Lecture 4: Closure and NFAs

CSC 320: Foundations of Computer Science

Quinton Yong

quintonyong@uvic.ca

University of Victoria

Closure Properties for Sets

- A set is **closed under some operation** if applying the operation to elements of the set returns **another element of the set**
- **Example:** The set of **even numbers** is closed under **addition** since adding even numbers returns another even number

Closure Properties for Language Classes

- We can also have **closure properties** for **classes of languages** (e.g. regular languages)

- That is, if we apply **a set operation** to languages in a **certain class**, the resulting language is **also in that class**

- We will prove that performing the **union**, **intersection**, and **concatenation** of two regular languages $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ results in another regular language

- i.e. Regular languages are **closed under** union, intersection, and concatenation

Regular Languages are Closed under Union

Theorem: If $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ are **regular languages** over alphabet Σ, then the language $\boldsymbol{L_1} \cup \boldsymbol{L_2}$ is a **regular language**

Proof: Since $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ are regular languages, then there exist DFAs $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$ where $\boldsymbol{L_1} = \boldsymbol{L(M_1)}$ and $\boldsymbol{L_2} = \boldsymbol{L(M_2)}$

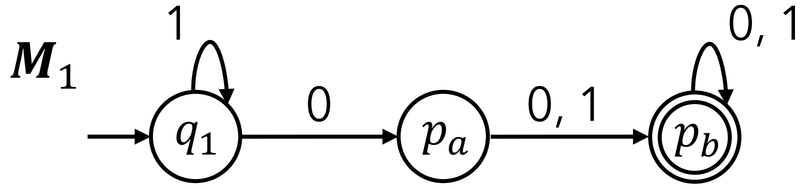
Proof idea: Construct a DFA $\boldsymbol{M}$ that accepts exactly the strings accepted by $\boldsymbol{M_1}$ as well as the strings accepted by $\boldsymbol{M_2}$

We do this by creating a DFA which **simulates both machines concurrently**, and accepts if at least one of them accepts

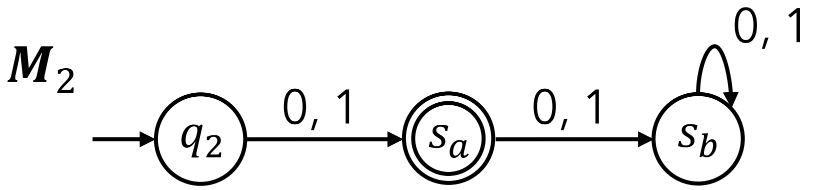
Regular Languages are Closed under Union

Example: $\Sigma = \{0, 1\}$

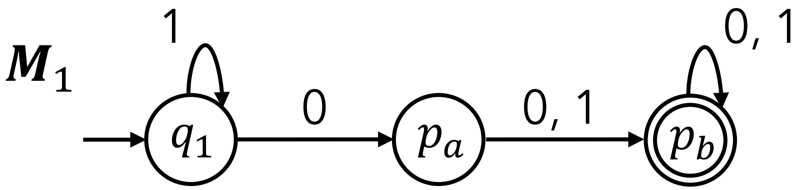
- $\boldsymbol{L_1}$: set of strings that, after a **possible prefix of 1s**, consists of **at least one 0** followed by **at least one symbol**

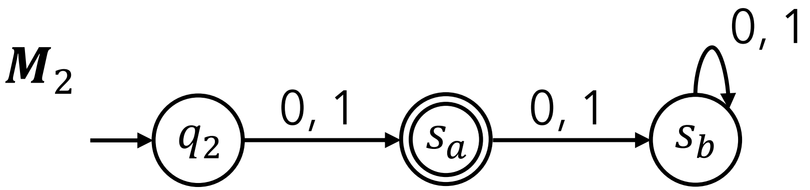


- $\boldsymbol{L_2}$: set of all strings of **length exactly 1**



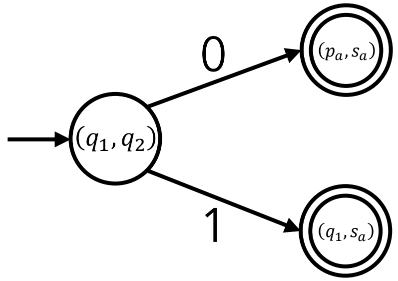
Regular Languages are Closed under Union





Union idea for M:

- Use pairs of states of $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$ as states in $\boldsymbol{M}$



- Transitions for a symbol $\boldsymbol{a} \in \boldsymbol{\Sigma}$ is the state pair corresponding to where $\boldsymbol{M_1}$ goes with $\boldsymbol{a}$ and where $\boldsymbol{M_2}$ goes with $\boldsymbol{a}$ (simulate both DFAs)

- State is an accept state if either resulting state in $\boldsymbol{M_1}$ or $\boldsymbol{M_2}$ is an accept state

Regular Languages are Closed under Union

Proof continued:

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be DFAs.

We construct DFA $M = (Q, \Sigma, \delta, q_0, F)$ as follows:

- $Q = \{ (r_1, r_2) | \, r_1 \in Q_1, r_2 \in Q_2 \}$,
  - all pairs of states where one is from M_1 and one is from M_2
- For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$ define transition function
$$\delta : \delta\big((r_1, r_2), a\big) = \big(\delta_1(r_1, a), \delta_2(r_2, a)\big)$$
- $q_0 = (q_1, q_2)$
  - Start from the state pair with starts states of M_1 and M_2
- $F = \{ (r_1, r_2) | \, r_1 \in F_1 \text{ or } r_2 \in F_2 \}$
  - Strings are accepted if at least one DFA accepts

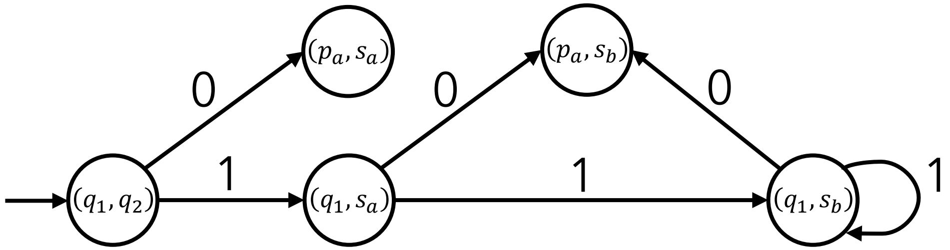
$\boxed{M \text{ recognizes } L_1 \cup L_2}$

Regular Languages are Closed under Union

- Same **example** as before (fully worked out):

- For each $(\boldsymbol{r_1}, \boldsymbol{r_2}) \in \boldsymbol{Q}$ and each $\boldsymbol{a} \in \boldsymbol{\Sigma}$ define transition function

$$\boldsymbol{\delta} : \boldsymbol{\delta}\big((\boldsymbol{r_1}, \boldsymbol{r_2}), \boldsymbol{a}\big) = \big(\boldsymbol{\delta_1}(\boldsymbol{r_1}, \boldsymbol{a}), \boldsymbol{\delta_2}(\boldsymbol{r_2}, \boldsymbol{a})\big)$$

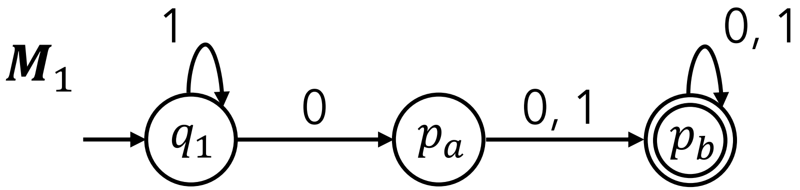


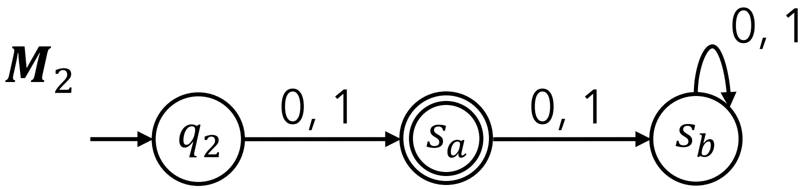
Transition table for $\boldsymbol{M}$

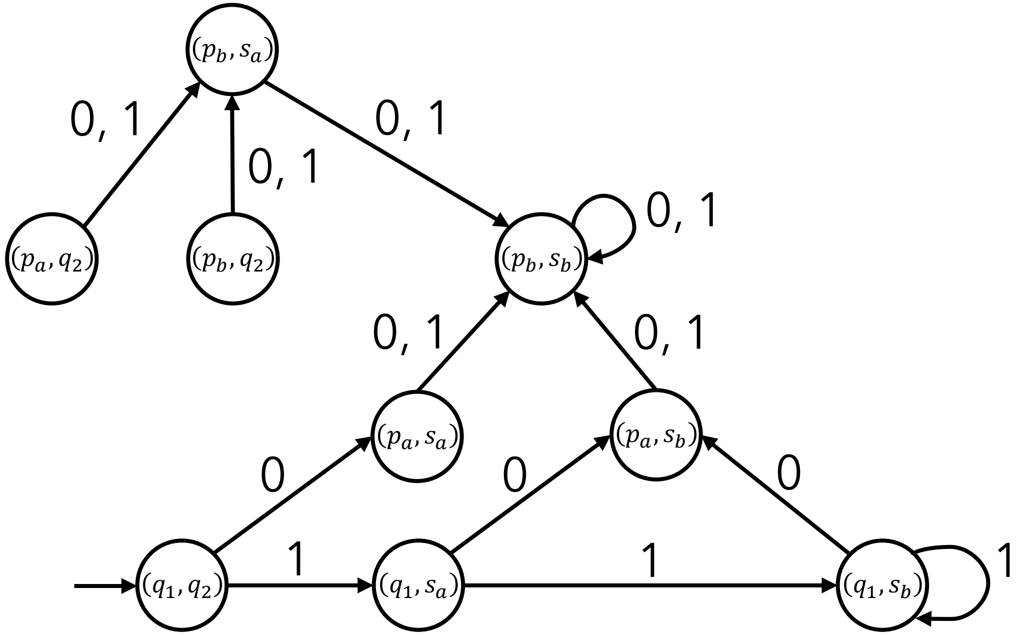
$\boldsymbol{Q}$	$\mathbf{0}$	$\mathbf{1}$
(q_1, q_2)	(p_a, s_a)	(q_1, s_a)
(q_1, s_a)	(p_a, s_b)	(q_1, s_b)
(q_1, s_b)	(p_a, s_b)	(q_1, s_b)
(p_a, q_2)		
(p_a, s_a)		
(p_a, s_b)		
(p_b, q_2)		
(p_b, s_a)		
(p_b, s_b)		

Regular Languages are Closed under Union

- Same **example** as before (fully worked out):





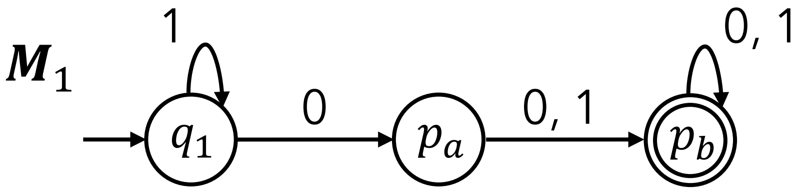


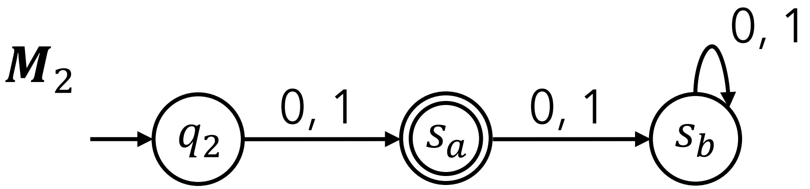
Transition table for M

Q	0	1
→ (q_1, q_2)	(p_a, s_a)	(q_1, s_a)
(q_1, s_a)	(p_a, s_b)	(q_1, s_b)
(q_1, s_b)	(p_a, s_b)	(q_1, s_b)
(p_a, q_2)	(p_b, s_a)	(p_b, s_a)
(p_a, s_a)	(p_b, s_b)	(p_b, s_b)
(p_a, s_b)	(p_b, s_b)	(p_b, s_b)
(p_b, q_2)	(p_b, s_a)	(p_b, s_a)
(p_b, s_a)	(p_b, s_b)	(p_b, s_b)
(p_b, s_b)	(p_b, s_b)	(p_b, s_b)

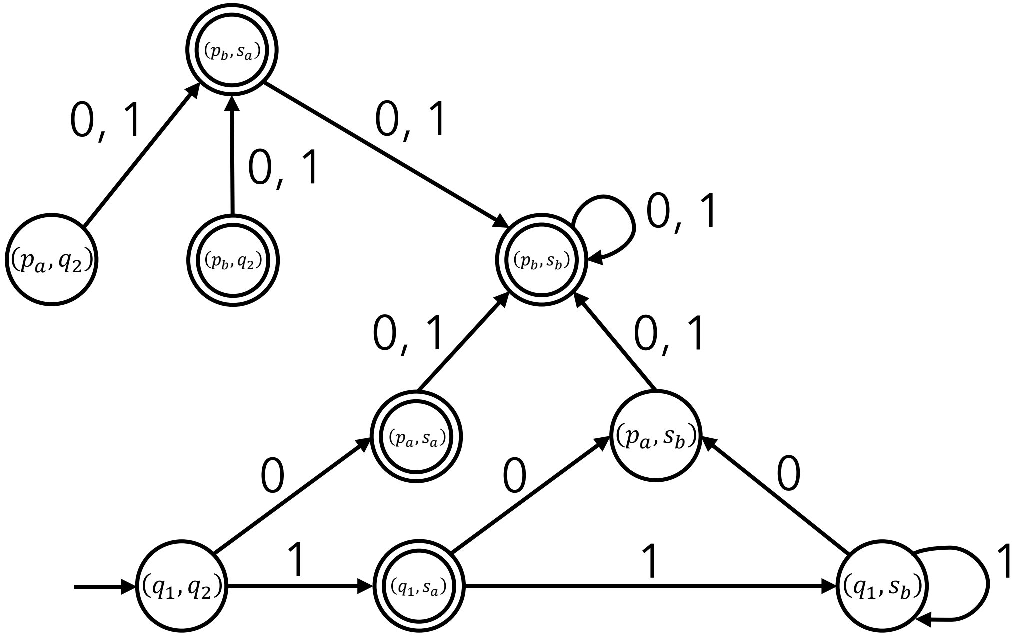
Regular Languages are Closed under Union

- Same **example** as before (fully worked out):





- $F = \{\ (r_1, r_2) |\ r_1 \in F_1 \text{ or } r_2 \in F\ \}$



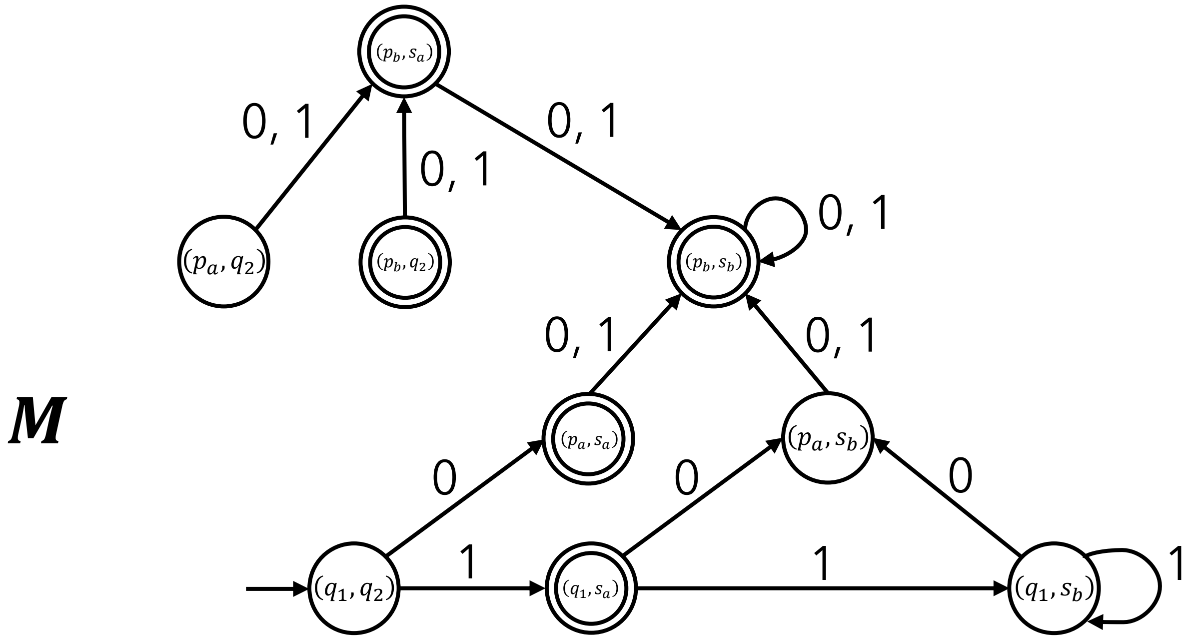
Transition table for M

Accept states

Q	0	1
$\to (q_1, q_2)$	(p_a, s_a)	(q_1, s_a)
(q_1, s_a)	(p_a, s_b)	(q_1, s_b)
(q_1, s_b)	(p_a, s_b)	(q_1, s_b)
(p_a, q_2)	(p_b, s_a)	(p_b, s_a)
(p_a, s_a)	(p_b, s_b)	(p_b, s_b)
(p_a, s_b)	(p_b, s_b)	(p_b, s_b)
(p_b, q_2)	(p_b, s_a)	(p_b, s_a)
(p_b, s_a)	(p_b, s_b)	(p_b, s_b)
(p_b, s_b)	(p_b, s_b)	(p_b, s_b)

Regular Languages are Closed under Union

$$L(M) = L_1 \cup L_2$$



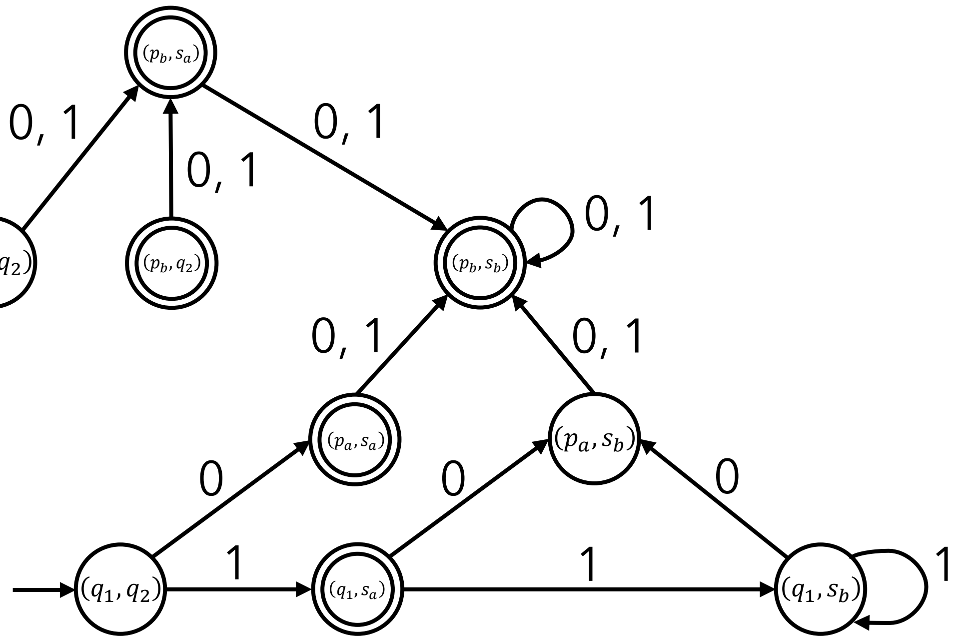
Transition table for M

Q	0	1
(q_1, q_2)	(p_a, s_a)	(q_1, s_a)
(q_1, s_a)	(p_a, s_b)	(q_1, s_b)
(q_1, s_b)	(p_a, s_b)	(q_1, s_b)
(p_a, q_2)	(p_b, s_a)	(p_b, s_a)
(p_a, s_a)	(p_b, s_b)	(p_b, s_b)
(p_a, s_b)	(p_b, s_b)	(p_b, s_b)
(p_b, q_2)	(p_b, s_a)	(p_b, s_a)
(p_b, s_a)	(p_b, s_b)	(p_b, s_b)
(p_b, s_b)	(p_b, s_b)	(p_b, s_b)

Regular Languages are Closed under Union

Recall:

- $\boldsymbol{L_1}$: set of strings that, after a **possible prefix of 1s**, consists of **at least one 0** followed by **at least one symbol**
- $\boldsymbol{L_2}$: set of all strings of **length exactly 1**
- Verify that **0** $\in \boldsymbol{L_1} \cup \boldsymbol{L_2}$ and also **11100** $\in \boldsymbol{L_1} \cup \boldsymbol{L_2}$ are accepted by $\boldsymbol{M}$

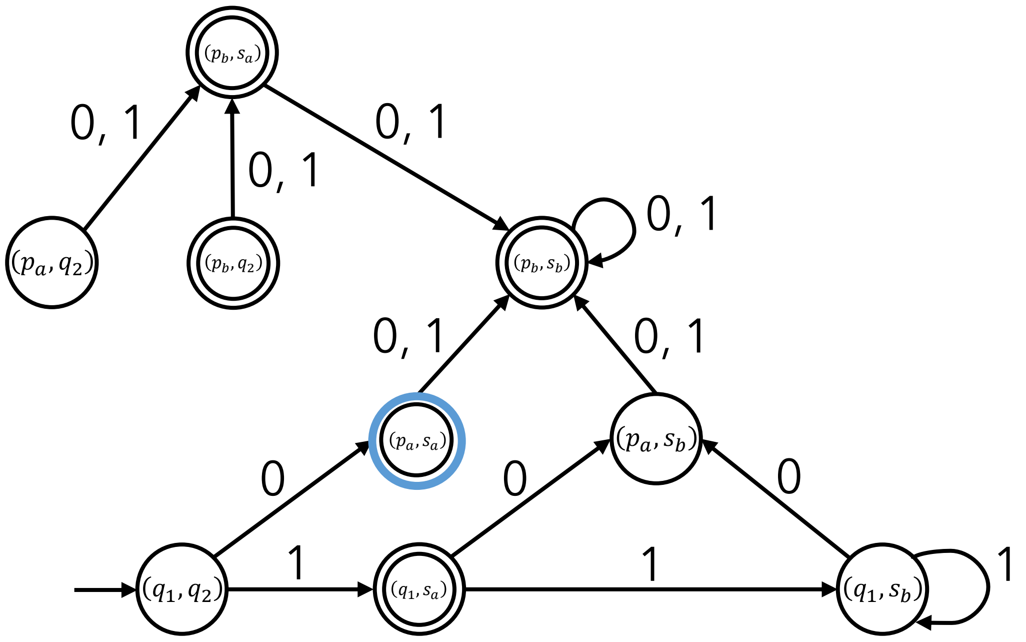


Regular Languages are Closed under Union

Recall:

- L_1: set of strings that, after a **possible prefix of 1s**, consists of **at least one 0** followed by **at least one symbol**
- L_2: set of all strings of **length exactly 1**
- Verify that **0** $\in L_1 \cup L_2$ and also **11100** $\in L_1 \cup L_2$ are accepted by M

0

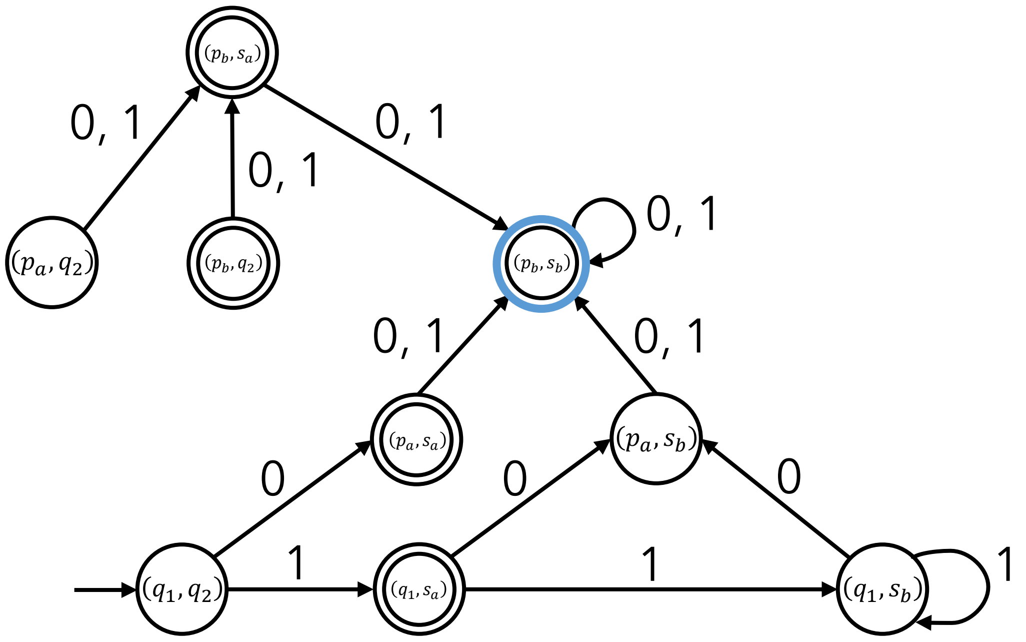


Regular Languages are Closed under Union

Recall:

- L_1: set of strings that, after a **possible prefix of 1s**, consists of **at least one 0** followed by **at least one symbol**
- L_2: set of all strings of **length exactly 1**
- Verify that **0** $\in L_1 \cup L_2$ and also **11100** $\in L_1 \cup L_2$ are accepted by M

11100



Regular Languages are Closed under Intersection

Theorem: If $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ are **regular languages** over alphabet Σ, then the language $\boldsymbol{L_1} \cap \boldsymbol{L_2}$ is a **regular language**

Proof: Since $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ are regular languages, then there exist DFAs $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$ where $\boldsymbol{L_1} = \boldsymbol{L(M_1)}$ and $\boldsymbol{L_2} = \boldsymbol{L(M_2)}$

Proof idea: We can create a DFA which **simulates both machines concurrently** in the exact same way as the **union closure** proof.

However, now the strings that are accepted by the new DFA are the strings which are accepted by **both $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$** (instead of either one)

Regular Languages are Closed under Intersection

Proof continued:

Let $\boldsymbol{M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)}$ and $\boldsymbol{M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)}$ be DFAs.

We construct DFA $\boldsymbol{M = (Q, \Sigma, \delta, q_0, F)}$ as follows:

- $\boldsymbol{Q} = \{\, (\boldsymbol{r_1}, \boldsymbol{r_2}) \mid \boldsymbol{r_1} \in \boldsymbol{Q_1}, \boldsymbol{r_2} \in \boldsymbol{Q_2} \,\}$,
- For each $(\boldsymbol{r_1}, \boldsymbol{r_2}) \in \boldsymbol{Q}$ and each $\boldsymbol{a} \in \boldsymbol{\Sigma}$ define transition function

$$\boldsymbol{\delta} : \boldsymbol{\delta}\big((\boldsymbol{r_1}, \boldsymbol{r_2}), \boldsymbol{a}\big) = \big(\boldsymbol{\delta_1}(\boldsymbol{r_1}, \boldsymbol{a}), \boldsymbol{\delta_2}(\boldsymbol{r_2}, \boldsymbol{a})\big)$$

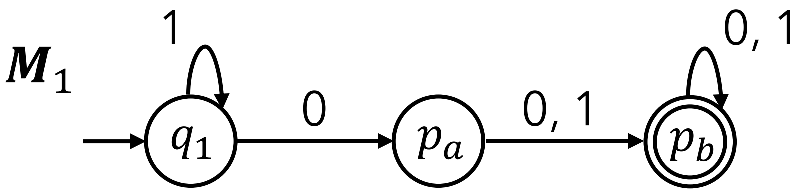
- $\boldsymbol{q_0} = (\boldsymbol{q_1}, \boldsymbol{q_2})$
- $\boxed{\boldsymbol{F} = \{\, (\boldsymbol{r_1}, \boldsymbol{r_2}) \mid \boldsymbol{r_1} \in \boldsymbol{F_1} \textbf{ and } \boldsymbol{r_2} \in \boldsymbol{F_2} \,\}}$

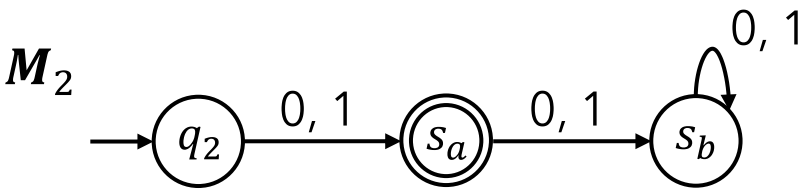
The DFA construction is exactly the same as before, except for which states we make **accept states**

$\boxed{\boldsymbol{M} \text{ recognizes } \boldsymbol{L_1} \cap \boldsymbol{L_2}}$

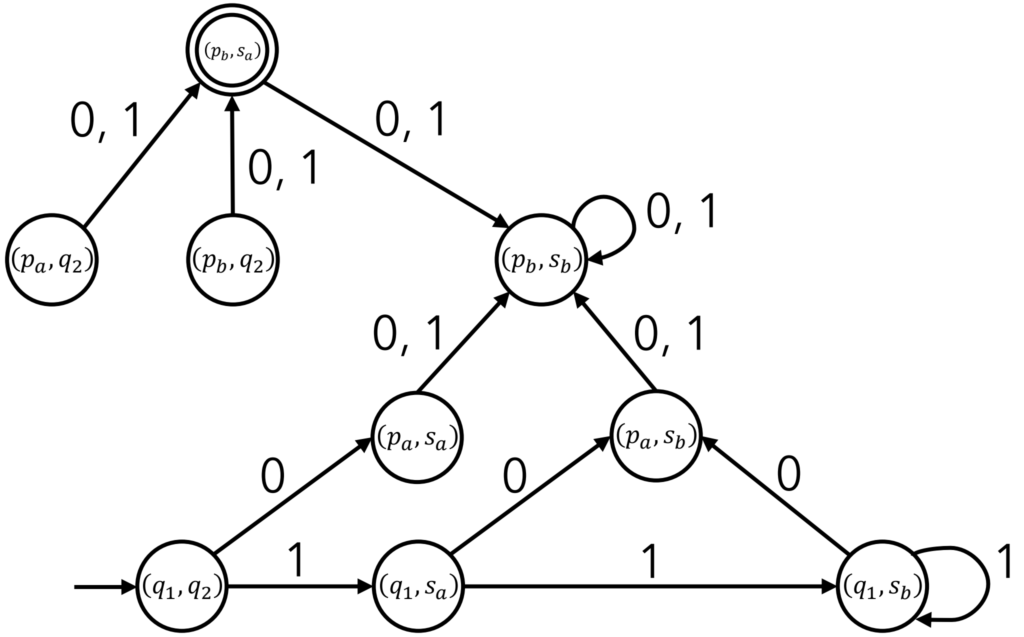
Regular Languages are Closed under Intersection

- $L(M) = L_1 \cap L_2$ (where L_1 and L_2 are same as union closure example)





- $F = \{\ (r_1, r_2) \mid r_1 \in F_1$ **and** $r_2 \in F_2\ \}$



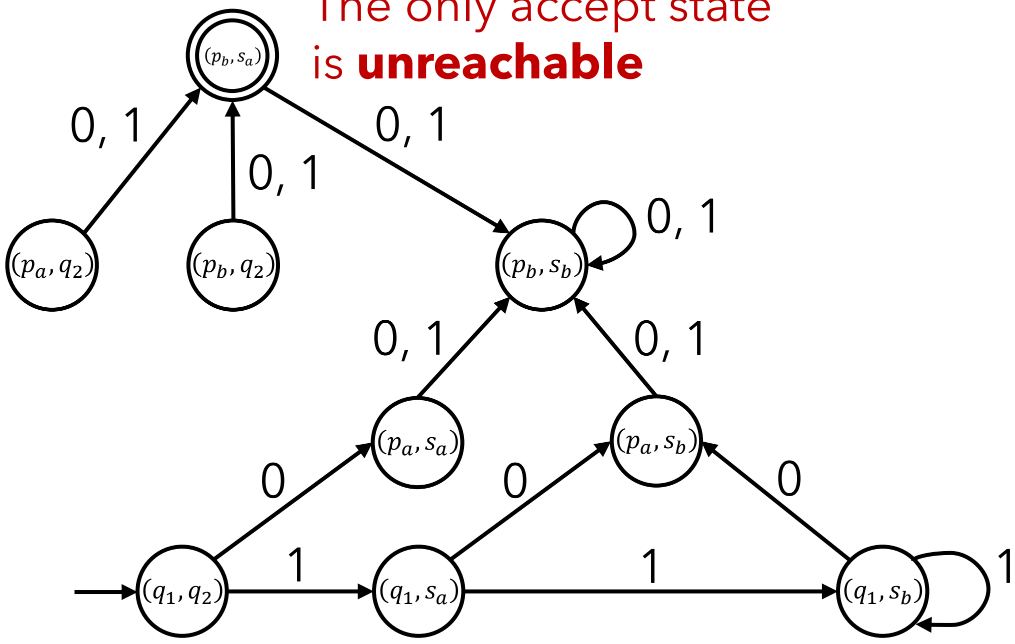
Transition table for M

Accept states

Q	0	1
→ (q_1, q_2)	(p_a, s_a)	(q_1, s_a)
(q_1, s_a)	(p_a, s_b)	(q_1, s_b)
(q_1, s_b)	(p_a, s_b)	(q_1, s_b)
(p_a, q_2)	$\boldsymbol{(p_b, s_a)}$	$\boldsymbol{(p_b, s_a)}$
(p_a, s_a)	(p_b, s_b)	(p_b, s_b)
(p_a, s_b)	(p_b, s_b)	(p_b, s_b)
(p_b, q_2)	$\boldsymbol{(p_b, s_a)}$	$\boldsymbol{(p_b, s_a)}$
$\boldsymbol{(p_b, s_a)}$	(p_b, s_b)	(p_b, s_b)
(p_b, s_b)	(p_b, s_b)	(p_b, s_b)

Regular Languages are Closed under Intersection

- L_1: set of strings that, after a **possible prefix of 1s**, consists of **at least one 0** followed by **at least one symbol**
- L_2: set of all strings of **length exactly 1**
- Notice that the strings in L_1 all have length **at least 2** and the strings in L_2 have length **exactly 1**, so the $L_1 \cap L_2 = \emptyset$



Regular Languages are Closed under Concatenation

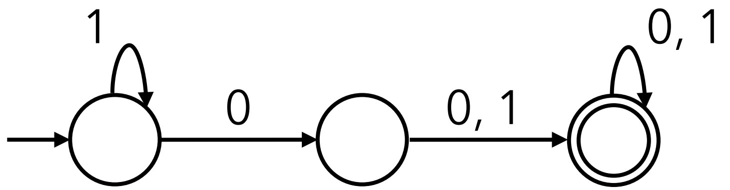
- How do we create a DFA which accepts the concatenation of two regular languages?

- We will introduce a different computational model which makes this easier **but has the same computational power** as a DFA

Deterministic Computation

Computation on a **DFA** is **deterministic**

- When computing a string, there is **no ambiguity** (deterministic) for the current state is and the next state when reading a character
- **Single execution path** for every string

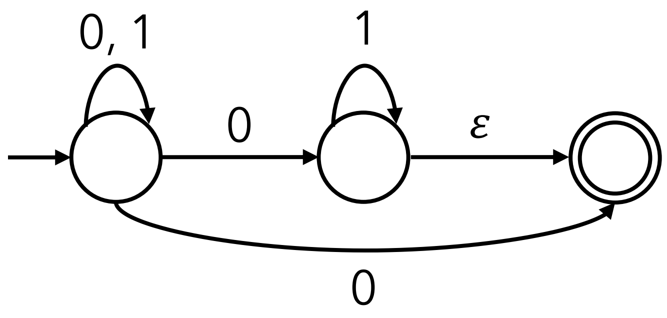


Nondeterministic Computation

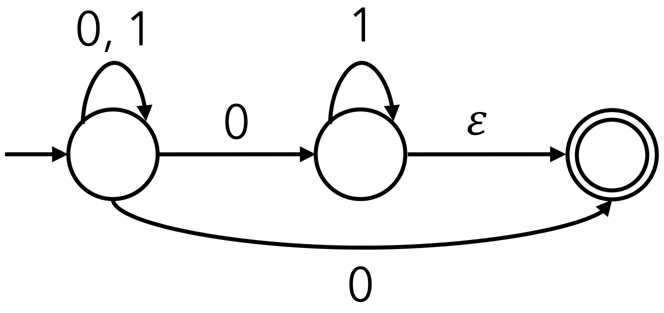
We describe another computational model with the **same computational power** as a DFA, but has **nondeterministic** computation

Nondeterminism:

- Can have **multiple** (simultaneous) **execution paths**
- Strings are accepted if there exists **at least one execution path that accepts** (still need to **read all symbols of string**)



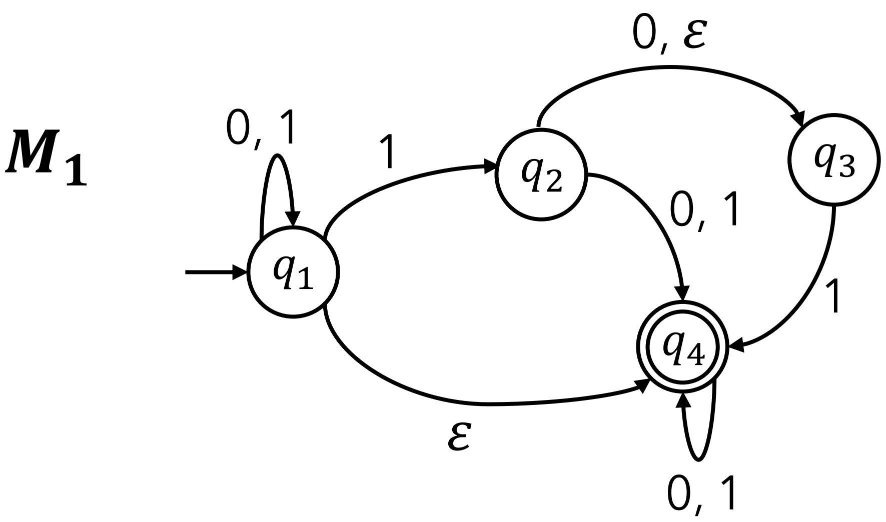
Nondeterministic Finite Automata (NFA)



Differences between **NFA** and **DFA** state diagrams:

1. Transitions go from states to **sets of states**
   - From a state $\boldsymbol{q}$, reading symbol $\boldsymbol{a}$ can transition to **more than one state**
   - We can also **not have transitions** for a symbol $\boldsymbol{a}$ out of a state $\boldsymbol{q}$
   - Formally $\boldsymbol{\delta}: Q \times (\Sigma \cup \{\varepsilon\}) \to \boldsymbol{\mathcal{P}}(Q)$
2. We allow **empty transitions** ($\boldsymbol{\varepsilon}$-transitions) $\xrightarrow{\boldsymbol{\varepsilon}}$
   - Can take this transition **without reading any symbol**
   - Essentially a **free transition** from a state to another state

NFA Example 1

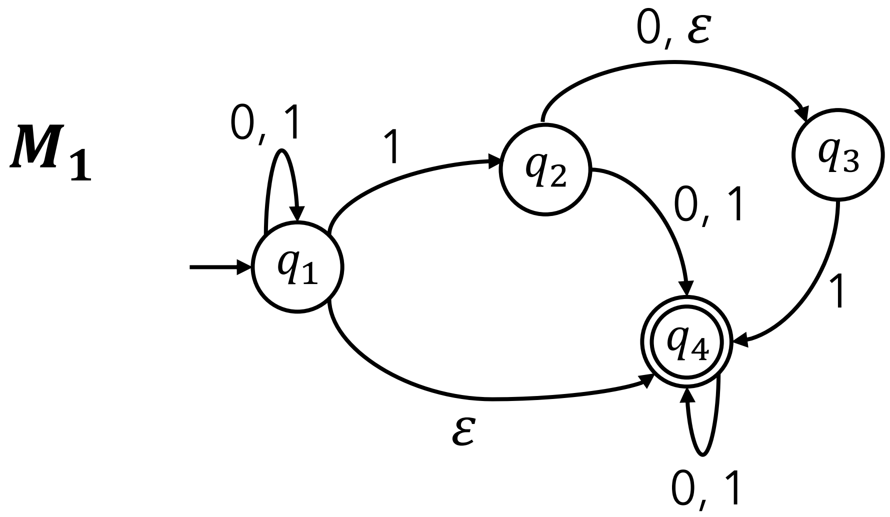


$\boldsymbol{M_1} = (\{\boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3}, \boldsymbol{q_4}\}, \{\mathbf{0}, \mathbf{1}\}, \boldsymbol{\delta}, \boldsymbol{q_1}, \{\boldsymbol{q_4}\})$ with $\boldsymbol{\delta}: \boldsymbol{Q} \times (\boldsymbol{\Sigma} \cup \{\boldsymbol{\varepsilon}\}) \to \boldsymbol{\mathcal{P}}(\boldsymbol{Q})$ defined by

$\boldsymbol{\delta}$	**0**	**1**	$\boldsymbol{\varepsilon}$
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\{q_4\}$
q_2	$\{q_3, q_4\}$	$\{q_4\}$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

State diagram **omits** transitions into $\emptyset$

NFA Example 1

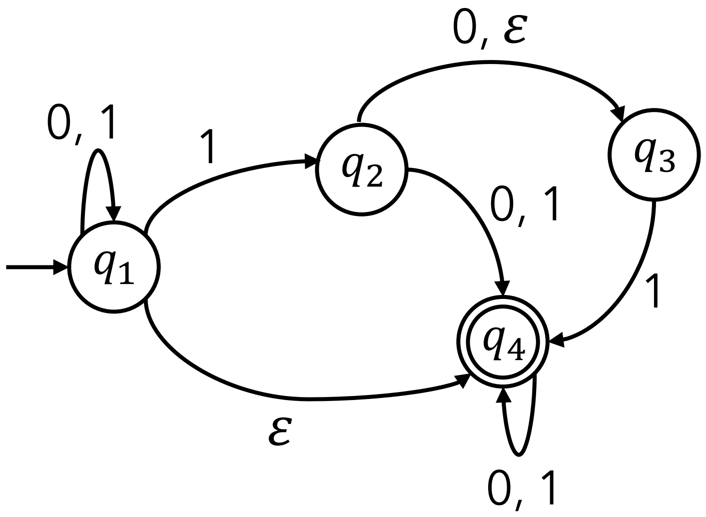


Is the string $\boldsymbol{w} = \mathbf{1}$ accepted by $\boldsymbol{M_1}$? **Yes**

Is the string $\boldsymbol{w} = \mathbf{101}$ accepted by $\boldsymbol{M_1}$? **Yes**

NFA Example 1

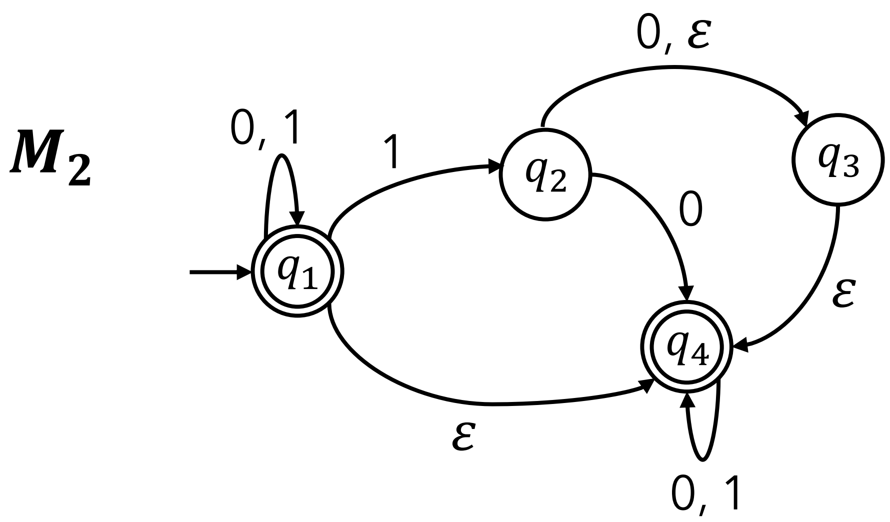
M_1



What is the $L(M_1)$?

- Every string in Σ^* is accepted
- $L(M_1) = \Sigma^*$

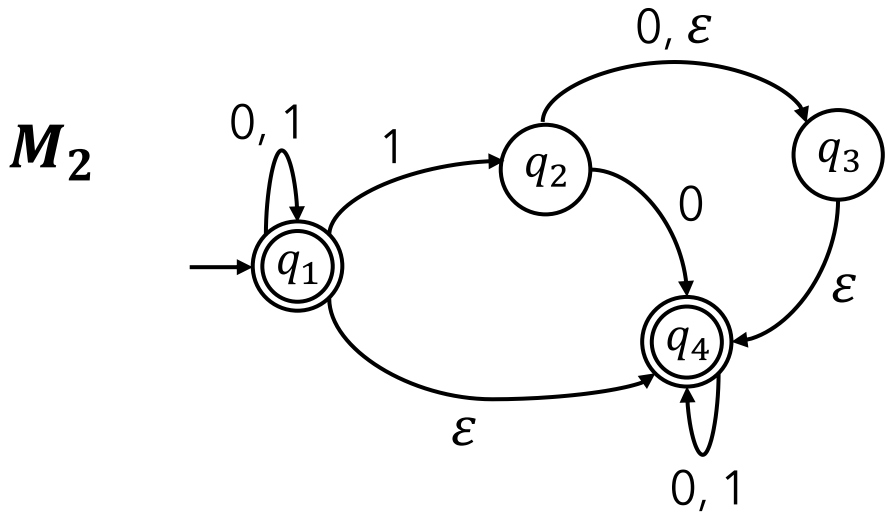
NFA Example 2



$\boldsymbol{M_2} = (\{\boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3}, \boldsymbol{q_4}\}, \{\mathbf{0}, \mathbf{1}\}, \boldsymbol{\delta}, \boldsymbol{q_1}, \{\boldsymbol{q_1}, \boldsymbol{q_4}\})$ with $\boldsymbol{\delta}: \boldsymbol{Q} \times (\boldsymbol{\Sigma} \cup \{\boldsymbol{\varepsilon}\}) \to \boldsymbol{\mathcal{P}}(\boldsymbol{Q})$ defined by

$\boldsymbol{\delta}$	$\mathbf{0}$	$\mathbf{1}$	$\boldsymbol{\varepsilon}$
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\{q_4\}$
q_2	$\{q_3, q_4\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\emptyset$	$\{q_4\}$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

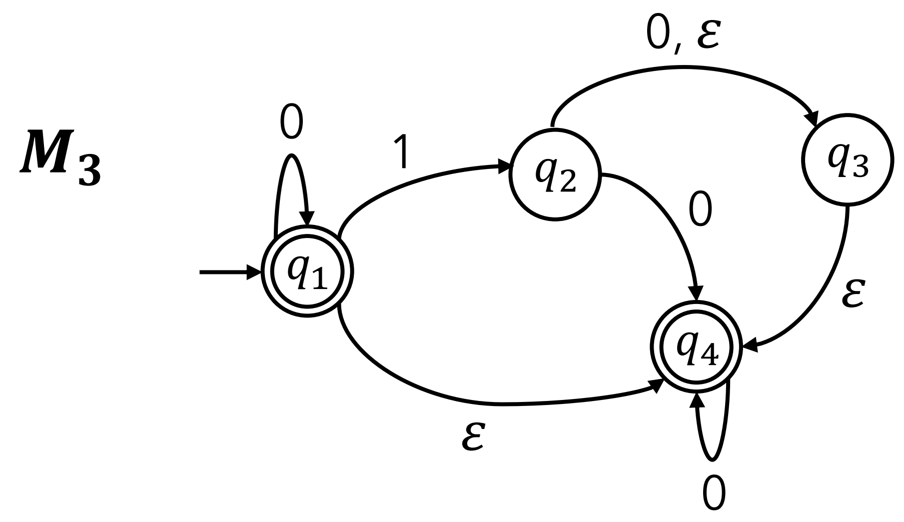
NFA Example 2



Does $L(M_2) = L(M_1)$?

- Every string in Σ^* is also accepted by M_2
- $L(M_2) = L(M_1)$

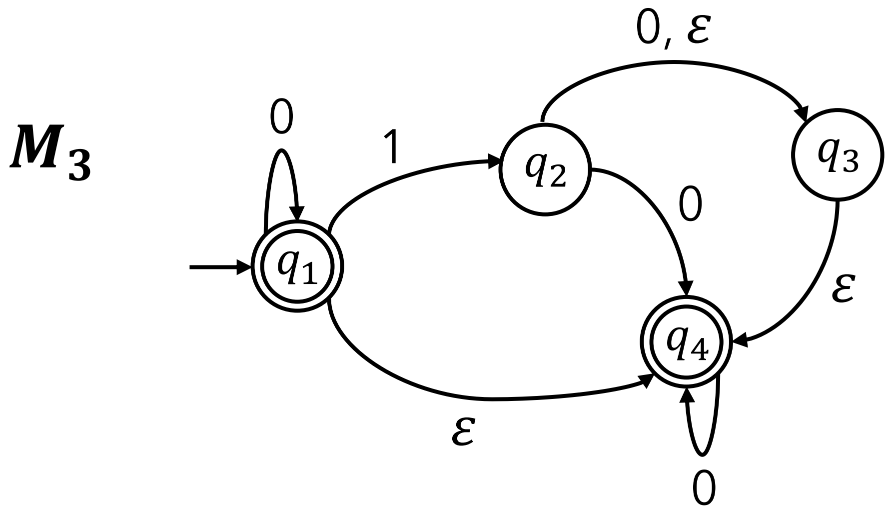
NFA Example 3



$M_3 = (\{q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_1, \{q_1, q_4\})$ with $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ defined by

δ	0	1	ε
q_1	$\{q_1\}$	$\{q_2\}$	$\{q_4\}$
q_2	$\{q_3, q_4\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\emptyset$	$\{q_4\}$
q_4	$\{q_4\}$	$\emptyset$	$\emptyset$

NFA Example 3



Does $L(M_3) = L(M_1)$?

- **No**, not every string in Σ^* is accepted by M_3
- E.g. $w = 11$ is not accepted by M_3

Formal Definition: Nondeterministic Finite Automaton

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

- Q is a **finite set** called the **states**
- Σ is a **finite set** called the **alphabet**
- $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$ is the **transition function**
- $q_0 \in Q$ is the **start state**
- $F \subseteq Q$ is the **set of accept states**

NFA: Formal Definition of Computation

Let $\boldsymbol{M} = (\boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{F})$ be an NFA and let $\boldsymbol{w} = \boldsymbol{w_1 w_2 \dots w_n}$ be a string over $\boldsymbol{\Sigma}$

Then $\boldsymbol{M}$ **accepts** $\boldsymbol{w}$ if

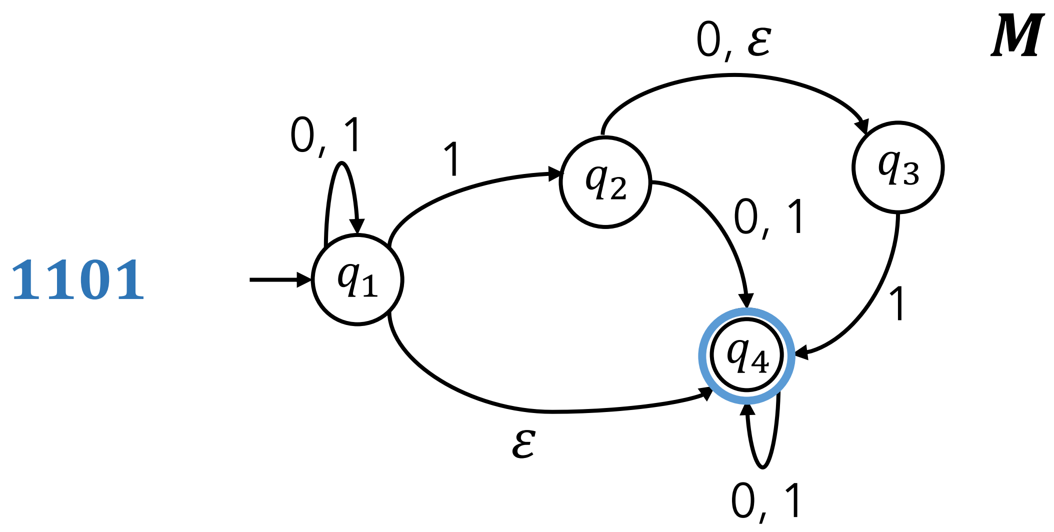
- we can write $\boldsymbol{w} = \boldsymbol{y_1 y_2 \dots y_m}$ with $\boldsymbol{y_i} \in \boldsymbol{\Sigma} \cup \{\boldsymbol{\varepsilon}\}$
- and there is a sequence of states $\boldsymbol{r_0, r_1, r_2, \dots, r_m}$ in $\boldsymbol{Q}$ such that

1. $\boldsymbol{r_0 = q_0}$
2. $\boldsymbol{\delta(r_i, y_{i+1}) = r_{i+1}}$
3. $\boldsymbol{r_m} \in \boldsymbol{F}$

$\boldsymbol{M}$ recognizes language $\boldsymbol{L}$ if $\boldsymbol{L} = \boldsymbol{L(M)} = \{\, \boldsymbol{w} \in \boldsymbol{\Sigma}^* \mid \boldsymbol{M} \text{ accepts } \boldsymbol{w} \,\}$

NFA Computation Example

Is the string $\boldsymbol{w} = \mathbf{1101}$ in $\boldsymbol{L(M)}$?



- We can rewrite $\mathbf{1101}$ as $\boldsymbol{w} = \mathbf{1}\boldsymbol{\varepsilon}\mathbf{101}$ which gives state sequence $q_1, q_1, q_4, q_4, q_4, q_4$
  - (There are also other execution paths from start state to accept state)
  - Note that the state sequence q_1, q_1, q_1, q_1, q_1 does not yield acceptance, but $\boldsymbol{w} \in \boldsymbol{L(M)}$

- Since there is at least one accepting execution path, $\boldsymbol{w} \in \boldsymbol{L(M)}$