Lecture 10: CNF and PDAs

CSC 320: Foundations of Computer Science

Quinton Yong

quintonyong@uvic.ca

University of Victoria

Ambiguous Grammars

- A string is derived **ambiguously** in a CFG if it has at **least two different leftmost derivations**
  - i.e. can derive the string in multiple ways even if always substituting the leftmost variable

- A context-free grammar is **ambiguous** if there **exists a string** that can be derived ambiguously

Ambiguous Grammars Example

Consider grammar $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{\Sigma}, \boldsymbol{R}, \boldsymbol{E})$ with $\boldsymbol{V} = \{\boldsymbol{E}, \boldsymbol{F}, \boldsymbol{T}\}$, $\boldsymbol{\Sigma} = \{\boldsymbol{a},\ +,\ \cdot\ ,\ (\ ,\ )\ \}$, and $\boldsymbol{R}$ is given by:

$$\boldsymbol{E} \rightarrow \boldsymbol{E} + \boldsymbol{E} \mid \boldsymbol{E} \cdot \boldsymbol{E} \mid (\boldsymbol{E}) \mid \boldsymbol{a}$$

Is this grammar ambiguous? **Yes**

Consider the string $\boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{a}$

- Leftmost derivation 1:

$$\boldsymbol{E} \Rightarrow \boldsymbol{E} \cdot \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{E} + \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{a}$$

- Leftmost derivation 2:

$$\boldsymbol{E} \Rightarrow \boldsymbol{E} + \boldsymbol{E} \Rightarrow \boldsymbol{E} \cdot \boldsymbol{E} + \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{E} + \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{a}$$

Ambiguous Grammars

- Often, we can **rewrite an ambiguous grammar** for a context free language in an **equivalent unambiguous way**

- However, not every context-free language has an unambiguous grammar which describes it (called **inherently ambiguous languages**)
  - Example: $\boldsymbol{L} = \{\mathbf{0}^{\boldsymbol{m}}\mathbf{1}^{\boldsymbol{n}}\mathbf{2}^{\boldsymbol{k}} \mid \boldsymbol{m} = \boldsymbol{n} \text{ or } \boldsymbol{m} = \boldsymbol{k}\}$

- We will learn how write a CFG in **Chomsky Normal Form** (**CNF**), which can often make the grammar unambiguous

Chomsky Normal Form (CNF)

- A context-free grammar $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{\Sigma}, \boldsymbol{R}, \boldsymbol{S})$ is in **Chomsky Normal Form** (**CNF**) if every rule is of the form

$$\boldsymbol{A} \to \boldsymbol{BC} \text{ or } \boldsymbol{A} \to \boldsymbol{a}$$

Right side can be **two variables** or **one terminal**, nothing else

where:

  - $\boldsymbol{a} \in \boldsymbol{\Sigma}$
  - $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in \boldsymbol{V}$
  - $\boldsymbol{B}, \boldsymbol{C}$ may not be the start variable

    Start variable **cannot be on the right side** of a rule
  - $\boldsymbol{S} \to \boldsymbol{\epsilon}$ is permitted where $\boldsymbol{S}$ is the start variable

    No other $\boldsymbol{\varepsilon}$-substitutions permitted

Chomsky Normal Form (CNF)

Theorem: Any context-free language is generated by a context-free grammar in Chomsky Normal Form (CNF)

Proof: We will show how to convert any context-free grammar into CNF without changing the language

If any rule in a CFG **violates a CNF condition**, replace it with **equivalent rule(s)** that satisfy CNF

1. Add a **new start variable** $\boldsymbol{S_0}$
2. Eliminate all **$\boldsymbol{\varepsilon}$-rules** of form $\boldsymbol{A} \rightarrow \boldsymbol{\varepsilon}$
3. Eliminate all **unit rules** of form $\boldsymbol{A} \rightarrow \boldsymbol{B}$
4. Convert remaining rules

Converting a CFG to CNF

1. Add a **new start variable** $\boldsymbol{S_0}$

- Let $\boldsymbol{S} \in \boldsymbol{V}$ be the previous start variable
- Let $\boldsymbol{S_0} \notin \boldsymbol{V}$

- Add new start variable $\boldsymbol{S_0}$ and rule $\boldsymbol{S_0} \rightarrow \boldsymbol{S}$

- Thus, the new start variable will **not appear on the right** of a rule

Converting a CFG to CNF

2. Eliminate all **ε-rules** of form $A \to \varepsilon$

Repeat until **all ε-rules not involving** S_0 are eliminated:

- Given $A \to \varepsilon$ where $A \neq S_0$
- For each $W \to uAv$ where u, v are strings of variables and terminals, **add new rule** $W \to uv$
- For $W \to A$, **add** $W \to \varepsilon$ (which we will **remove later**) unless $W \to \varepsilon$ was previously removed
- Remove $A \to \varepsilon$

Converting a CFG to CNF

3. Eliminate all **unit rules** of form $A \to B$

Repeat until **all unit rules** are eliminated:

- Given $A \to B$
- For each rule $B \to u$, add $A \to u$ (unless $A \to u$ was previously removed)
  - As before, u is a **string of variables and terminals**
  - Basically, just copying $B \to$ rules into $A \to$ rules
- Remove $A \to B$

Converting a CFG to CNF

4. Convert remaining rules

- Replace rules $\boldsymbol{A} \to \boldsymbol{bC}$ with $\boldsymbol{A} \to \boldsymbol{BC}$, $\boldsymbol{B} \to \boldsymbol{b}$
- Replace rules $\boldsymbol{A} \to \boldsymbol{Bc}$ with $\boldsymbol{A} \to \boldsymbol{BC}$, $\boldsymbol{C} \to \boldsymbol{c}$
- Replace rules $\boldsymbol{A} \to \boldsymbol{bc}$ with $\boldsymbol{A} \to \boldsymbol{BC}$, $\boldsymbol{B} \to \boldsymbol{b}$, $\boldsymbol{C} \to \boldsymbol{c}$
- Replace rules $\boldsymbol{A} \to \boldsymbol{u_1 u_2 \ldots u_k}$, where $\boldsymbol{k \geq 3}$ and each $\boldsymbol{u_i}$ is a **variable** / **terminal**, with $\boldsymbol{A} \to \boldsymbol{u_1 A_1}$, $\boldsymbol{A_1} \to \boldsymbol{u_2 A_2}$, $\boldsymbol{A_2} \to \boldsymbol{u_3 A_3}$, … , and $\boldsymbol{A_{k-2}} \to \boldsymbol{u_{k-1} u_k}$
  - Replace any **terminal** $\boldsymbol{u_i}$ with variable $\boldsymbol{U_i}$ and add rule $\boldsymbol{U_i} \to \boldsymbol{u_i}$
  - E.g. $\boldsymbol{A} \to \boldsymbol{BcD}$ becomes $\boldsymbol{A} \to \boldsymbol{BA_1}$, $\boldsymbol{A_1} \to \boldsymbol{U_1 D}$, $\boldsymbol{U_1} \to \boldsymbol{c}$

Example: Converting a CFG to CNF

Convert the following grammar $\boldsymbol{G}$ into CNF:

$$\boldsymbol{S} \rightarrow \boldsymbol{ASA} \mid \boldsymbol{aB}$$

$$\boldsymbol{A} \rightarrow \boldsymbol{B} \mid \boldsymbol{S}$$

$$\boldsymbol{B} \rightarrow \boldsymbol{b} \mid \boldsymbol{\varepsilon}$$

Example: Converting a CFG to CNF

Convert the following grammar $\boldsymbol{G}$ into CNF:

$$S \to ASA \mid aB$$

$$A \to B \mid S$$

$$B \to b \mid \varepsilon$$

$$S_0 \to S$$

$$S \to ASA \mid aB$$

$$A \to B \mid S$$

$$B \to b \mid \varepsilon$$

1. Add a **new start variable** S_0

Example: Converting a CFG to CNF

Convert the following grammar $\boldsymbol{G}$ into CNF:

$$S_0 \to S$$

$$S \to ASA \mid aB$$

$$A \to B \mid S$$

$$B \to b \mid \varepsilon$$

2. Eliminate all **ε-rules** of form $A \to \varepsilon$

Example: Converting a CFG to CNF

Removed:
$B \to \varepsilon$

Convert the following grammar G into CNF:

$S_0 \to S$

$S \to ASA \mid a\boxed{B}$

$A \to \boxed{B} \mid S$

$\boxed{B} \to b \mid \boxed{\varepsilon}$

$S_0 \to S$

$S \to ASA \mid aB \mid a$

$A \to B \mid S \mid \varepsilon$

$B \to b$

- For each $W \to uAv$ where u, v are strings of variables and terminals, **add new rule** $W \to uv$
- For $W \to A$, **add** $W \to \varepsilon$ (which we will **remove later**) unless $W \to \varepsilon$ was previously removed
- Remove $A \to \varepsilon$

Example: Converting a CFG to CNF

Removed:
$B \to \varepsilon$, $A \to \varepsilon$

Convert the following grammar G into CNF:

$S_0 \to S$

$S \to \boxed{A}S\boxed{A} \mid aB \mid a$

$\boxed{A} \to B \mid S \mid \boxed{\varepsilon}$

$B \to b$

$S_0 \to S$

$S \to ASA \mid aB \mid a \mid SA \mid AS \cancel{\mid S}$

$A \to B \mid S$

$B \to b$

- For each $W \to uAv$ where u, v are strings of variables and terminals, **add new rule** $W \to uv$
- For $W \to A$, **add** $W \to \varepsilon$ (which we will **remove later**) unless $W \to \varepsilon$ was previously removed
- Remove $A \to \varepsilon$

Example: Converting a CFG to CNF

Removed:
$\boldsymbol{B \to \varepsilon}$, $\boldsymbol{A \to \varepsilon}$

Convert the following grammar $\boldsymbol{G}$ into CNF:

$$S_0 \to S$$

$$S \to ASA \mid aB \mid a \mid SA \mid AS$$

$$A \to B \mid S$$

$$B \to b$$

3. Eliminate all **unit rules** of form $A \to B$

Example: Converting a CFG to CNF

Removed:
$B \rightarrow \varepsilon$, $A \rightarrow \varepsilon$,
$S_0 \rightarrow S$

Convert the following grammar G into CNF:

$\boxed{S_0} \rightarrow \boxed{S}$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow B \mid S$

$B \rightarrow b$

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow B \mid S$

$B \rightarrow b$

- For each rule $B \rightarrow u$, add $A \rightarrow u$ (unless $A \rightarrow u$ was previously removed)
  - As before, u is a **string of variables and terminals**
  - Basically, just copying $B \rightarrow$ rules into $A \rightarrow$ rules
- Remove $A \rightarrow B$

Example: Converting a CFG to CNF

Removed:
$B \to \varepsilon$, $A \to \varepsilon$,
$S_0 \to S$, $A \to B$

Convert the following grammar G into CNF:

$S_0 \to ASA \mid aB \mid a \mid SA \mid AS$

$S \to ASA \mid aB \mid a \mid SA \mid AS$

$\boxed{A} \to \boxed{B} \mid S$

$B \to b$

$S_0 \to ASA \mid aB \mid a \mid SA \mid AS$

$S \to ASA \mid aB \mid a \mid SA \mid AS$

$A \to S \mid b$

$B \to b$

- For each rule $B \to u$, add $A \to u$ (unless $A \to u$ was previously removed)
    - As before, u is a **string of variables and terminals**
    - Basically, just copying $B \to$ rules into $A \to$ rules
- Remove $A \to B$

Example: Converting a CFG to CNF

Removed:
$B \to \varepsilon$, $A \to \varepsilon$,
$S_0 \to S$, $A \to B$, $A \to S$

Convert the following grammar G into CNF:

$S_0 \to ASA \mid aB \mid a \mid SA \mid AS$

$S \to ASA \mid aB \mid a \mid SA \mid AS$

$\boxed{A} \to \boxed{S} \mid b$

$B \to b$

$S_0 \to ASA \mid aB \mid a \mid SA \mid AS$

$S \to ASA \mid aB \mid a \mid SA \mid AS$

$A \to b \mid ASA \mid aB \mid a \mid SA \mid AS$

$B \to b$

- For each rule $B \to u$, add $A \to u$ (unless $A \to u$ was previously removed)
  - As before, u is a **string of variables and terminals**
  - Basically, just copying $B \to$ rules into $A \to$ rules
- Remove $A \to B$

Example: Converting a CFG to CNF

Convert the following grammar $\boldsymbol{G}$ into CNF:

$$S_0 \to ASA \mid aB \mid a \mid SA \mid AS$$

$$S \to ASA \mid aB \mid a \mid SA \mid AS$$

$$A \to b \mid ASA \mid aB \mid a \mid SA \mid AS$$

$$B \to b$$

4. Convert remaining rules

Example: Converting a CFG to CNF

Convert the following grammar $\boldsymbol{G}$ into CNF:

$S_0 \to \boxed{ASA} \mid aB \mid a \mid SA \mid AS$

$S \to \boxed{ASA} \mid aB \mid a \mid SA \mid AS$

$A \to b \mid \boxed{ASA} \mid aB \mid a \mid SA \mid AS$

$B \to b$

$S_0 \to AA_1 \mid aB \mid a \mid SA \mid AS$

$S \to AA_1 \mid aB \mid a \mid SA \mid AS$

$A \to b \mid AA_1 \mid aB \mid a \mid SA \mid AS$

$B \to b$

$A_1 \to SA$

- Replace rules $A \to u_1 u_2 \ldots u_k$, where $k \geq 3$ and each u_i is a **variable** / **terminal**, with $A \to u_1 A_1$, $A_1 \to u_2 A_2$, $A_2 \to u_3 A_3$, … , and $A_{k-2} \to u_{k-1} u_k$
  - Replace any **terminal** u_i with variable U_i and add rule $U_i \to u_i$
  - E.g. $A \to BcD$ becomes $A \to BA_1$, $A_1 \to U_1 D$, $U_1 \to c$

Example: Converting a CFG to CNF

Convert the following grammar $\boldsymbol{G}$ into CNF:

$S_0 \rightarrow AA_1 \mid \boxed{aB} \mid a \mid SA \mid AS$

$S \rightarrow AA_1 \mid \boxed{aB} \mid a \mid SA \mid AS$

$A \rightarrow b \mid AA_1 \mid \boxed{aB} \mid a \mid SA \mid AS$

$B \rightarrow b$

$A_1 \rightarrow SA$

$S_0 \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$

$S \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$

$A \rightarrow b \mid AA_1 \mid UB \mid a \mid SA \mid AS$

$B \rightarrow b$

$A_1 \rightarrow SA$

$U \rightarrow a$

- Replace rules $A \rightarrow bC$ with $A \rightarrow BC$, $B \rightarrow b$

Example: Converting a CFG to CNF

Convert the following grammar $\boldsymbol{G}$ into CNF:

$$S \to ASA \mid aB$$
$$A \to B \mid S$$
$$B \to b \mid \varepsilon$$

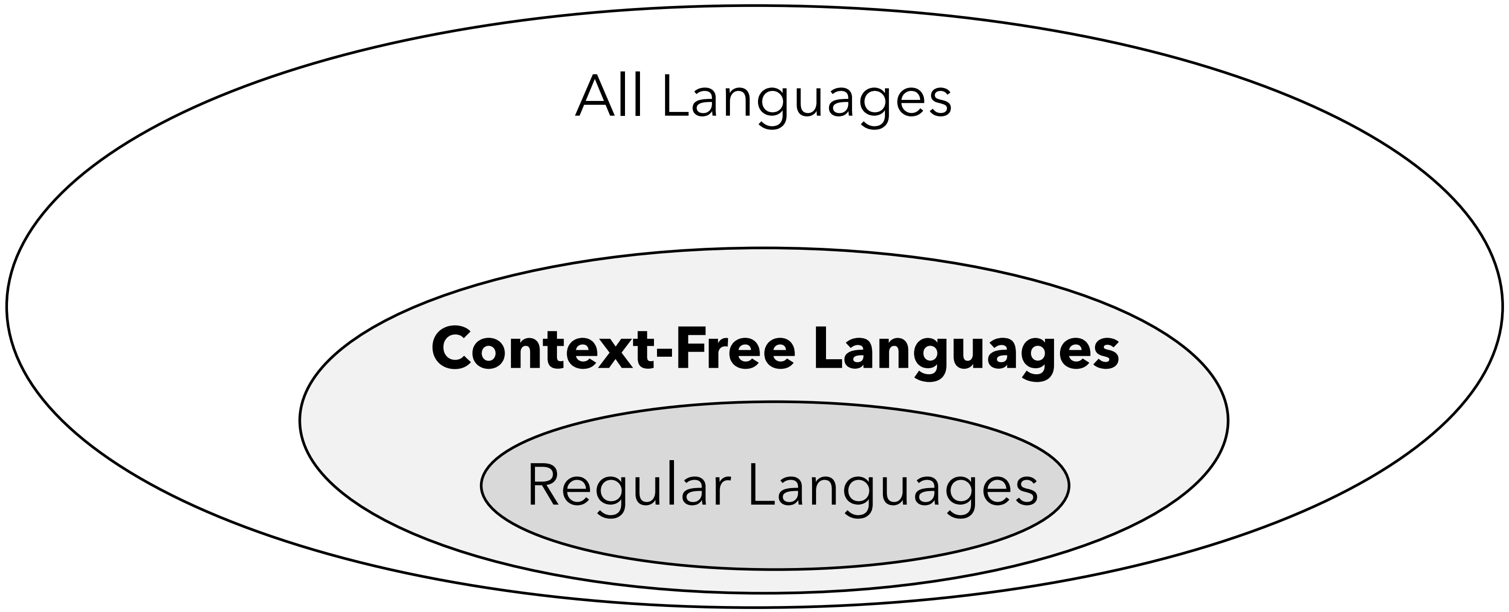
$$\Rightarrow$$

$$S_0 \to AA_1 \mid UB \mid a \mid SA \mid AS$$
$$S \to AA_1 \mid UB \mid a \mid SA \mid AS$$
$$A \to b \mid AA_1 \mid UB \mid a \mid SA \mid AS$$
$$B \to b$$
$$A_1 \to SA$$
$$U \to a$$

A context-free grammar is in **Chomsky Normal Form** (**CNF**) if every rule is of the form $A \to BC$ or $A \to a$ where:

- $a \in \Sigma$
- $A, B, C \in V$
- B, C may not be the start variable
- $S \to \epsilon$ is permitted where S is the start variable

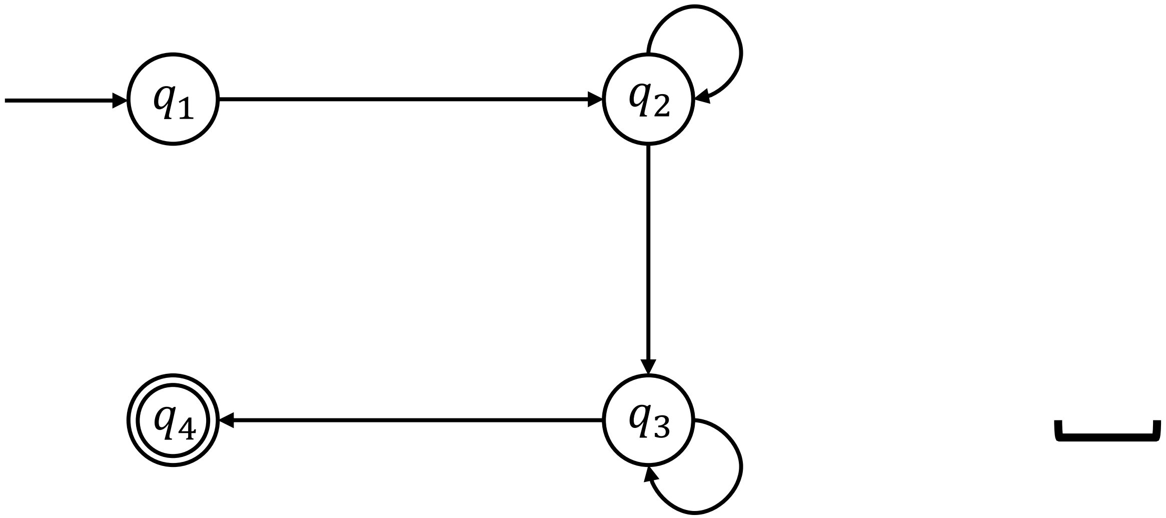
Pushdown Automata



- The class of context-free languages is the class of languages recognized by context-free grammars

- Next, we want to show that the set of context-free languages is exactly the set of languages recognized by **pushdown automata**

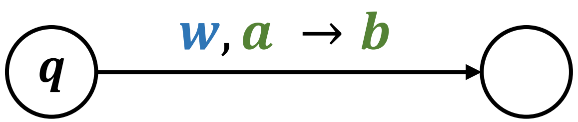
Pushdown Automata (PDA)



- A **pushdown automaton** (**PDA**) is essentially an **NFA with a stack**
- The stack provides **additional memory** to recognize more complex languages

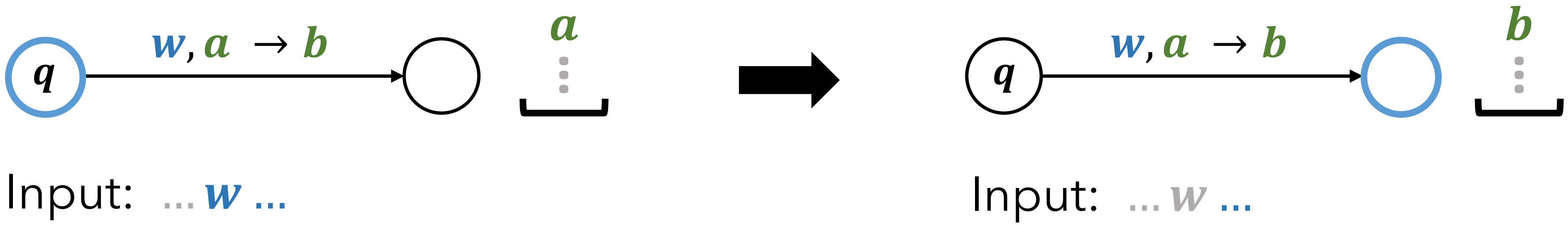
Pushdown Automata Transitions

Pushdown automata **transitions** use the following notation:



In state q, we can take the transition when:

- **reading symbol** w from input **and top stack symbol is** a
- **pop** a from the top of the stack and **push** b onto the top the stack



Pushdown Automata ε-Transitions

- If $w = \varepsilon$, then input symbol is ignored

$$\xrightarrow{\quad \varepsilon, a \ \to b \quad}$$

- If $a = \varepsilon$, then **top stack symbol is ignored** and b **is pushed onto the stack**

$$\xrightarrow{\quad w, \varepsilon \ \to b \quad}$$

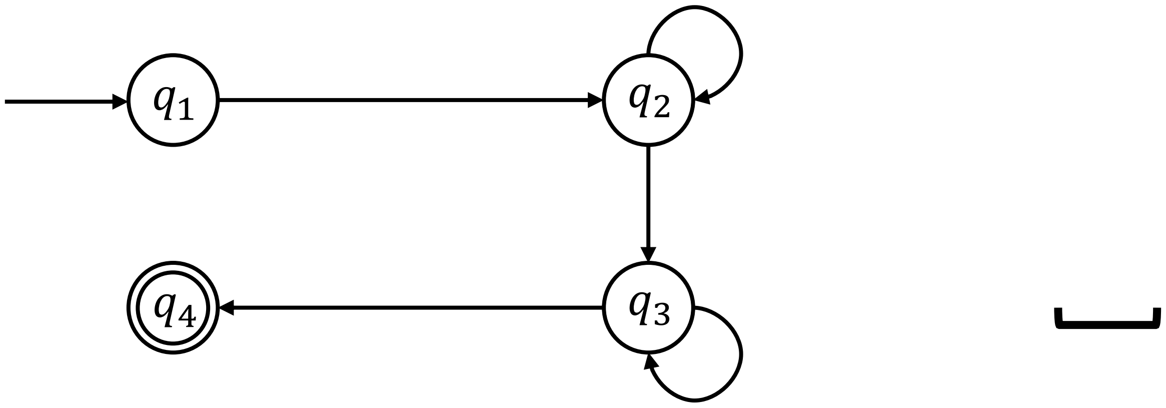
- If $b = \varepsilon$, then **top stack symbol is removed** from the stack

$$\xrightarrow{\quad w, a \ \to \varepsilon \quad}$$

- If $w = a = b = \varepsilon$, then the transition is a **free transition**

$$\xrightarrow{\quad \varepsilon, \varepsilon \ \to \varepsilon \quad}$$

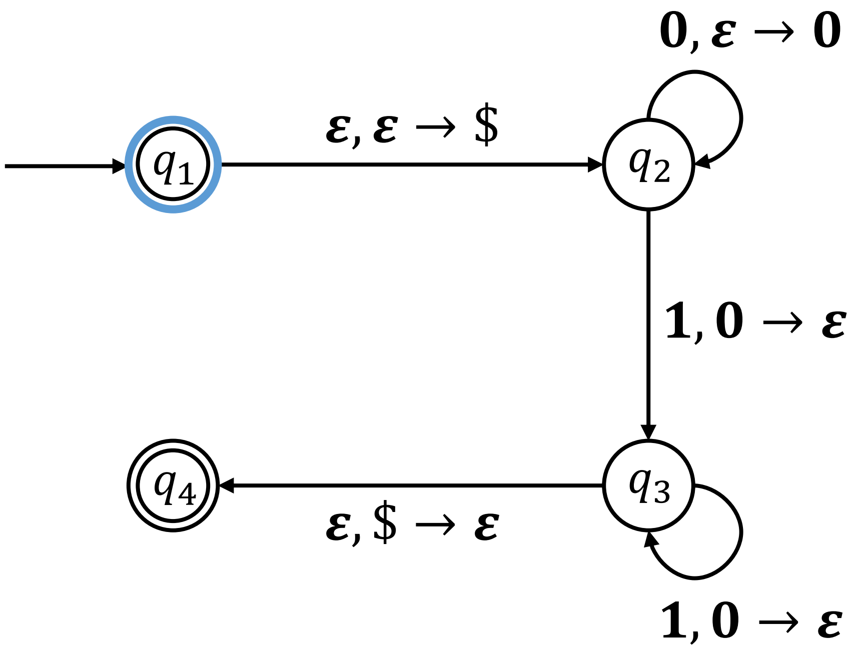
Pushdown Automata Acceptance



- PDA $\boldsymbol{M}$ accepts an input string $\boldsymbol{w}$ similar to an NFA
- $\boldsymbol{M}$ **accepts** $\boldsymbol{w}$ if:
  - starting with **empty stack**
  - there is any execution path that **reads the entire input** using transitions
  - and end on an **accept state**
- All strings which are accepted by a PDA $\boldsymbol{M}$ is the **language recognized by** $\boldsymbol{M}$, denoted $\boldsymbol{L}(\boldsymbol{M})$

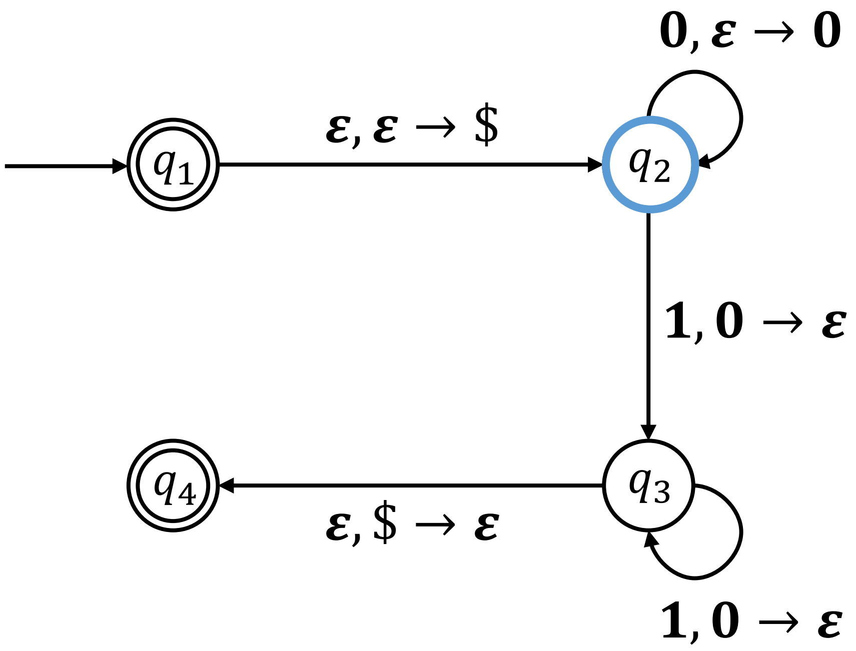
Pushdown Automata Example

$w = 0011$



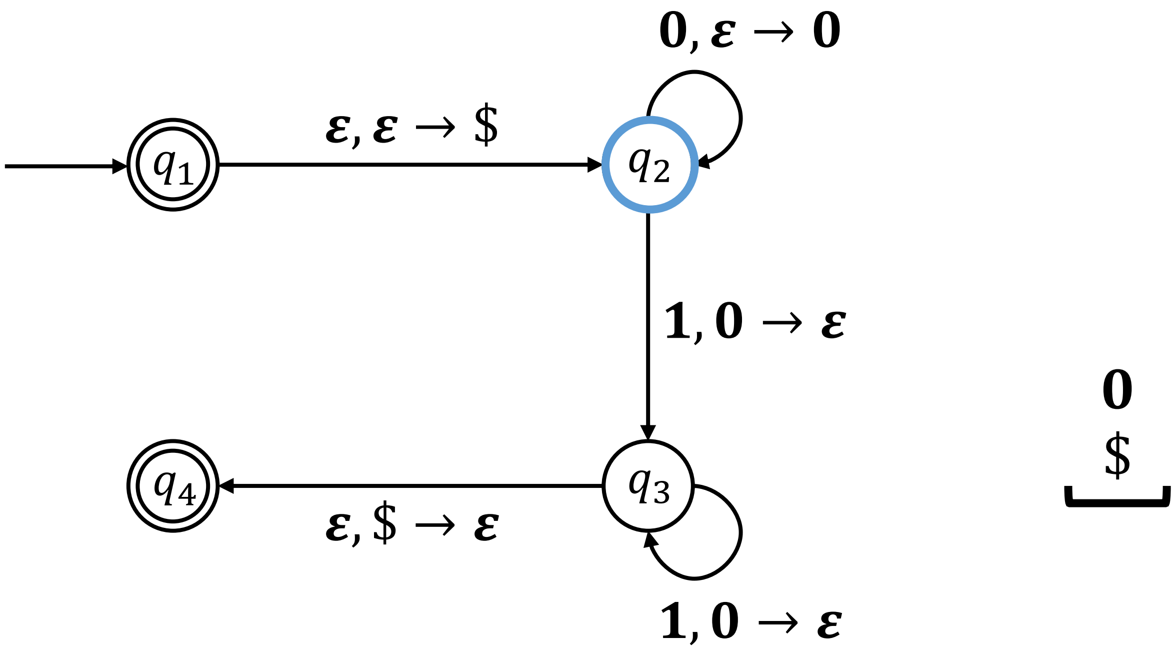
Pushdown Automata Example

$w = \mathbf{0011}$



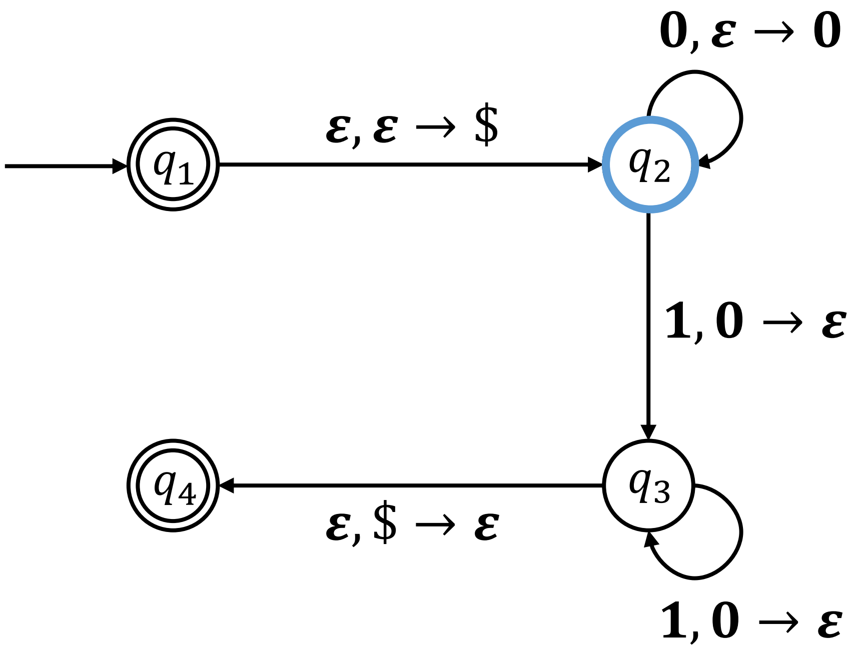
Pushdown Automata Example

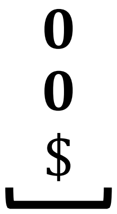
$w = 0011$



Pushdown Automata Example

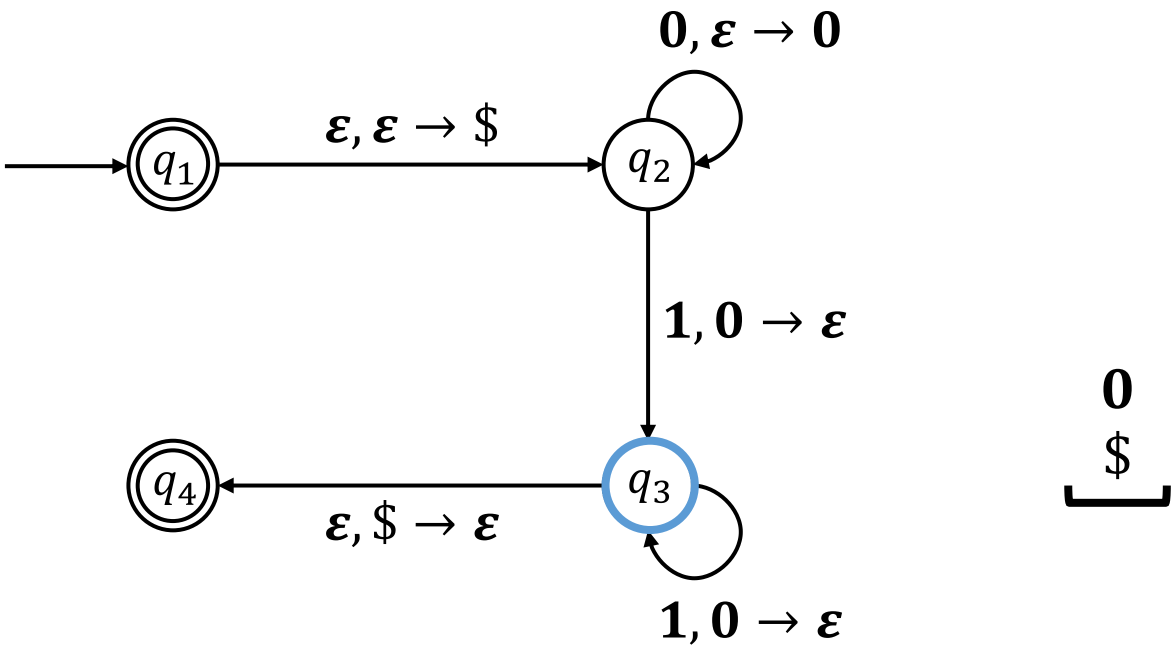
$w = 0011$





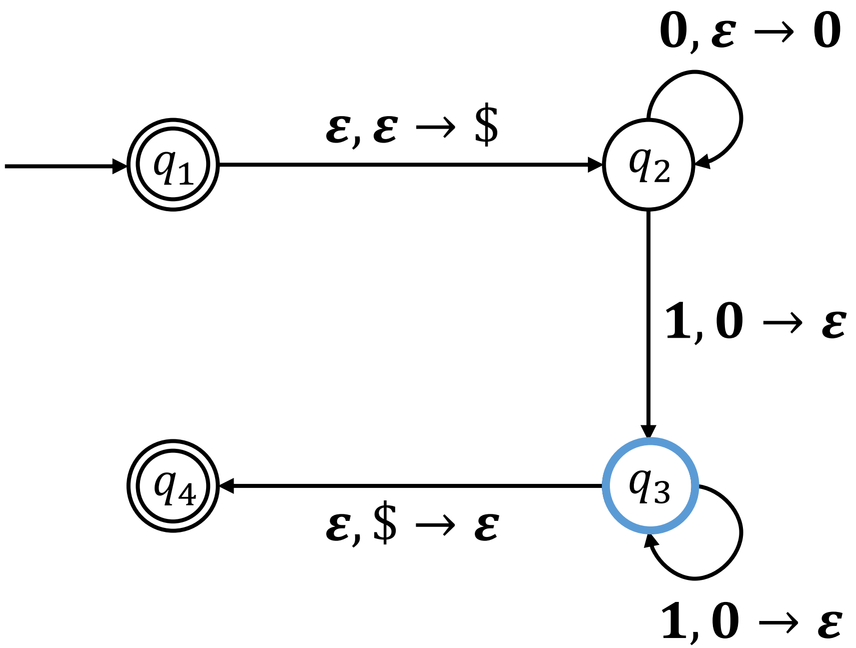
Pushdown Automata Example

$w = 0011$



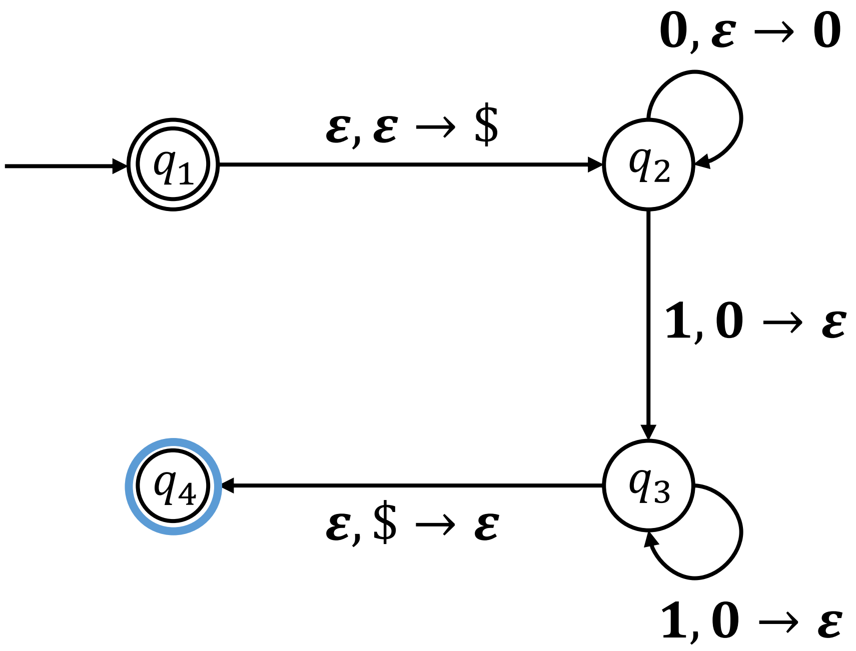
Pushdown Automata Example

$w = 0011$

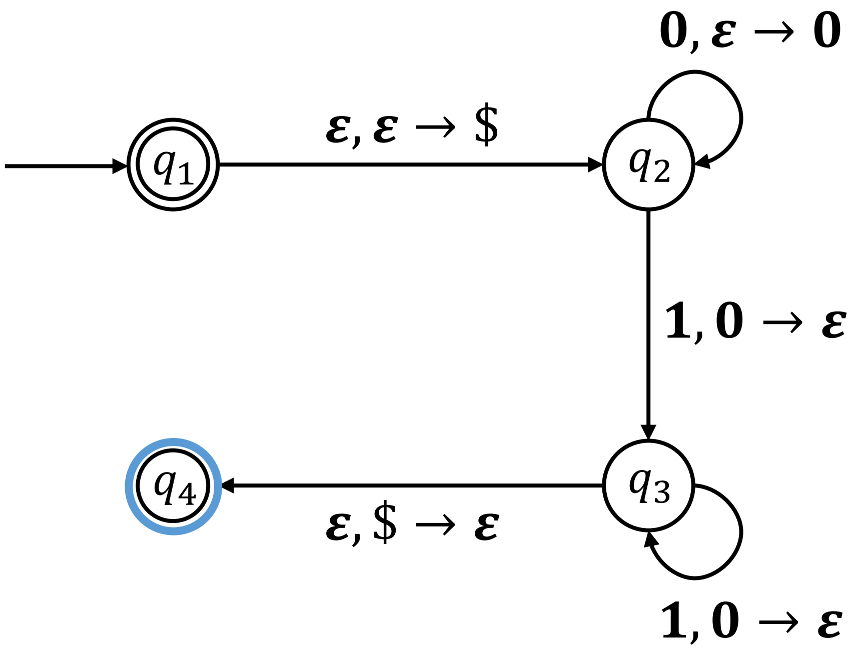


Pushdown Automata Example

$w = 0011$



Pushdown Automata Example

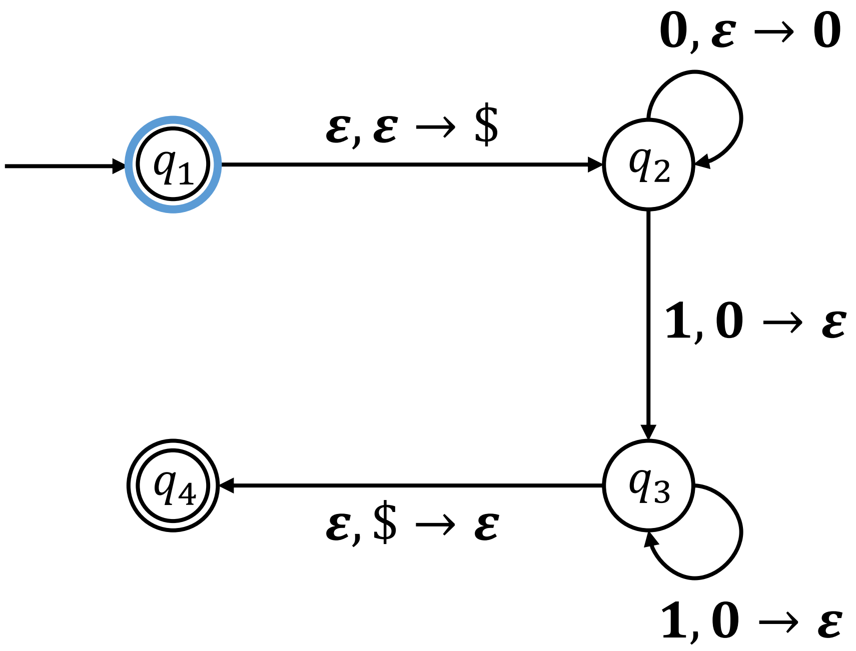


$\boldsymbol{w} = \mathbf{0011}$

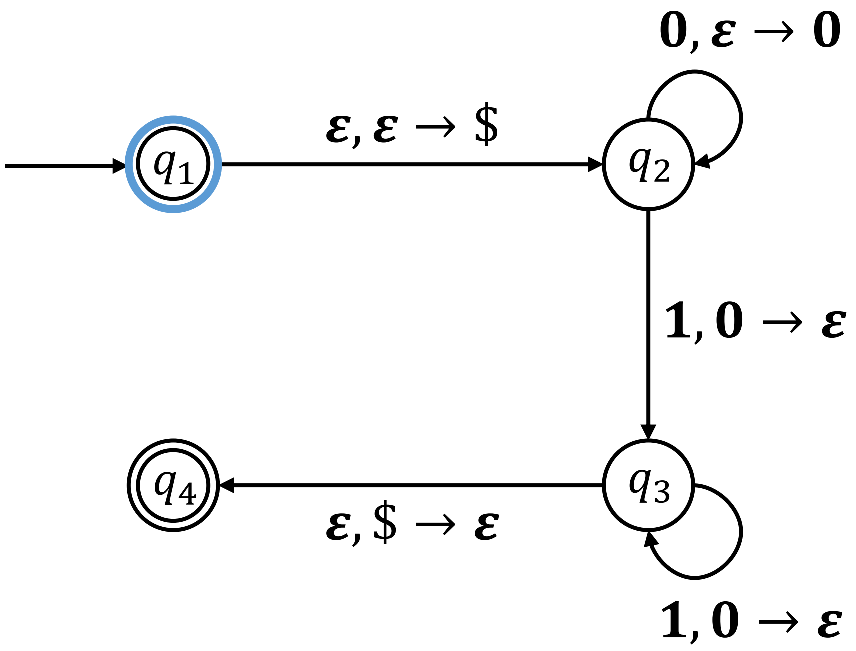
- $\boldsymbol{w} = \mathbf{0011}$ is accepted

Pushdown Automata Example

$w = \varepsilon$



Pushdown Automata Example

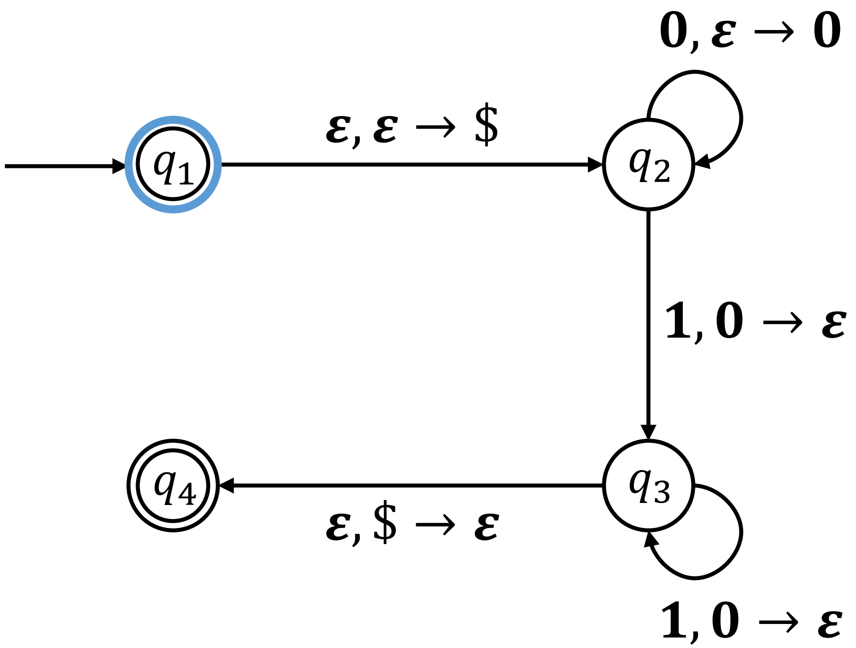


$\boldsymbol{w} = \boldsymbol{\varepsilon}$

- $\boldsymbol{w} = \boldsymbol{\varepsilon}$ is accepted

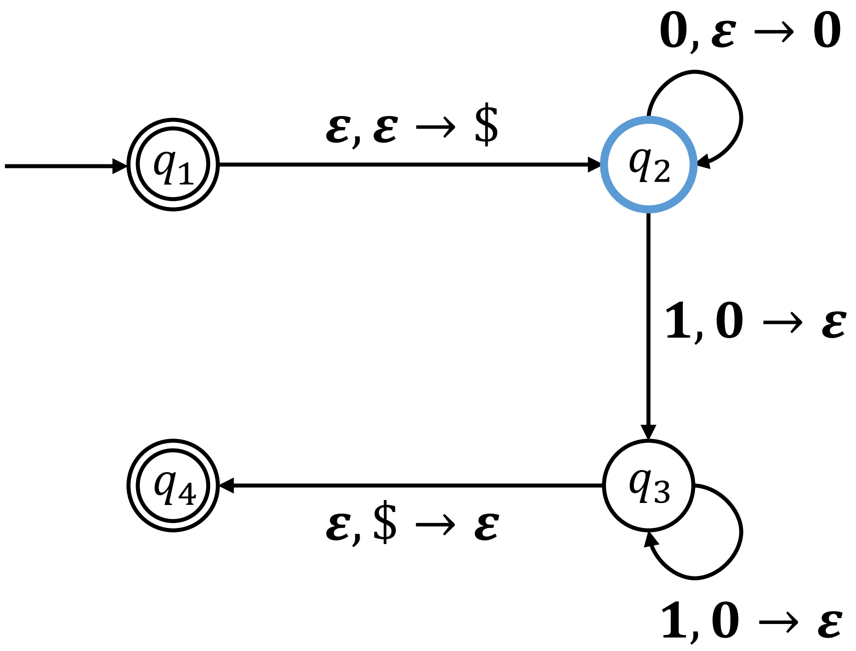
Pushdown Automata Example

$w = 00011$



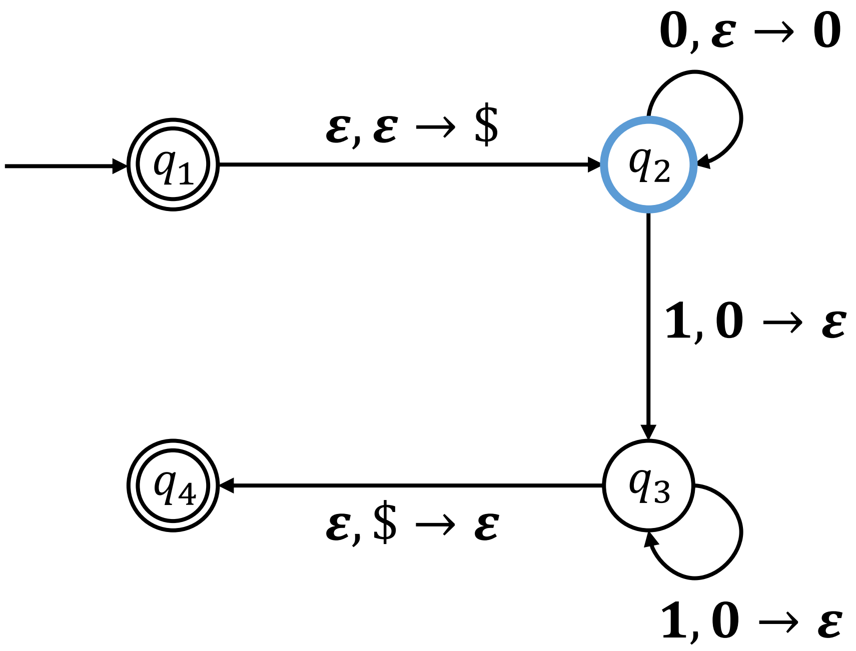
Pushdown Automata Example

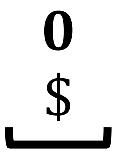
$w = 00011$



Pushdown Automata Example

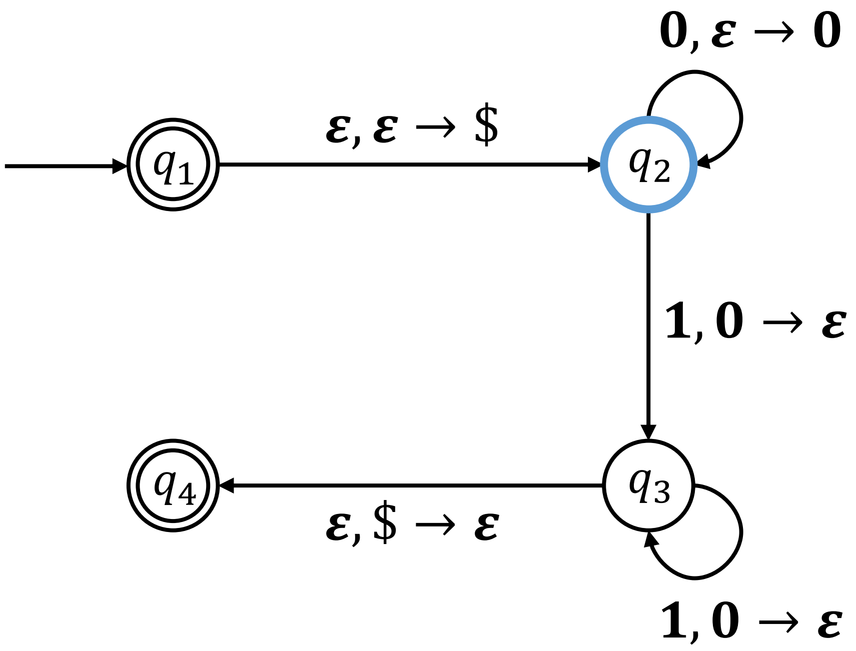
$w = 00011$

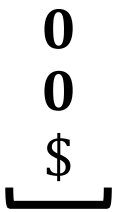




Pushdown Automata Example

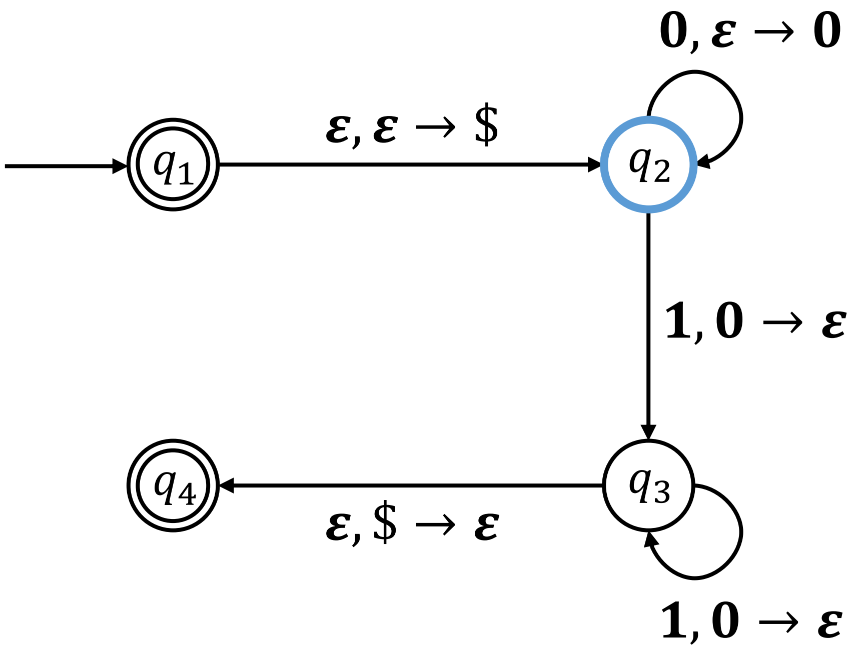
$w = 00011$

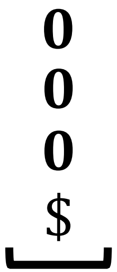




Pushdown Automata Example

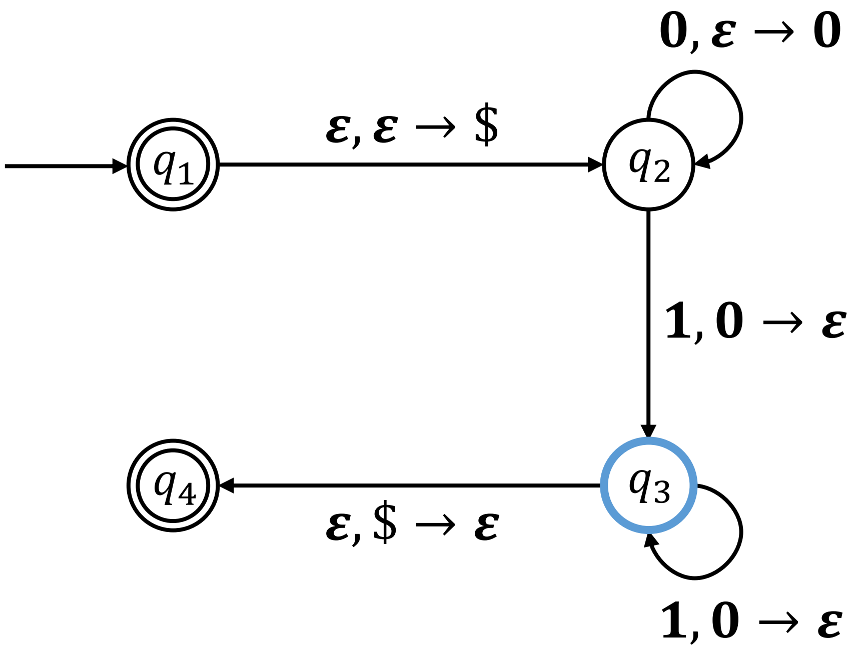
$w = 00011$





Pushdown Automata Example

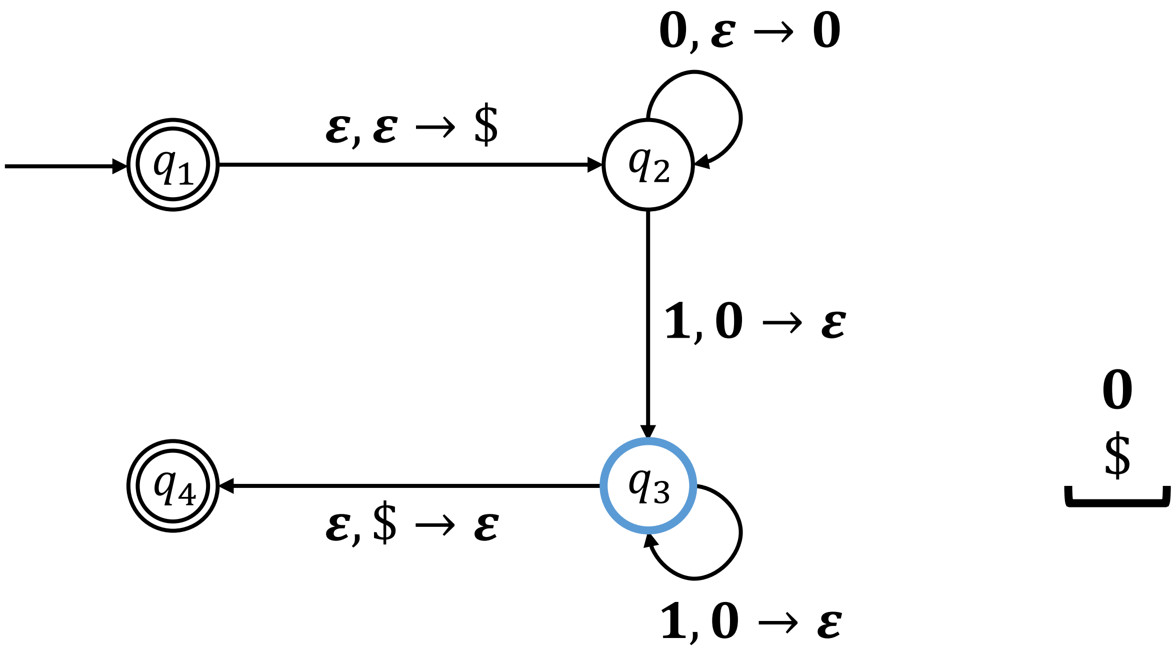
$w = 00011$



0
0
\$

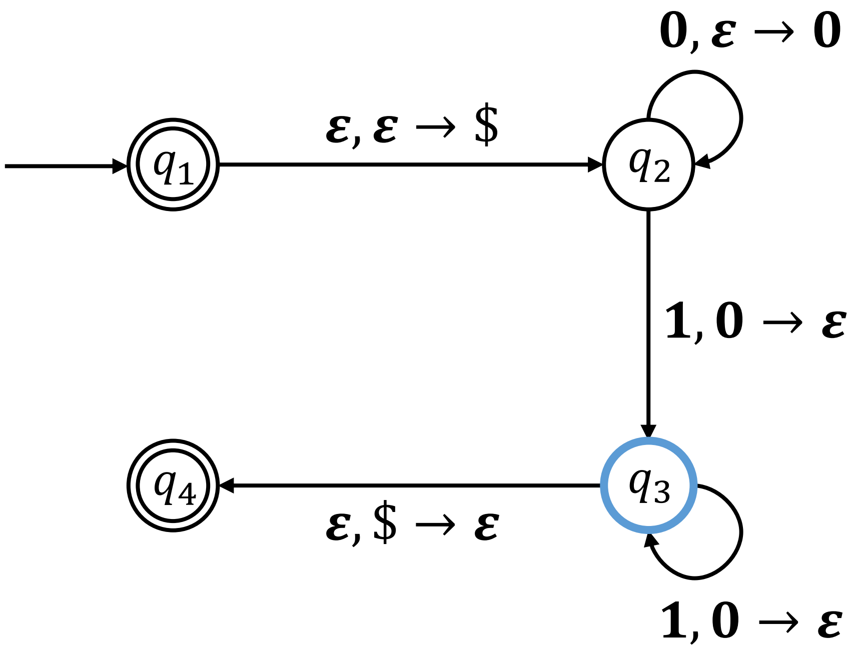
Pushdown Automata Example

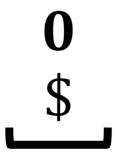
$w = 00011$



Pushdown Automata Example

$w = \mathbf{00011}$

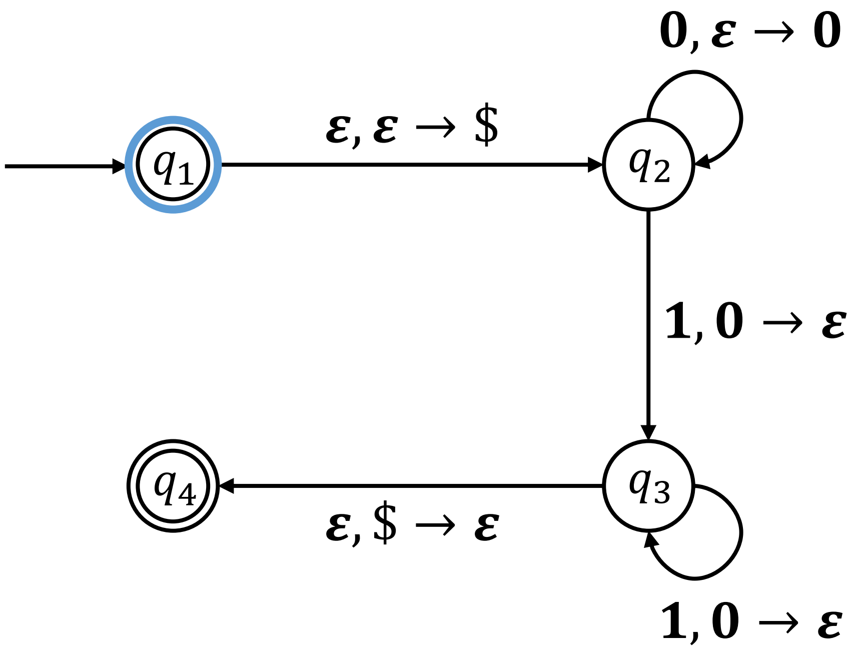




- $w = \mathbf{00011}$ is not accepted

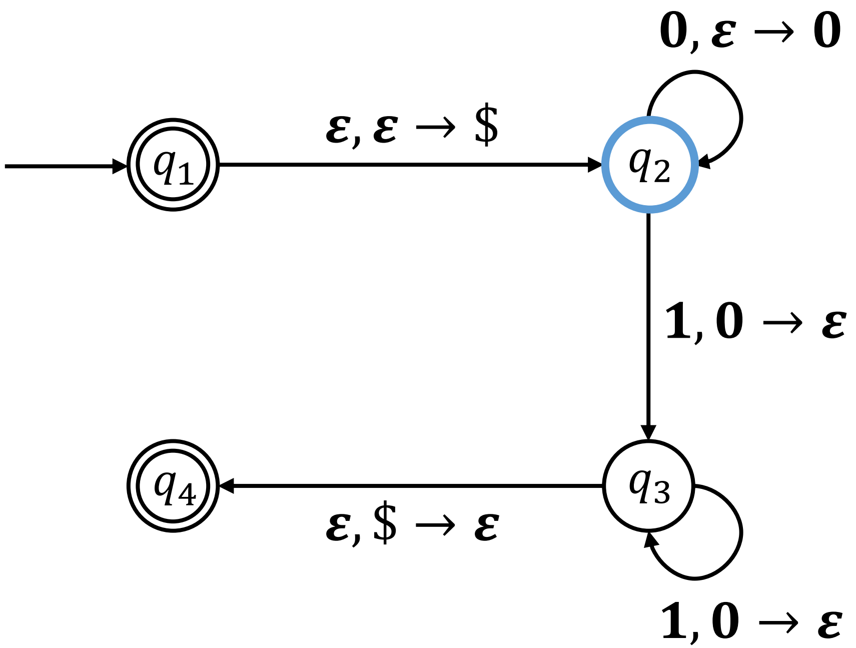
Pushdown Automata Example

$w = 00111$



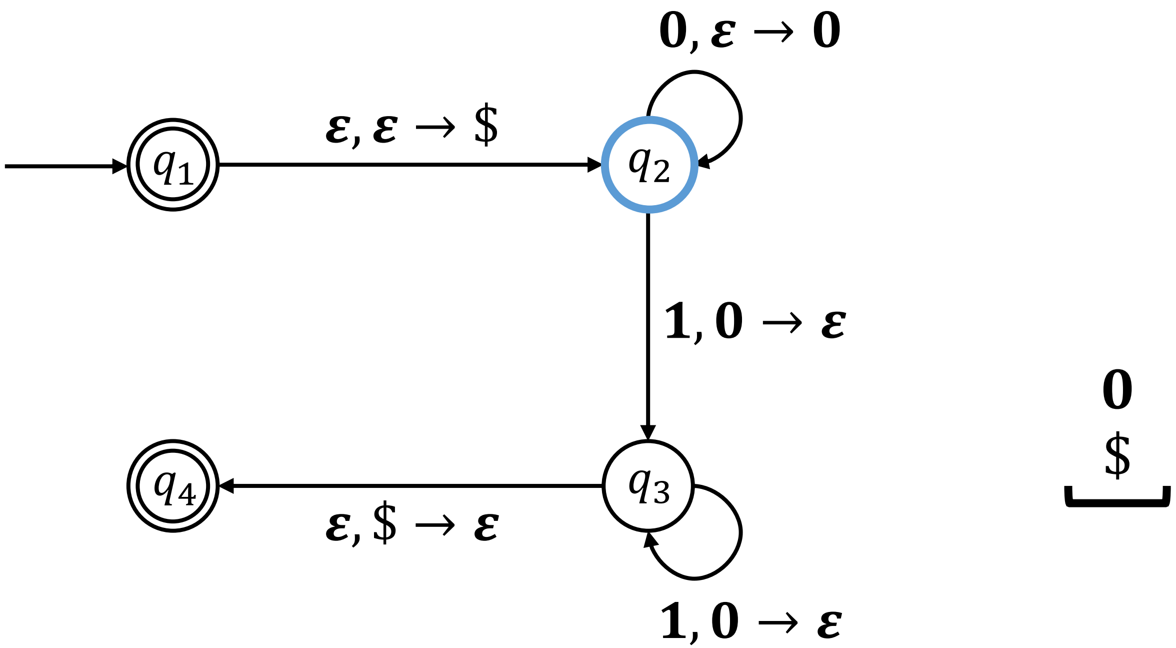
Pushdown Automata Example

$w = 00111$



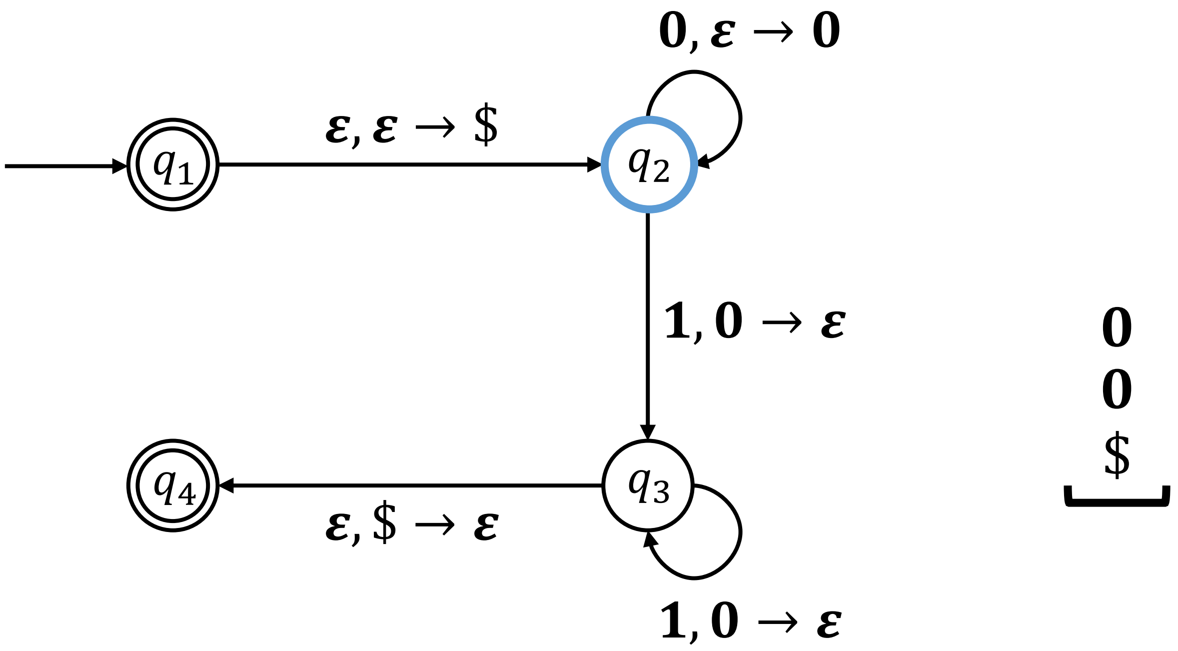
Pushdown Automata Example

$w = 00111$



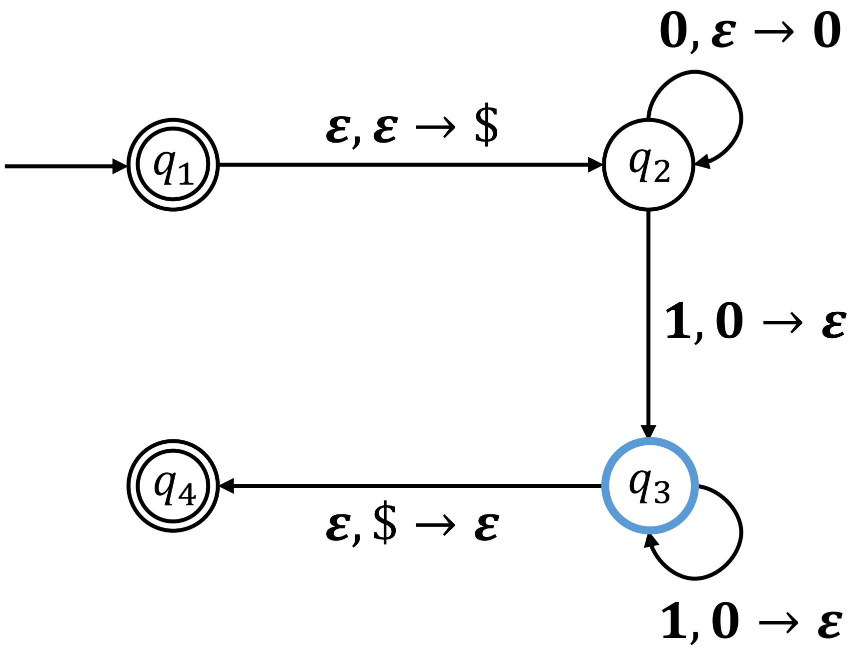
Pushdown Automata Example

$w = 00111$



Pushdown Automata Example

$w = \color{gray}{001}\mathbf{11}$

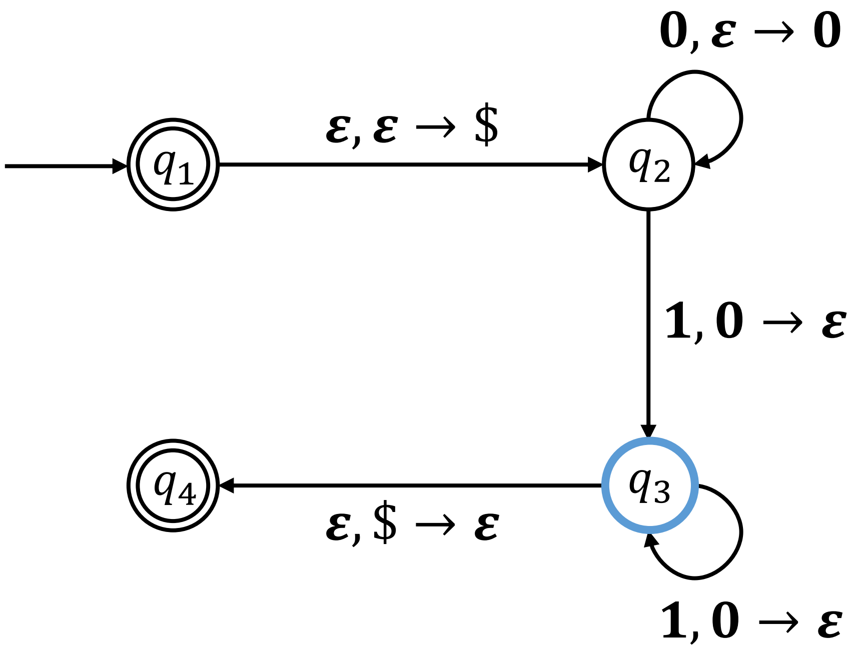


$\mathbf{0}$

$\$$

Pushdown Automata Example

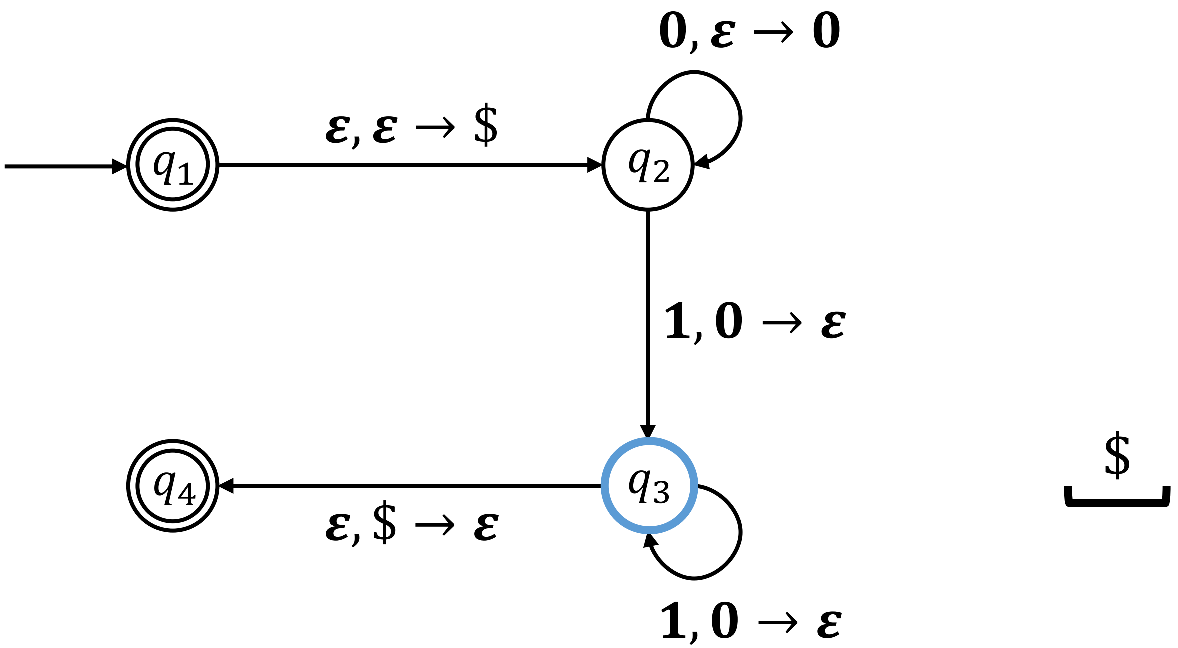
$w = 00111$



$\$$

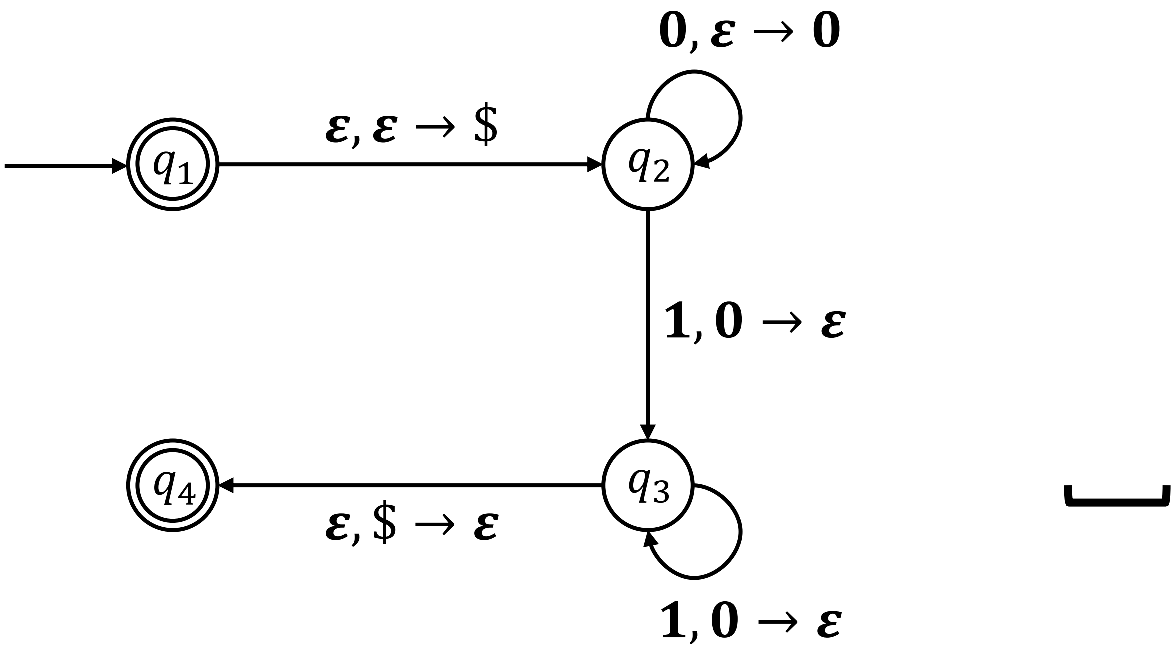
Pushdown Automata Example

$w = \mathbf{00111}$



- $w = \mathbf{00111}$ is not accepted

Pushdown Automata Example



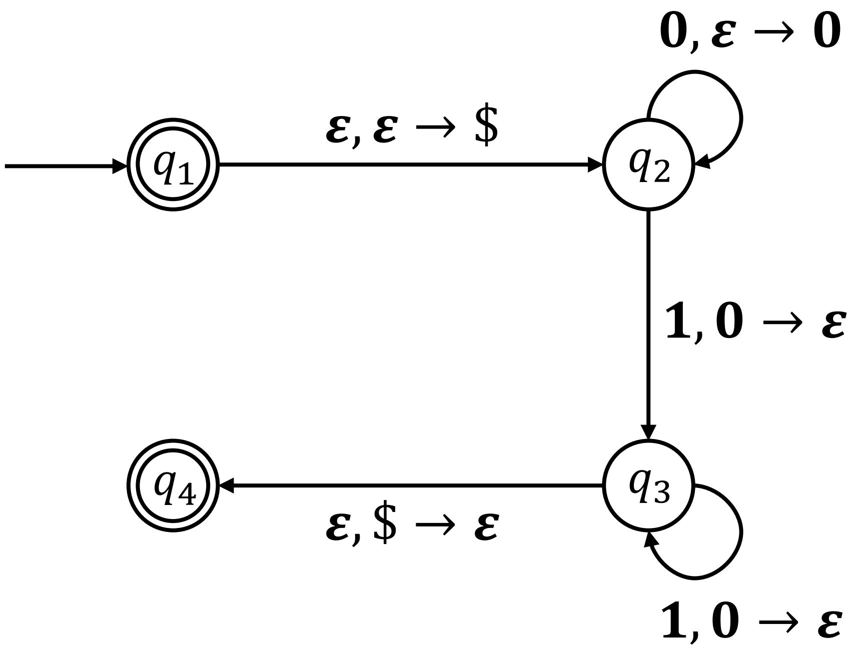
- What is $L(M)$? $\{ \mathbf{0^n 1^n} \mid n \geq \mathbf{0} \}$

Formal Definition: Pushdown Automata

A **pushdown automaton** is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

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

Pushdown Automata Example



- $\mathbf{\Sigma} = \{\mathbf{0}, \mathbf{1}\}, \mathbf{\Gamma} = \{\mathbf{0}, \$\}$

Pushdown Automata Example 2

Create a PDA which recognizes the language

$$\boldsymbol{L = \{\, ww^r \mid w \in \{0,1\}^* \,\} \setminus \{\varepsilon\}}$$

- Try $\boldsymbol{w = 10100101}$
- Try $\boldsymbol{w = 1010010}$