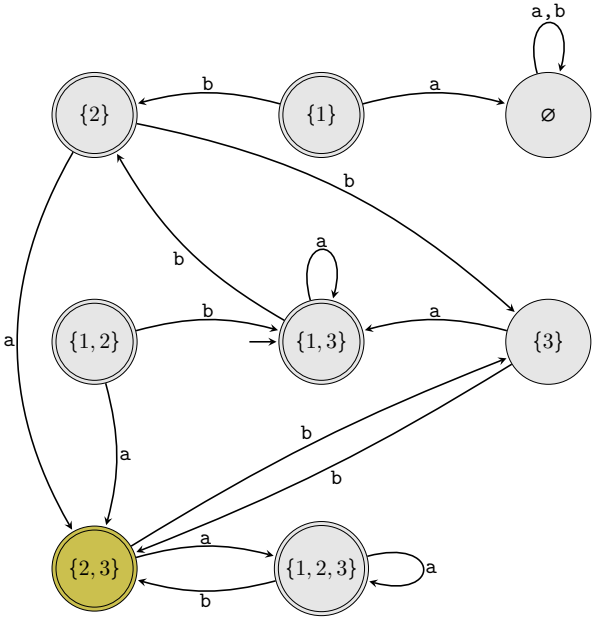
Converting our example to a DFA

N:



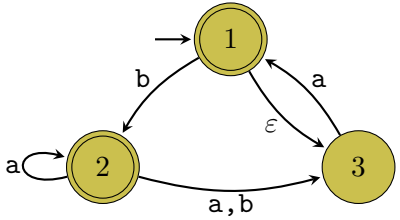
abaababb

M:



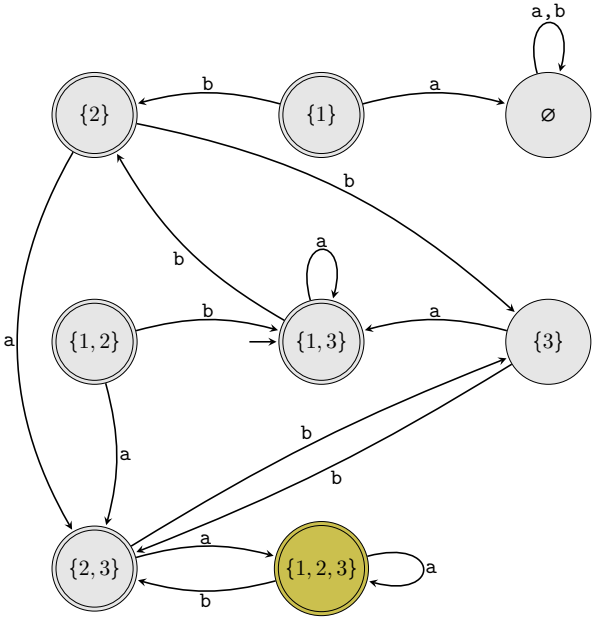
Converting our example to a DFA

N:



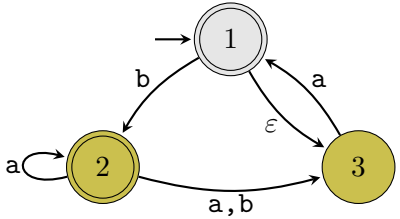
abaabbabb

M:



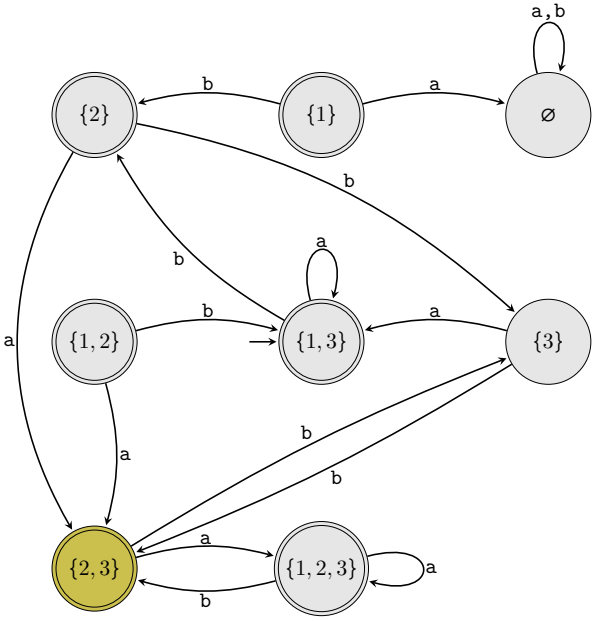
Converting our example to a DFA

N:



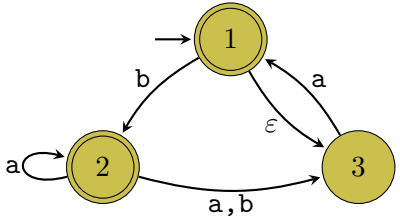
abaababb

M:



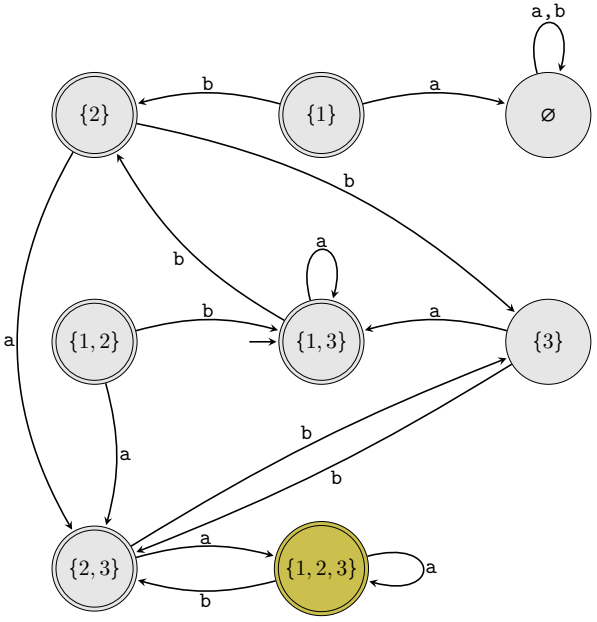
Converting our example to a DFA

N:



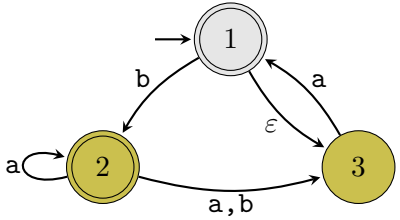
abaababb

M:



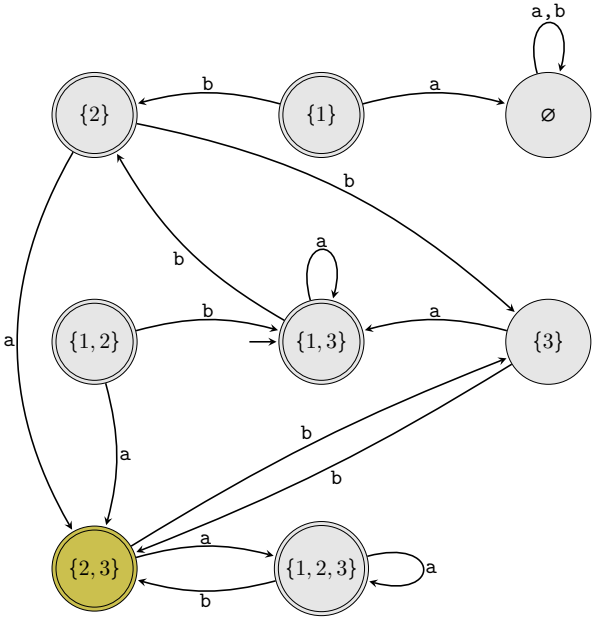
Converting our example to a DFA

N:



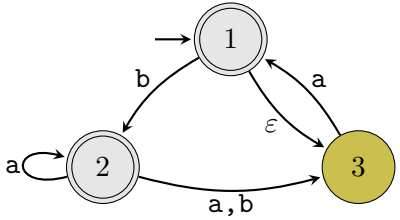
abaababb

M:



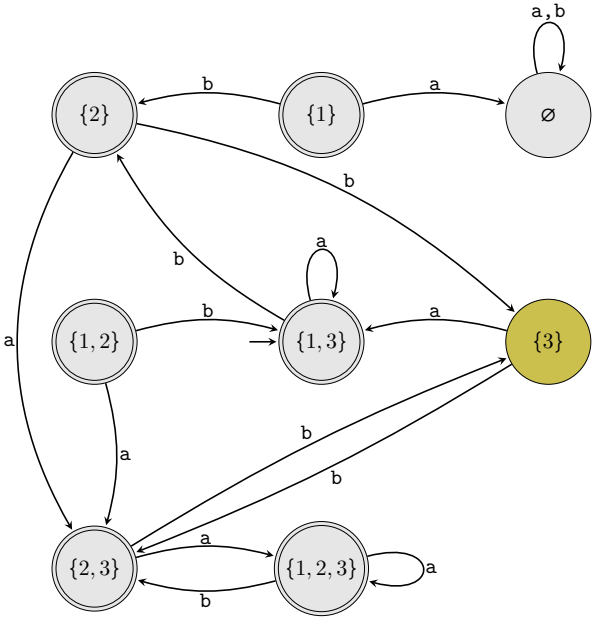
Converting our example to a DFA

N:



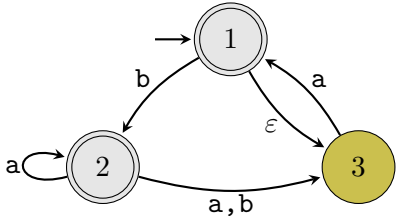
abaababb

M:



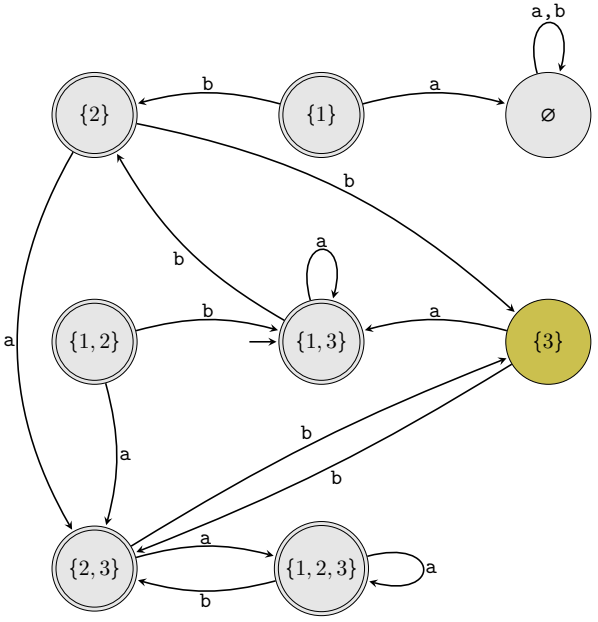
Converting our example to a DFA

N:



abaababb ✖Rejected

M:



Regular languages

Theorem

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

Proof.

$\Longrightarrow$

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.

$\Longleftarrow$

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 $A_1, A_2, \ldots, A_n$ are regular, then $f(A_1, A_2, \ldots, A_n)$ 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 $A^* = \{w_1 w_2 \cdots w_k \mid k \geq 0 \text{ and } w_i \in A \text{ for all } i\}$

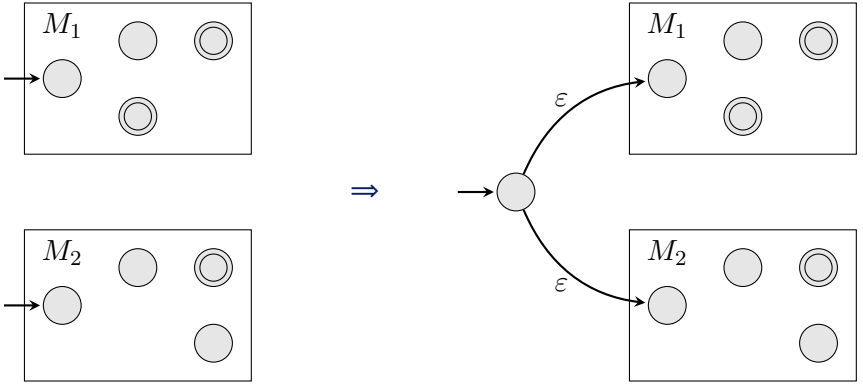
Theorem

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

$$M_1 = (Q_1, \Sigma, \delta, q_1, F_1)$$
$$M_2 = (Q_2, \Sigma, \delta, q_2, F_2).$$

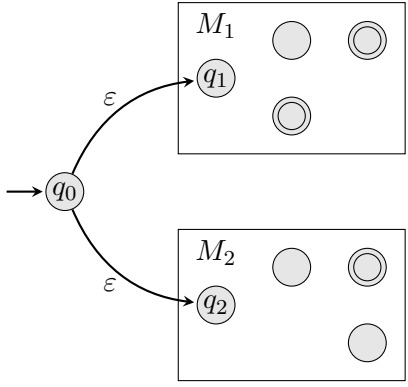
Build NFA $N = (Q, \Sigma, \delta, q_0, F)$ where

$$Q = Q_1 \cup Q_2 \cup \{q_0\}$$
$$F = F_1 \cup F_2$$

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

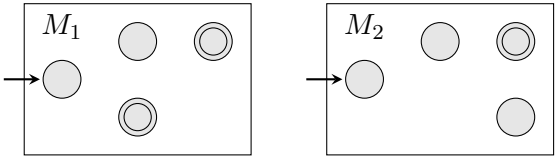
$$\delta(q, t) = \begin{cases} \varnothing & \text{if } q = q_0 \\ \{\delta_1(q, t)\} & \text{for } q \in Q_1 \\ \{\delta_2(q, t)\} & \text{for } q \in Q_2 \end{cases}$$

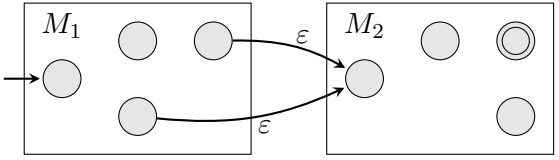
□



Concatenation

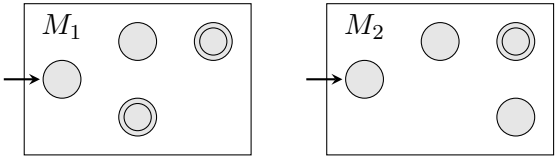
Let A and B be regular languages recognized by DFAs M_1 and M_2



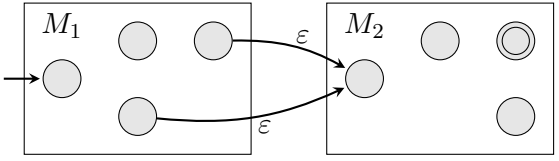


Concatenation

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



$\Downarrow$



Let

$$M_1 = (Q_1, \Sigma, \delta, q_1, F_1)$$
$$M_2 = (Q_2, \Sigma, \delta, q_2, F_2).$$

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

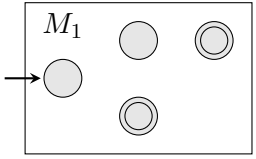
$$Q = Q_1 \cup Q_2$$

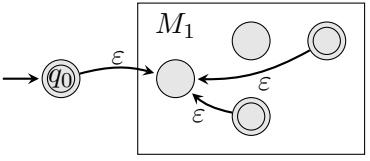
$$\delta(q, \varepsilon) = \begin{cases} \{q_2\} & \text{if } q \in F_1 \\ \varnothing & \text{otherwise} \end{cases}$$

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

Kleene Star

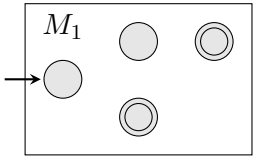
Let A be a regular language recognized by DFA M_1

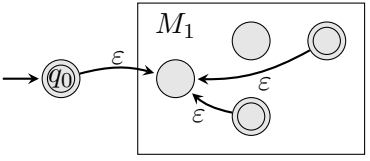




Kleene Star

Let A be a regular language recognized by DFA M_1





Let $M_1 = (Q_1, \Sigma, \delta, q_1, F_1)$.
Build NFA $N = (Q, \Sigma, \delta, q_0, F)$ where

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

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

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

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

Let's build some NFAs!

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