

PDA and CFG conversions

CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Fall 2019

Chinese University of Hong Kong

L has a context-free grammar if and only if it is accepted by some pushdown automaton.



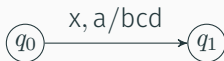
Will first convert CFG to PDA

Convention

A sequence of transitions like



will be abbreviated as



replace a by bcd on stack

Converting a CFG to a PDA

Idea: Use PDA to simulate derivations

Example:

$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

Rules:

1. Write the start symbol A onto the stack
2. Rewrite variable on top of stack (in reverse) according to production

PDA control		stack	input
write start variable	$\epsilon, \epsilon/A$	$\$A$	00#11
replace by production in reverse	$\epsilon, A/1A0$	$\$1A0$	00#11

Converting a CFG to a PDA

Idea: Use PDA to simulate derivations

Example:

$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

Rules:

1. Write the start symbol A onto the stack
2. Rewrite variable on top of stack (in reverse) according to production
3. Pop top terminal if it matches input

PDA control		stack	input
write start variable	$\epsilon, \epsilon/A$	$\$A$	00#11
replace by production in reverse	$\epsilon, A/1A0$	$\$1A0$	00#11
pop terminal and match	$0, 0/\epsilon$	$\$1A$	0#11
replace by production in reverse	$\epsilon, A/1A0$	$\$11A0$	0#11
	\vdots		

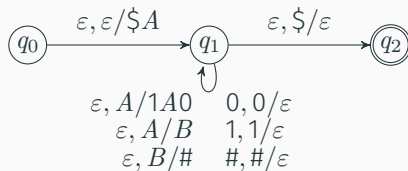
Converting a CFG to a PDA

CFG

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$



input
stack

00#11
\$A

00#11
\$1A0

00#11
\$1A

00#11
\$11A0

00#11
\$11A

00#11
\$11B

00#11
\$11#

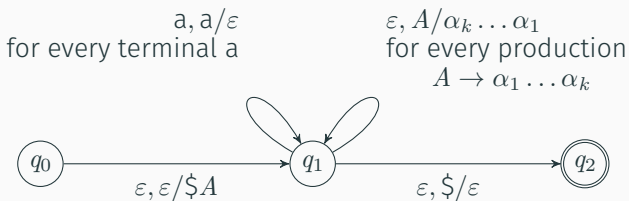
00#11
\$11

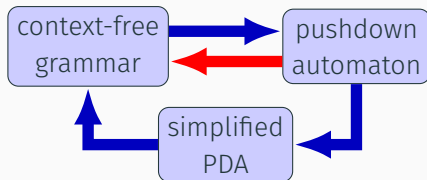
00#11
\$1

00#11
\$

$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$

General CFG to PDA conversion





Simplified pushdown automaton:

- Has a **single accepting state**
- **Empties its stack** before accepting
- Each transition is either a push, or a pop, but not both

Simplifying the PDA

Single accepting state



Empties its stack before accepting

$\epsilon, a/\epsilon$ for every stack symbol a



Simplifying the PDA

Each transition either pushes or pops, but not both



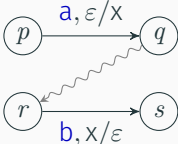


For every pair (q, r) of states in PDA, introduce variable A_{qr} in CFG

Intention:

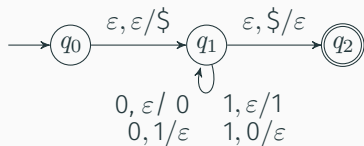
A_{qr} generates all strings that allow the PDA to go from q to r
(with empty stack both at q and at r)

Simplified PDA to CFG

PDA	CFG
	$A_{qq} \rightarrow \varepsilon$
	$A_{pr} \rightarrow A_{pq}A_{qr}$
	$A_{ps} \rightarrow aA_{qr}b$ $a = \varepsilon$ or $b = \varepsilon$ allowed

Start variable: A_{pq} (initial state p , accepting state q)

Example: Simplified PDA to CFG

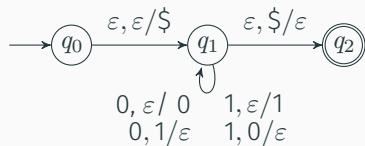


productions:

variables:

start variable:

Example: Simplified PDA to CFG



productions:

$$A_{02} \rightarrow A_{01} A_{12}$$

$$A_{01} \rightarrow A_{01} A_{11}$$

$$A_{12} \rightarrow A_{11} A_{12}$$

$$A_{11} \rightarrow A_{11} A_{11}$$

$$A_{11} \rightarrow 0 A_{11} 1$$

$$A_{11} \rightarrow 1 A_{11} 0$$

$$A_{02} \rightarrow A_{11}$$

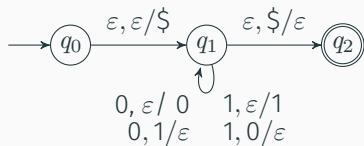
$$A_{00} \rightarrow \epsilon, A_{11} \rightarrow \epsilon,$$

$$A_{22} \rightarrow \epsilon$$

variables: $A_{00}, A_{11}, A_{22},$
 A_{01}, A_{02}, A_{12}

start variable: A_{02}

Example: Simplified PDA to CFG



variables: $A_{00}, A_{11}, A_{22},$
 A_{01}, A_{02}, A_{12}

start variable: A_{02}

productions:

$A_{02} \rightarrow A_{01} A_{12}$

$A_{01} \rightarrow A_{01} A_{11}$

$A_{12} \rightarrow A_{11} A_{12}$

$A_{11} \rightarrow A_{11} A_{11}$

$A_{11} \rightarrow 0 A_{11} 1$

$A_{11} \rightarrow 1 A_{11} 0$

$A_{02} \rightarrow A_{11}$

$A_{00} \rightarrow \epsilon, A_{11} \rightarrow \epsilon,$

$A_{22} \rightarrow \epsilon$

