Lecture 11: CFG and PDA Equivalence

CSC 320: Foundations of Computer Science

Quinton Yong

quintonyong@uvic.ca

University of Victoria

Equivalence of PDAs

We will prove that set of languages **recognizable by PDAs** is the same as the **context-free languages** (produced by CFGs)

Theorem: A language is context-free if and only if some PDA recognizes it

Proof consists of two parts:

1. If a language is **context-free** (can be produced by a CFG), then it can be **recognized by a PDA**
2. If a language can be **recognized by a PDA**, then it is **context-free** (can be produced by a CFG)

Equivalence of PDAs

1. If a language is **context-free** (can be produced by a CFG), then it can be **recognized by a PDA**

Idea:

- Construct a PDA which can **replicate the derivation** a string using the CFG
- The PDA builds **intermediate strings** (steps in leftmost derivation) on the stack nondeterministically

- Pop a **CFG variable** off the stack and **push** the right side of a rule
- Pop a **CFG terminal** off the stack to **read** an input string terminal

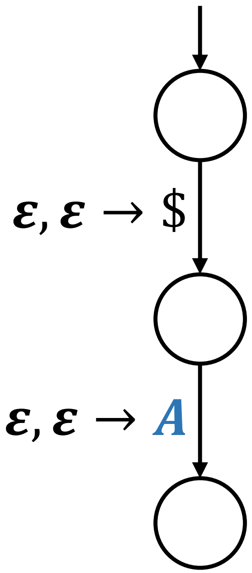
- **Accepts** input string if it can be **built** using the CFG

CFG to PDA (Partial Example)

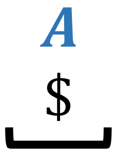
$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$



Start by **pushing $** and **starting variable** A onto stack

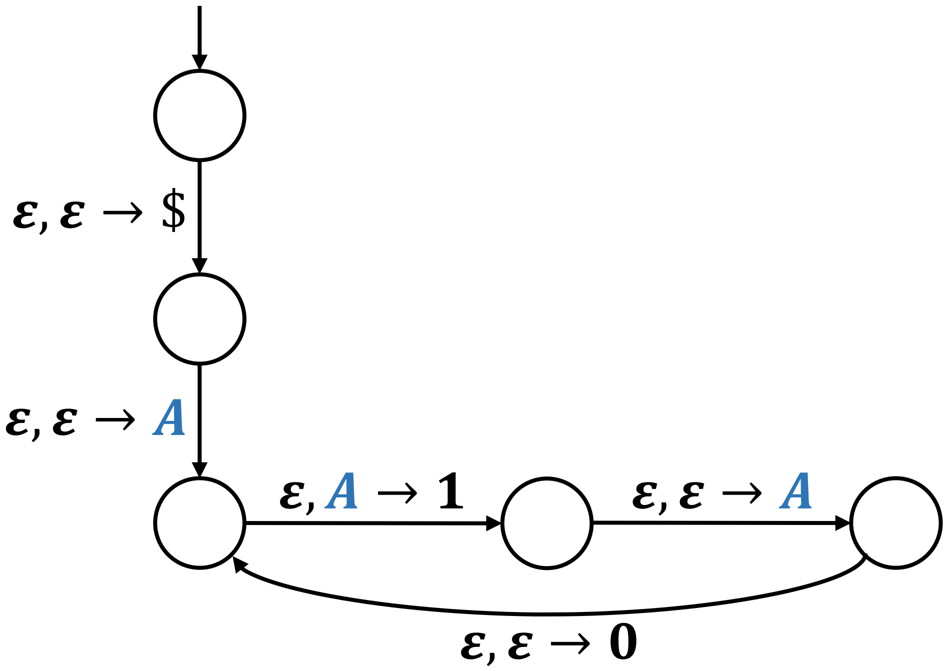


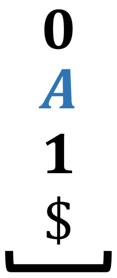
CFG to PDA (Partial Example)

$\boxed{A \to 0A1}$

$A \to B$

$B \to \#$





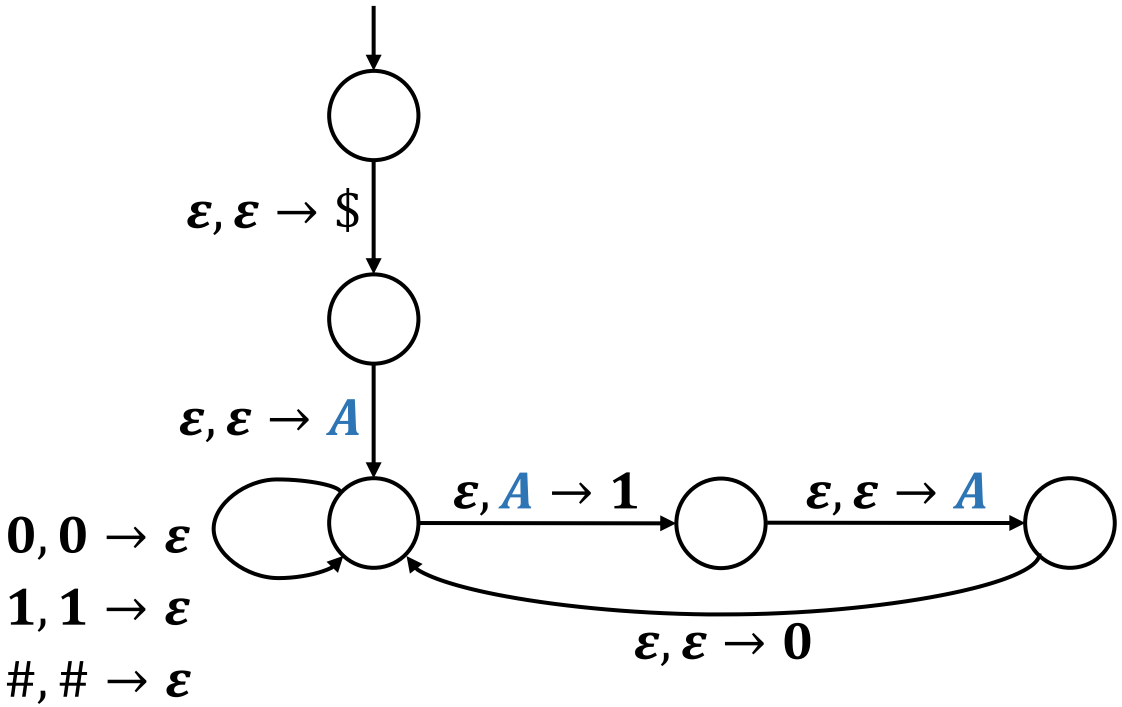
Replicate rule $\boldsymbol{A \to 0A1}$:
pop an $\boldsymbol{A}$ and push $\mathbf{1}$ then $\boldsymbol{A}$ then $\mathbf{0}$

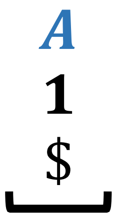
CFG to PDA (Partial Example)

$A \to 0A1$

$A \to B$

$B \to \#$





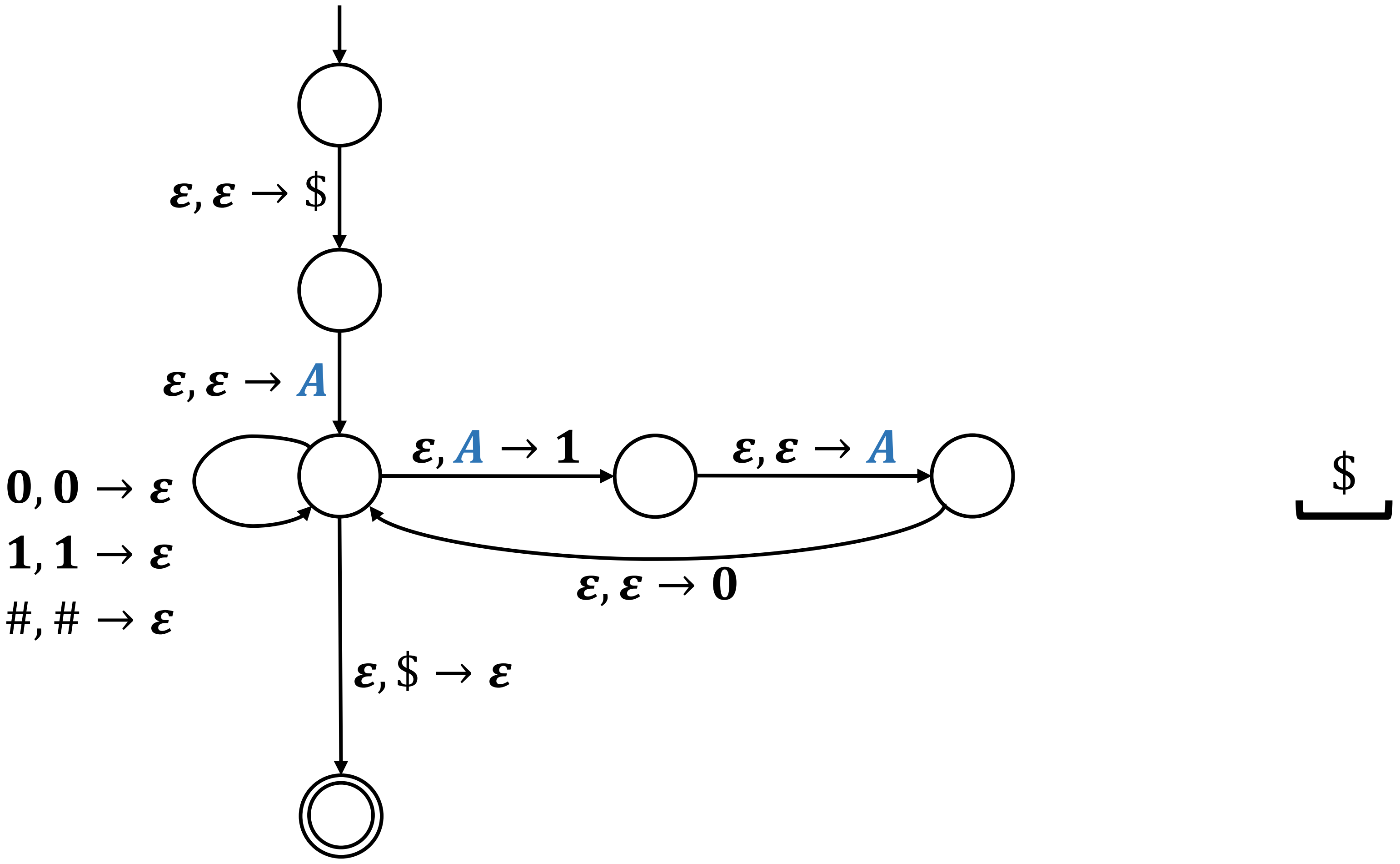
Loop to read input:
if it **can be built** by CFG on stack, then
we can **read** and **pop terminals**

CFG to PDA (Partial Example)

$A \rightarrow 0A1$

$A \rightarrow B$

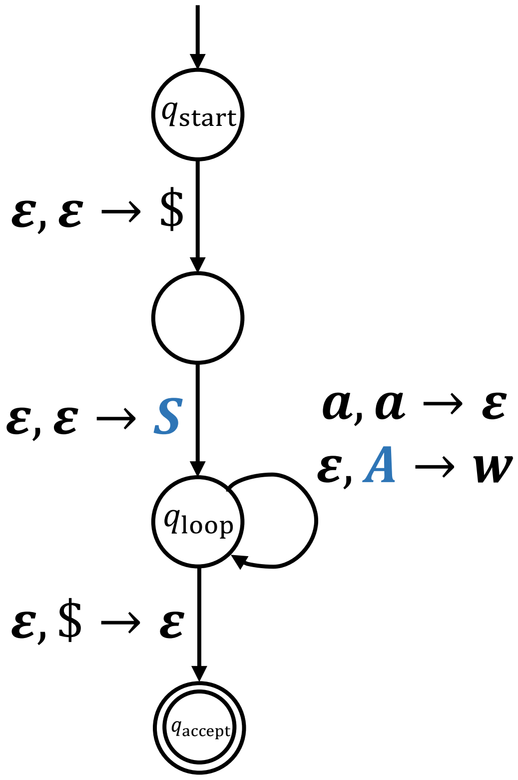
$B \rightarrow \#$



Equivalence of PDAs

Proof: For any CFG $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{\Sigma}, \boldsymbol{R}, \boldsymbol{S})$, design PDA $\boldsymbol{M} = (\boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\Gamma}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{F})$ with $\boldsymbol{L}(\boldsymbol{G}) = \boldsymbol{L}(\boldsymbol{M})$

- Place marker symbol \$ and start variable $\boldsymbol{S}$ onto empty stack
- For each top stack symbol:
  - If **terminal** $\boldsymbol{a}$: read next input symbol $\boldsymbol{w}$ and pop stack symbol if $\boldsymbol{w} = \boldsymbol{a}$
  - If **variable** $\boldsymbol{A}$: choose some rule $\boldsymbol{A} \to \boldsymbol{\alpha_1}, \boldsymbol{\alpha_2}, \dots, \boldsymbol{\alpha_k}$ and substitute $\boldsymbol{A}$ with $\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2}, \dots, \boldsymbol{\alpha_k}$ (α_1 is top stack symbol)
  - If \$: go to accept state

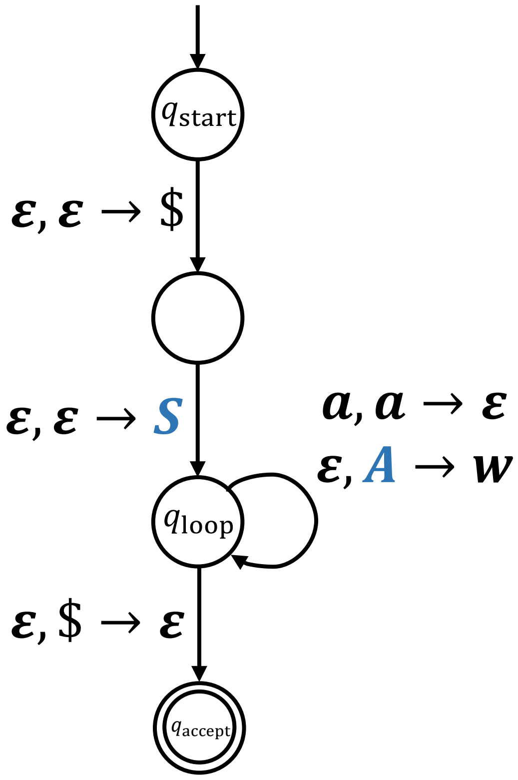


Equivalence of PDAs

Proof: For any CFG $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{\Sigma}, \boldsymbol{R}, \boldsymbol{S})$, design PDA $\boldsymbol{M} = (\boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\Gamma}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{F})$ with $\boldsymbol{L}(\boldsymbol{G}) = \boldsymbol{L}(\boldsymbol{M})$

6-tuple contents of $\boldsymbol{M}$:

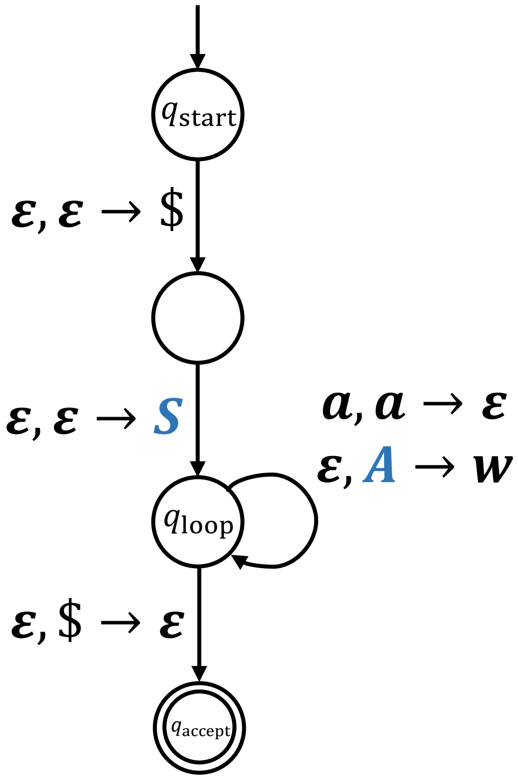
- $\boldsymbol{Q} = \{q_{\text{start}}, q_{\text{loop}}, q_{\text{accept}}\}$ ∪ auxiliary states for pushing $\boldsymbol{S}$ and right side of rules in $\boldsymbol{R}$
- $\boldsymbol{\Gamma} = \boldsymbol{V} \cup \boldsymbol{\Sigma} \cup \{\$\}$
- $\boldsymbol{q_0} = q_{\text{start}}$
- $\boldsymbol{F} = \{q_{\text{accept}}\}$



Equivalence of PDAs

$\boldsymbol{M}$ transitions:

- In $\boldsymbol{q_{\text{start}}}$, when reading $\boldsymbol{\varepsilon}$ and top symbol $\boldsymbol{\varepsilon}$, push \$ and then $\boldsymbol{S}$ onto the stack, then move onto $\boldsymbol{q_{\text{loop}}}$
- In $\boldsymbol{q_{\text{loop}}}$, for each rule $\boldsymbol{A} \to \boldsymbol{\alpha_1\alpha_2 \dots \alpha_k}$ in $\boldsymbol{R}$, replace $\boldsymbol{A}$ by $\boldsymbol{\alpha_1\alpha_2 \dots \alpha_k}$ ($\boldsymbol{\alpha_1}$ new top stack symbol) and stay in $\boldsymbol{q_{\text{loop}}}$
- In $\boldsymbol{q_{\text{loop}}}$, for each terminal $\boldsymbol{a} \in \boldsymbol{\Sigma}$ if $\boldsymbol{a}$ is the top stack symbol then read $\boldsymbol{a}$, pop $\boldsymbol{a}$, and remain in $\boldsymbol{q_{\text{loop}}}$
- In $\boldsymbol{q_{\text{loop}}}$, if \$ is the top stack symbol, pop \$ and move to $\boldsymbol{q_{\text{accept}}}$

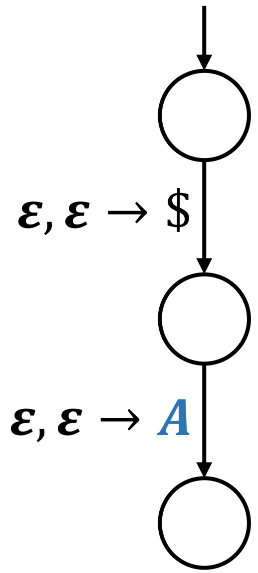


CFG to PDA Example

$A \to 0A1$

$A \to B$

$B \to \#$

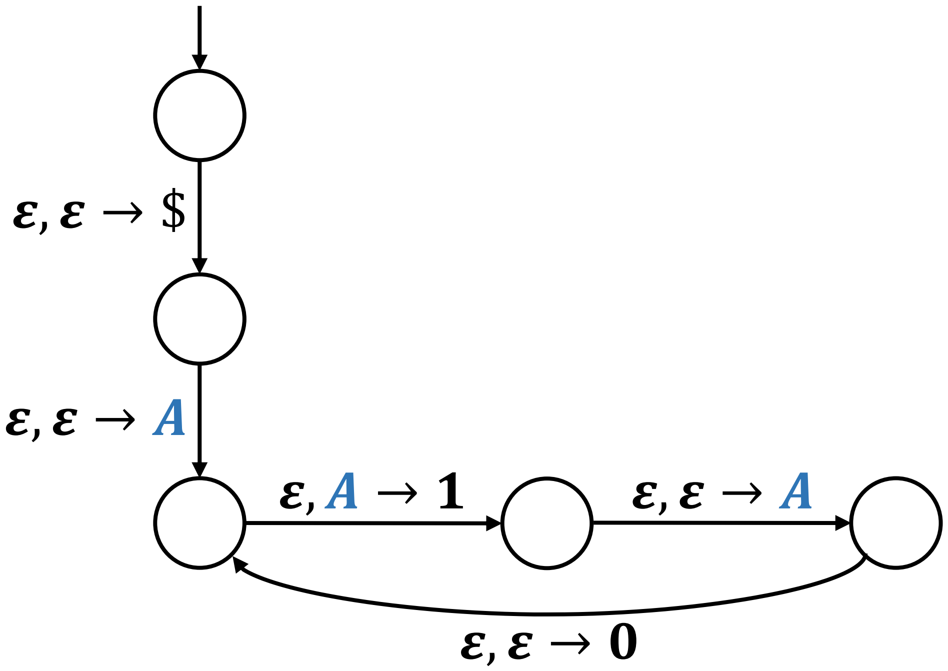


CFG to PDA Example

$\boxed{A \rightarrow 0A1}$

$A \rightarrow B$

$B \rightarrow \#$

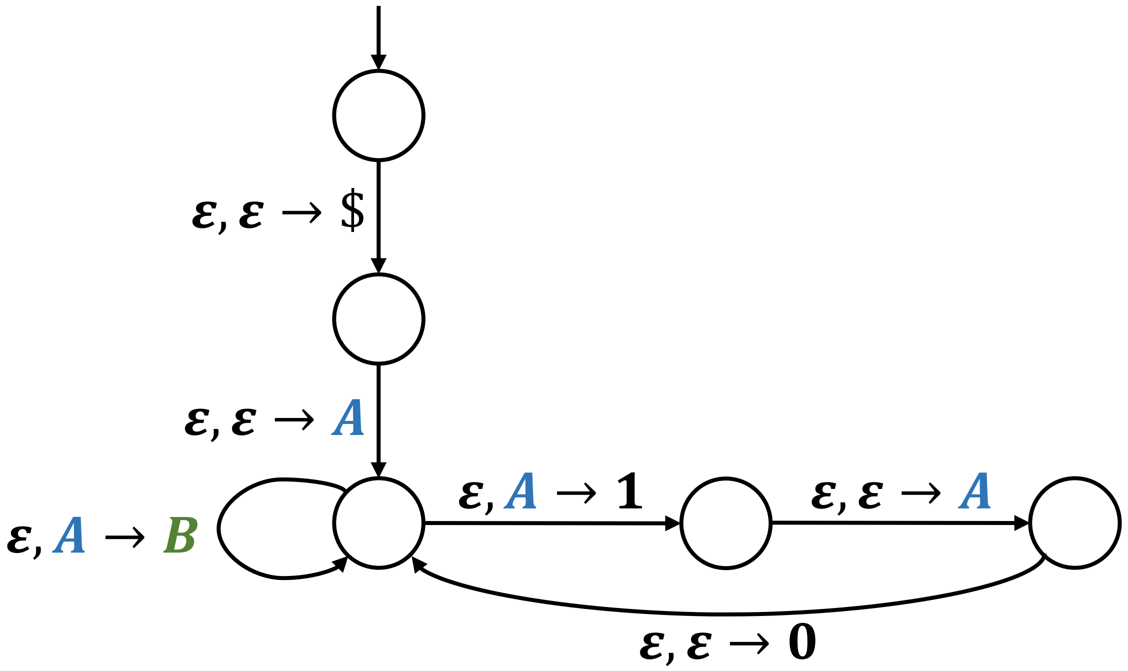


CFG to PDA Example

$A \rightarrow 0A1$

$\boxed{A \rightarrow B}$

$B \rightarrow \#$

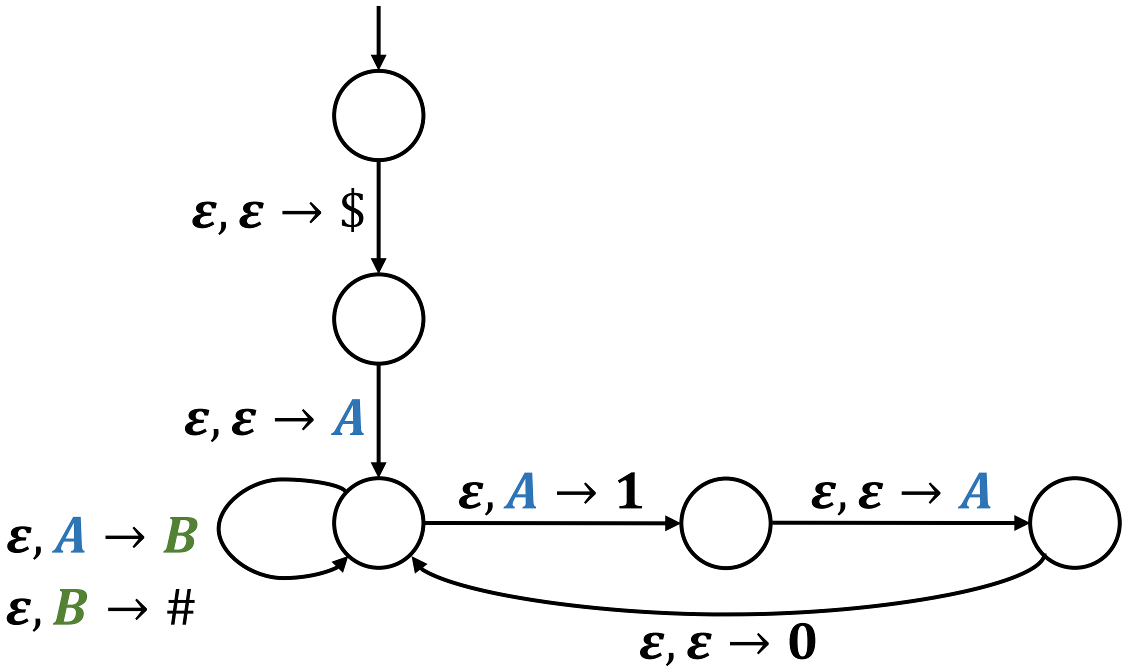


CFG to PDA Example

$A \rightarrow 0A1$

$A \rightarrow B$

$\boxed{B \rightarrow \#}$

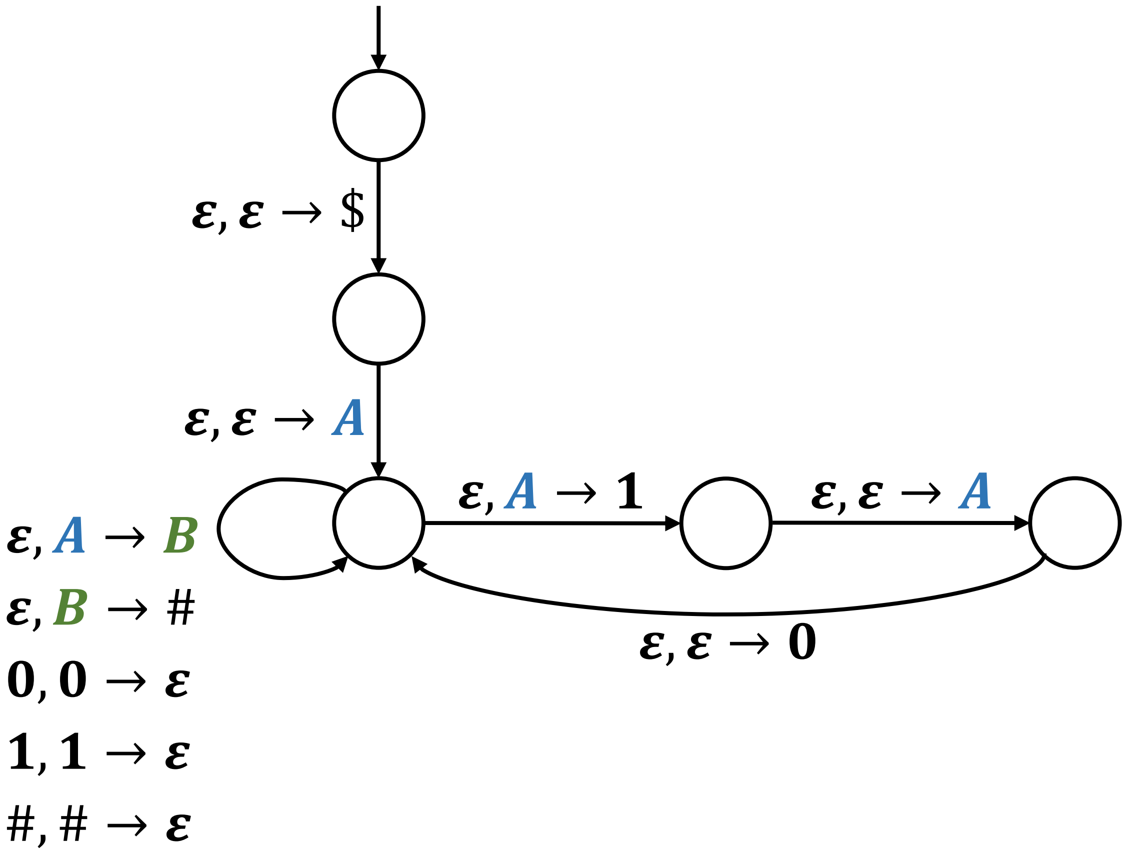


CFG to PDA Example

$A \to 0A1$

$A \to B$

$B \to \#$

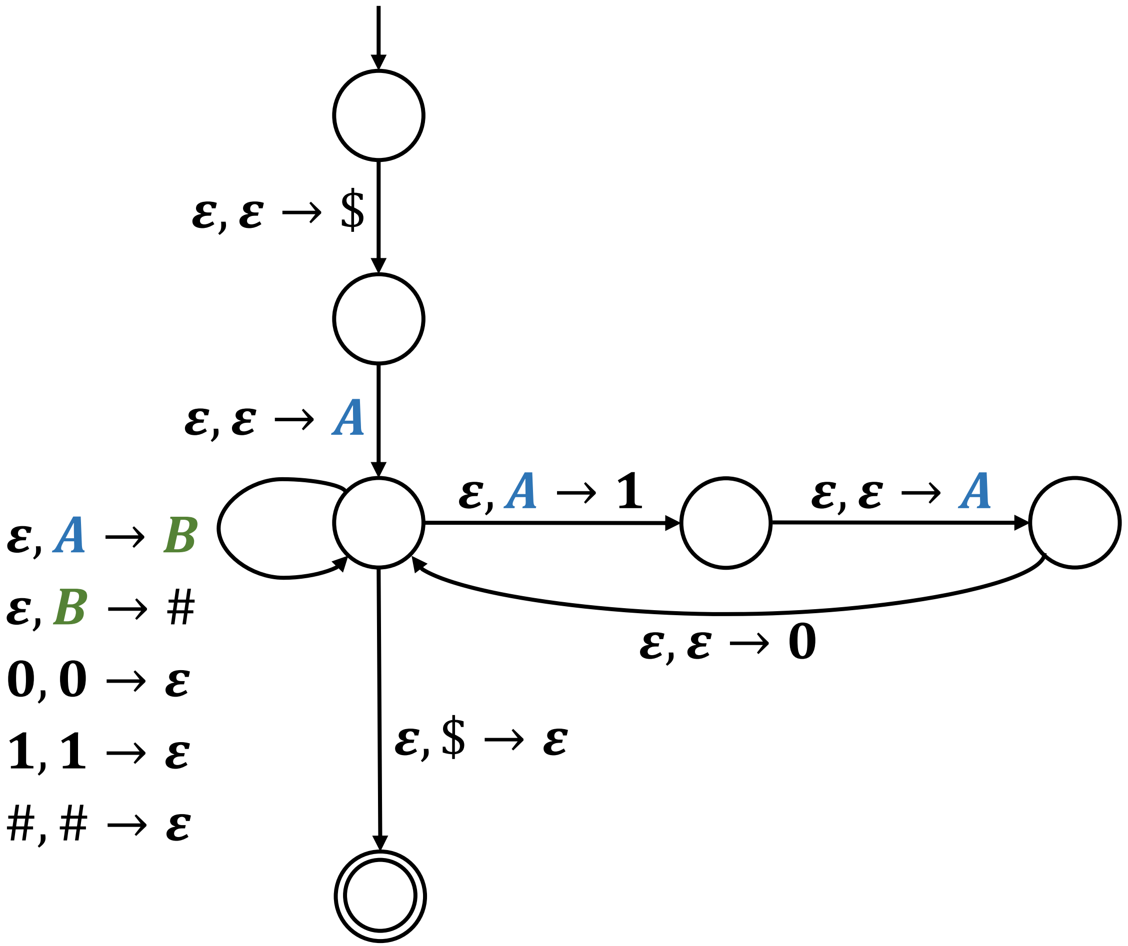


CFG to PDA Example

$A \to 0A1$

$A \to B$

$B \to \#$



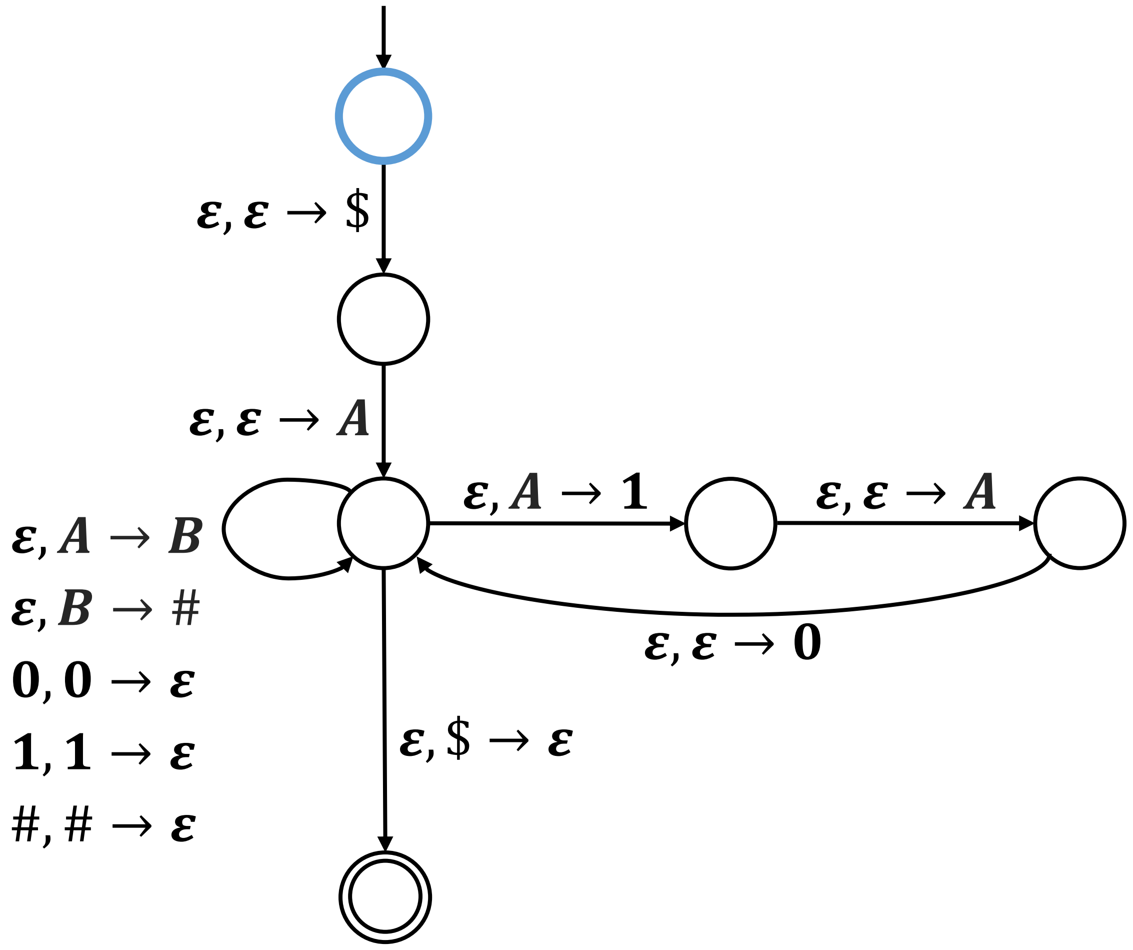
CFG to PDA Example

$A \to 0A1$

$A \to B$

$B \to \#$

$\boxed{w = 00\#11}$



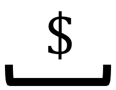
CFG to PDA Example

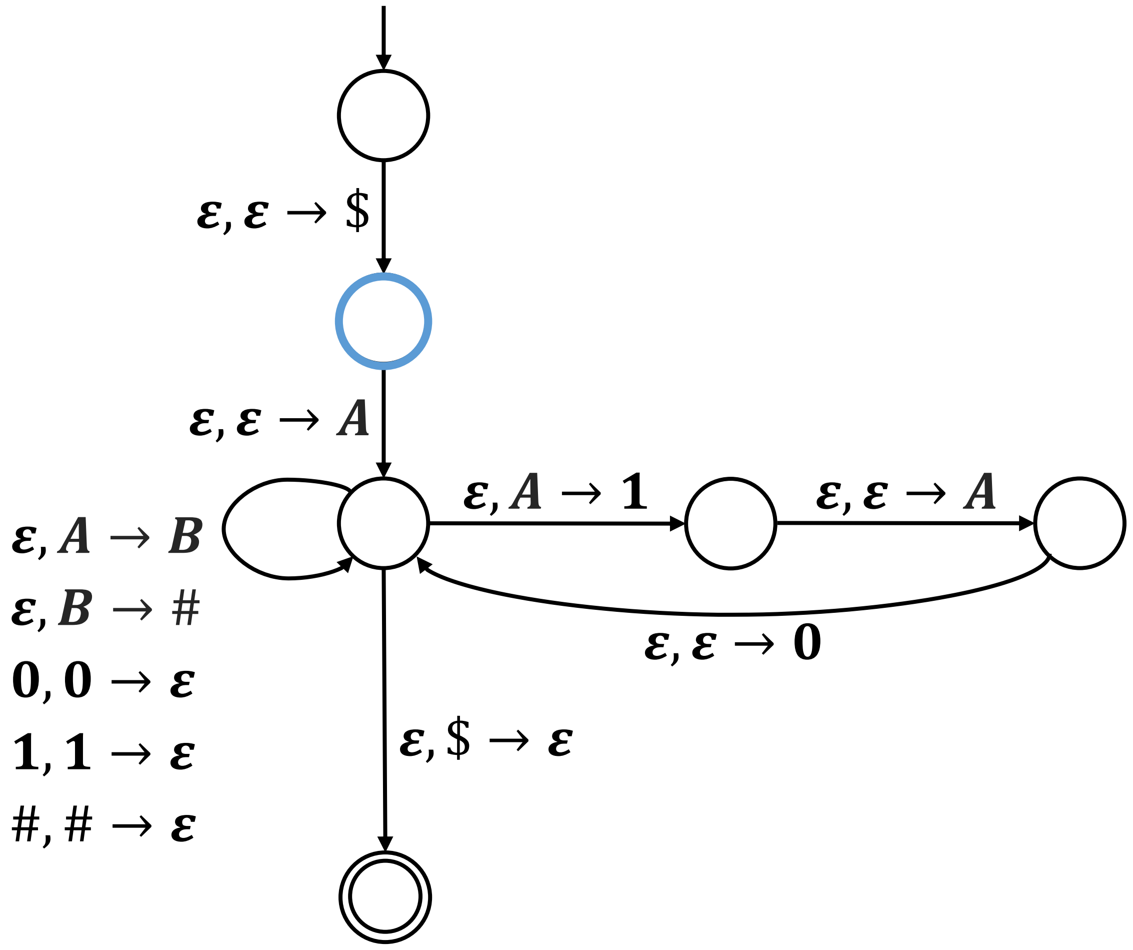
$A \to 0A1$

$A \to B$

$B \to \#$

$\boxed{w = 00\#11}$





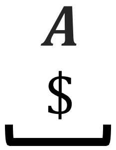
CFG to PDA Example

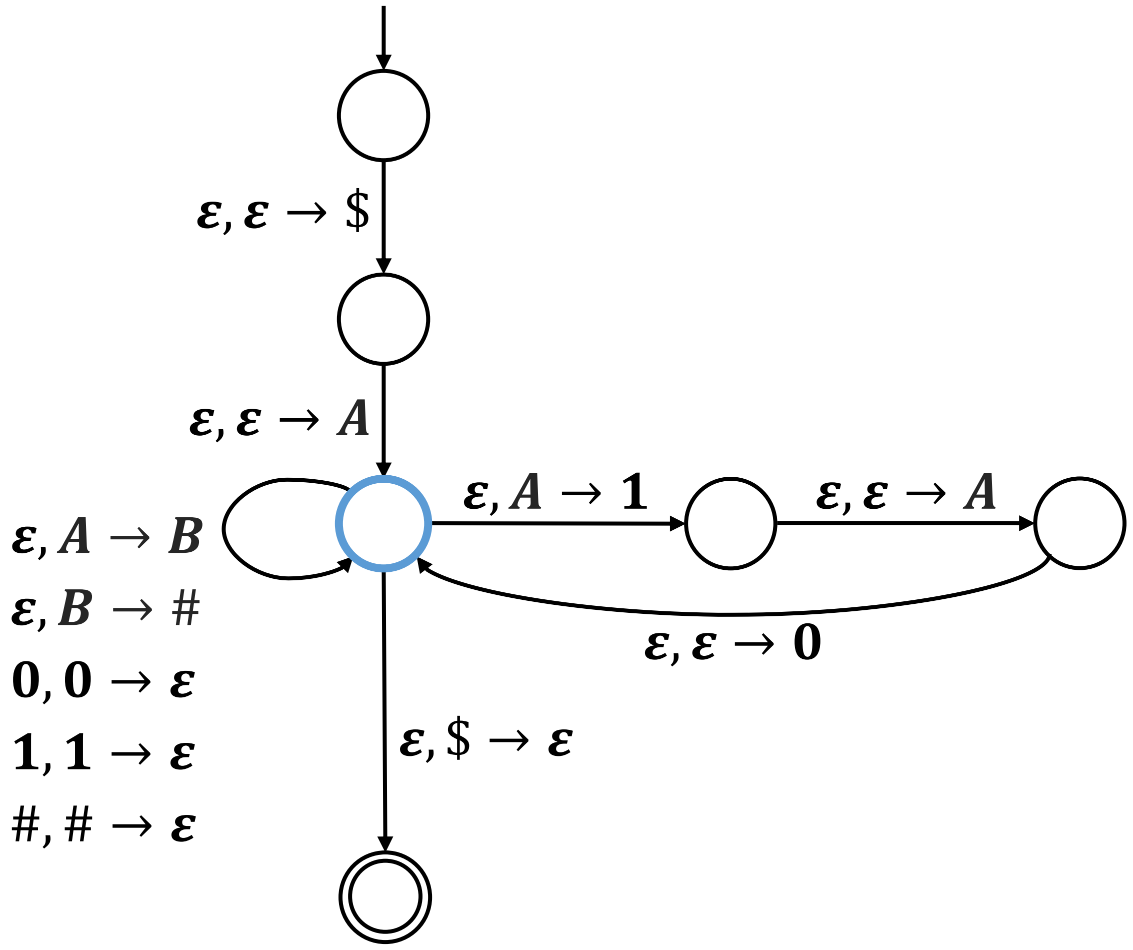
$A \to 0A1$

$A \to B$

$B \to \#$

$\boxed{w = 00\#11}$





CFG to PDA Example

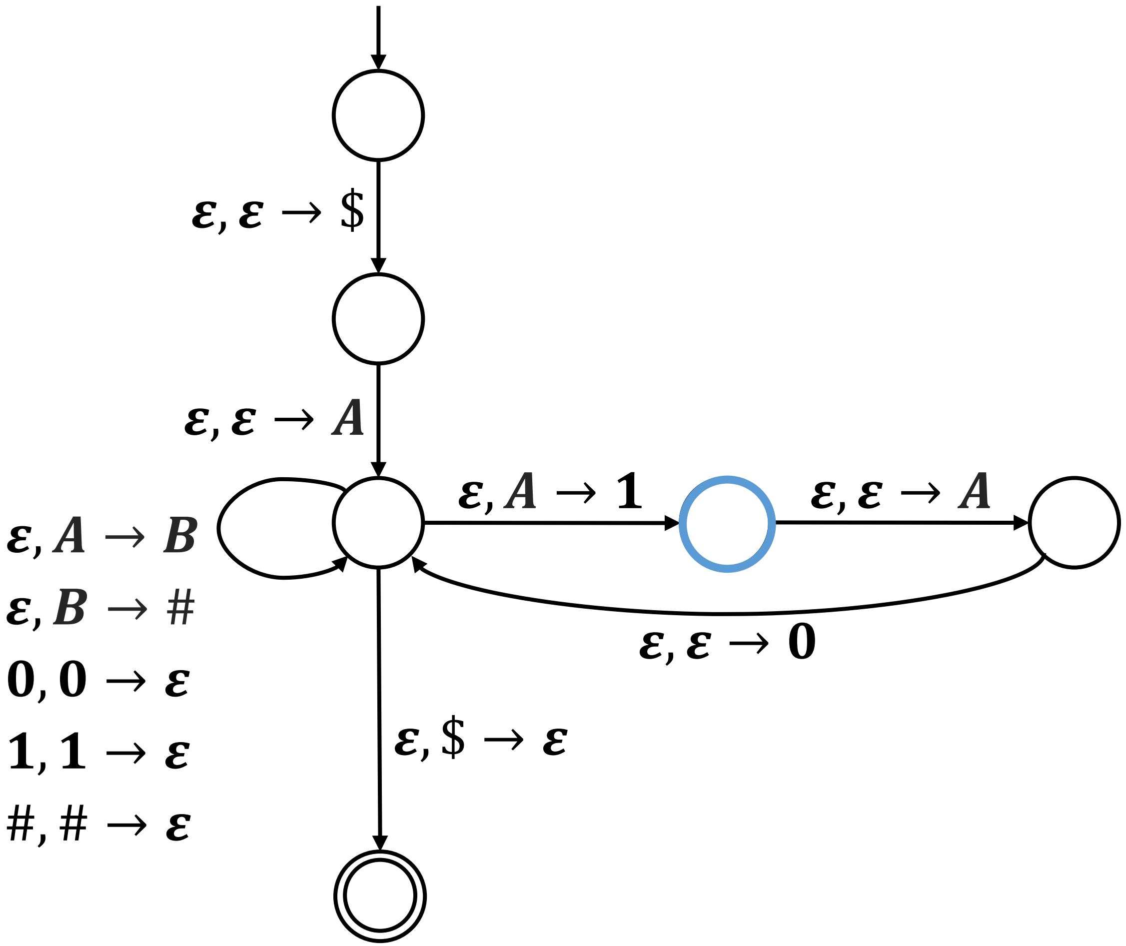
$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

$\boxed{w = 00\#11}$

1
$



CFG to PDA Example

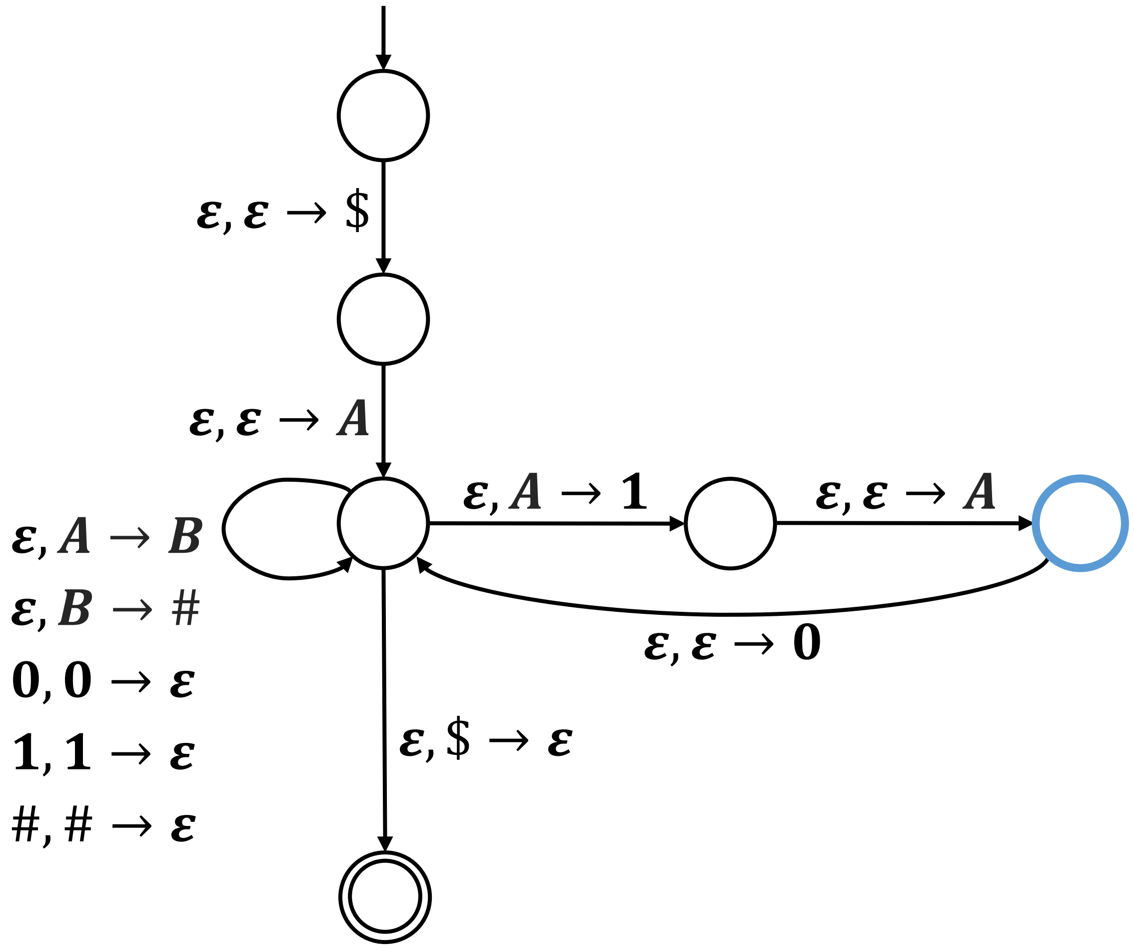
$A \to 0A1$

$A \to B$

$B \to \#$

$\boxed{w = 00\#11}$

A
1
$\$$



CFG to PDA Example

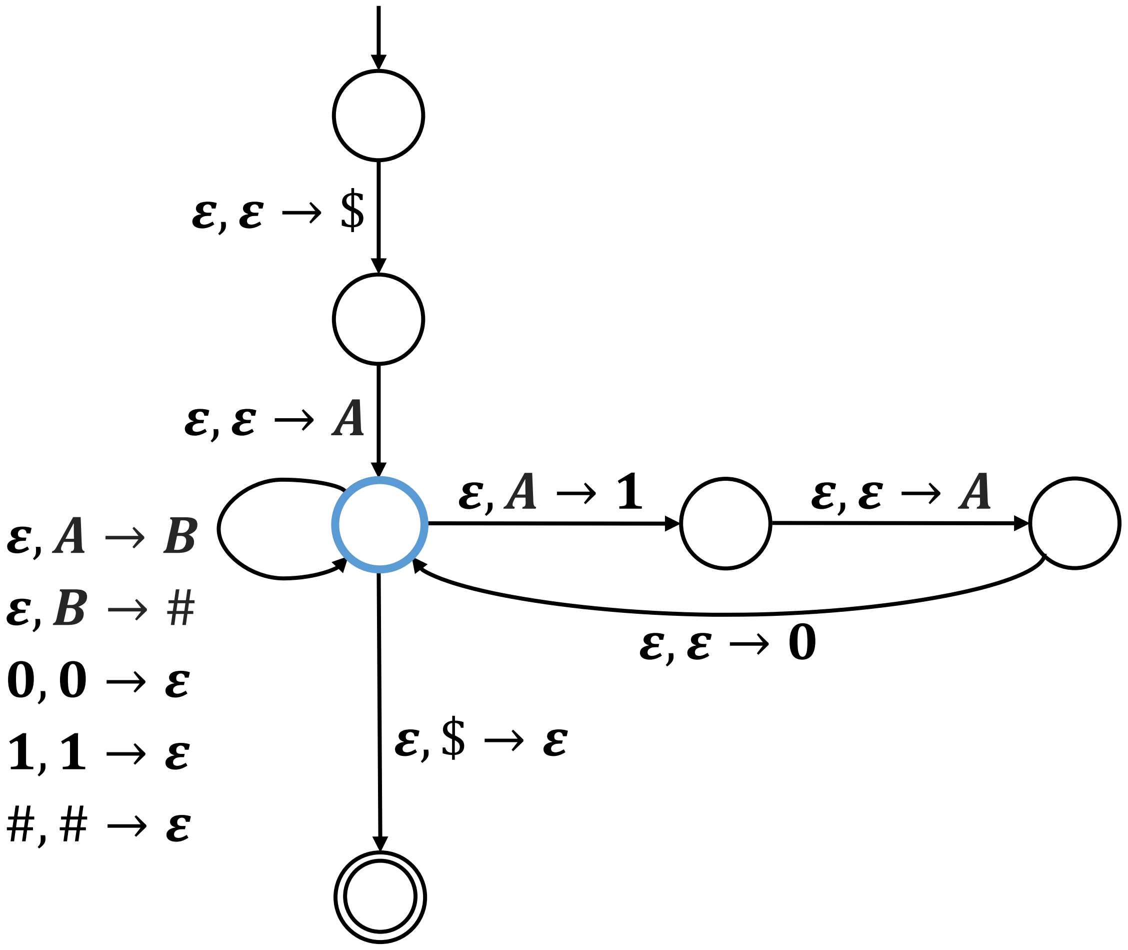
$A \to 0A1$

$A \to B$

$B \to \#$

$\boxed{w = 00\#11}$

0
A
1
$\$$



CFG to PDA Example

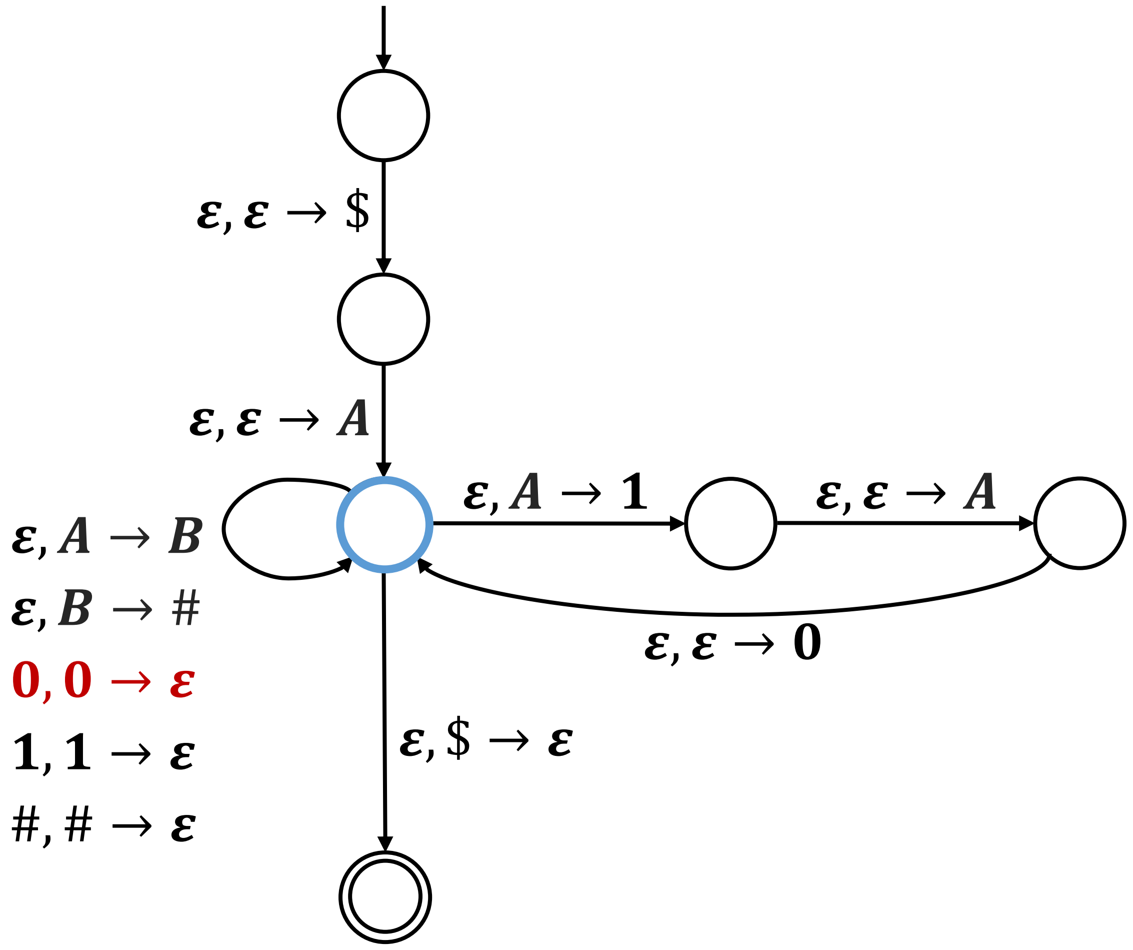
$A \to 0A1$

$A \to B$

$B \to \#$

$\boxed{w = 00\#11}$

A
1
$\$$



CFG to PDA Example

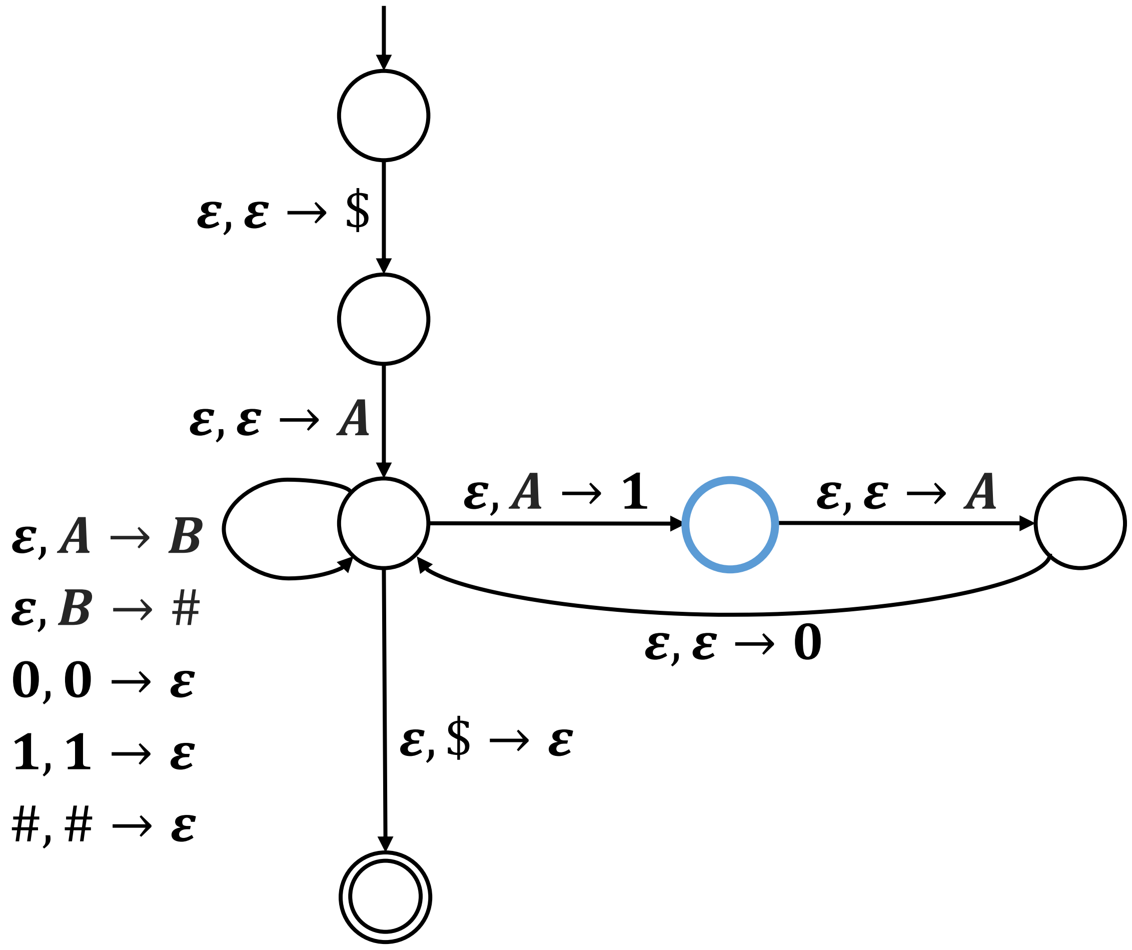
$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

$\boxed{w = 00\#11}$

1
1
$



CFG to PDA Example

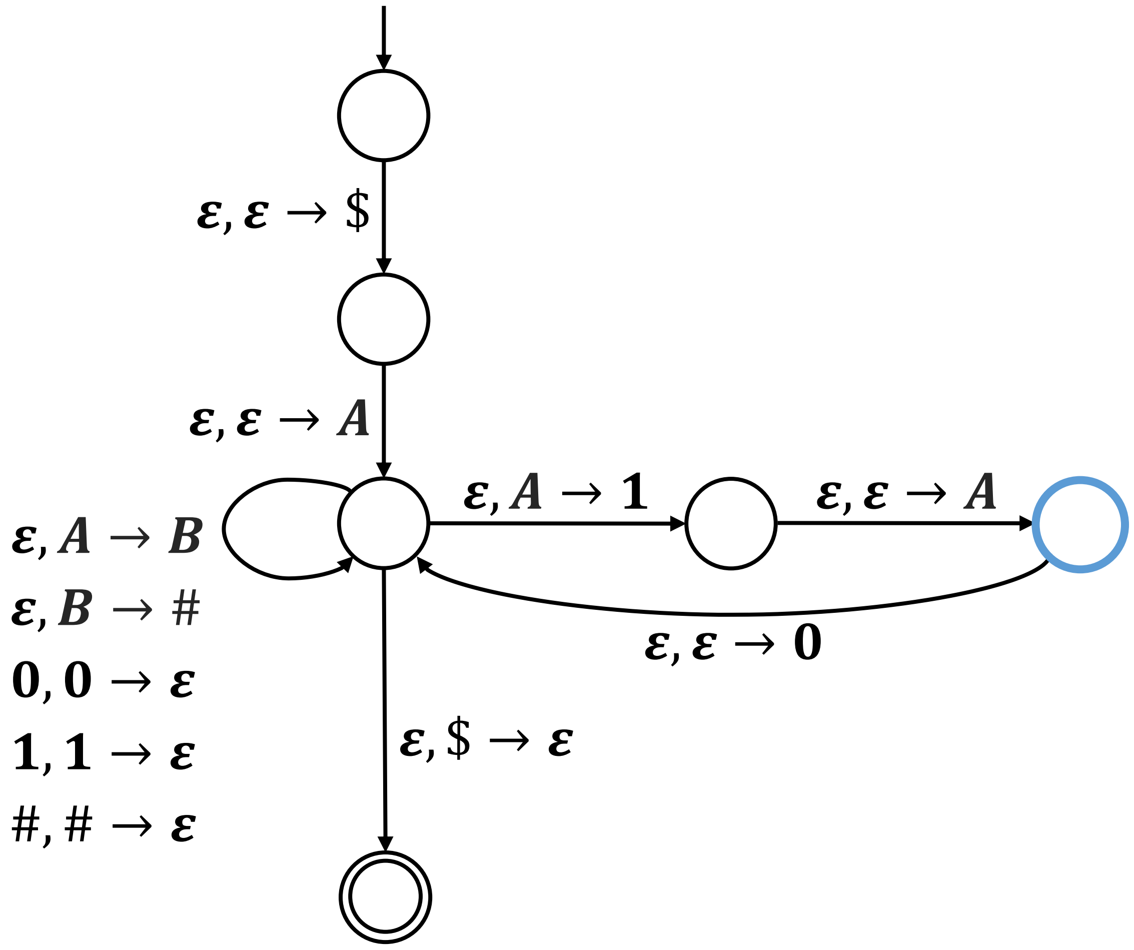
$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

A
1
1
$\$$

$\boxed{w = 00\#11}$



CFG to PDA Example

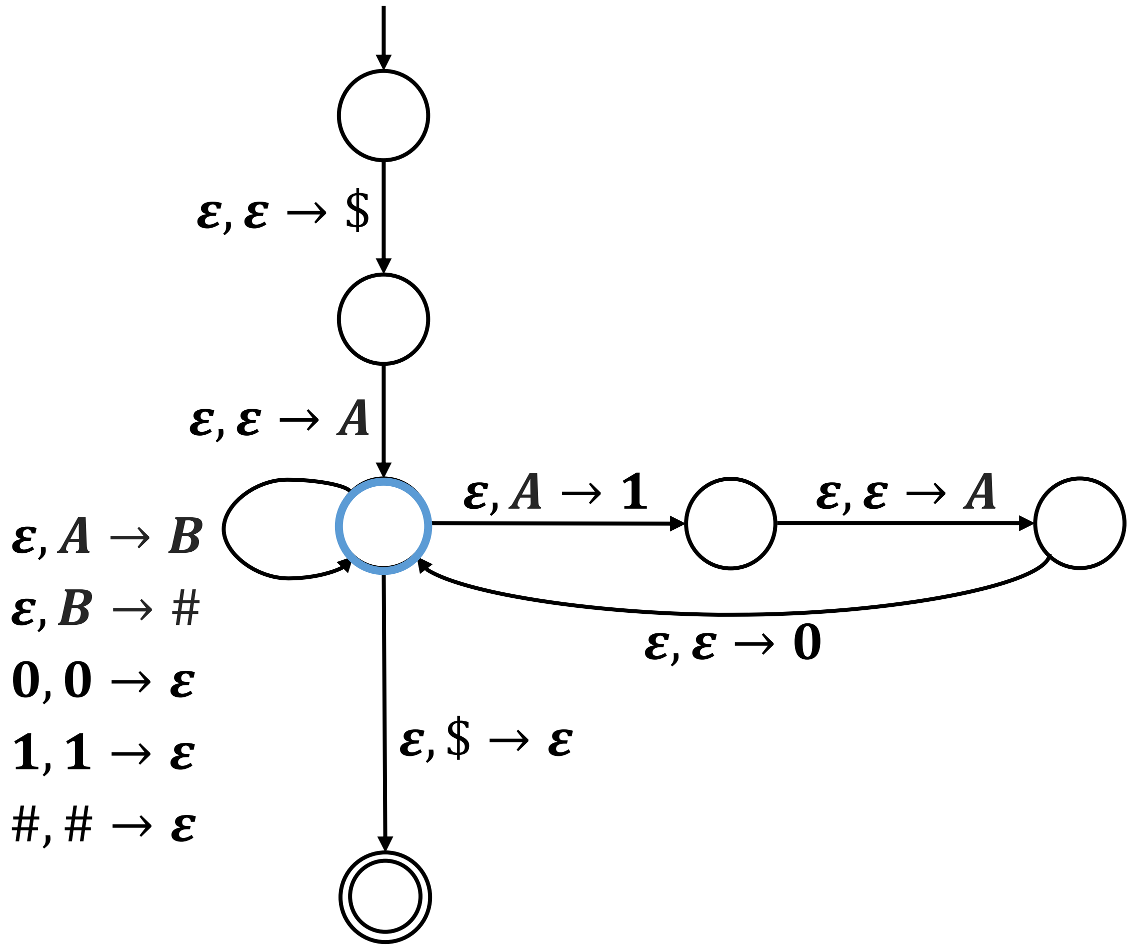
$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

$\boxed{w = 00\#11}$

0
A
1
1
$\$$



CFG to PDA Example

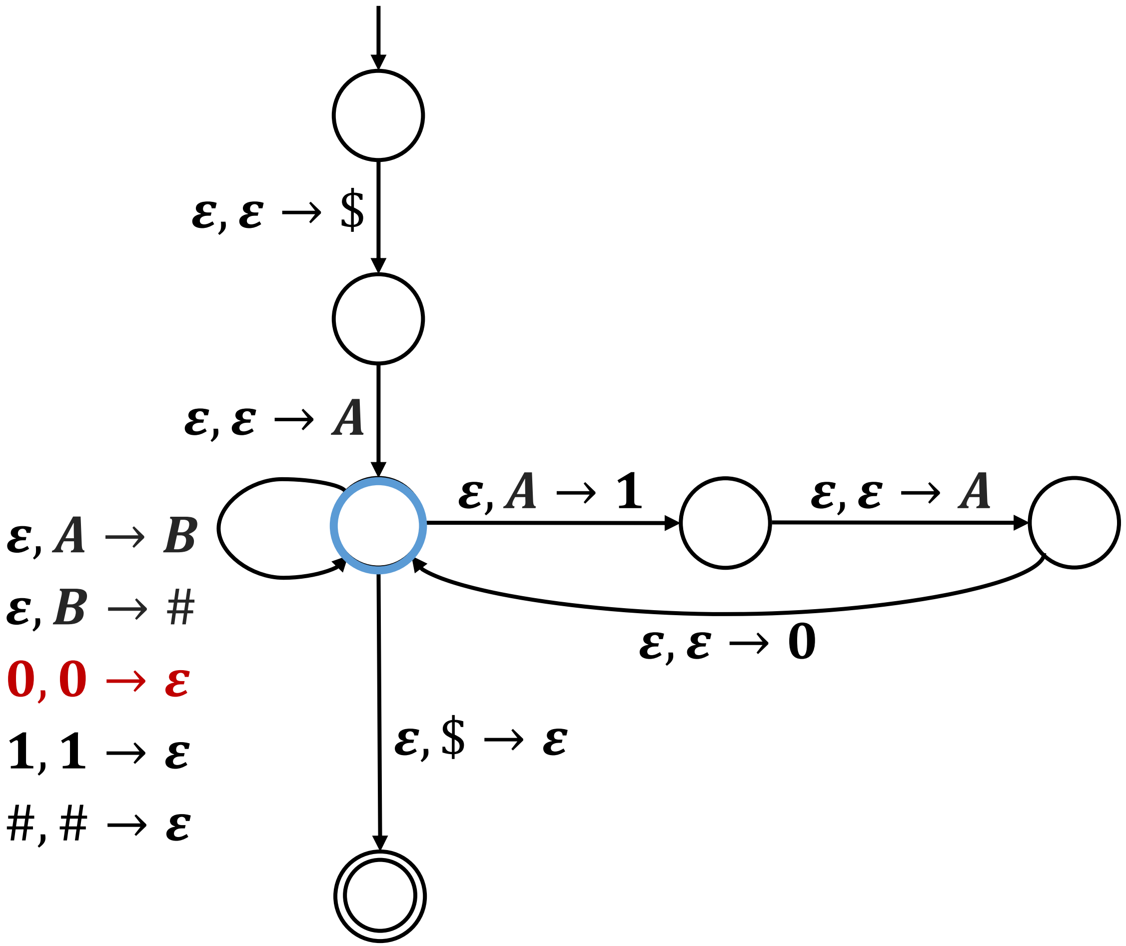
$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

A
1
1
$\$$

$\boxed{w = 00\#11}$



CFG to PDA Example

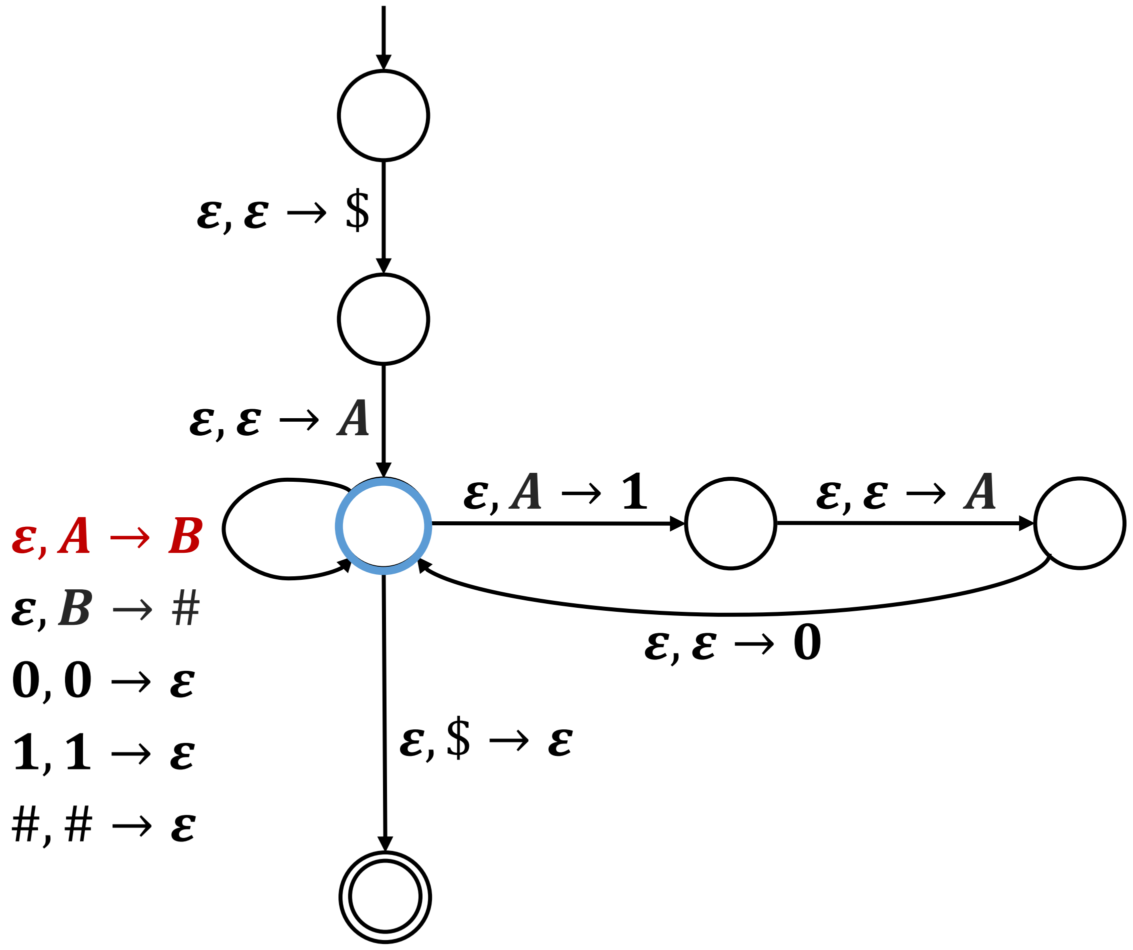
$A \to 0A1$

$A \to B$

$B \to \#$

$w = 00\#11$

B
1
1
$\$$



CFG to PDA Example

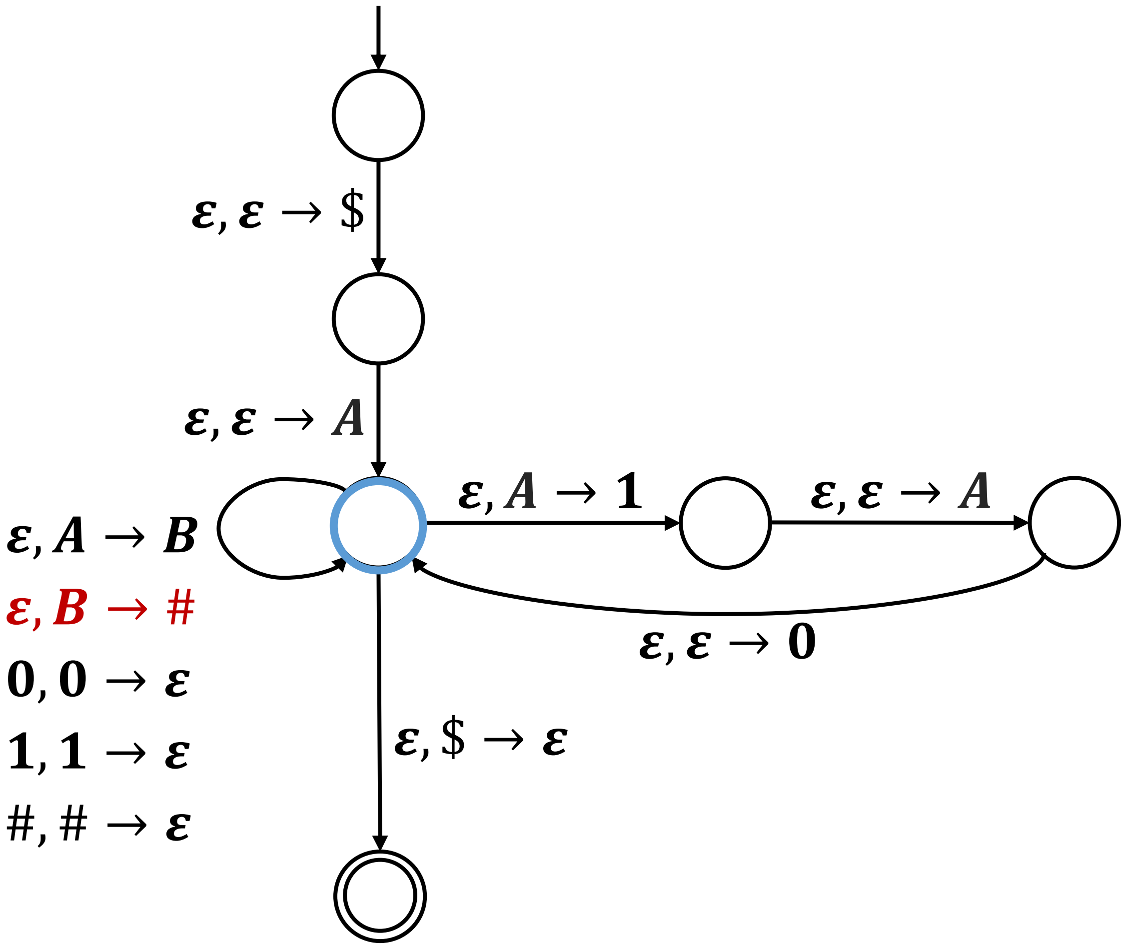
$A \to 0A1$

$A \to B$

$B \to \#$

$\boxed{w = 00\#11}$

$\#$
1
1
$\$$



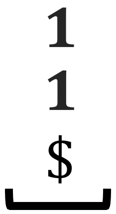
CFG to PDA Example

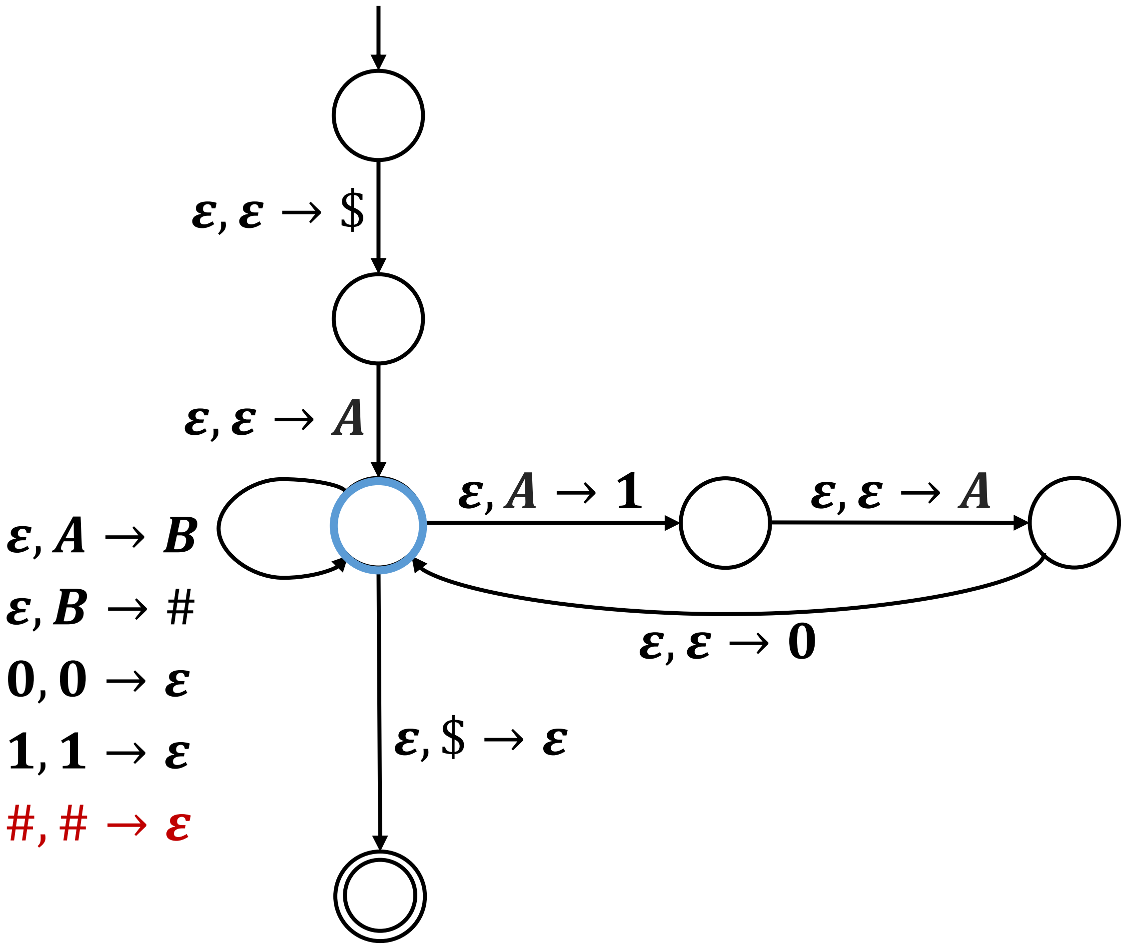
$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

$\boxed{w = 00\#11}$





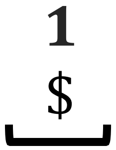
CFG to PDA Example

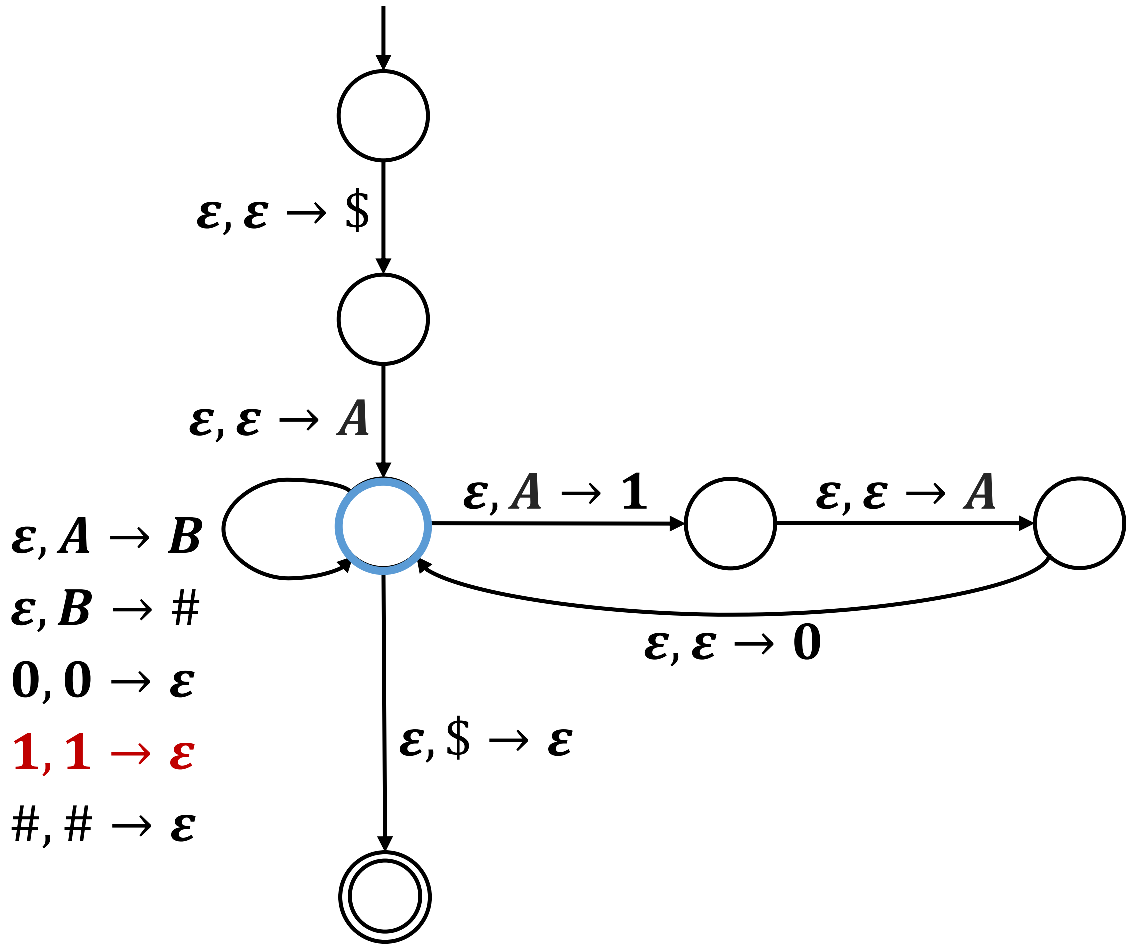
$A \to 0A1$

$A \to B$

$B \to \#$

$\boxed{w = 00\#11}$





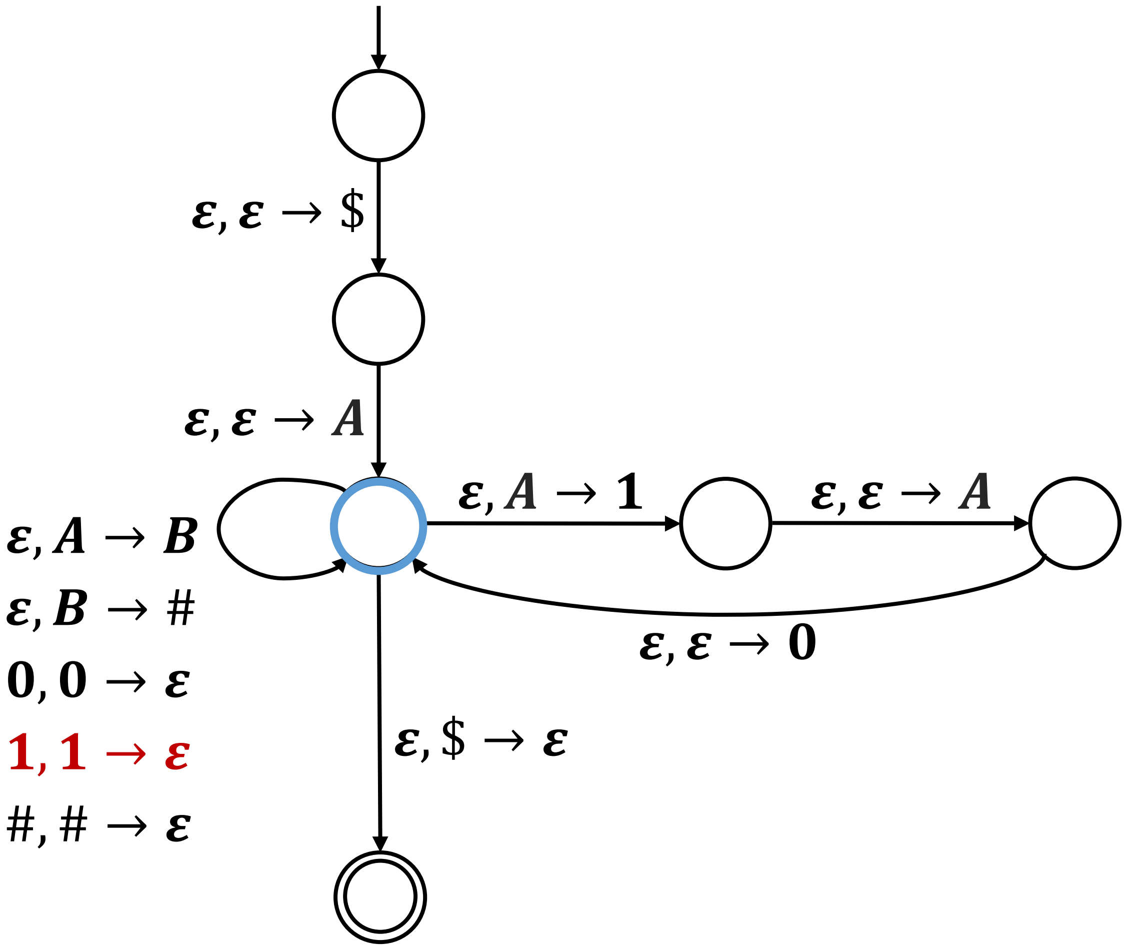
CFG to PDA Example

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

$\boxed{w = 00\#11}$



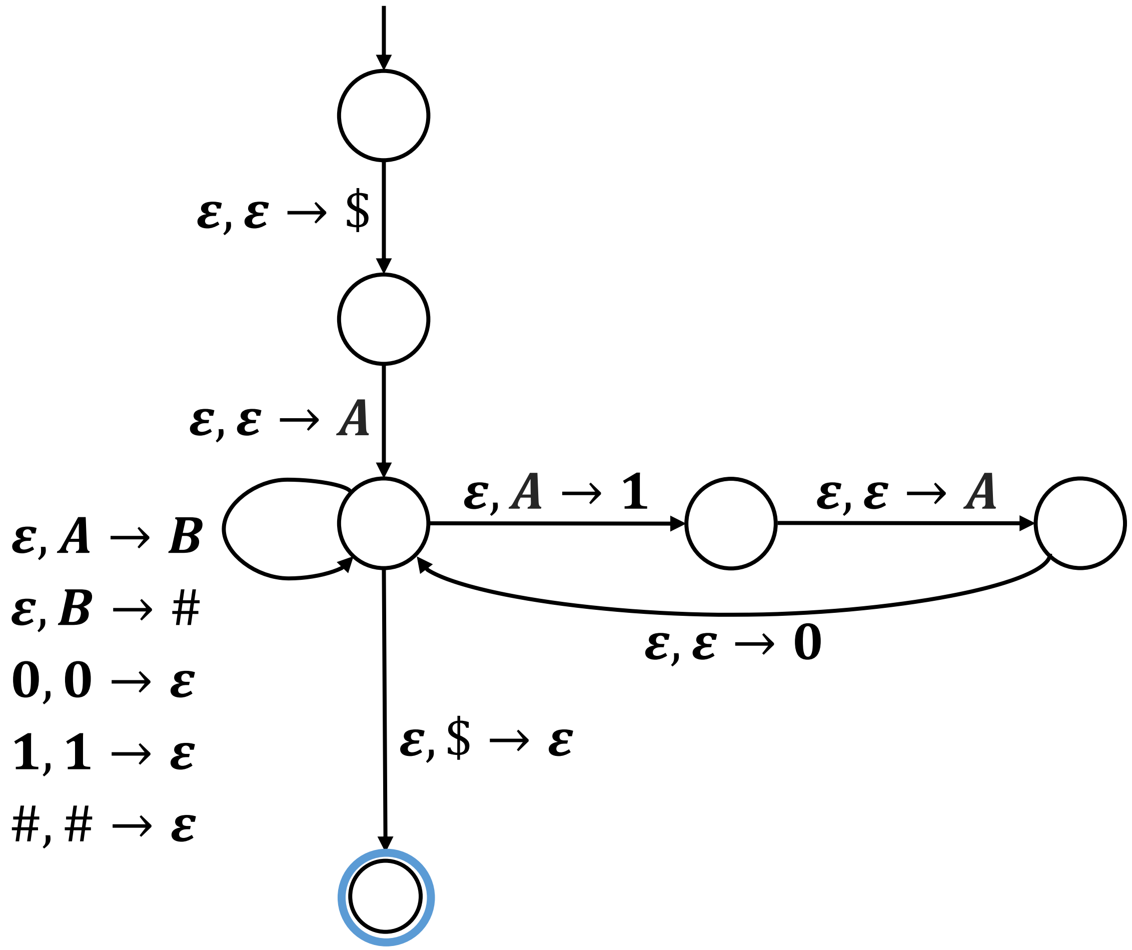
CFG to PDA Example

$A \to 0A1$

$A \to B$

$B \to \#$

$\boxed{w = 00\#11}$



Equivalence of PDAs

We will prove that set of languages **recognizable by PDAs** is the same as the **context-free languages** (produced by CFGs)

Theorem: A language is context-free if and only if some PDA recognizes it

Proof consists of two parts:

1. If a language is **context-free** (can be produced by a CFG), then it can be **recognized by a PDA**
2. If a language can be **recognized by a PDA**, then it is **context-free** (can be produced by a CFG)

Equivalence of PDAs

2. If a language can be **recognized by a PDA**, then it is **context-free** (can be produced by a CFG)

We will only introduce the proof idea

Step 1: Simplify the PDA $\boldsymbol{M}$

- **One accept state** (make new accept state)
- Only transition from **start state**: $\varepsilon, \varepsilon \rightarrow \$$
- Transitions to new **accept state**: $\varepsilon, \$ \rightarrow \varepsilon$
- All transitions must **either be push or pop**, not both (replace these by two separate transitions)

Step 2: Design grammar from simplified PDA

Not Context-Free Languages

Is the following language context-free?

$$L = \{\, \mathbf{0}^n \mathbf{1}^n \mathbf{2}^n \mid n \geq 0 \,\}$$

A language is context-free if and only if a **context-free grammar** or **pushdown automaton** recognizes it

Consider how we would make a PDA to recognize L:

- Can use stack to **push** $\mathbf{0}$'s when reading $\mathbf{0}$'s and **pop** $\mathbf{0}$'s when reading $\mathbf{1}$'s
- Then we're left with an empty stack…

L is **not context-free**, but how can we prove that there is no context-free grammar or pushdown automaton that recognizes it?

Pumping Lemma for Context-Free Languages

If $\boldsymbol{L}$ is a context-free language, then there is a number $\boldsymbol{p}$ (pumping length of $\boldsymbol{L}$) such that for **every string $\boldsymbol{s} \in \boldsymbol{L}$** of **length at least $\boldsymbol{p}$**, $\boldsymbol{s}$ can be **divided into five parts $\boldsymbol{s = uvxyz}$** satisfying the following:

1. $|\boldsymbol{vy}| > \boldsymbol{0}$ (i.e. $\boldsymbol{v}$ and $\boldsymbol{y}$ cannot both be empty)
2. $|\boldsymbol{vxy}| \leq \boldsymbol{p}$
3. $\boldsymbol{uv^i xy^i z} \in \boldsymbol{L}$ for each $\boldsymbol{i} \geq \boldsymbol{0}$