BBM401-Lecture 11: Pushdown Automata

Lecturer: Lale Özkahya

Resources for the presentation: https://courses.engr.illinois.edu/cs373/fa2010/

Computing Using a Stack **Definition** Examples of Pushdown Automata

K ロ ▶ K 何 ▶

Ε

× Þ. 一 (語)

 299

哇

Restricted Infinite Memory: The Stack

Agha-Viswanathan CS373

Computing Using a Stack **Definition** Examples of Pushdown Automata

 \leftarrow \Box

 200

Restricted Infinite Memory: The Stack

So far we considered automata with finite memory or machines with infinite memory

 Ω

- So far we considered automata with finite memory or machines with infinite memory
- Today: automata with access to an infinite stack infinite memory but restricted access

 Ω

- So far we considered automata with finite memory or machines with infinite memory
- **•** Today: automata with access to an infinite stack infinite memory but restricted access
- **The stack can contain an unlimited number of characters.**

 Ω

- So far we considered automata with finite memory or machines with infinite memory
- **•** Today: automata with access to an infinite stack infinite memory but restricted access
- **The stack can contain an unlimited number of characters.** But
	- can read/erase only the top of the stack: pop

つくへ

- So far we considered automata with finite memory or machines with infinite memory
- **•** Today: automata with access to an infinite stack infinite memory but restricted access
- **The stack can contain an unlimited number of characters.** But
	- can read/erase only the top of the stack: pop
	- can add to only the top of the stack: push

 2990

- So far we considered automata with finite memory or machines with infinite memory
- **•** Today: automata with access to an infinite stack infinite memory but restricted access
- **The stack can contain an unlimited number of characters.** But
	- can read/erase only the top of the stack: pop
	- can add to only the top of the stack: push
- On longer inputs, automaton may have more items in the stack

Computing Using a Stack **Definition** Examples of Pushdown Automata

4 0 8

 \sim

Þ

 \rightarrow

Ε

 $2Q$

Keeping Count Using the Stack

An automaton can use the stack to recognize $\{0^n1^n\}$

Computing Using a Stack **Definition** Examples of Pushdown Automata

 \leftarrow

 Ω

Keeping Count Using the Stack

An automaton can use the stack to recognize $\{0^n1^n\}$

On reading a 0, push it onto the stack

Computing Using a Stack **Definition** Examples of Pushdown Automata

∢⊡

 Ω

- An automaton can use the stack to recognize $\{0^n1^n\}$
	- On reading a 0, push it onto the stack
	- After the 0s, on reading each 1, pop a 0

Computing Using a Stack Definition Examples of Pushdown Automata

 Ω

- An automaton can use the stack to recognize $\{0^n1^n\}$
	- On reading a 0, push it onto the stack
	- After the 0s, on reading each 1, pop a 0
	- (If a 0 comes after a 1, reject)

Computing Using a Stack Definition Examples of Pushdown Automata

つへへ

- An automaton can use the stack to recognize $\{0^n1^n\}$
	- On reading a 0, push it onto the stack
	- After the 0s, on reading each 1, pop a 0
	- (If a 0 comes after a 1, reject)
	- If attempt to pop an empty stack, reject

Computing Using a Stack Definition Examples of Pushdown Automata

つくへ

- An automaton can use the stack to recognize $\{0^n1^n\}$
	- On reading a 0, push it onto the stack
	- After the 0s, on reading each 1, pop a 0
	- (If a 0 comes after a 1, reject)
	- If attempt to pop an empty stack, reject
	- **If stack not empty at the end, reject**

Computing Using a Stack Definition Examples of Pushdown Automata

つくへ

- An automaton can use the stack to recognize $\{0^n1^n\}$
	- On reading a 0, push it onto the stack
	- After the 0s, on reading each 1, pop a 0
	- (If a 0 comes after a 1, reject)
	- If attempt to pop an empty stack, reject
	- **If stack not empty at the end, reject**
	- **•** Else accept

Computing Using a Stack **Definition** Examples of Pushdown Automata

4 0 8

Þ

 \rightarrow

Ε

 $2Q$

Matching Parenthesis Using the Stack

An automaton can use the stack to recognize balanced parenthesis

Computing Using a Stack **Definition** Examples of Pushdown Automata

€⊡

重き

 Ω

- An automaton can use the stack to recognize balanced parenthesis
- \bullet e.g. $(())()$ is balanced, but $()()()$ and $(()$ are not

Computing Using a Stack Definition Examples of Pushdown Automata

 200

- An automaton can use the stack to recognize balanced parenthesis
- \bullet e.g. $(())()$ is balanced, but $()()()$ and $(()$ are not
	- On seeing a (push it on the stack

 Ω

- An automaton can use the stack to recognize balanced parenthesis
- \bullet e.g. $(())()$ is balanced, but $()()()$ and $(()$ are not
	- On seeing a (push it on the stack
	- On seeing a) pop a (from the stack

 Ω

- An automaton can use the stack to recognize balanced parenthesis
- \bullet e.g. $(())()$ is balanced, but $()()()$ and $(()$ are not
	- On seeing a (push it on the stack
	- On seeing a) pop a (from the stack
	- If attempt to pop an empty stack, reject

 2990

- An automaton can use the stack to recognize balanced parenthesis
- \bullet e.g. $(())()$ is balanced, but $()()()$ and $(()$ are not
	- On seeing a (push it on the stack
	- On seeing a) pop a (from the stack
	- If attempt to pop an empty stack, reject
	- If stack not empty at the end, reject

 2990

- An automaton can use the stack to recognize balanced parenthesis
- e.g. $(())()$ is balanced, but $()()()$ and $()$ are not
	- On seeing a (push it on the stack
	- On seeing a) pop a (from the stack
	- If attempt to pop an empty stack, reject
	- **If stack not empty at the end, reject**
	- Else accept

Computing Using a Stack **Definition** Examples of Pushdown Automata

← ロ ▶ → イ 冊

Þ

 \rightarrow × Þ. ×. 后 $\,$ 哇 299

Pushdown Automata (PDA)

A Pushdown Automaton

Computing Using a Stack **Definition** Examples of Pushdown Automata

4 0 8

∍

Þ

つへへ

Pushdown Automata (PDA)

A Pushdown Automaton

• Like an NFA with ϵ -transitions, but with a stack

Computing Using a Stack Definition Examples of Pushdown Automata

4 O F

 Ω

Pushdown Automata (PDA)

A Pushdown Automaton

- Like an NFA with ϵ -transitions, but with a stack
	- Stack depth unlimited: not a finite-state machine

Computing Using a Stack Definition Examples of Pushdown Automata

Pushdown Automata (PDA)

A Pushdown Automaton

- Like an NFA with ϵ -transitions, but with a stack
	- Stack depth unlimited: not a finite-state machine
	- Non-deterministic: accepts if any thread of execution accepts

 Ω

Computing Using a Stack **Definition** Examples of Pushdown Automata

Ε おす 高々 299

准

Pushdown Automata (PDA)

Has a non-deterministic finite-state control

Computing Using a Stack **Definition** Examples of Pushdown Automata

4 0 8

- ∢ n⊽ \sim ∍ Þ. 299

Ε

一 (語)

- Has a non-deterministic finite-state control
- At every step:

Computing Using a Stack Definition Examples of Pushdown Automata

€⊡

Þ

-b

Ε

 299

- **Has a non-deterministic finite-state control**
- At every step:
	- Consume next input symbol (or none)

Computing Using a Stack **Definition** Examples of Pushdown Automata

∢⊡

 Ω

- **Has a non-deterministic finite-state control**
- At every step:
	- Consume next input symbol (or none) and pop the top symbol on stack (or none)

Computing Using a Stack Definition Examples of Pushdown Automata

 Ω

- **Has a non-deterministic finite-state control**
- At every step:
	- Consume next input symbol (or none) and pop the top symbol on stack (or none)
	- Based on current state, consumed input symbol and popped stack symbol, do (non-deterministically):

Computing Using a Stack Definition Examples of Pushdown Automata

 2990

- **Has a non-deterministic finite-state control**
- At every step:
	- Consume next input symbol (or none) and pop the top symbol on stack (or none)
	- Based on current state, consumed input symbol and popped stack symbol, do (non-deterministically):
		- 1 push a symbol onto stack (or push none)

Computing Using a Stack **Definition** Examples of Pushdown Automata

 Ω

- **Has a non-deterministic finite-state control**
- At every step:
	- Consume next input symbol (or none) and pop the top symbol on stack (or none)
	- Based on current state, consumed input symbol and popped stack symbol, do (non-deterministically):
		- 1 push a symbol onto stack (or push none)
		- 2 change to a new state

Computing Using a Stack Definition Examples of Pushdown Automata

Pushdown Automata (PDA)

- Has a non-deterministic finite-state control
- At every step:
	- Consume next input symbol (or none) and pop the top symbol on stack (or none)
	- Based on current state, consumed input symbol and popped stack symbol, do (non-deterministically):
		- 1 push a symbol onto stack (or push none)
		- 2 change to a new state

If at q_1 , with next input symbol a and top of stack x, then can consume a, pop x, push y onto stack and move to q_2

 QQ

Computing Using a Stack Definition Examples of Pushdown Automata

Pushdown Automata (PDA)

- Has a non-deterministic finite-state control
- At every step:
	- Consume next input symbol (or none) and pop the top symbol on stack (or none)
	- Based on current state, consumed input symbol and popped stack symbol, do (non-deterministically):
		- 1 push a symbol onto stack (or push none)
		- 2 change to a new state

If at q_1 , with next input symbol a and top of stack x, then can consume a, pop x, push y onto stack and move to q_2 (any of a, x, y may be ϵ)

 QQ

Computing Using a Stack Definition Examples of Pushdown Automata

つへへ

Pushdown Automata (PDA): Formal Definition

A PDA
$$
P = (Q, \Sigma, \Gamma, \delta, q_0, F)
$$
 where

- \bullet Q = Finite set of states
- $\bullet \Sigma$ = Finite input alphabet
- $\bullet \Gamma =$ Finite stack alphabet
- $q_0 =$ Start state
- $F \subset Q =$ Accepting/final states
- $\bullet \delta : Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\epsilon\}))$
Computing Using a Stack Examples of Pushdown Automata

4 0 8

∢ ⊕ \sim 299

₿

Þ

-b

Matching Parenthesis: PDA construction

Computing Using a Stack Examples of Pushdown Automata

4 **D** F

つへへ

Matching Parenthesis: PDA construction

• First push a "bottom-of-the-stack" symbol \$ and move to q

Computing Using a Stack Definition Examples of Pushdown Automata

 Ω

Matching Parenthesis: PDA construction

• First push a "bottom-of-the-stack" symbol \$ and move to q

• On seeing a (push it onto the stack

Computing Using a Stack Definition Examples of Pushdown Automata

 Ω

Matching Parenthesis: PDA construction

- First push a "bottom-of-the-stack" symbol \$ and move to q
- On seeing a (push it onto the stack
- On seeing a) pop if a (is in the stack

Computing Using a Stack Definition Examples of Pushdown Automata

Matching Parenthesis: PDA construction

- First push a "bottom-of-the-stack" symbol $\frac{1}{2}$ and move to q
- On seeing a (push it onto the stack
- On seeing a) pop if a (is in the stack
- Pop \$ and move to final state q_F

つへへ

Computing Using a Stack Examples of Pushdown Automata

> 4 0 8 一个问

∍

 \rightarrow × Þ. × \rightarrow

后

 299

哇

Computing Using a Stack Examples of Pushdown Automata

4 0 8

同

× \rightarrow $2Q$

∍

Þ

 \sim

哇

Computing Using a Stack Examples of Pushdown Automata

Matching Parenthesis: PDA execution

4 0 8

同

× \rightarrow ∍ \sim $2Q$

∍

Þ

哇

Computing Using a Stack Examples of Pushdown Automata

Matching Parenthesis: PDA execution

4 0 8

 \mathcal{A} a \sim $2Q$

∍

Þ

 \sim

Ε

Computing Using a Stack Examples of Pushdown Automata

4 0 8

 \mathcal{A} a \sim 299

₿

∍

Þ

 \sim

Computing Using a Stack Examples of Pushdown Automata

Computing Using a Stack Examples of Pushdown Automata

Computing Using a Stack Examples of Pushdown Automata

K ロ ▶ K 何 ▶

Ε

× Þ. \leftarrow

活 \rightarrow Ε 299

Palindrome: PDA construction

Computing Using a Stack Examples of Pushdown Automata

4 D F

 299

Palindrome: PDA construction

First push a "bottom-of-the-stack" symbol \$ and move to a pushing state

Computing Using a Stack Examples of Pushdown Automata

∢⊡

 Ω

Palindrome: PDA construction

- First push a "bottom-of-the-stack" symbol \$ and move to a pushing state
- Push input symbols onto the stack

Computing Using a Stack Definition Examples of Pushdown Automata

つくへ

Palindrome: PDA construction

- First push a "bottom-of-the-stack" symbol \$ and move to a pushing state
- Push input symbols onto the stack
- Non-deterministically move to a popping state (with or without consuming a single input symbol)

Computing Using a Stack Definition Examples of Pushdown Automata

Palindrome: PDA construction

- First push a "bottom-of-the-stack" symbol \$ and move to a pushing state
- Push input symbols onto the stack
- Non-deterministically move to a popping state (with or without consuming a single input symbol)
- If next input symbol is same as top of stack, pop

つくへ

Computing Using a Stack Definition Examples of Pushdown Automata

つくへ

Palindrome: PDA construction

- First push a "bottom-of-the-stack" symbol \$ and move to a pushing state
- Push input symbols onto the stack
- Non-deterministically move to a popping state (with or without consuming a single input symbol)
- **If next input symbol is same as top of stack, pop**
- If \$ on top of stack move to accept state

Definition Examples of Pushdown Automata

重

メメ 高々

K ロ ▶ K @ ▶ K 동

 299

Definition Examples of Pushdown Automata

重

おす 高々

K ロ ▶ | K @ ▶ | K ミ

 299

Computing Using a Stack Definition Examples of Pushdown Automata

K ロ ▶ K 何 ▶ K

Ε

一 (語) \rightarrow

重

 299

Computing Using a Stack Definition Examples of Pushdown Automata

K ロ ▶ K 何 ▶ K

Ε

一 (語) \rightarrow

重

 299

Computing Using a Stack Definition Examples of Pushdown Automata

Palindrome: PDA execution

K ロ ▶ K 何 ▶ K

Ε

一 (語) \rightarrow

重

 299

Computing Using a Stack Definition Examples of Pushdown Automata

Palindrome: PDA execution

K ロ ▶ K 何 ▶ K

Ε

メモト \rightarrow

重

 299

Computing Using a Stack Definition Examples of Pushdown Automata

Computation Language Recognized Expressive Power

4 0 8

Ε

 Ω

Instantaneous Description

In order to describe a machine's execution, we need to capture a "snapshot" of the machine that completely determines future behavior

Computation Language Recognized Expressive Power

 2990

Instantaneous Description

In order to describe a machine's execution, we need to capture a "snapshot" of the machine that completely determines future behavior

• In the case of an NFA (or DFA), it is the state

Computation Language Recognized Expressive Power

つくへ

Instantaneous Description

In order to describe a machine's execution, we need to capture a "snapshot" of the machine that completely determines future behavior

- In the case of an NFA (or DFA), it is the state
- In the case of a TM, it is the state, head position, and tape contents

Computation Language Recognized Expressive Power

つくへ

Instantaneous Description

In order to describe a machine's execution, we need to capture a "snapshot" of the machine that completely determines future behavior

- In the case of an NFA (or DFA), it is the state
- In the case of a TM, it is the state, head position, and tape contents
- \bullet In the case of a PDA, it is the state $+$ stack contents

Computation Language Recognized Expressive Power

Instantaneous Description

In order to describe a machine's execution, we need to capture a "snapshot" of the machine that completely determines future behavior

- In the case of an NFA (or DFA), it is the state
- In the case of a TM, it is the state, head position, and tape contents
- \bullet In the case of a PDA, it is the state $+$ stack contents

Definition

An instantaneous description of a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is a pair $\langle q, \sigma \rangle$, where $q \in Q$ and $\sigma \in \Gamma^*$

つくへ

Computation Language Recognized Expressive Power

∢ ロ ⊁ (何) (ミ) (ミ) (

 2990

Computation

Definition

For a PDA $P=(Q,\Sigma,\Gamma,\delta,q_0,F),$ string $w\in\Sigma^*$, and instantaneous descriptions $\langle q_1, \sigma_1 \rangle$ and $\langle q_2, \sigma_2 \rangle$, we say $\langle q_1, \sigma_1\rangle \stackrel{w}{\longrightarrow}_P \langle q_2, \sigma_2\rangle$ iff there is a sequence of instanteous descriptions $\langle r_0, s_0 \rangle$, $\langle r_1, s_1 \rangle$, . . . $\langle r_k, s_k \rangle$ and a sequence $x_1, x_2, \ldots x_k$, where for each $i, x_i \in \Sigma \cup \{\epsilon\}$, such that

$$
\bullet \ \ w = x_1 x_2 \cdots x_k,
$$

Computation Language Recognized Expressive Power

∢ ロ ⊁ (何) (ミ) (ミ) (ニ)

 2990

Computation

Definition

For a PDA $P=(Q,\Sigma,\Gamma,\delta,q_0,F),$ string $w\in\Sigma^*$, and instantaneous descriptions $\langle q_1, \sigma_1 \rangle$ and $\langle q_2, \sigma_2 \rangle$, we say $\langle q_1, \sigma_1\rangle \stackrel{w}{\longrightarrow}_P \langle q_2, \sigma_2\rangle$ iff there is a sequence of instanteous descriptions $\langle r_0, s_0 \rangle$, $\langle r_1, s_1 \rangle$, . . . $\langle r_k, s_k \rangle$ and a sequence $x_1, x_2, \ldots x_k$, where for each $i, x_i \in \Sigma \cup \{\epsilon\}$, such that

$$
\bullet \ \ w = x_1 x_2 \cdots x_k,
$$

•
$$
r_0 = q_1
$$
, and $s_0 = \sigma_1$,

Computation Language Recognized Expressive Power

∢ ロ ⊁ (何) (ミ) (ミ) (ニ)

 2990

Computation

Definition

For a PDA $P=(Q,\Sigma,\Gamma,\delta,q_0,F),$ string $w\in\Sigma^*$, and instantaneous descriptions $\langle q_1, \sigma_1 \rangle$ and $\langle q_2, \sigma_2 \rangle$, we say $\langle q_1, \sigma_1\rangle \stackrel{w}{\longrightarrow}_P \langle q_2, \sigma_2\rangle$ iff there is a sequence of instanteous descriptions $\langle r_0, s_0 \rangle$, $\langle r_1, s_1 \rangle$, ... $\langle r_k, s_k \rangle$ and a sequence $x_1, x_2, \ldots x_k$, where for each $i, x_i \in \Sigma \cup \{\epsilon\}$, such that

$$
\bullet \ \ w = x_1 x_2 \cdots x_k,
$$

$$
\bullet \ \ r_0=q_1, \text{ and } s_0=\sigma_1,
$$

•
$$
r_k = q_2
$$
, and $s_k = \sigma_2$,

Computation Language Recognized Expressive Power

Computation

Definition

For a PDA $P=(Q,\Sigma,\Gamma,\delta,q_0,F),$ string $w\in\Sigma^*$, and instantaneous descriptions $\langle q_1, \sigma_1 \rangle$ and $\langle q_2, \sigma_2 \rangle$, we say $\langle q_1, \sigma_1\rangle \stackrel{w}{\longrightarrow}_P \langle q_2, \sigma_2\rangle$ iff there is a sequence of instanteous descriptions $\langle r_0, s_0 \rangle$, $\langle r_1, s_1 \rangle$, ... $\langle r_k, s_k \rangle$ and a sequence $x_1, x_2, \ldots x_k$, where for each $i, x_i \in \Sigma \cup \{\epsilon\}$, such that

$$
\bullet \ \ w = x_1 x_2 \cdots x_k,
$$

$$
\bullet \ \ r_0=q_1, \text{ and } s_0=\sigma_1,
$$

•
$$
r_k = q_2
$$
, and $s_k = \sigma_2$,

for every i , $(r_{i+1}, b) \in \delta(r_i, x_{i+1}, a)$ such that $s_i = as$ and $s_{i+1} = bs$, where $a, b \in \Gamma \cup \{\epsilon\}$ and $s \in \Gamma^*$

←ロ→ → 何→ → ヨ→ → ヨ→ →

 2990

Computation Language Recognized Expressive Power

4 0 8

Ε 重

 \rightarrow

 \sim

Ε

 299

Example of Computation

Example

 $\langle q_0, \epsilon \rangle \stackrel{(1)}{\longrightarrow} \langle q, ((\text{\$}) \text{ because})$

Computation Language Recognized Expressive Power

Example of Computation

Example

 $\langle q_0, \epsilon \rangle \stackrel{(1)}{\longrightarrow} \langle q, ((\text{\$}) \text{ because})$ $\langle q_0, \epsilon \rangle \stackrel{x_1=\epsilon}{\longrightarrow} \langle q, \$ \rangle \stackrel{x_2=\left(\ \langle q, (\$ \rangle \stackrel{x_3=\left(\ \langle q, ((\$ \rangle \stackrel{x_4=\left>\ }{\longrightarrow} \langle q, (\$ \rangle \stackrel{x_5=\left(\ \langle q, ((\$ \rangle \rangle \langle q, (\$ \rangle \langle$

メロメ メ母メ メミメ メミメー

 Ω
Language Recognized Expressive Power

メロメ メ御 メメ きょく ミメー

准

 $2Q$

Acceptance/Recognition

Definition

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts a string $w \in \Sigma^*$ iff

Computation Language Recognized Expressive Power

イロト イ母 トイヨ トイヨ トー

Ε

 Ω

Acceptance/Recognition

Definition

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts a string $w \in \Sigma^*$ iff for some $q \in F$ and $\sigma \in \Gamma^*, \langle q_0, \epsilon \rangle \stackrel{w}{\longrightarrow} _P \langle q, \sigma \rangle$

Computation Language Recognized Expressive Power

メロト メ母 トメ ミト メミトー

 2990

Acceptance/Recognition

Definition

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts a string $w \in \Sigma^*$ iff for some $q \in F$ and $\sigma \in \Gamma^*, \langle q_0, \epsilon \rangle \stackrel{w}{\longrightarrow} _P \langle q, \sigma \rangle$

Definition

The language recognized/accepted by a PDA $P=(Q,\Sigma,\Gamma,\delta,q_0,F)$ is $L(P)=\{w\in\Sigma^* \mid P \text{ accepts } w\}.$

Computation Language Recognized Expressive Power

→ イラン イヨン イヨン

 2990

Acceptance/Recognition

Definition

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts a string $w \in \Sigma^*$ iff for some $q \in F$ and $\sigma \in \Gamma^*, \langle q_0, \epsilon \rangle \stackrel{w}{\longrightarrow} _P \langle q, \sigma \rangle$

Definition

The language recognized/accepted by a PDA $P=(Q, \Sigma, \Gamma, \delta, q_0, F)$ is $L(P)=\{w\in \Sigma^* \mid P \text{ accepts } w\}$. A language L is said to be accepted/recognized by P if $L = L(P)$.

Computation Language Recognized Expressive Power

> 4 0 8 → 母→

∍ メメ者を Ε

 299

Expressive Power of CFGs and PDAs

CFGs and PDAs have equivalent expressive powers. More formally,

. . .

Computation Language Recognized Expressive Power

→ 手下

 \leftarrow \Box

 2990

Expressive Power of CFGs and PDAs

CFGs and PDAs have equivalent expressive powers. More formally,

. . .

Theorem

For every CFG G, there is a PDA P such that $L(G) = L(P)$. In addition, for every PDA P, there is a CFG G such that $L(P) = L(G)$.

Computation Language Recognized Expressive Power

つくへ

Expressive Power of CFGs and PDAs

CFGs and PDAs have equivalent expressive powers. More formally,

. . .

Theorem

For every CFG G, there is a PDA P such that $L(G) = L(P)$. In addition, for every PDA P, there is a CFG G such that $L(P) = L(G)$. Thus, L is context-free iff there is a PDA P such that $L = L(P)$.

Computation Language Recognized Expressive Power

つくへ

Expressive Power of CFGs and PDAs

CFGs and PDAs have equivalent expressive powers. More formally,

. . .

Theorem

For every CFG G, there is a PDA P such that $L(G) = L(P)$. In addition, for every PDA P, there is a CFG G such that $L(P) = L(G)$. Thus, L is context-free iff there is a PDA P such that $L = L(P)$.

Proof.

Skipped. \Box