CS 483 - Introduction to the Theory of Computation

Lecture 3

From Regex to DFA: Concatenation?

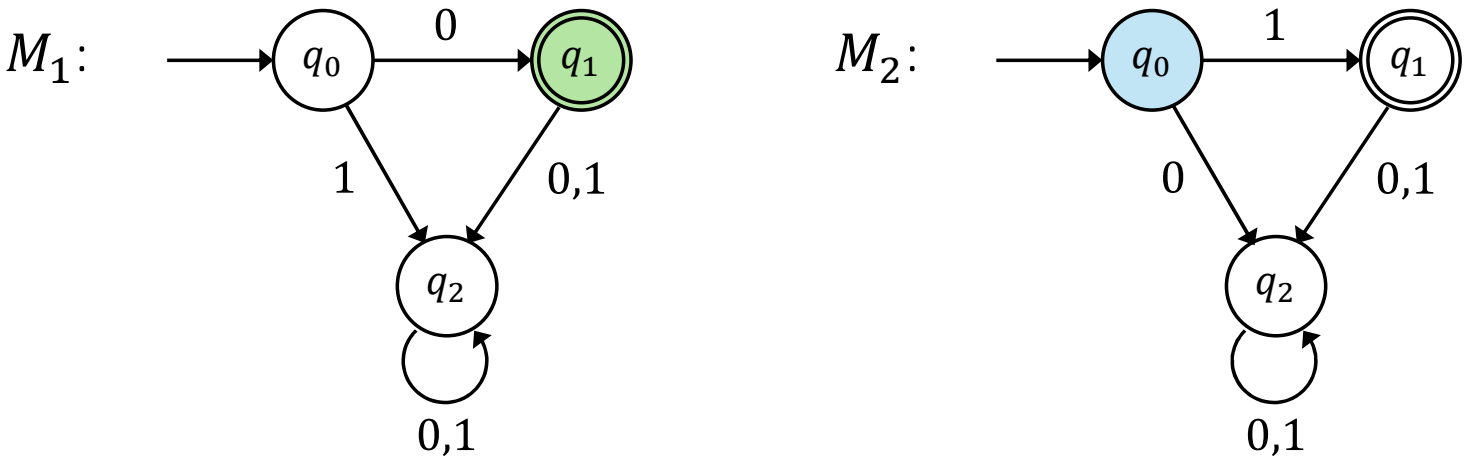
Suppose that R_1 is computed by M_1 and R_2 is computed by M_2.

How to construct a DFA M that computes $R_1 \circ R_2 = L(M_1) \circ L(M_2)$?

From Regex to DFA: Concatenation?

Suppose that R_1 is computed by M_1 and R_2 is computed by M_2.
How to construct a DFA M that computes $R_1 \circ R_2 = L(M_1) \circ L(M_2)$?

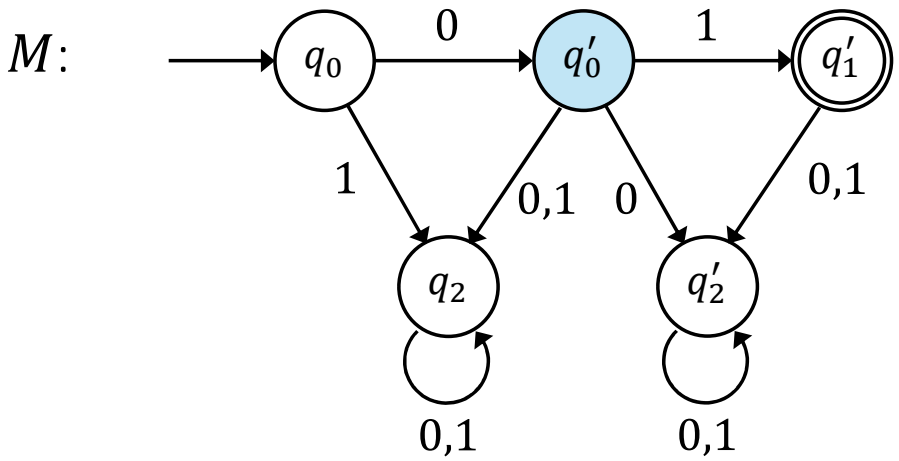
Idea: identify the accepting states of M_1 with the initial state of M_2



From Regex to DFA: Concatenation?

Suppose that R_1 is computed by M_1 and R_2 is computed by M_2.
How to construct a DFA M that computes $R_1 \circ R_2 = L(M_1) \circ L(M_2)$?

Idea: identify the accepting states of M_1 with the initial state of M_2

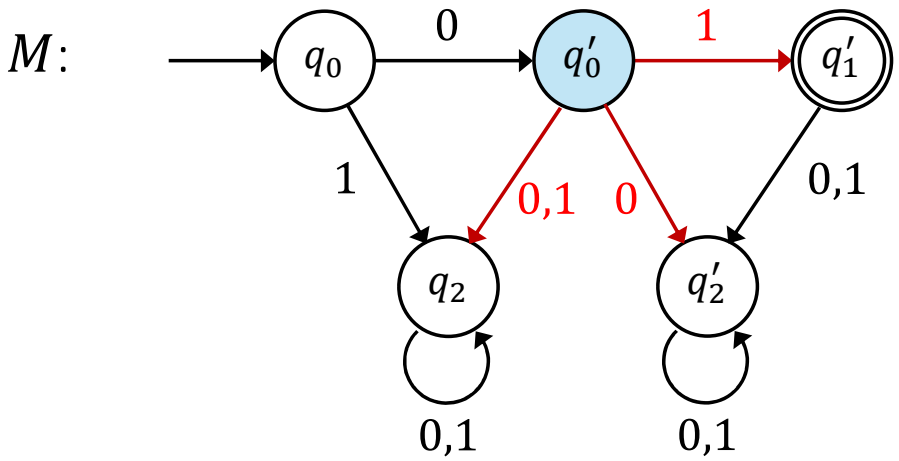


From Regex to DFA: Concatenation?

Suppose that R_1 is computed by M_1 and R_2 is computed by M_2.

How to construct a DFA M that computes $R_1 \circ R_2 = L(M_1) \circ L(M_2)$?

Idea: identify the accepting states of M_1 with the initial state of M_2



Nondeterministic Finite Automata

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

- Q is the finite set of states
- Σ is the finite set of symbols (alphabet)
- $\delta: Q \times \Sigma \rightarrow \mathbf{2^Q}$ is the transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of accept states

Nondeterministic Finite Automata

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

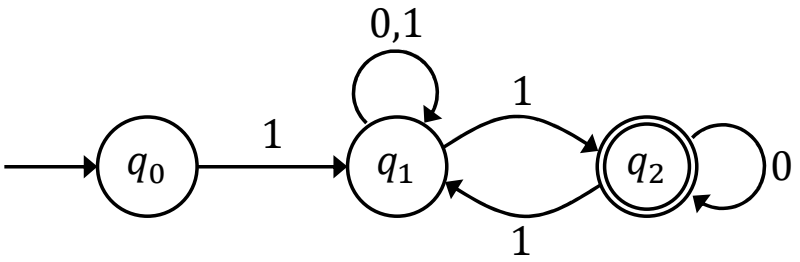
- Q is the finite set of states
- Σ is the finite set of symbols (alphabet)
- $\delta: Q \times \Sigma \to \mathbf{2^Q}$ is the transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of accept states

For each $q \in Q$ and $a \in \Sigma$, $\delta(q, a) \subseteq Q$ is a (possibly empty) set of states.

Nondeterministic Finite Automata

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

- Q is the finite set of states
- Σ is the finite set of symbols (alphabet)
- $\delta: Q \times \Sigma \rightarrow \mathbf{2^Q}$ is the transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of accept states



For each $q \in Q$ and $a \in \Sigma$, $\delta(q, a) \subseteq Q$ is a (possibly empty) set of states.

The state diagram contains an edge (q_i, q_j) labeled with a if $q_j \in \delta(q_i, a)$.

Execution of NFA

On input $w = w_1 w_2 \dots w_n$, the NFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts if and only if there exists a sequence of states $r_0, r_1, \dots, r_n \in Q$ such that:

$r_0 = q_0$

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

$r_n \in F$.

Execution of NFA

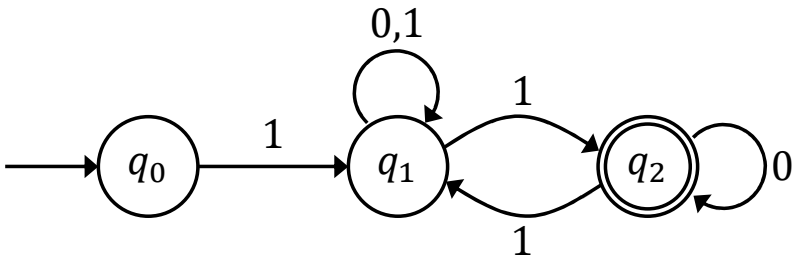
On input $w = w_1 w_2 \ldots w_n$, the NFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts if and only if there exists a sequence of states $r_0, r_1, \ldots, r_n \in Q$ such that:

$r_0 = q_0$

$r_i \in \delta(r_{i-1}, w_i)$ for $i = 1,2, \ldots, n$

$r_n \in F$.

(i.e., there exists a path in the state diagram from q_0 to some $q \in F$)



Execution of NFA

On input $w = w_1 w_2 \ldots w_n$, the NFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts if and only if there exists a sequence of states $r_0, r_1, \ldots, r_n \in Q$ such that:

$r_0 = q_0$

$r_i \in \delta(r_{i-1}, w_i)$ for $i = 1,2, \ldots, n$

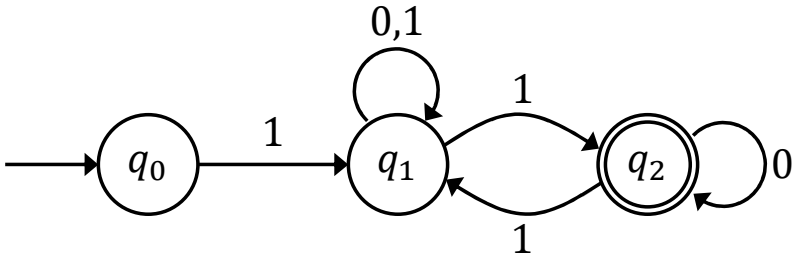
$r_n \in F$.

(i.e., there exists a path in the state diagram from q_0 to some $q \in F$)

For example, $1011 \in L(M)$

$100 \notin L(M)$

$0011 \notin L(M)$

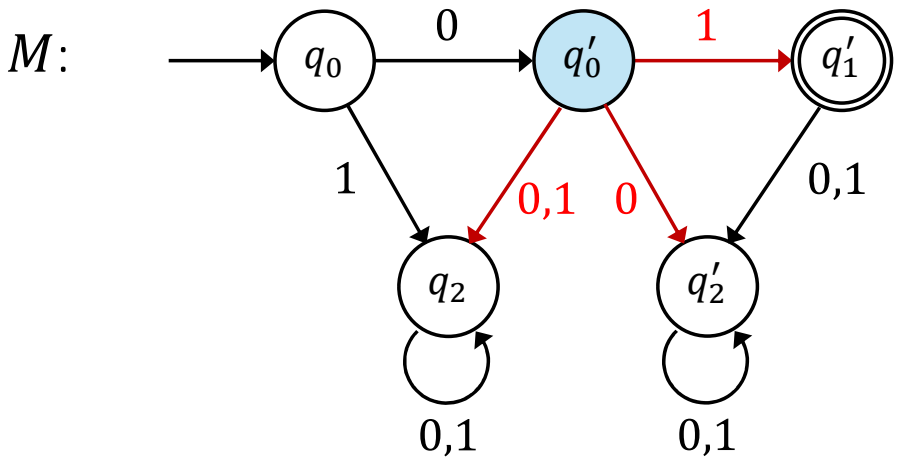


From Regex to NFA: Concatenation?

Suppose that R_1 is computed by M_1 and R_2 is computed by M_2.

How to construct a NFA M that computes $R_1 \circ R_2 = L(M_1) \circ L(M_2)$?

Idea: identify the accepting states of M_1 with the initial state of M_2

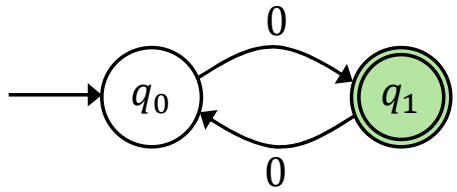


From Regex to NFA: Concatenation?

Suppose that R_1 is computed by M_1 and R_2 is computed by M_2.
How to construct a NFA M that computes $R_1 \circ R_2 = L(M_1) \circ L(M_2)$?

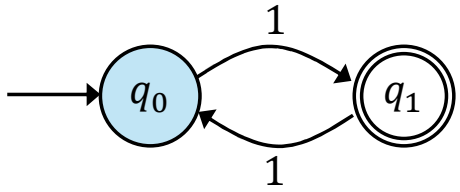
Idea: identify the accepting states of M_1 with the initial state of M_2

M_1:



$L(M_1) = 0(00)^*$

M_2:



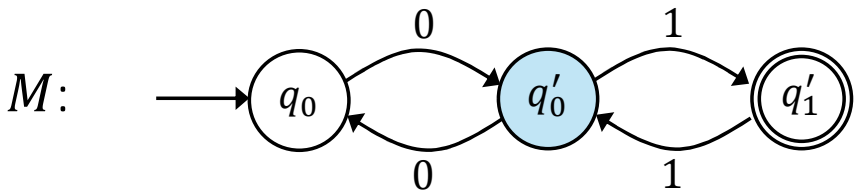
$L(M_2) = 1(11)^*$

From Regex to NFA: Concatenation?

Suppose that R_1 is computed by M_1 and R_2 is computed by M_2.

How to construct a NFA M that computes $R_1 \circ R_2 = L(M_1) \circ L(M_2)$?

Idea: identify the accepting states of M_1 with the initial state of M_2



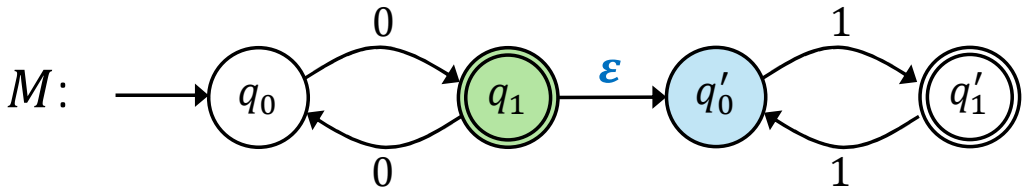
$L(M) \neq 0(00)^*1(11)^*$

From Regex to NFA: Concatenation?

Suppose that R_1 is computed by M_1 and R_2 is computed by M_2.

How to construct a NFA M that computes $R_1 \circ R_2 = L(M_1) \circ L(M_2)$?

Idea: ~~identify~~ the accepting states of M_1 with the initial state of M_2
connect



ε-Nondeterministic Finite Automata

An **$\boldsymbol{\varepsilon}$-NFA** is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$:

- Q is the finite set of states
- Σ is the finite set of symbols (alphabet)
- $\delta: Q \times (\boldsymbol{\Sigma} \cup \{\boldsymbol{\varepsilon}\}) \rightarrow \boldsymbol{2^Q}$ is the transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of accept states

ε-Nondeterministic Finite Automata

An $\boldsymbol{\varepsilon}$-NFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$:

- Q is the finite set of states
- Σ is the finite set of symbols (alphabet)
- $\delta: Q \times (\boldsymbol{\Sigma} \cup \{\boldsymbol{\varepsilon}\}) \rightarrow \boldsymbol{2^Q}$ is the transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of accept states

An ε-transition $\delta(q_i, \varepsilon) = q_j$ does not read the next symbol and is intrinsically nondeterministic.

Execution of ε-NFA

On input $w = w_1 w_2 \ldots w_n$, the ε-NFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts if and only if there exists a padded $y = y_1 y_2 \ldots y_m$ such that

$$y_i \in \Sigma \cup \{\varepsilon\}, i = 1,2, \ldots, m$$

and $y = w$, and a sequence of states $r_0, r_1, \ldots, r_m \in Q$ and

$r_0 = q_0$

$r_i \in \delta(r_{i-1}, y_i)$ for $i = 1,2, \ldots, m$

$r_m \in F$.

Execution of ε-NFA

Examples:

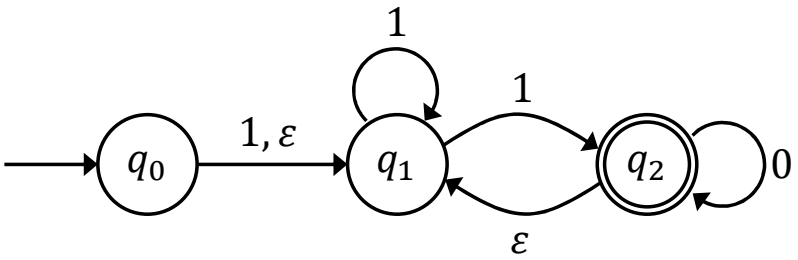
- $1 \in L(M)$

  By $y = \varepsilon 1$

- $1101 \in L(M)$

  By $y = 110\varepsilon 1$

- $000 \notin L(M)$



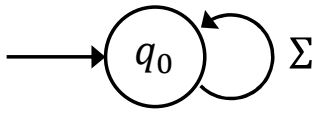
From Regex to ε-NFA: Base Cases

We now prove by induction that for every regex R, there exists an ε-NFA M such that $L(M) = R$.

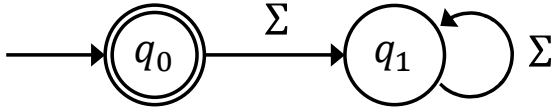
From Regex to ε-NFA: Base Cases

We now prove by induction that for every regex R, there exists an ε-NFA M such that $L(M) = R$.

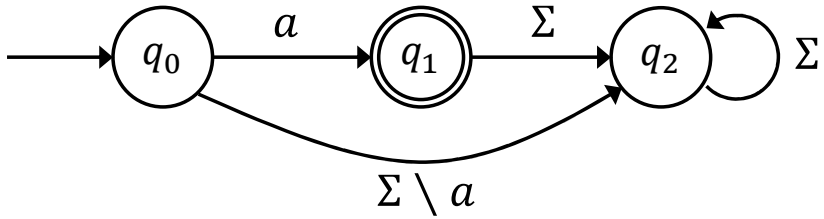
- If $R = \emptyset$:



- If $R = \varepsilon$:

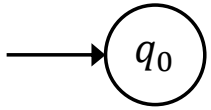


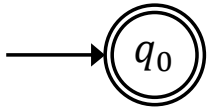
- If $R = a$ for some $a \in \Sigma$:

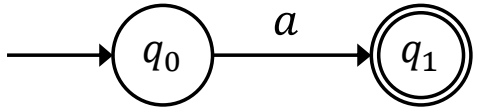


From Regex to ε-NFA: Base Cases

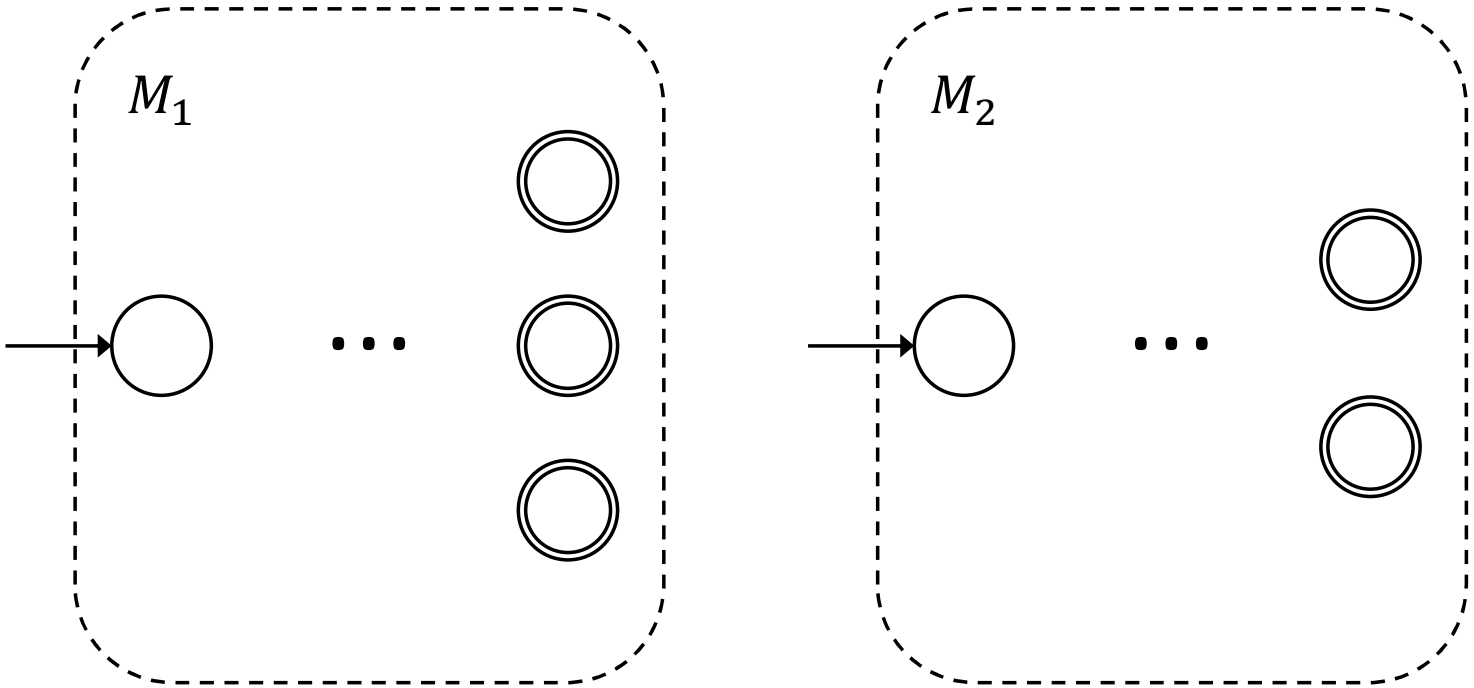
We now prove by induction that for every regex R, there exists an ε-NFA M such that $L(M) = R$.

- If $R = \emptyset$: 

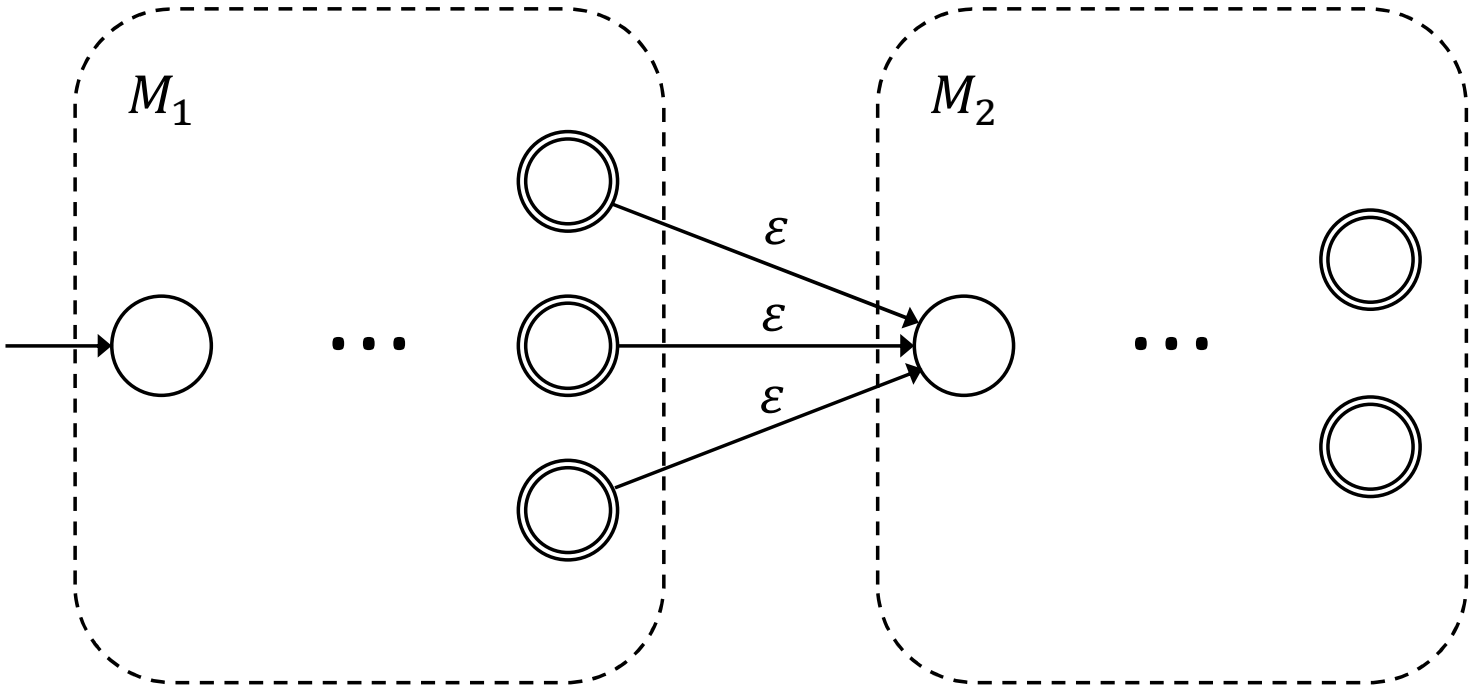
- If $R = \varepsilon$: 

- If $R = a$ for some $a \in \Sigma$: 

From Regex to ε-NFA: Concatenation



From Regex to ε-NFA: Concatenation



From Regex to ε-NFA: Concatenation

Let $M_1 = (Q_1, \Sigma, \delta_1, q_0^{(1)}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^{(2)}, F_2)$.
Define $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \sqcup Q_2$ ⟵ disjoint union

From Regex to ε-NFA: Concatenation

Let $M_1 = (Q_1, \Sigma, \delta_1, q_0^{(1)}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^{(2)}, F_2)$.
Define $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \sqcup Q_2$

- $\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 \setminus F_1 \text{ or } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_0^{(2)}\} & \text{if } q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & \text{if } q \in Q_2 \end{cases}$

From Regex to ε-NFA: Concatenation

Let $M_1 = (Q_1, \Sigma, \delta_1, q_0^{(1)}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^{(2)}, F_2)$.
Define $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = Q_1 \sqcup Q_2$
- $\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{if } q \in Q_1 \setminus F_1 \text{ or } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_0^{(2)}\} & \text{if } q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & \text{if } q \in Q_2 \end{cases}$
- $q_0 = q_0^{(1)}$
- $F = F_2$.

From Regex to ε-NFA: Concatenation

To prove $L(M) = L(M_1) \circ L(M_2)$, note that every accepting path in M (path from $q_0^{(1)}$ to F_2) must use the ε-transition once

$$\delta(q, \varepsilon) = q_0^{(2)}, q \in F_1$$

to transit from Q_1 to Q_2.

The paths before the transition corresponds to strings in $L(M_1)$ and the paths after the transition corresponds to strings in $L(M_2)$.

From Regex to ε-NFA: Concatenation

Formally, for every $w \in L(M)$, let $y = y_1 \dots y_m$ be the ε-padded w, and let $r_0 = q_0^{(1)}, r_1 \dots, r_{m-1}, r_m \in F_2$ be the accepting sequence.

Let r_k be the first state in the sequence that is in Q_2, then:

From Regex to ε-NFA: Concatenation

Formally, for every $w \in L(M)$, let $y = y_1 \dots y_m$ be the ε-padded w, and let $r_0 = q_0^{(1)}, r_1 \dots, r_{m-1}, r_m \in F_2$ be the accepting sequence.

Let r_k be the first state in the sequence that is in Q_2, then:

- $r_0, \dots, r_{k-1} \in Q_1$, $r_k, \dots, r_m \in Q_2$;
- $r_{k-1} \in F_1$, $r_k = q_0^{(2)}$, $y_k = \varepsilon$.

So $y_1 \dots y_{k-1} \in L(M_1)$ and $y_{k+1} \dots y_m \in L(M_2)$.

From Regex to ε-NFA: Concatenation

Formally, for every $w \in L(M)$, let $y = y_1 \dots y_m$ be the ε-padded w, and let $r_0 = q_0^{(1)}, r_1 \dots, r_{m-1}, r_m \in F_2$ be the accepting sequence.

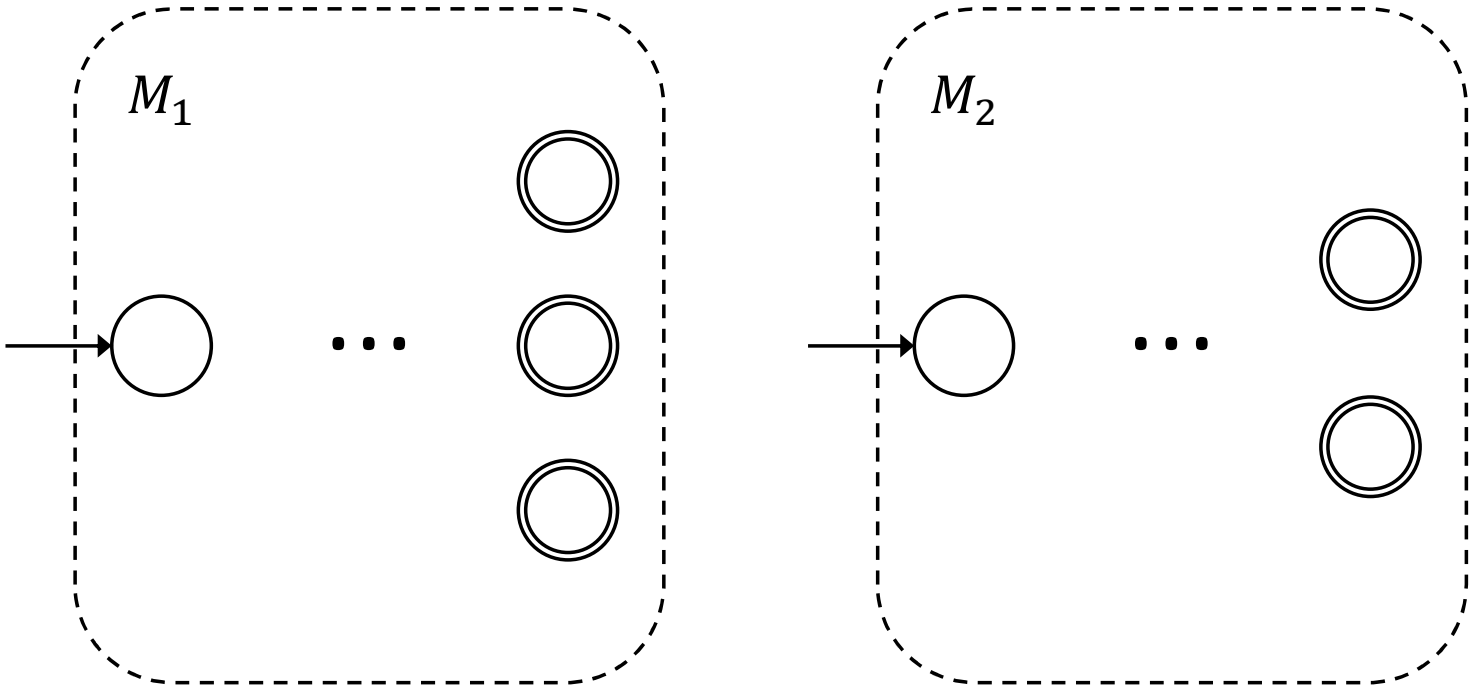
Let r_k be the first state in the sequence that is in Q_2, then:

- $r_0, \dots, r_{k-1} \in Q_1$, $r_k, \dots, r_m \in Q_2$;
- $r_{k-1} \in F_1$, $r_k = q_0^{(2)}$, $y_k = \varepsilon$.

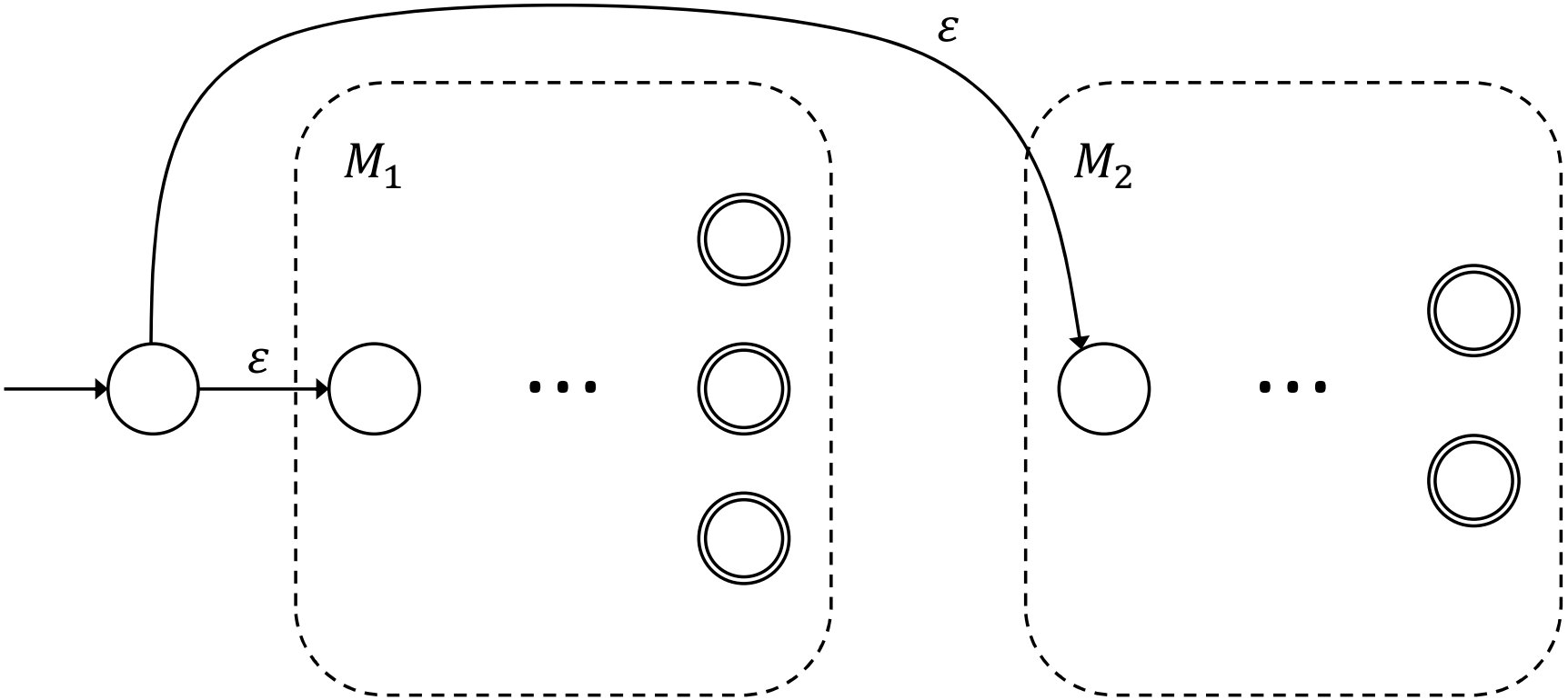
So $y_1 \dots y_{k-1} \in L(M_1)$ and $y_{k+1} \dots y_m \in L(M_2)$.

Conversely, for every $w_1 \in L(M_1)$ and $w_2 \in L(M_2)$, …

From Regex to ε-NFA: Union



From Regex to ε-NFA: Union



From Regex to ε-NFA: Union

Let $M_1 = (Q_1, \Sigma, \delta_1, q_0^{(1)}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^{(2)}, F_2)$.

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

- $Q = Q_1 \sqcup Q_2 \sqcup \{q_0\}$

From Regex to ε-NFA: Union

Let $M_1 = (Q_1, \Sigma, \delta_1, q_0^{(1)}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^{(2)}, F_2)$.

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

- $Q = Q_1 \sqcup Q_2 \sqcup \{q_0\}$

- $\delta(q, a) = \begin{cases} \delta_1(q, a) & \text{if } q \in Q_1 \\ \delta_2(q, a) & \text{if } q \in Q_2 \\ \{q_0^{(1)}, q_0^{(2)}\} & \text{if } q = q_0 \text{ and } a = \varepsilon \\ \emptyset & \text{if } q = q_0 \text{ and } a \neq \varepsilon \end{cases}$

- $F = F_1 \cup F_2$.

From Regex to ε-NFA: Union

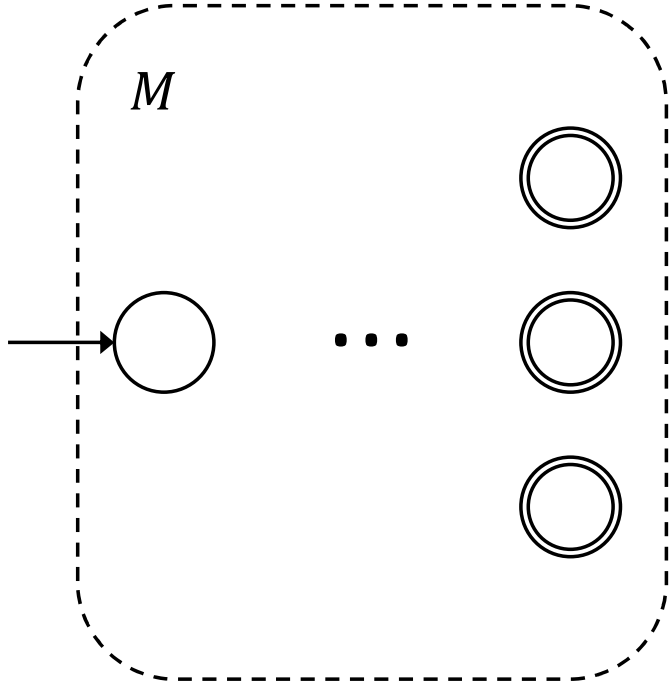
To prove $L(M) = L(M_1) \cup L(M_2)$, note that every accepting path in M must use the ε-transitions

$$\delta(q_0, \varepsilon) = \{q_0^{(1)}, q_0^{(2)}\}$$

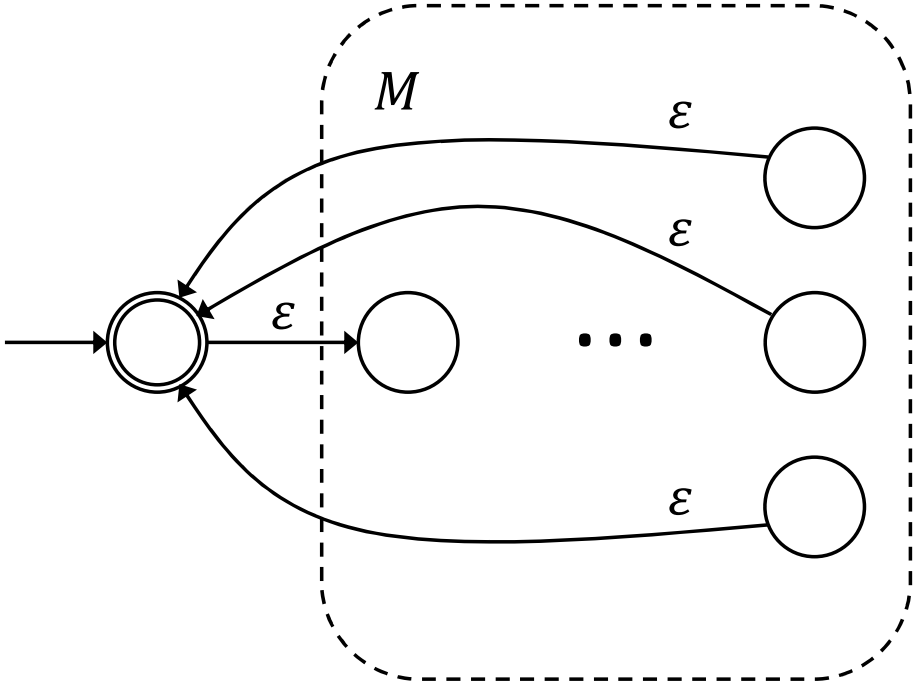
at the start.

- If the second state is $q_0^{(1)}$, the rest computes exactly $L(M_1)$;
- If the second state is $q_0^{(2)}$, the rest computes exactly $L(M_2)$.

From Regex to ε-NFA: Kleene Star



From Regex to ε-NFA: Kleene Star



From Regex to ε-NFA: Kleene Star

Let $M = (Q, \Sigma, \delta, q_0, F)$, define $M' = (Q', \Sigma, \delta', q_0', F')$ where

- $Q' = Q \sqcup \{q_0'\}$
- $\delta'(q,a) = \begin{cases} \delta(q,a) & \text{if } q \in Q \setminus F \text{ or } a \neq \varepsilon \\ \delta(q,a) \cup \{q_0'\} & \text{if } q \in F \text{ and } a = \varepsilon \\ \{q_0\} & \text{if } q = q_0' \text{ and } a = \varepsilon \\ \emptyset & \text{if } q = q_0' \text{ and } a \neq \varepsilon \end{cases}$
- $F' = \{q_0'\}$.

From Regex to ε-NFA: Kleene Star

To prove $L(M') = L(M)^*$, consider an accepting path in M':

$$q_0', (\dots, q_0', \dots, q_0', \dots \dots \dots, q_0')$$

where q_0' appears $k \geq 1$ times.

Each part of the path between two appearances of q_0' is a path from q_0 to F, which corresponds to a string in $L(M)$.

From Regex to ε-NFA

What we have shown are called closure properties:

Lemma. The class of languages that can be computed by ε-NFAs is closed under union, concatenation and Kleene star.

From Regex to ε-NFA

What we have shown are called closure properties:

Lemma. The class of languages that can be computed by ε-NFAs is closed under union, concatenation and Kleene star.

So far, we have shown: