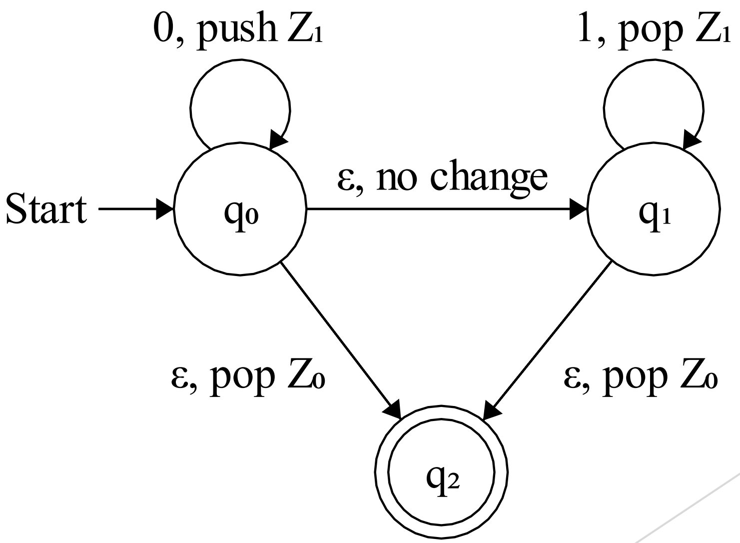
Pushdown Automata

COMP 455 – 002, Spring 2019

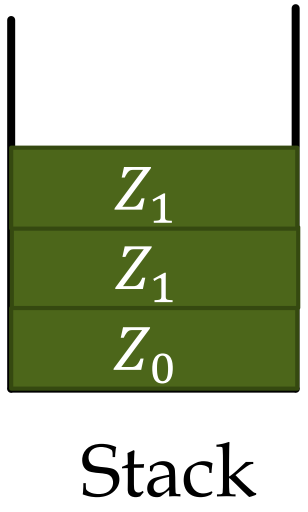
Pushdown Automata (PDAs)

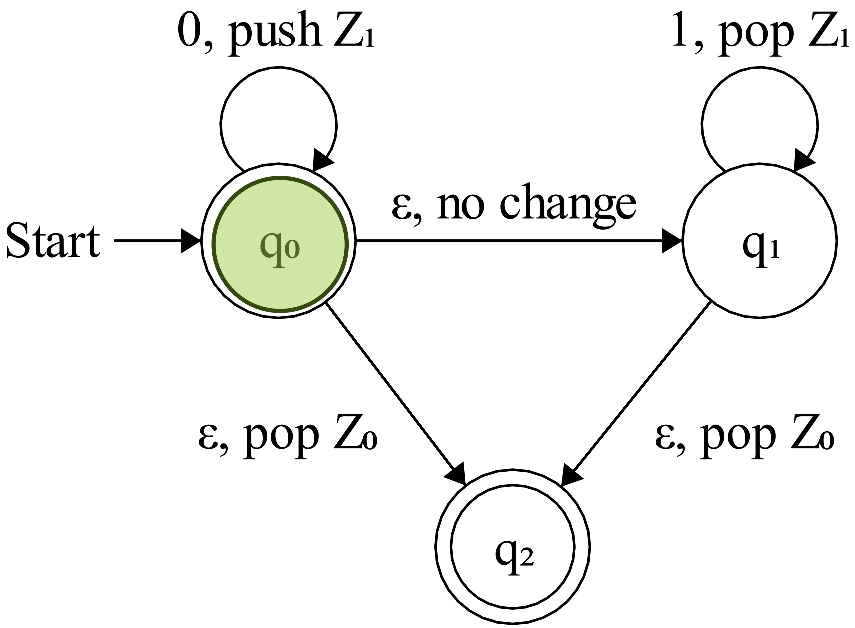
- A *pushdown automaton* (PDA) is essentially a finite automaton with a stack.
- Example PDA accepting $L = \{0^n 1^n \mid n \geq 0\}$:



- Initially, the symbol Z_0 is on the stack.
- Acceptance can be by final state or empty stack.

Example PDA Execution





Input string: $\boxed{0}011$

Current input

Formal Definition of a PDA

- A PDA can be defined by a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$.
  - Q: A finite set of states
  - Σ: The input alphabet
  - Γ: The stack alphabet
  - δ: The transition function: $\boxed{Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma} \rightarrow \boxed{2^{Q \times \Gamma^*}}$
  - q_0: The start state
  - $Z_0 \in \Gamma$: The initial stack symbol
  - F: The set of accepting states

Takes a state, an input symbol or ε, and a stack symbol.

Returns a set of (state, stack symbol string) pairs.

Moves in a PDA

$$\delta(\underline{q}, \underline{a}, \underline{Z}) = \{(p_1, \gamma_1, \dots, (p_m, \gamma_m)\} \equiv$$

If:

- The current state is q, **and**
- The current input symbol is a, **and**
- The symbol Z is on the top of the stack

…

Moves in a PDA

$$\delta(q, a, Z) = \{(p_1, \gamma_1), \ldots, (p_m, \gamma_m)\} \equiv$$

If the current state is q, the current input symbol is a, and the symbol Z is on the top of the stack,

Then choose a value i, (nondeterministic!)

- Enter state p_i,
- Replace the symbol on top of the stack with γ_i, and
- Advance to the next input symbol.

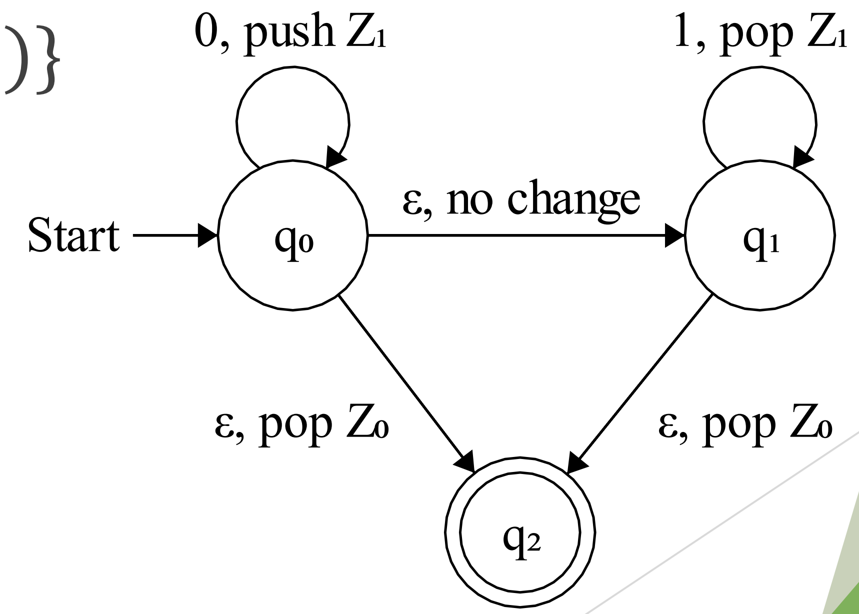
ε-Transitions in a PDA

$$\delta(q, \varepsilon, Z) = \{(p_1, \gamma_1, \ldots, (p_m, \gamma_m)\} \equiv$$

- The rules here are identical for the states and the stack, but no input is consumed.
  - This adds further nondeterminism.

Example: PDA Move Notation

- $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{Z_0, Z_1\}, \delta, q_0, Z_0, \{q_2\})$
- $\delta(q_0, 0, Z_0) = \{(q_0, Z_1 Z_0)\}$
- $\delta(q_0, 0, Z_1) = \{(q_0, Z_1 Z_1)\}$
- $\delta(q_0, \varepsilon, Z_0) = \{(q_2, \varepsilon), (q_1, Z_0)\}$
- $\delta(q_0, \varepsilon, Z_1) = \{(q_1, Z_1)\}$
- $\delta(q_1, 1, Z_1) = \{(q_1, \varepsilon)\}$
- $\delta(q_1, \varepsilon, Z_0) = \{(q_2, \varepsilon)\}$

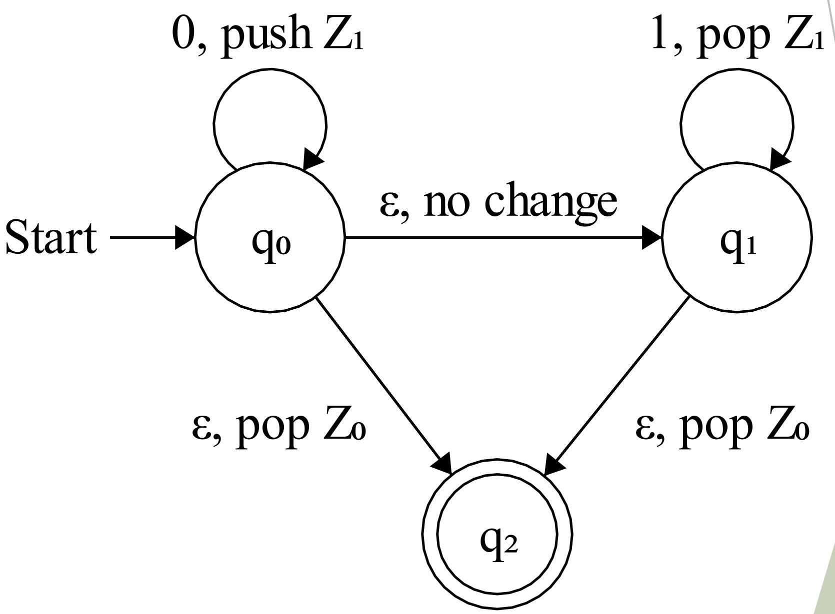


Instantaneous Description

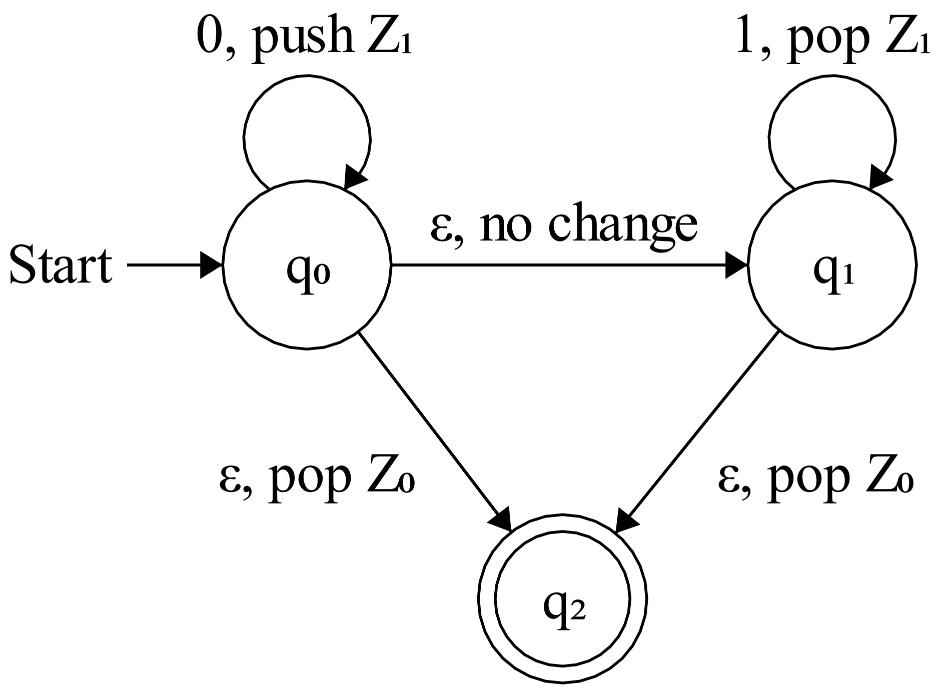
- An *instantaneous description* (ID) describes the "global state" of a PDA.
- An instantaneous description is given by (q, w, γ):
  - q: The current state
  - w: The remaining input
  - γ: The current contents of the stack
- We write $(q, aw, Z\alpha) \underset{M}{\vdash} (p, w, \beta\alpha)$ if $\delta(q, a, Z)$ contains (p, β).
- We can also write $\overset{i}{\underset{M}{\vdash}}$, $\overset{*}{\underset{M}{\vdash}}$, $\vdash$, $\overset{i}{\vdash}$, $\overset{*}{\vdash}$.

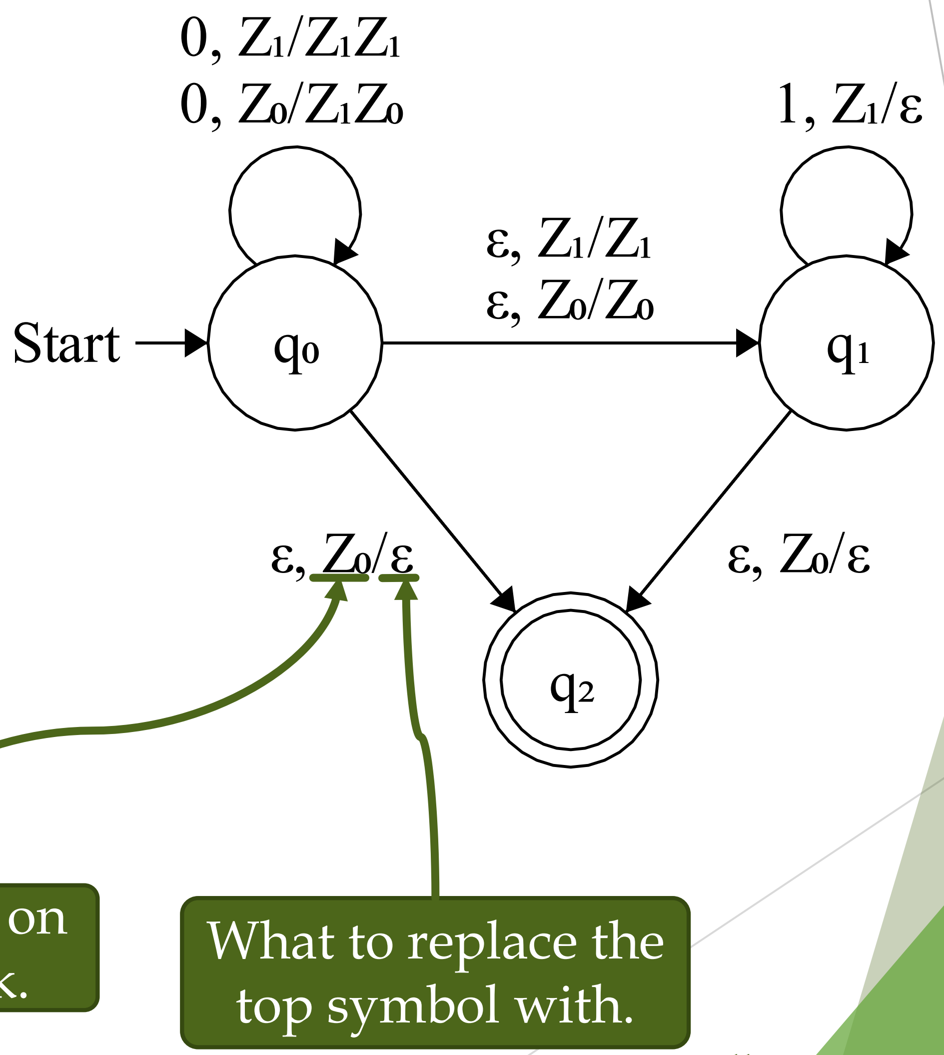
Instantaneous Description Example

$$
\begin{aligned}
(q_0, 0011, Z_0) \;&\vdash (q_0, 011, Z_1Z_0) \\
&\vdash (q_0, 11, Z_1Z_1Z_0) \\
&\vdash (q_1, 11, Z_1Z_1Z_0) \\
&\vdash (q_1, 1, Z_1Z_0) \\
&\vdash (q_1, \varepsilon, Z_0) \\
&\vdash (q_2, \varepsilon, \varepsilon)
\end{aligned}
$$



Formalizing Graphical Notation





The Language of a PDA

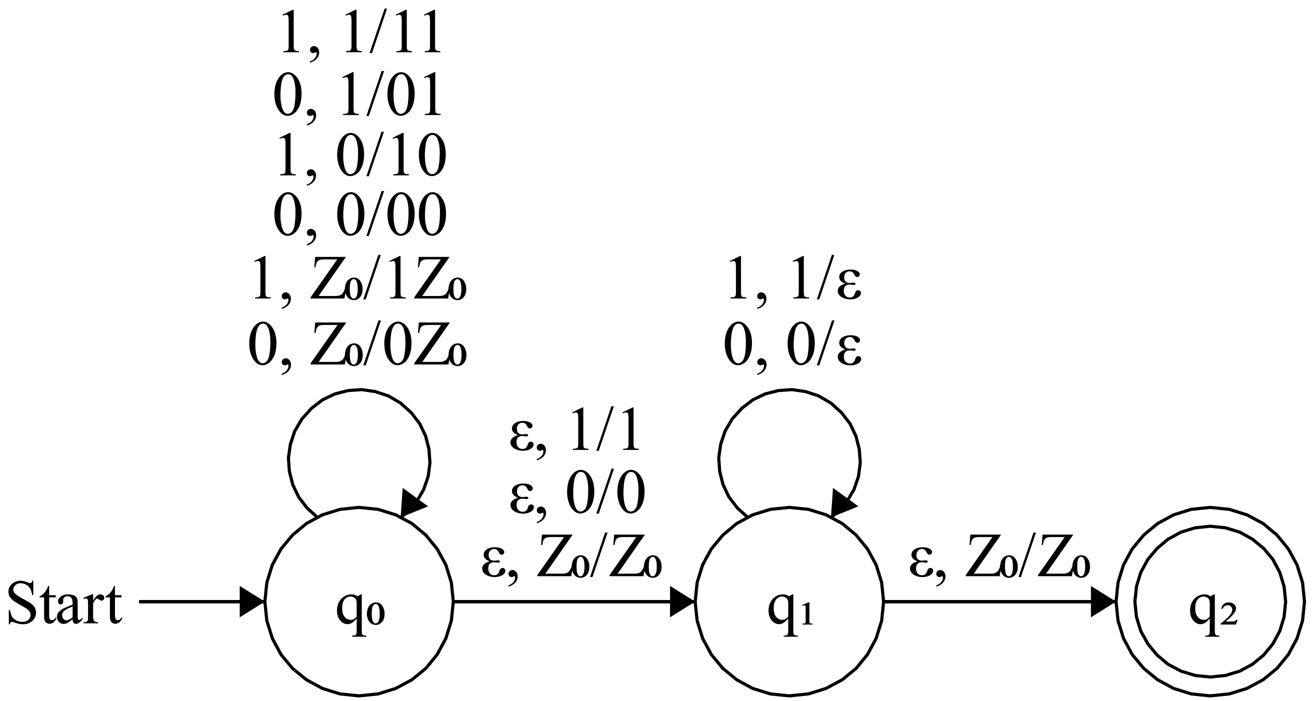
Let M be a PDA: $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

M defines *two* (possibly different) languages:

- The language accepted by *final state:*
  - $L(M) \equiv \left\{ w \mid (q_0, w, Z_0) \overset{*}{\vdash} (p, \varepsilon, \gamma),\ p \in F,\ \gamma \in \Gamma^* \right\}$
- The language accepted by *empty stack*:
  - $N(M) \equiv \left\{ w \mid (q_0, w, Z_0) \overset{*}{\vdash} (p, \varepsilon, \varepsilon),\ p \in Q \right\}$

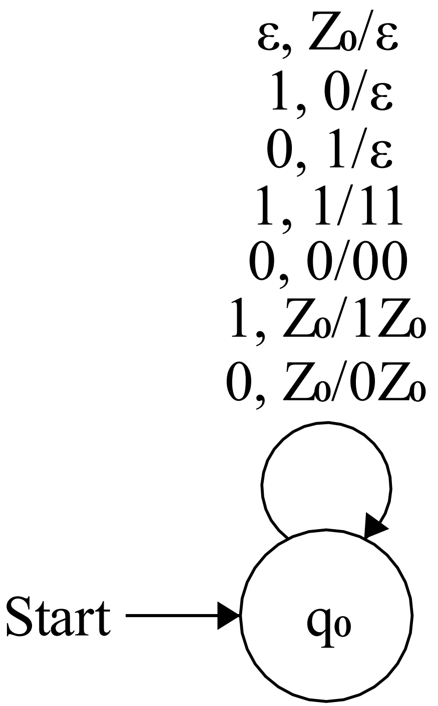
More PDA Examples

- $L = \{xx^R \mid x \in (\mathbf{0} + \mathbf{1})^*\}$



More PDA Examples

- L is the language consisting of strings with an equal number of 0s and 1s.



This accepts by empty stack. (You could also convert it to accept by final state, instead.)

Converting Empty Stack → Final State

Theorem 6.9: If $L = N(P_N)$ for some PDA P_N, then $L = L(P_F)$ for some PDA P_F.

- P_N: Accepts by empty stack
- P_F: Accepts by final state

Proof:

- Let $P_N = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$
- Then, $P_F = \left(Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\}\right)$, where…

Accepting states are unnecessary when accepting by empty stack.

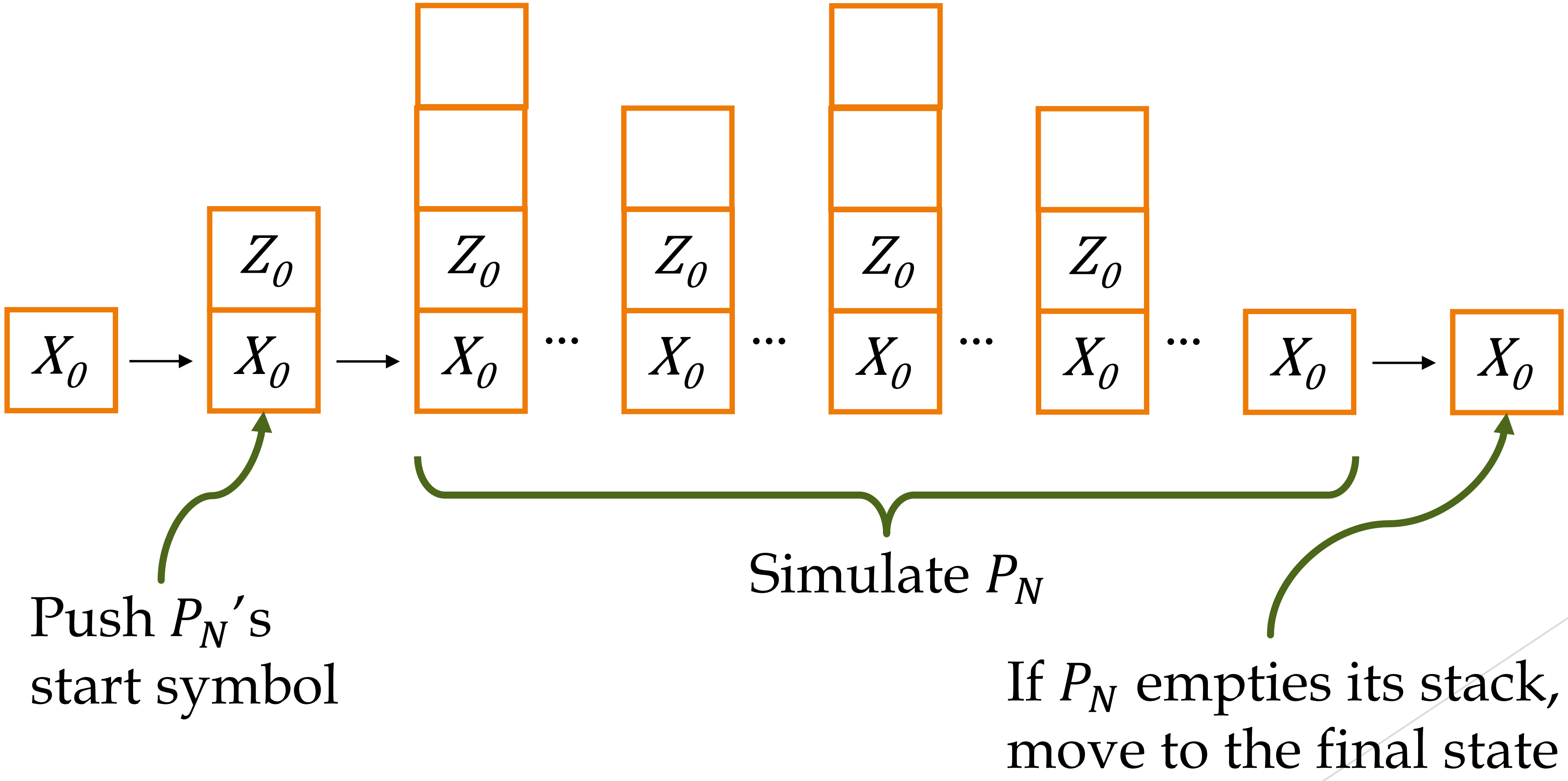
Converting Empty Stack → Final State

$P_N = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$
$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{\mathrm{X}_0\}, \delta_F, p_0, \mathrm{X}_0, \{p_f\})$

- $\delta_F(p_0, \varepsilon, \mathrm{X}_0) = \{(q_0, Z_0 X_0)\}$
  - P_F starts by pushing P_N's start symbol onto the stack. P_F will find its own start symbol (X_0) on top of the stack only when P_N's stack is empty.
- For all $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $\Upsilon \in \Gamma$, $\delta_F(q, a, \Upsilon)$ contains all pairs in $\delta_N(q, a, \Upsilon)$.
  - P_F just simulates P_N after initializing the stack.
- For all $q \in Q$, $\delta_F(q, \varepsilon, X_0)$ contains (p_f, ε).
  - P_F can accept by final state whenever P_N would by empty stack.

Converting Empty Stack → Final State

- The behavior of P_F (just showing the stack):



Converting Final State → Empty Stack

Theorem 6.11: If $L = L(P_F)$ for some PDA P_F, then $L = N(P_N)$ for some PDA P_N.

- P_F: Accepts by final state
- P_N: Accepts by empty stack

Proof:

- Let $P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
- Then, $P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0, \emptyset\}$, where…

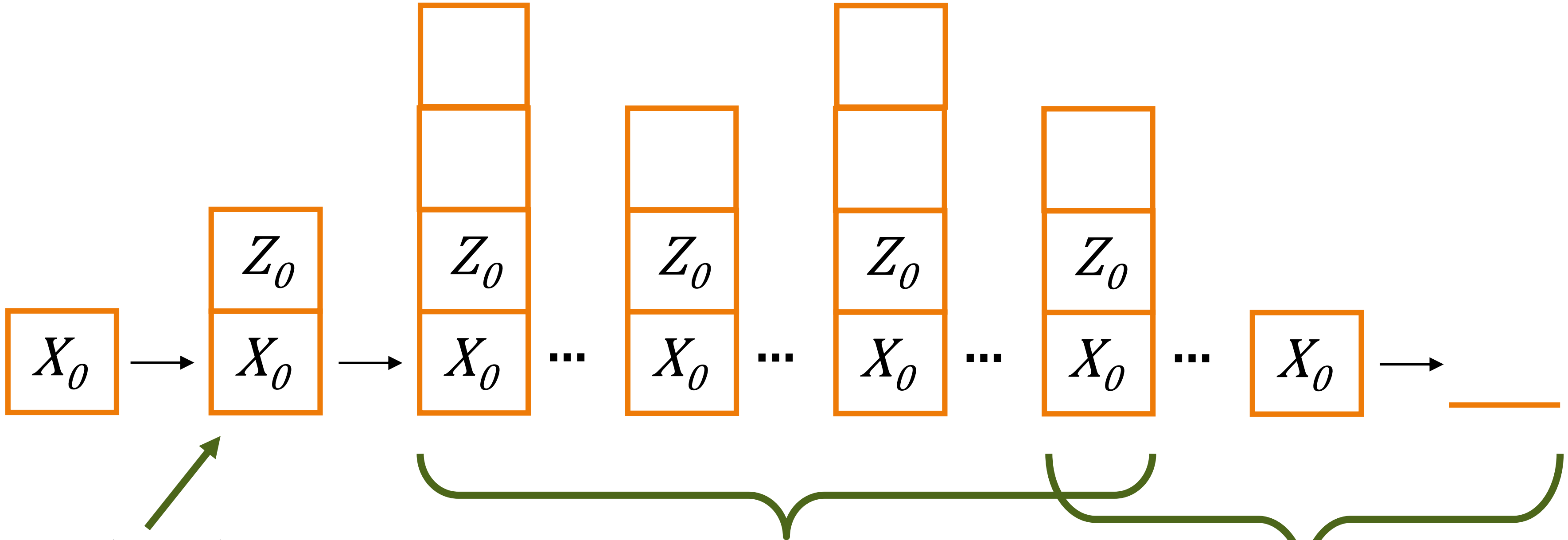
Converting Final State → Empty Stack

$P_F = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
$P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{\mathrm{X}_0\}, \delta_N, p_0, \mathrm{X}_0, \emptyset)$

- $\delta_N(p_0, \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$.
  - P_N starts by pushing P_F's start symbol onto the stack. X_0 ensures that P_N doesn't prematurely clear the stack.
- For all $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $\Upsilon \in \Gamma$, $\delta_F(q, a, \Upsilon)$ contains all pairs in $\delta_N(q, a, \Upsilon)$.
  - P_N simulates P_F after initializing the stack.
- For all $q \in F$, $\Upsilon \in \Gamma \cup X_0$, $\delta_N(q, \varepsilon, \Upsilon)$ contains (p, ε).
  - If P_F can accept by final state, P_N can begin emptying the stack.
- For all $\Upsilon \in \Gamma \cup X_0$, $\delta_N(p, \varepsilon, \Upsilon) = \{(p, \varepsilon)\}$.
  - P_N continues to empty its stack until it is completely empty.

Converting Final State → Empty Stack

- The behavior of P_N (just showing the stack):



Push P_F's start symbol on the stack.

Simulate P_F

If P_F enters a final state, go to state p and empty the stack.

Converting CFGs to PDAs

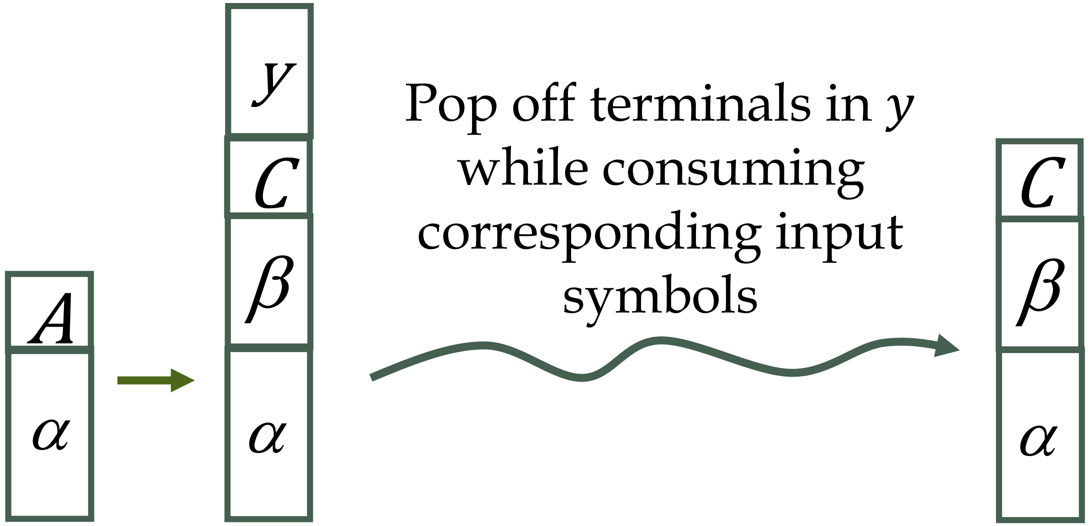
Given a CFG G, we will construct a PDA that simulates *leftmost derivations* in G.

Main ideas:

- Each left-sentential form of G is of the form $xA\alpha$, where:
  - x is all terminals,
  - A is the variable that will be replaced in the next derivation step, and
  - α is some string that can include both variables and terminals.
- If you look at the “state” of the PDA at this point,
  - $A\alpha$ will be on its stack, and
  - input x will have already been consumed.

Converting CFGs to PDAs

Apply the production $A \to yC\beta$ like this:



Replace A (on top of the stack) with all of the terminals and variables it produces.

Example PDA Execution for a CFG

Productions:

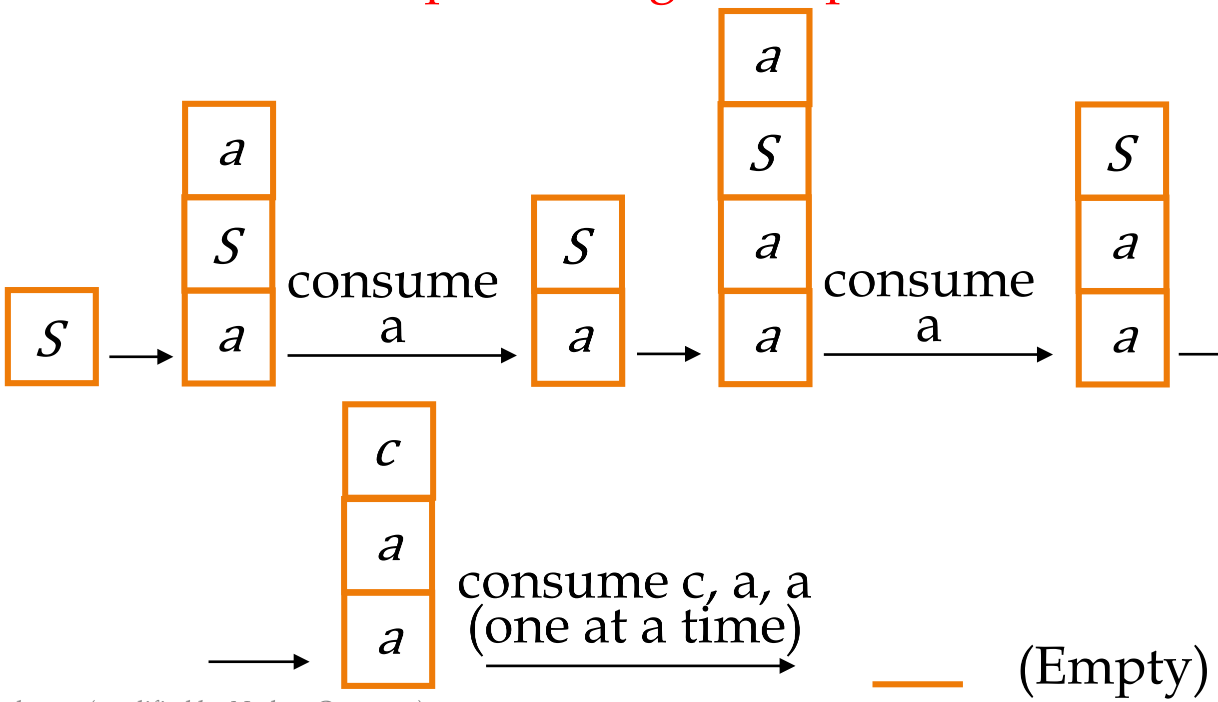
- $S \rightarrow SS \mid \varepsilon \mid aSa \mid bA$
- $A \rightarrow c$

PDA transition function:

- $\delta(q, \varepsilon, S) = \{(q, SS), (q, \varepsilon), (q, aSa), \ (q, bA)\}$
- $\delta(q, \varepsilon, A) = \{(q, c)\}$
- $\delta(q, d, d) = \{(q, \varepsilon)\}$, where $d \in \{a, b, c\}$

A LM derivation: $S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabAaa \Rightarrow aabcaa$.

The PDA's stack when processing the input *aabcaa*:



Formal Definition of a PDA for a CFG

Define a PDA P accepting the language defined by CFG G:

- $P = (\{q\},\ T,\ V \cup T,\ \delta,\ q,\ S,\ \emptyset)$
- δ is defined by:
  - **Rule 1**: For each variable A: $\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \rightarrow \beta$ is a production$\}$
  - **Rule 2**: For each terminal a:
  $$\delta(q, a, a) = \{(q, \varepsilon)\}$$
- Note: this means that any CFG can be accepted by a PDA (by empty stack) *with only one state*.

Reminder:
- PDA definition: $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
- CFG definition: (V, T, S, P)

Proving CFG → PDA Correctness

Theorem 6.13: $N(P) = L(G)$, where P and G are defined as before.

Shorthand for the language of P accepted by empty stack.

Proof: We will show that w is in $N(P)$ if and only if it is in $L(G)$.

"If": If w is in $L(G)$ then it has a LM derivation $S = \gamma_1 \underset{lm}{\Rightarrow} \gamma_2 \underset{lm}{\Rightarrow} \dots \underset{lm}{\Rightarrow} \gamma_n = w$.

We will show by induction on i that $(q, w, S) \overset{*}{\vdash} (q, y_i, \alpha_i)$, where $\gamma_i = x_i \alpha_i$, and $x_i y_i = w$.

Proving CFG → PDA Correctness

Base case: $i = 1$, so $\gamma_1 = S$. This means $x_1 = \varepsilon$, $y_1 = w$, and $\alpha_1 = S$. So, $(q, w, S) \overset{*}{\vdash} (q_1, y_1, \alpha_1)$.

Inductive step: Assume $(q, w, S) \overset{*}{\vdash} (q, y_i, \alpha_i)$ by the inductive hypothesis. We need to show that $(q, w, S) \overset{*}{\vdash} (q, y_{i+1}, \alpha_{i+1})$.

Reminder: After simulating the derivation up to γ_i, y_i is the unconsumed input and α_i is the stack contents.

We need to show that P can make moves that simulate the next step.

Proving CFG → PDA Correctness

This follows from the definition of δ:

- α_i is of the form A ..., where A is a variable.
- The derivation step $\gamma_i \Rightarrow \gamma_{i+1}$ involves replacing A by some string β.
- By **Rule 1** in the definition of δ, we can replace A by β on top of the stack.
- By **Rule 2** in the definition of δ, any leading terminals in β can be popped off the stack and the corresponding input consumed.

Proving CFG → PDA Correctness

"Only if": We want to prove that if w is in $N(M)$, then w is in $L(G)$. To do so, we prove something more general:

Claim: If $(q, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$, then $A \underset{G}{\overset{*}{\Rightarrow}} x$.

In words: Consider running P when A is on top of the stack, and continue running until A (and anything that replaced it) has been popped off the stack. If x is the string of input symbols consumed while doing this, then x is derivable from A.

Example: From the example PDA, $(q, abca, S) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$ and $S \underset{G}{\overset{*}{\Rightarrow}} abca$.

Proving CFG → PDA Correctness

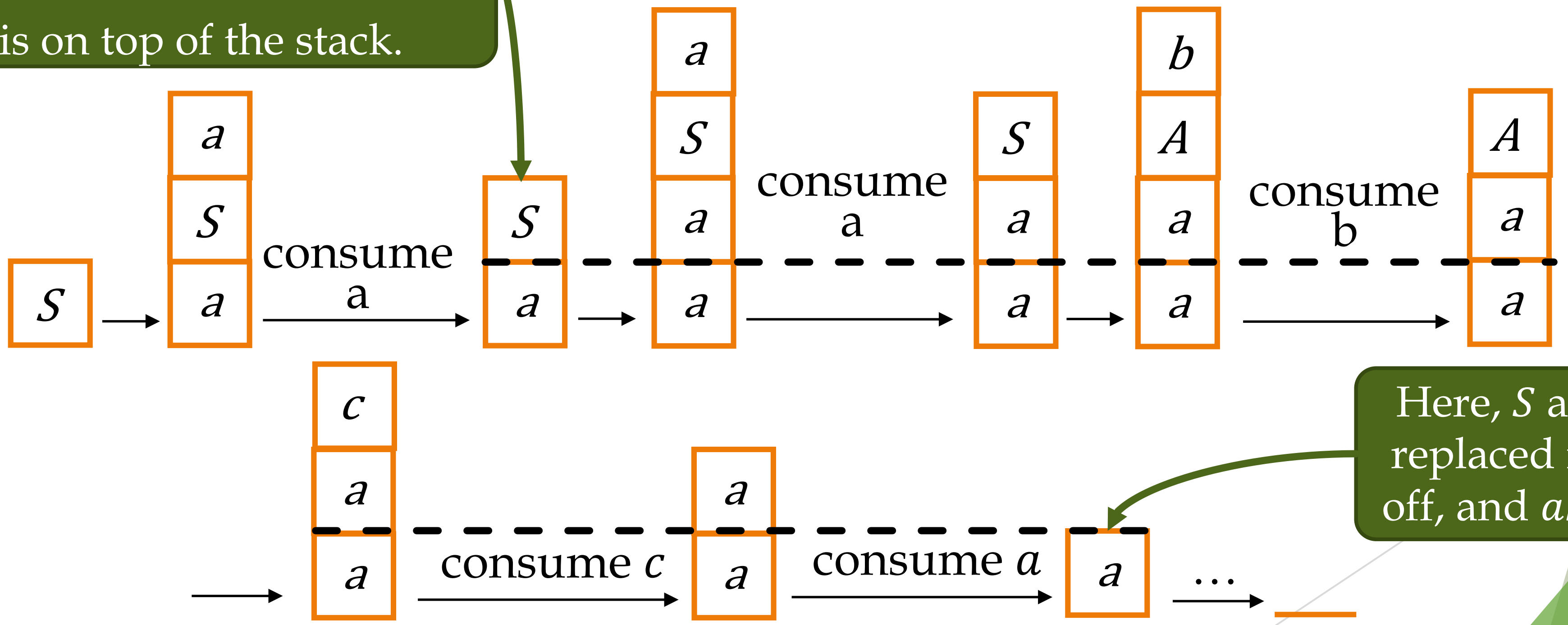
Revisiting the example from before:

$(q, abca, S) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$ and $S \underset{G}{\overset{*}{\Rightarrow}} abca$

Productions:

- $S \rightarrow SS \mid \varepsilon \mid aSa \mid bA$
- $A \rightarrow c$

We can see the derivation for $S \underset{G}{\overset{*}{\Rightarrow}} abca$ starting here, where S is on top of the stack.



Here, S and everything that replaced it has been popped off, and $abca$ was consumed.

Proving CFG → PDA Correctness

Claim: If $(q, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$, then $A \underset{G}{\overset{*}{\Rightarrow}} x$.

We will prove this by induction on the number of moves made by P.

Base case: P makes one move. A can be popped off the stack directly only if $A \to \varepsilon$ is a production (in which case $x = \varepsilon$). In this case, the claim that $A \underset{G}{\overset{*}{\Rightarrow}} \varepsilon$ follows.

Proving CFG → PDA Correctness

Claim:
$(q, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$, then $A \underset{G}{\overset{*}{\Rightarrow}} x$

Claim: If $(q, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$, then $A \underset{G}{\overset{*}{\Rightarrow}} x$.

Inductive step: Consider n moves, and assume that the claim is true for fewer than n moves.

The first move must be defined because of a production of the form $A \rightarrow Y_1 Y_2 \ldots Y_k$, where each Y_i is either a variable or terminal.

If Y_1 is a terminal symbol, then the only thing that P can do is pop it off the stack and consume the corresponding symbol in the input.

Proving CFG → PDA Correctness

Claim:
$(q, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$, then $A \underset{G}{\overset{*}{\Rightarrow}} x$

Inductive step, continued:

The first move must be defined because of a production of the form $A \rightarrow Y_1 Y_2 \ldots Y_k$, where each Y_i is either a variable or terminal.

If $\mathbf{Y_1}$ is a variable, then consider the behavior of P until Y_1, or whatever ends up replacing it, is erased from the stack. If y_1 is the string of input symbols consumed while doing this, then by the inductive hypothesis $Y_1 \overset{*}{\Rightarrow} y_1$.

In turn, this argument can be applied to $Y_2 \ldots Y_k$.

Proving CFG → PDA Correctness

Claim:
$(q, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$, then $A \underset{G}{\overset{*}{\Rightarrow}} x$

From the previous slide, it follows that $x = x_1 x_2 \ldots x_k$, where $x_i = Y_i$ if Y_i is a terminal and $x_i = y_i$ if Y_i is a variable. By concatenating these various derivations, we have $A \overset{*}{\Rightarrow} x$.

Converting PDAs to CFGs

Theorem 6.14: If $L = N(P)$ for some PDA P, then $L = L(G)$ for some CFG G.

These symbols will end up generating all strings that cause X to be popped from the stack while moving from state q to p.

Proof:

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$.

Define $G = (V, \Sigma, R, S)$ such that V contains $\boxed{[qXp]}$ for all $q, p \in Q$, and all $X \in \Gamma$.

R is defined on the next slide…

Converting PDAs to CFGs

R is defined as follows:

- $S \to [q_0 Z_0 p]$, for all $p \in Q$.
- If $\delta(q, a, X)$ contains $(r, Y_1 Y_2 \ldots Y_k)$, then R contains the rule $[qXr_k] \to a[rY_1r_1][r_1Y_2r_2] \ldots [r_{k-1}Y_kr_k]$.
  - a is an input symbol or ε
  - X and each Y_i are stack symbols
  - $r_1, r_2, \ldots r_k$ can be any list of states, so we need to add a production rule for *all possible lists* of k states.
- If $\delta(q, a, X)$ contains (r, ε), then R contains the rule $[qXr] \to a$.

Intuition Behind PDA→CFG Method

A leftmost derivation in G of a string w simulates P with w as an input.

$$S \underset{lm}{\overset{*}{\Rightarrow}} 00[q_0Xq_1][q_1Xq_1][q_1Z_0q_1] \underset{lm}{\overset{*}{\Rightarrow}} 00011$$

(This comes from the example in the following slides).

Intuition Behind PDA→CFG Method

A leftmost derivation in G of a string w simulates P with w as an input.

$$S \underset{lm}{\overset{*}{\Rightarrow}} \boxed{00}[\boxed{q_0}X\boxed{q_1}][q_1Xq_1][q_1Z_0q_1] \underset{lm}{\overset{*}{\Rightarrow}} 00011$$

The input consumed by P

P's current state

The state P will be in after X or the symbols that replace X have been erased from the stack.

Intuition Behind PDA→CFG Method

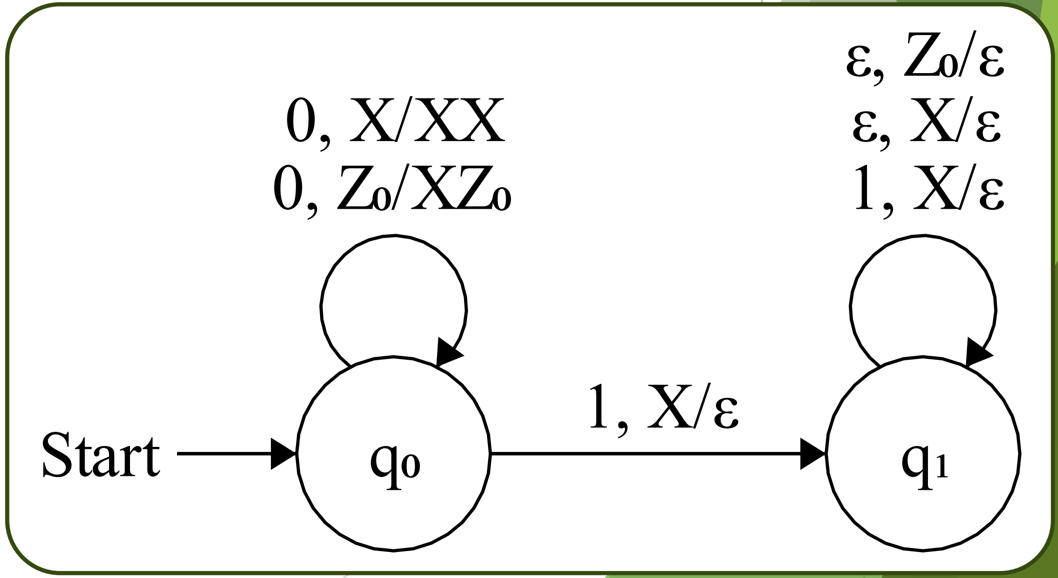
A leftmost derivation in G of a string w simulates P with w as an input.

$$S \overset{*}{\underset{lm}{\Rightarrow}} 00[q_0\boxed{X}q_1][q_1\boxed{X}q_1][q_1\boxed{Z_0}q_1] \overset{*}{\underset{lm}{\Rightarrow}} 00011$$

P's current stack:
XXZ_0

Converting PDA→CFG: Example

- We will convert the following PDA P to a CFG.
- $N(P) = \{0^i 1^j \mid i \geq j \geq 1\}$.
  - $P = (\{\mathrm{q}_0, \mathrm{q}_1\}, \{0, 1\}, \{X, Z_0\}, \delta, q_0, Z_0, \emptyset)$
  - $\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$
  - $\delta(q_0, 0, X) = \{(q_0, XX)\}$
  - $\delta(q_0, 1, X) = \{(q_1, \varepsilon)\}$
  - $\delta(q_1, 1, X) = \{(q_1, \varepsilon)\}$
  - $\delta(q_1, \varepsilon, X) = \{(q_1, \varepsilon)\}$
  - $\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$



Converting PDA→CFG: Example

The variables in the CFG created from this example PDA will be:

$$V = \{[q_0 X q_0], [q_0 X q_1], [q_1 X q_0], [q_1 X q_1], [q_0 Z_0 q_0], [q_0 Z_0 q_1], [q_1 Z_0 q_0], [q_1 Z_0 q_1]\}$$

Converting PDA→CFG: Example

- Production rules from the start state S:
  - $S \rightarrow [q_0 Z_0 q_0]$
  - $S \rightarrow [q_0 Z_0 q_1]$

$\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$
$\delta(q_0, 0, X) = \{(q_0, XX)\}$
$\delta(q_0, 1, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, 1, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$

Converting PDA→CFG: Example

- Production rules from $\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$:
  - $[q_0 Z_0 q_0] \rightarrow 0[q_0 X q_0][q_0 Z_0 q_0]$
    - $r_1 = q_0$ and $r_2 = q_0$
  - $[q_0 Z_0 q_1] \rightarrow 0[q_0 X q_0][q_0 Z_0 q_1]$
    - $r_1 = q_0$ and $r_2 = q_1$
  - $[q_0 Z_0 q_0] \rightarrow 0[q_0 X q_1][q_1 Z_0 q_0]$
    - $r_1 = q_1$ and $r_2 = q_0$
  - $[q_0 Z_0 q_1] \rightarrow 0[q_0 X q_1][q_1 Z_0 q_1]$
    - $r_1 = q_1$ and $r_2 = q_1$

$\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$
$\delta(q_0, 0, X) = \{(q_0, XX)\}$
$\delta(q_0, 1, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, 1, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$

Converting PDA→CFG: Example

- Production rules from $\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$:
  - $[q_0 Z_0 q_0] \rightarrow 0[q_0 X q_0][q_0 Z_0 q_0]$
    - $r_1 = q_0$ and $r_2 = q_0$
  - $[q_0 Z_0 q_1] \rightarrow 0[q_0 X q_0][q_0 Z_0 q_1]$
    - $r_1 = q_0$ and $r_2 = q_1$
  - $[q_0 Z_0 q_0] \rightarrow 0[q_0 X q_1][q_1 Z_0 q_0]$
    - $r_1 = q_1$ and $r_2 = q_0$
  - $[q_0 Z_0 q_1] \rightarrow 0[q_0 X q_1][q_1 Z_0 q_1]$
    - $r_1 = q_1$ and $r_2 = q_1$

$\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$
$\delta(q_0, 0, X) = \{(q_0, XX)\}$
$\delta(q_0, 1, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, 1, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$

These four rules are due to the number of possible lists of $r_1, \ldots, r_k$. In this case $k = 2$, and we have two states, so we end up with four possible lists containing two states.

Converting PDA→CFG: Example

- Production rules from $\delta(q_0, 0, X) = \{(q_0, XX)\}$:
  - $[q_0 X q_0] \rightarrow 0[q_0 X q_0][q_0 X q_0]$
    - $r_1 = q_0$ and $r_2 = q_0$
  - $[q_0 X q_1] \rightarrow 0[q_0 X q_0][q_0 X q_1]$
    - $r_1 = q_0$ and $r_2 = q_1$
  - $[q_0 X q_0] \rightarrow 0[q_0 X q_1][q_1 X q_0]$
    - $r_1 = q_1$ and $r_2 = q_0$
  - $[q_0 X q_1] \rightarrow 0[q_0 X q_1][q_1 X q_1]$
    - $r_1 = q_1$ and $r_2 = q_1$

$\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$
$\delta(q_0, 0, X) = \{(q_0, XX)\}$
$\delta(q_0, 1, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, 1, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, X) = \{(q_1, \varepsilon)\}$
$\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$

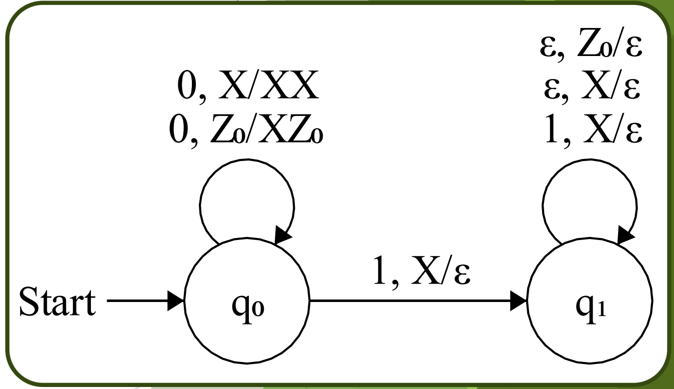
Converting PDA→CFG: Example

- Production rule from $\delta(q_0, 1, X) = \{(q_1, \varepsilon)\}$:
  - $[q_0 X q_1] \rightarrow 1$
- Production rule from $\delta(q_1, 1, X) = \{(q_1, \varepsilon)\}$:
  - $[q_1 X q_1] \rightarrow 1$
- Production rule from $\delta(q_1, \varepsilon, X) = \{(q_1, \varepsilon)\}$:
  - $[q_1 X q_1] \rightarrow \varepsilon$
- Production rule from $\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$:
  - $[q_1 Z_0 q_1] \rightarrow \varepsilon$

$$\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$$
$$\delta(q_0, 0, X) = \{(q_0, XX)\}$$
$$\delta(q_0, 1, X) = \{(q_1, \varepsilon)\}$$
$$\delta(q_1, 1, X) = \{(q_1, \varepsilon)\}$$
$$\delta(q_1, \varepsilon, X) = \{(q_1, \varepsilon)\}$$
$$\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$$

Converting PDA→CFG: Example

- We now have production rules for every variable except for $[q_1 Z_0 q_0]$ and $[q_1 X q_0]$.
  - These variables have *no production rules* — if they are ever produced we could never replace them with terminals.
  - Intuitively, this makes sense because the original PDA can't possibly transition from q_1 to q_0.
    - Remember that $[q_1 X q_0]$ corresponds to P transitioning from q_1 to q_0 while popping X off the stack.



Converting PDA→CFG: Example

- $S \rightarrow [q_0 Z_0 q_0]$
- $S \rightarrow [q_0 Z_0 q_1]$
- $[q_0 Z_0 q_0] \rightarrow 0[q_0 X q_0][q_0 Z_0 q_0]$
- $[q_0 Z_0 q_1] \rightarrow 0[q_0 X q_0][q_0 Z_0 q_1]$
- $[q_0 Z_0 q_0] \rightarrow 0[q_0 X q_1][q_1 Z_0 q_0]$
- $[q_0 Z_0 q_1] \rightarrow 0[q_0 X q_1][q_1 Z_0 q_1]$
- $[q_0 X q_0] \rightarrow 0[q_0 X q_0][q_0 X q_0]$
- $[q_0 X q_1] \rightarrow 0[q_0 X q_0][q_0 X q_1]$
- $[q_0 X q_0] \rightarrow 0[q_0 X q_1][q_1 X q_0]$
- $[q_0 X q_1] \rightarrow 0[q_0 X q_1][q_1 X q_1]$
- $[q_0 X q_1] \rightarrow 1$
- $[q_1 X q_1] \rightarrow 1$
- $[q_1 X q_1] \rightarrow \varepsilon$
- $[q_1 Z_0 q_1] \rightarrow \varepsilon$

This technique can result in many *useless* productions, which are either unreachable or will never lead to a terminal string.

Converting PDA→CFG: Example

- $S \to [q_0 Z_0 q_0]$
- $S \to [q_0 Z_0 q_1]$
- $[q_0 Z_0 q_0] \to 0[q_0 X q_0][q_0 Z_0 q_0]$
- $[q_0 Z_0 q_1] \to 0[q_0 X q_0][q_0 Z_0 q_1]$
- $\cancel{[q_0 Z_0 q_0] \to 0[q_0 X q_1][q_1 Z_0 q_0]}$
- $[q_0 Z_0 q_1] \to 0[q_0 X q_1][q_1 Z_0 q_1]$
- $[q_0 X q_0] \to 0[q_0 X q_0][q_0 X q_0]$
- $[q_0 X q_1] \to 0[q_0 X q_0][q_0 X q_1]$
- $\cancel{[q_0 X q_0] \to 0[q_0 X q_1][q_1 X q_0]}$
- $[q_0 X q_1] \to 0[q_0 X q_1][q_1 X q_1]$
- $[q_0 X q_1] \to 1$
- $[q_1 X q_1] \to 1$
- $[q_1 X q_1] \to \varepsilon$
- $[q_1 Z_0 q_1] \to \varepsilon$

We know that $[q_1 Z_0 q_0]$ and $[q_1 X q_0]$ can never lead to terminals due to having no productions, so any production of these variables is useless.

Converting PDA→CFG: Example

- $S \rightarrow [q_0 Z_0 q_0]$
- $S \rightarrow [q_0 Z_0 q_1]$
- $\cancel{[q_0 Z_0 q_0] \rightarrow 0[q_0 X q_0][q_0 Z_0 q_0]}$
- $[q_0 Z_0 q_1] \rightarrow 0[q_0 X q_0][q_0 Z_0 q_1]$
- $[q_0 Z_0 q_0] \rightarrow 0[q_0 X q_1][q_1 Z_0 q_0]$
- $[q_0 Z_0 q_1] \rightarrow 0[q_0 X q_1][q_1 Z_0 q_1]$
- $[q_0 X q_0] \rightarrow 0[q_0 X q_0][q_0 X q_0]$
- $[q_0 X q_1] \rightarrow 0[q_0 X q_0][q_0 X q_1]$
- $[q_0 X q_0] \rightarrow 0[q_0 X q_1][q_1 X q_0]$
- $[q_0 X q_1] \rightarrow 0[q_0 X q_1][q_1 X q_1]$
- $[q_0 X q_1] \rightarrow 1$
- $[q_1 X q_1] \rightarrow 1$
- $[q_1 X q_1] \rightarrow \varepsilon$
- $[q_1 Z_0 q_1] \rightarrow \varepsilon$

$[q_0 Z_0 q_0]$ has only one other production, which also produces itself, meaning that $[q_0 Z_0 q_0]$ can never be entirely replaced with terminals.

Converting PDA→CFG: Example

- $S \rightarrow [q_0 Z_0 q_0]$
- $S \rightarrow [q_0 Z_0 q_1]$
- $[q_0 Z_0 q_0] \rightarrow 0[q_0 X q_0][q_0 Z_0 q_0]$
- $[q_0 Z_0 q_1] \rightarrow 0[q_0 X q_0][q_0 Z_0 q_1]$
- $[q_0 Z_0 q_0] \rightarrow 0[q_0 X q_1][q_1 Z_0 q_0]$
- $[q_0 Z_0 q_1] \rightarrow 0[q_0 X q_1][q_1 Z_0 q_1]$
- $\cancel{[q_0 X q_0] \rightarrow 0[q_0 X q_0][q_0 X q_0]}$
- $[q_0 X q_1] \rightarrow 0[q_0 X q_0][q_0 X q_1]$
- $[q_0 X q_0] \rightarrow 0[q_0 X q_1][q_1 X q_0]$
- $[q_0 X q_1] \rightarrow 0[q_0 X q_1][q_1 X q_1]$
- $[q_0 X q_1] \rightarrow 1$
- $[q_1 X q_1] \rightarrow 1$
- $[q_1 X q_1] \rightarrow \varepsilon$
- $[q_1 Z_0 q_1] \rightarrow \varepsilon$

$[q_0 X q_0]$ has only one remaining production, which only produces itself. So, $[q_0 X q_0]$ can never be replaced with terminals and is also useless.

Converting PDA→CFG: Example

After removing all of the useless productions, we end up with the following production rules in the grammar:

- $S \rightarrow [q_0 Z_0 q_1]$
- $[q_0 Z_0 q_1] \rightarrow 0[q_0 X q_1][q_1 Z_0 q_1]$
- $[q_0 X q_1] \rightarrow 0[q_0 X q_1][q_1 X q_1]$
- $[q_0 X q_1] \rightarrow 1$
- $[q_1 X q_1] \rightarrow 1$
- $[q_1 X q_1] \rightarrow \varepsilon$
- $[q_1 Z_0 q_1] \rightarrow \varepsilon$

Converting PDA→CFG: Example

Remember that the language of the PDA and CFG is $\{0^i 1^j \mid i \geq j \geq 1\}$.
Example string in the language: 001.

Leftmost derivation in the constructed grammar:

$$
\begin{aligned}
S &\Rightarrow [q_0 Z_0 q_1] \\
&\Rightarrow 0[q_0 X q_1][q_1 Z_0 q_1] \\
&\Rightarrow 00[q_0 X q_1][q_0 X q_1][q_1 Z_0 q_1] \\
&\Rightarrow 001[q_0 X q_1][q_1 Z_0 q_1] \\
&\Rightarrow 001[q_1 Z_0 q_1] \\
&\Rightarrow 001
\end{aligned}
$$

Sequence of moves in the original PDA:

$$
\begin{aligned}
&(q_0, 001, Z_0) \\
&\vdash (q_0, 01, XZ_0) \\
&\vdash (q_0, 1, XXZ_0) \\
&\vdash (q_1, \varepsilon, XZ_0) \\
&\vdash (q_1, \varepsilon, Z_0) \\
&\vdash (q_1, \varepsilon, \varepsilon)
\end{aligned}
$$

$$
\begin{aligned}
S &\rightarrow [q_0 Z_0 q_1] \\
[q_0 Z_0 q_1] &\rightarrow 0[q_0 X q_1][q_1 Z_0 q_1] \\
[q_0 X q_1] &\rightarrow 0[q_0 X q_1][q_1 X q_1] \\
[q_0 X q_1] &\rightarrow 1 \\
[q_1 X q_1] &\rightarrow 1 \\
[q_1 X q_1] &\rightarrow \varepsilon \\
[q_1 Z_0 q_1] &\rightarrow \varepsilon
\end{aligned}
$$

$$
\begin{aligned}
\delta(q_0, 0, Z_0) &= \{(q_0, XZ_0)\} \\
\delta(q_0, 0, X) &= \{(q_0, XX)\} \\
\delta(q_0, 1, X) &= \{(q_1, \varepsilon)\} \\
\delta(q_1, 1, X) &= \{(q_1, \varepsilon)\} \\
\delta(q_1, \varepsilon, X) &= \{(q_1, \varepsilon)\} \\
\delta(q_1, \varepsilon, Z_0) &= \{(q_1, \varepsilon)\}
\end{aligned}
$$

Proving PDA→CFG Correctness

We will prove that our construction is correct by first proving that the variables mean what we say they do.

Claim: $[qXp] \overset{*}{\underset{G}{\Rightarrow}} w$ if and only if $(q, w, X) \overset{*}{\underset{P}{\vdash}} (p, \varepsilon, \varepsilon)$.

Remember that this is the condition for determining w's membership in the language of a PDA by empty stack.

- "We can derive w from $[qXp]$ if and only if we can move from state q to p while consuming w and popping X from the stack."
- Special case: $[q_0 Z_0 p] \overset{*}{\Rightarrow} w$ if and only if $\boxed{(q_0, w, Z_0) \overset{*}{\vdash} (p, \varepsilon, \varepsilon)}$
- Because $S \Rightarrow [q_0 Z_0 p]$ (by our construction), this implies $w \in L(G)$ if and only if $w \in N(P)$.

Proving PDA→CFG Correctness

Claim (again): $[qXp] \overset{*}{\underset{G}{\Rightarrow}} w$ if and only if $(q, w, X) \overset{*}{\underset{P}{\vdash}} (p, \varepsilon, \varepsilon)$.

"If": We show by induction on the number of PDA moves that $(q, w, X) \overset{i}{\vdash} (p, \varepsilon, \varepsilon)$ implies $[qXp] \overset{*}{\Rightarrow} w$.

Base case: $i = 1$. This means that $w \in \Sigma \cup \{\varepsilon\}$.

The only way the PDA can accept a string in one step is if the string is a single symbol or ε.

$\delta(q, w, X)$ must contain (p, ε). Therefore, $[qXp] \rightarrow w$ is a production due to our construction of the CFG.

Proving PDA→CFG Correctness

$[qXp] \overset{*}{\underset{G}{\Rightarrow}} w$ if and only if $(q, w, X) \overset{*}{\underset{P}{\vdash}} (p, \varepsilon, \varepsilon)$

"If", Inductive step: i steps, where $i > 1$.

- $(q, ax, X) \vdash (r_0, x, Y_1 \dots Y_k) \overset{*}{\vdash} (p, \varepsilon, \varepsilon)$
  - Here, $w = ax$, where $a \in \Sigma \cup \{\varepsilon\}$
  - This means that $(r_0, Y_1 \dots Y_k) \in \delta(q, a, X)$, so the CFG will have a production $[qXr_k] \rightarrow a[r_0Y_1r_1] \dots [r_{k-1}Y_kr_k]$, where
    - $r_k = p$, and
    - $r_1, r_2 \dots r_{k-1}$ are any states in Q.

Proving PDA→CFG Correctness

"If", Inductive step, continued

- $(q, ax, X) \vdash (r_0, x, Y_1 \ldots Y_k) \overset{*}{\vdash} (p, \varepsilon, \varepsilon)$
  - We have a production $[qXr_k] \rightarrow a[r_0Y_1r_1] \ldots [r_{k-1}Y_kr_k]$, where
    - $r_k = p$, and
    - $r_1, r_2 \ldots r_{k-1}$ are any states in Q.
- For each Y_i in $Y_1 \ldots Y_k$:
  - Let r_i be the state of the PDA when Y_i is popped off the stack.
  - Let w_i be the input consumed when popping Y_i off the stack.
    - In this example, $x = w_1 \ldots w_k$
  - Putting the above points another way, $(r_{i-1}, w_i, Y_i) \overset{*}{\vdash} (r_i, \varepsilon, \varepsilon)$.
- Any set of moves going from r_{i-1} to r_i will take fewer than n moves, so the inductive hypothesis tells us that $[r_{i-1}Y_ir_i] \overset{*}{\Rightarrow} w_i$.
- Therefore, $a[r_0Y_1r_1] \ldots [r_{k-1}Y_kr_k] \overset{*}{\Rightarrow} aw_1 \ldots w_k = ax = w$.

Proving PDA→CFG Correctness

Claim (again): $[qXp] \overset{*}{\underset{G}{\Rightarrow}} w$ if and only if $(q, w, X) \overset{*}{\underset{P}{\vdash}} (p, \varepsilon, \varepsilon)$.

"Only if": We show by induction on the number of derivation steps that $[qXp] \overset{i}{\Rightarrow} w$ implies $(q, w, X) \overset{*}{\vdash} (p, \varepsilon, \varepsilon)$.

Base case: $i = 1$. $[qXp] \rightarrow w$, where $w \in \Sigma \cup \{\varepsilon\}$.

From the construction of the CFG, we know that $\delta(q, w, X)$ must contain (p, ε).

Proving PDA→CFG Correctness

"Only if" claim: $[qX\boxed{p}] \overset{i}{\Rightarrow} w$ implies $(q, w, X) \overset{*}{\vdash} (p, \varepsilon, \varepsilon)$

$r_k = p$

Inductive step: $i > 1$.

$[qX\boxed{r_k}] \Rightarrow a[r_0Y_1r_1] \dots [r_{k-1}Y_k\boxed{r_k}] \overset{i-1}{\Longrightarrow} w.$

Let $w = aw_1w_2 \dots w_k$, where $[r_{i-1}Y_ir_i] \overset{*}{\Rightarrow} w_j$ and $1 \le i \le k$.

By the inductive hypothesis, $(r_{i-1}, w_i, Y_i) \overset{*}{\vdash} (r_i, \varepsilon, \varepsilon)$, and $1 \le i \le k$.

Therefore, $(r_{i-1}, w_i, Y_iY_{i+1} \dots Y_k) \overset{*}{\vdash} (r_i, \varepsilon, Y_{i+1} \dots Y_k)$

From the first step of the derivation,

$(q, w, X) \vdash (r_0, w_1 \dots w_k, Y_1 \dots Y_k)$. So, $(q, w, X) \overset{*}{\vdash} (p, \varepsilon, \varepsilon)$.

Deterministic PDAs

A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is **deterministic** if and only if:

1. $\delta(q, a, X)$ has at most one member for any $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $X \in \Gamma$.
2. If $\delta(q, a, X)$ is nonempty for some $a \in \Sigma$, then $\delta(q, \varepsilon, X)$ must be empty.

- Deterministic PDAs (DPDAs) are useful in parsing.
- Unlike DFAs, which are just as powerful as NFAs, DPDAs are *less powerful than nondeterministic PDAs*.

There is at most one possible transition for every combination of state, input symbol, and stack symbol.

Regular Languages and DPDAs

Theorem 6.17: If L is a regular language, then $L = L(P)$ for some DPDA P.

Proof sketch: A DFA is a special case of a DPDA in which the stack is not used, i.e., all moves “replace” Z_0 by Z_0, where Z_0 is the start symbol.

DPDAs Accepting by Empty Stack

It is impossible for a DPDA that accepts by empty stack to accept both a string x and a string xy, where $y \neq \varepsilon$.

A language L has the **prefix property** if there are no two different strings x and y in L such that x is a prefix of y.

Theorem 6.19: A language L is $N(P)$ for some DPDA P if and only if L has the prefix property and L is $L(P')$ for some DPDA P'.

The DPDA would need to empty its stack while reading x, so it would fail when trying to process more input on an empty stack.

Relationships Between DPDAs and CFLs

“Proper” inclusion: The superset must have some elements not present in the subset.

Theorem: The languages accepted by DPDAs by final state properly include the regular languages, but are properly included in the context-free languages.

Proof sketch: Regular language inclusion (not necessarily *proper* inclusion) is implied by Theorem 6.17 (a regular language must be accepted by some DPDA).

DPDA Inclusion Proof Sketch, Continued

The fact that regular languages are properly included in the languages of DPDAs is because the language $\{w2w^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ is accepted by a DPDA by final state, but is not a regular language.

- The "2" tells the DPDA when to start looking for w^R, enabling it to be deterministic.

DPDA Inclusion Proof Sketch, Continued

The fact that the language of DPDAs is properly included by context-free languages is because the language $\{ww^R \mid w \in (\mathbf{0} + \mathbf{1})^*\}$ is accepted by a PDA but not by any DPDA.

- We will not prove in class that it's impossible for a DPDA to accept this, but the intuition is that the DPDA can't "know" when w ends and w^R begins.

DPDAs and Ambiguous Grammars

Theorem 6.20: If $L = N(P)$ for some DPDA P, then L has an unambiguous CFG.

Proof sketch:

A CFG is ambiguous if and only if multiple leftmost derivations are possible.

If we apply the previous PDA→CFG construction to a DPDA, we will end up with production rules where multiple derivations of the same string aren't possible.

DPDAs and Ambiguous Grammars

Theorem 6.21: If $L = L(P)$ for some DPDA P, then L has an unambiguous CFG.

Proof sketch:

Define $L\$ = \{x\$ \mid x \in L\}$, and \$ is a new symbol not in L's alphabet. This means that $L\$$ has the *prefix property*

If a string w contained a prefix in $L\$$, w and the prefix would both need to end with \$, which is not possible under our definition of $L\$$.

We can modify P to accept $L\$$ by final state. Therefore, by Theorem 6.19, $L\$ = N(P')$ for some DPDA P'.

DPDAs and Ambiguous Grammars

Theorem 6.21: If $L = L(P)$ for some DPDA P, then L has an unambiguous CFG.
Proof sketch, continued:

We know by Theorem 6.20 that, $L\$$ has an unambiguous CFG G, because $L\$ = N(P')$ for some DPDA P'.

However, in G, \$ is a terminal symbol. To fix this, define a grammar G' that is the same as G except \$ is a variable and the production $\$ \rightarrow \varepsilon$ is included.

Now, G' is unambiguous, and $L = L(G')$.