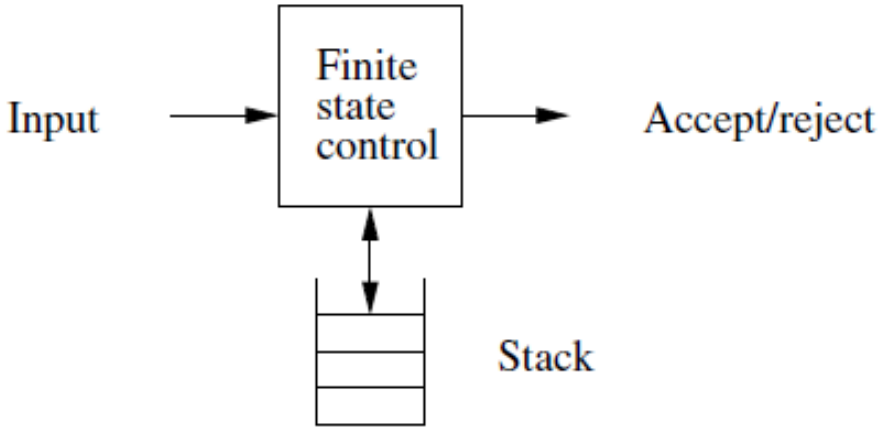
Pushdown Automata

Pushdown Automata

- **Pushdown automatons** accept exactly **context free languages**.
- A **pushdown automaton (PDA)** is essentially **an NFA with a stack**.
- **On a transition, a PDA:**
  1. **Consumes an input symbol (or ε-transition).**
  2. **Goes to a new state (or stays in the old).**
  3. **Replaces the top of the stack item by any string (does nothing, pops the stack, or pushes a string onto the stack)**



Pushdown Automata – Formal Definition

A pushdown Automata (PDA) is a seven-tuple:

$$P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F),$$

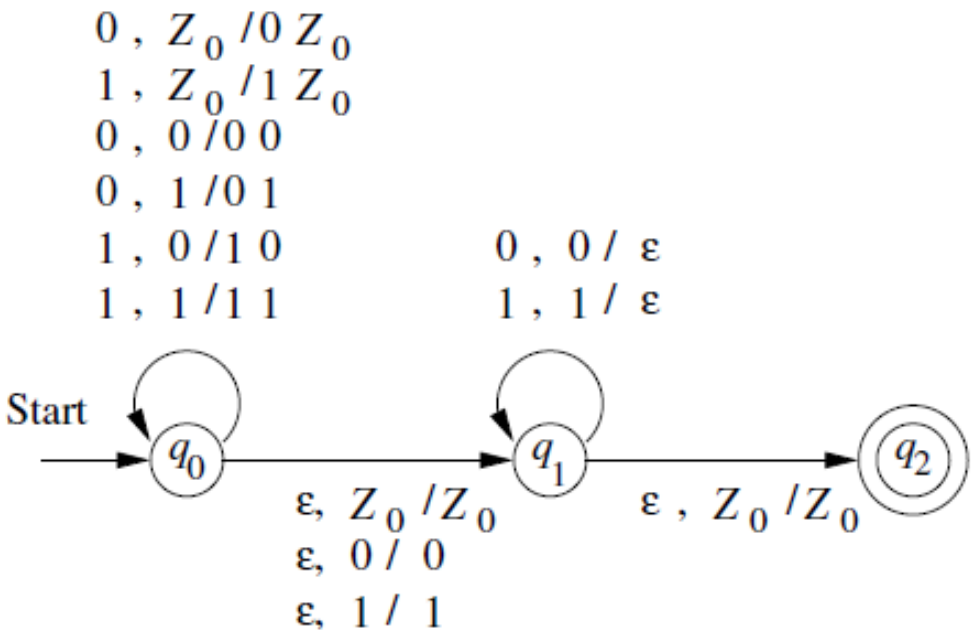
where

- Q is a finite set of states,
- Σ is a finite input alphabet,
- Γ is finite stack alphabet,
- δ: Q x $\Sigma \cup \{\varepsilon\}$ x $\Gamma \rightarrow 2^{Q x \Gamma^*}$ is the transition function,
- q_0 is a start state,
- Z_0 is the start symbol for the stack, and
- F is the set of accepting states.

Pushdown Automata – Example

- Consider a CFL $L_{ww^r} = \{ ww^r : w \in \{0,1\}^* \}$
- A pushdown automaton P for the language L_{ww^r} is as follows:

$$P = (\{q_0,q_1,q_2\}, \{0,1\}, \{0,1,Z_0\}, \delta, q_0, Z_0, \{q_2\})$$



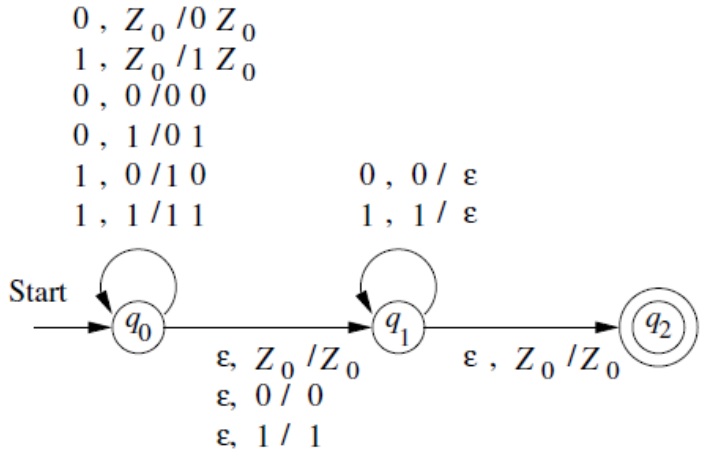
Pushdown Automata – Example

Table Representation of Transition Function

- Consider a CFL $L_{ww^r} = \{ ww^r : w \in \{0,1\}^* \}$
- A pushdown automaton P for the language $L_{ww^r} = \{ ww^r : w \in \{0,1\}^* \}$ is as actually a seven tuple:

$$P = (\{q_0,q_1,q_2\}, \{0,1\}, \{0,1,Z_0\}, \delta, q_0, Z_0, \{q_2\})$$

where its transition function can be also shown by a table.

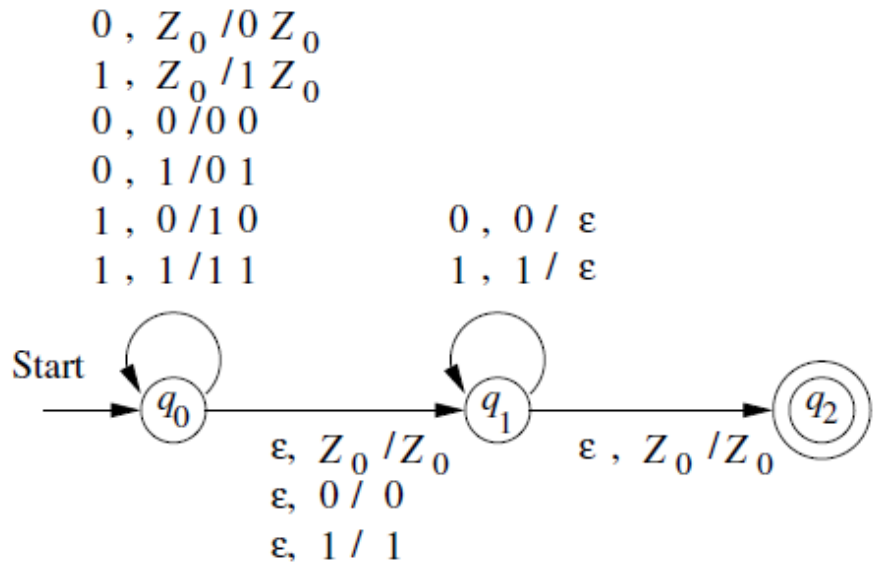


	$0, Z_0$	$1, Z_0$	$0,0$	$0,1$	$1,0$	$1,1$	ϵ, Z_0	$\epsilon, 0$	$\epsilon, 1$
$\rightarrow q_0$	$q_0, 0Z_0$	$q_0, 1Z_0$	$q_0, 00$	$q_0, 01$	$q_0, 10$	$q_0, 11$	q_1, Z_0	$q_1, 0$	$q_1, 1$
q_1			q_1, ϵ			q_1, ϵ	q_2, Z_0		
$\star q_2$									

Pushdown Automata – Example

A path for the input 0110

State	Input	Stack	Transition
q_0	0110	Z_0	0, Z_0 /0Z_0
q_0	110	0Z_0	1, 0 /10
q_0	ε10	10Z_0	ε, 1 /1
q_1	10	10Z_0	1, 1 / ε
q_1	0	0Z_0	0, 0 / ε
q_1	ε	Z_0	ε, Z_0 /Z_0
q_2	ε	Z_0	



Instantaneous Descriptions (ID) of a PDA

- A PDA goes from configuration to configuration when consuming input.

- **The configuration of a PDA is represented by a triple (q,w,γ) where**
  1. **q is a the state,**
  2. **w is the remaining input, and**
  3. **γ is the stack contents**

- **A configuration triple is called an instantaneous description, or ID, of the pushdown automaton.**

PDA Move - turnstile $\vdash$ Notation

- We need a notation that describes changes in the state, the input, and stack.
- Let $P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ be a PDA. We define a **move** $\vdash$ as follows.

- **If $(p,\alpha)\in\delta(q,a,X)$ where $q\in Q$, $a\in\Sigma\cup\{\varepsilon\}$, $X\in\Gamma$ and $\alpha\in\Gamma^*$. Then for all strings $w\in\Sigma^*$ and $\beta\in\Gamma^*$, we have a move (transition):**

  $(q,aw,X\beta) \vdash (p,w,\alpha\beta)$)

- This move reflects the idea that, by consuming **a** (which may be ε) from the input and replacing X on top of the stack by α we can go from state **q** to state **p**.
  - Note that what remains on the input, **w**, and what is below the top of the stack, β ,do not influence the action of the PDA, they are merely carried along.

PDA Moves - turnstile $\vdash^*$ Notation

Computation

- To represent a sequence of zero or more moves of the PDA, we use $\vdash^*$.
- **A sequence of moves is also called as computation.**

BASIS:

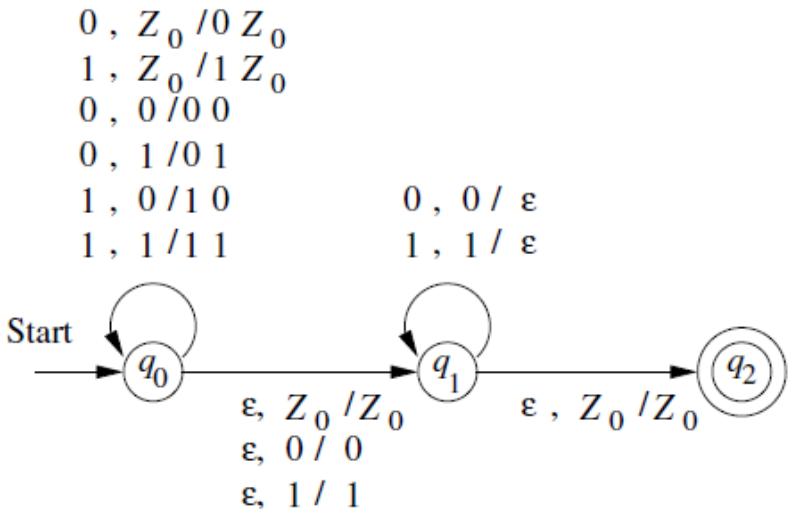
$I \vdash^* I$ for any ID I.

INDUCTION:

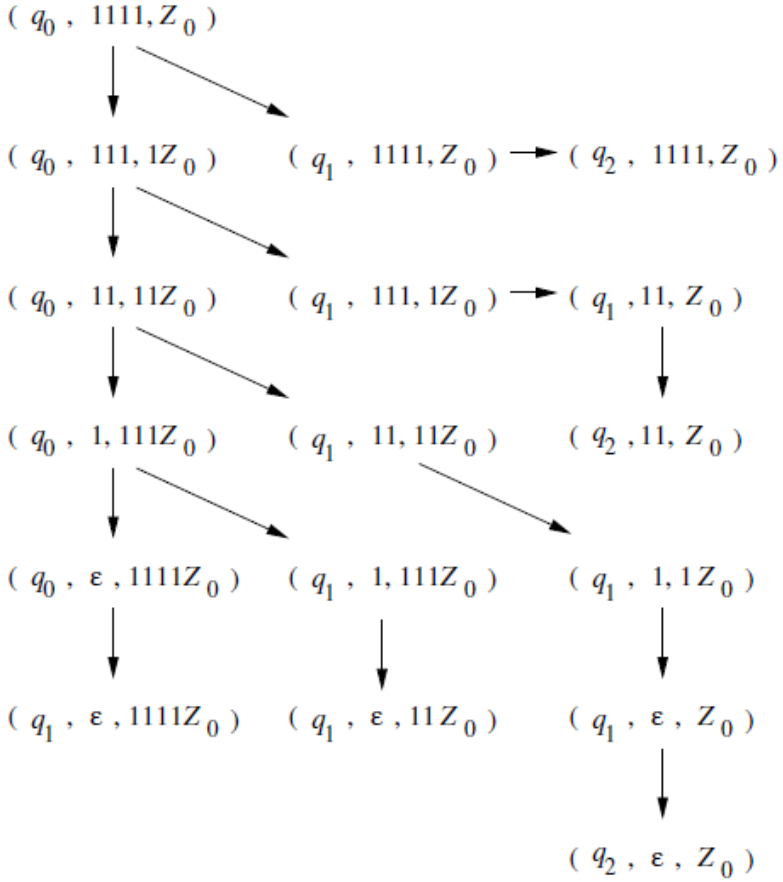
If $I \vdash^* J$ if there exists ID K such that $I \vdash K$ and $K \vdash^* J$.

PDA Move - Example

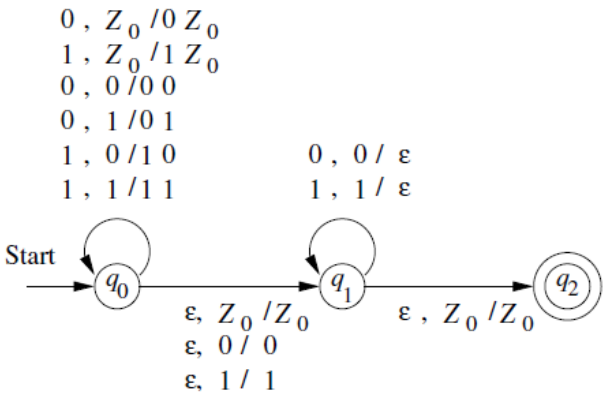
- On input 1111 the PDA has the following **computation sequences**:

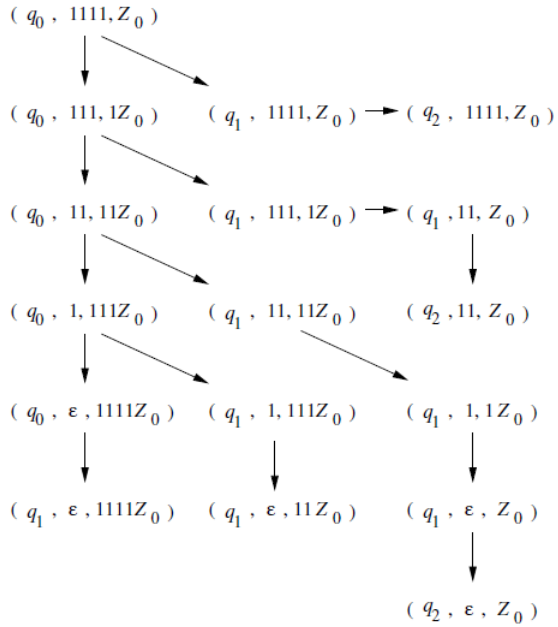


Arrows represent move ⊢ relation.



PDA Move - Example





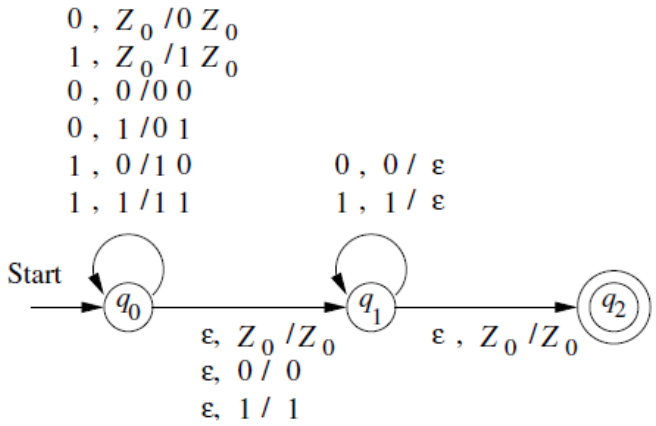
- A sequence of moves (a **computation**):

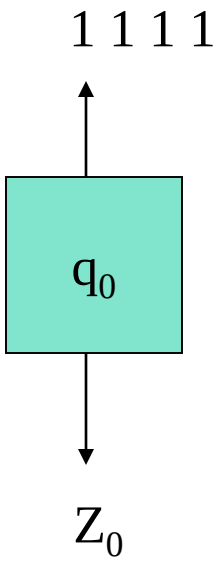
  $(q_0,1111,Z_0) \vdash (q_0,111,1Z_0) \vdash (q_0,11,11Z_0) \vdash (q_1,11,11Z_0) \vdash (q_1,1,1Z_0) \vdash (q_1,\varepsilon,Z_0) \vdash (q_2,\varepsilon,Z_0)$

- **Thus,** $(q_0,1111,Z_0) \vdash^* (q_2,\varepsilon,Z_0)$

Actions of Example PDA

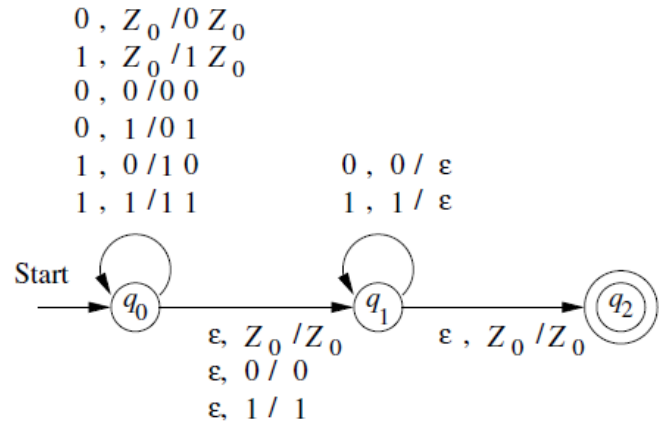
- Initial configuration (initial ID): **(q_0,1111,Z_0)**

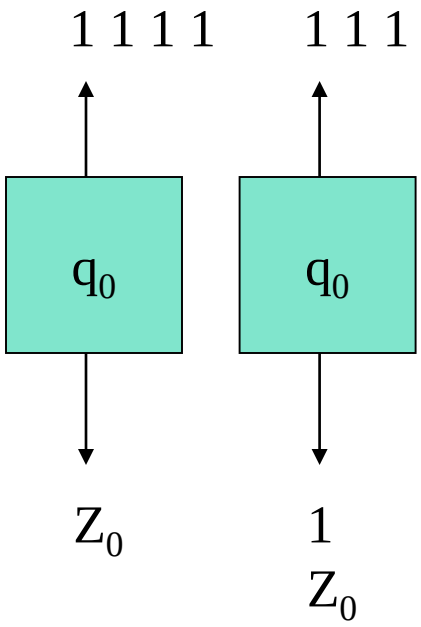




Actions of Example PDA

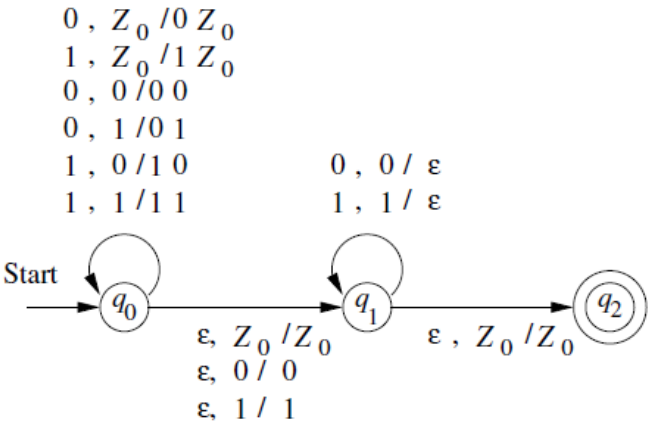
$(q_0,1111,Z_0) \vdash (q_0,111,1Z_0)$

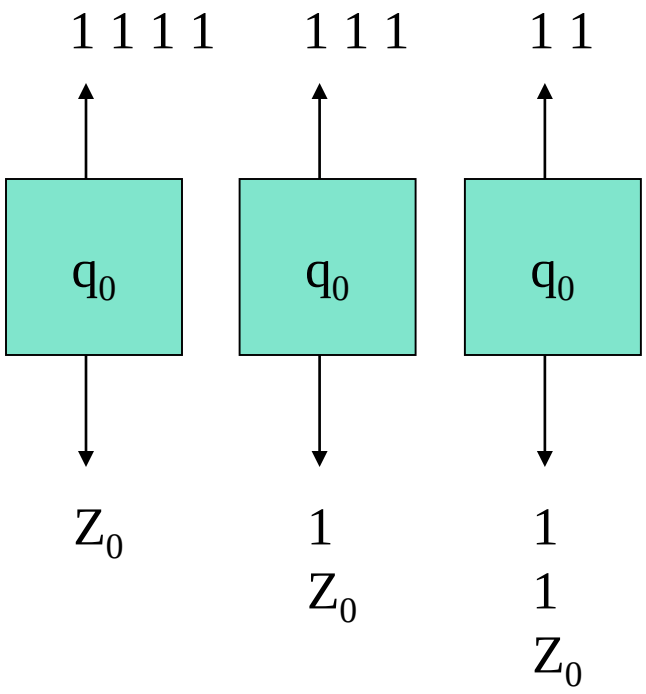




Actions of Example PDA

$(q_0,1111,Z_0) \vdash (q_0,111,1Z_0) \vdash (q_0,11,11Z_0)$

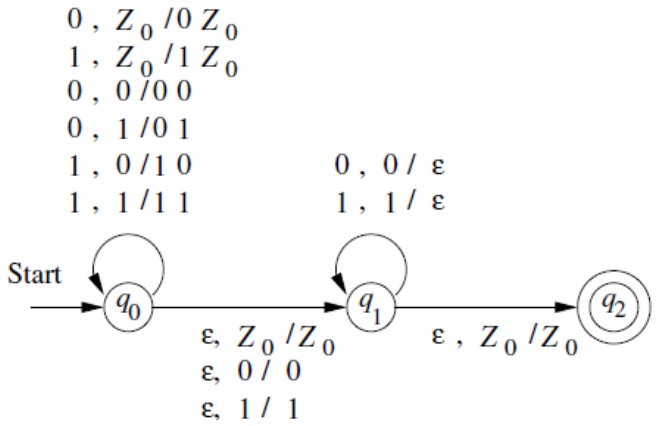


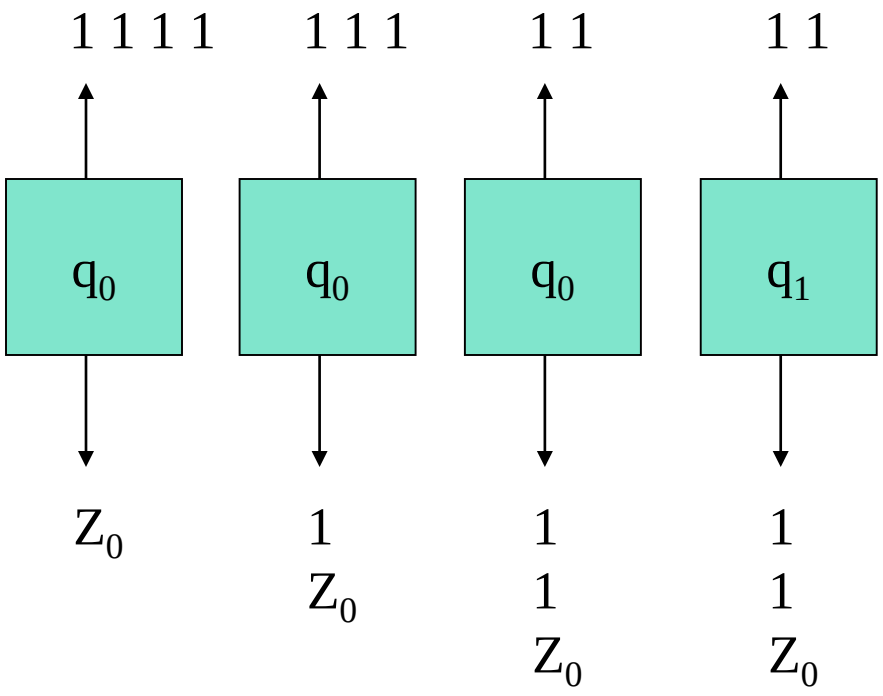


Actions of Example PDA

$(q_0,1111,Z_0) \vdash (q_0,111,1Z_0) \vdash (q_0,11,11Z_0)$

$\vdash (q_1,11,11Z_0)$

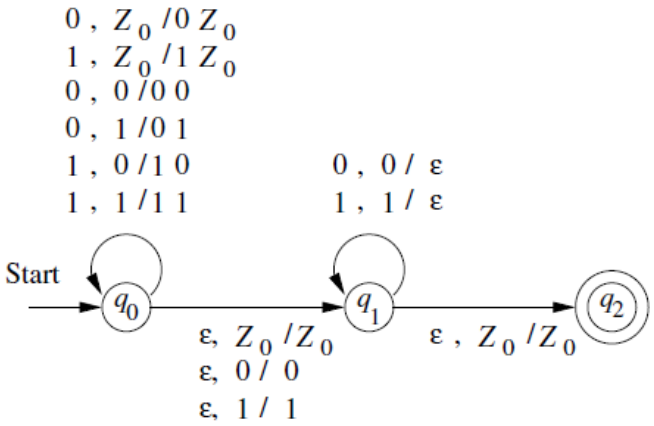


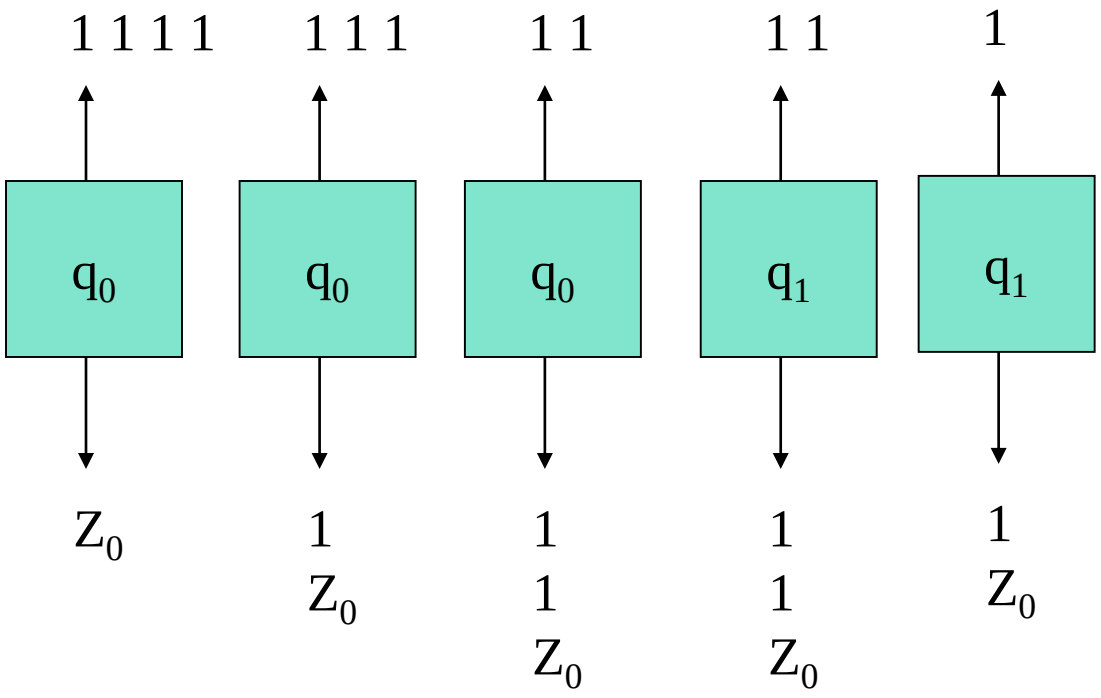


Actions of Example PDA

$(q_0,1111,Z_0) \vdash (q_0,111,1Z_0) \vdash (q_0,11,11Z_0)$

$\vdash (q_1,11,11Z_0) \vdash (q_1,1,1Z_0)$

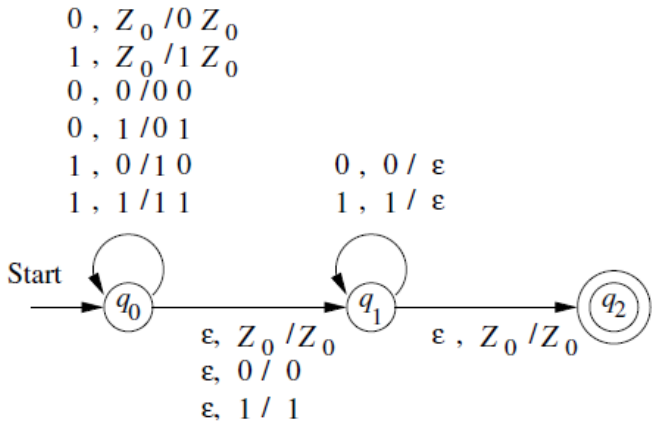


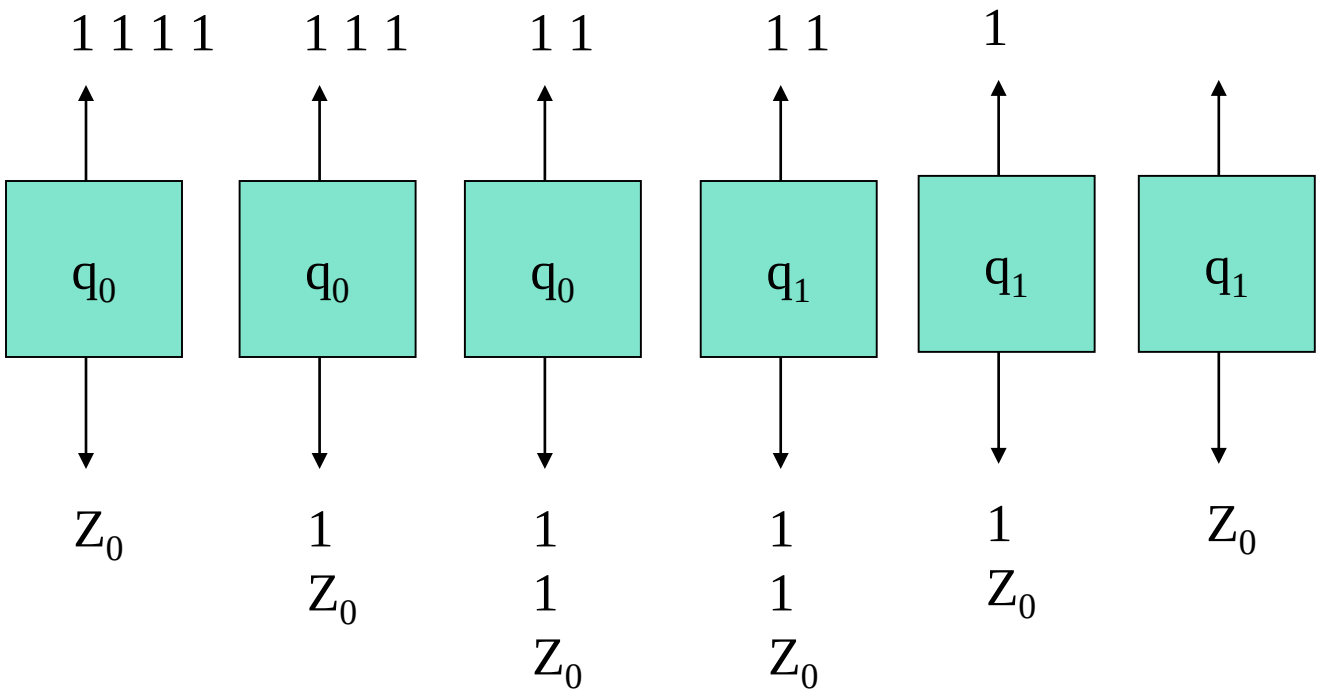


Actions of Example PDA

$(q_0,1111,Z_0) \vdash (q_0,111,1Z_0) \vdash (q_0,11,11Z_0)$

$\vdash (q_1,11,11Z_0) \vdash (q_1,1,1Z_0) \vdash (q_1,\varepsilon,Z_0)$

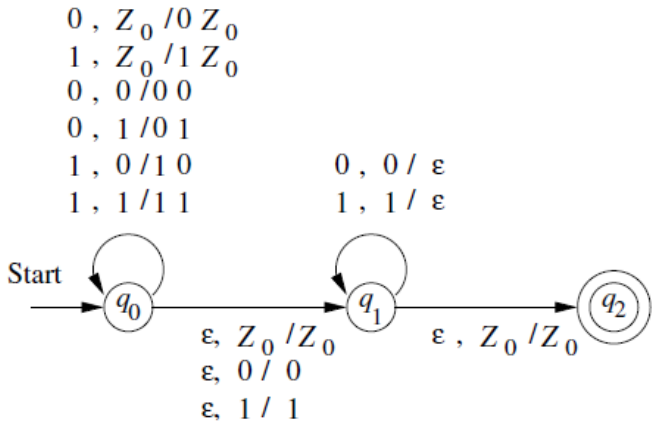


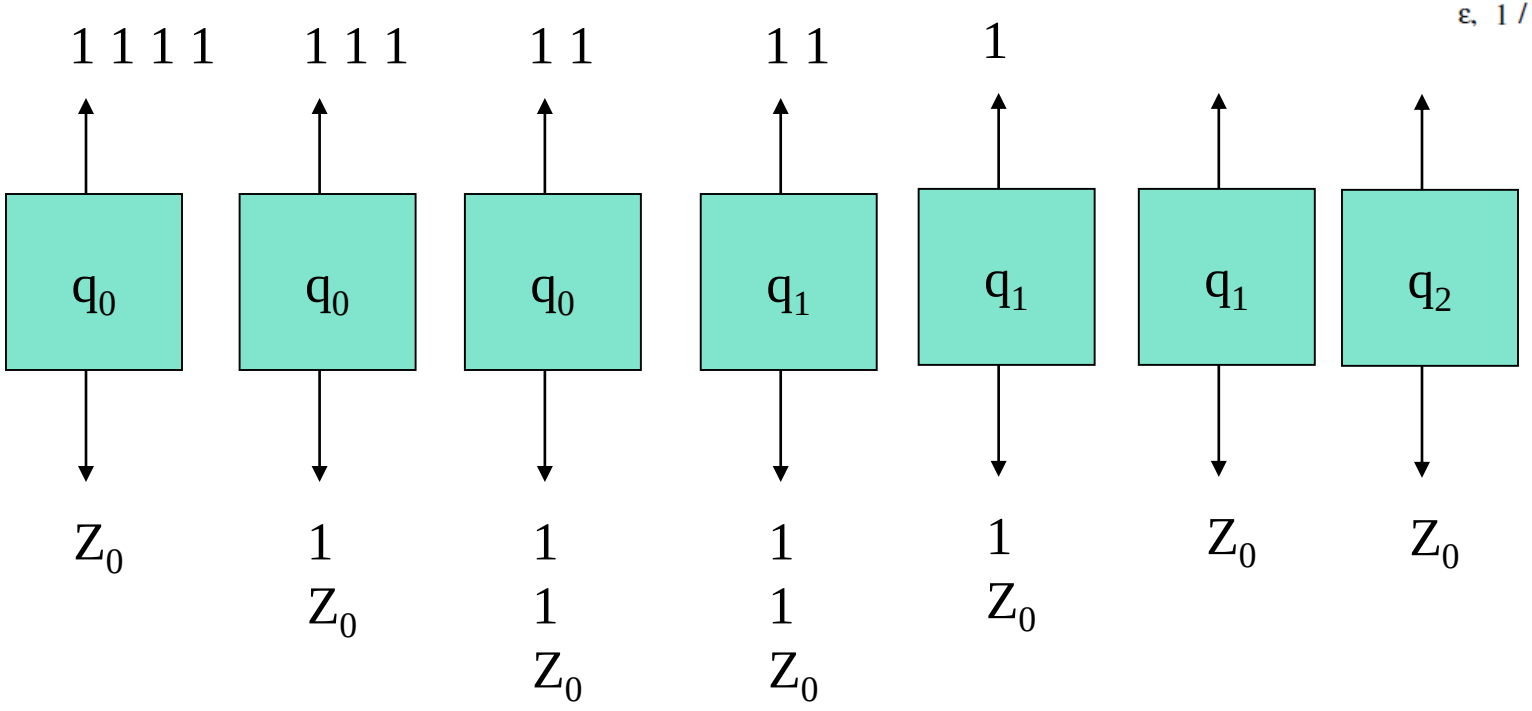


Actions of Example PDA

$(q_0,1111,Z_0) \vdash (q_0,111,1Z_0) \vdash (q_0,11,11Z_0)$

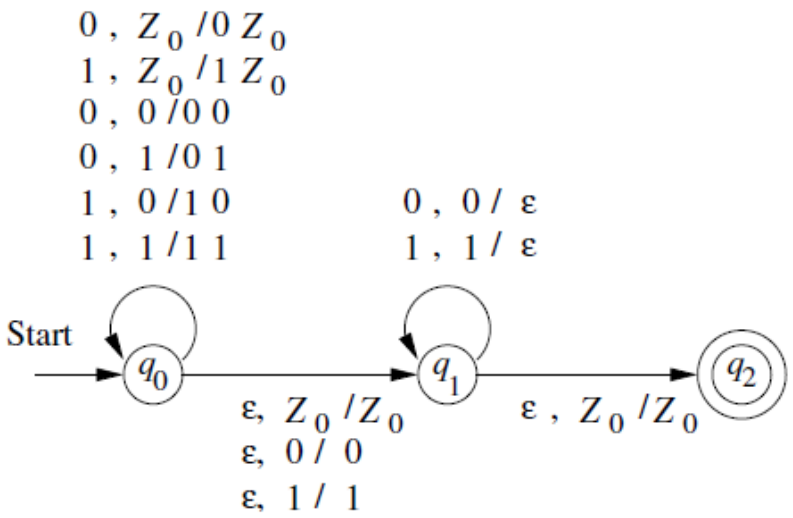
$\vdash (q_1,11,11Z_0) \vdash (q_1,1,1Z_0) \vdash (q_1,\varepsilon,Z_0) \vdash (q_2,\varepsilon,Z_0)$

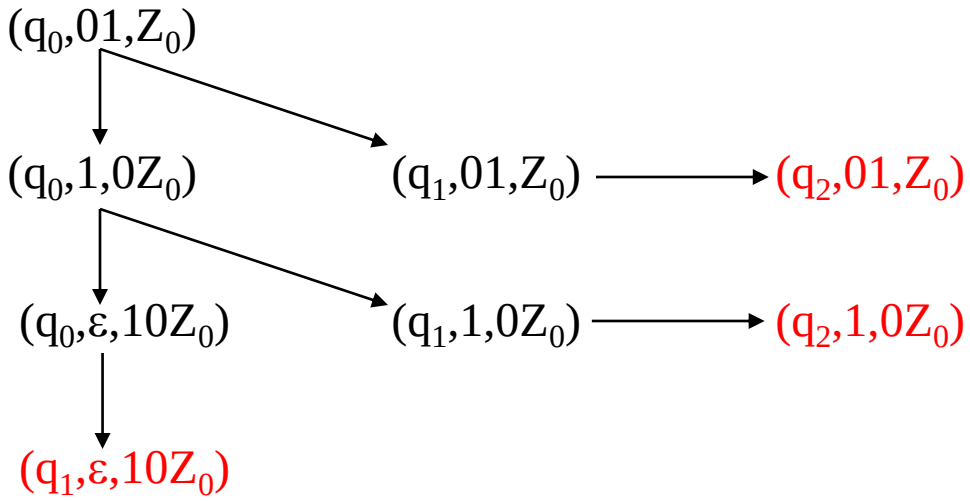




PDA Move - Example

- On input 01 the PDA has the following **computation sequences**:





- All computations end with an ID whose state is NOT a final state or the input string is NOT consumed. Thus, 01 is NOT accepted by this PDA.
  - $(q_1,\varepsilon,10Z_0)$ q_1 is NOT a final state, and q_1 does not have any move.
  - $(q_2,01,Z_0)$ the input (01) is NOT consumed, and q_2 does not have any move.
  - $(q_2,1,0Z_0)$ the input (1) is NOT consumed, and q_2 does not have any move.

Languages of PDA

The Languages of a PDA

- In our example**, we have assumed that a PDA accepts its input by consuming it and entering an accepting state.**
- This approach is called as "**Acceptance By Final State**".

- There is a second approach known as "**Accepted By Empty Stack**".
  - **The set of strings that cause the PDA to empty its stack, starting from the initial ID.**

- **These two methods are equivalent**:
  - **A language L has a PDA A that accepts it by final state if and only if L has a PDA B that accepts it by empty stack.**

The Languages of a PDA

Acceptance By Final State

Let $P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ be a PDA. Then L(P), the language of P which accepts by final state, is:

$$L(P) = \{ w : (q_0,w,Z_0) \vdash^* (q,\varepsilon,\alpha) \}$$

for some state q in F and any stack string α.

- Starting in the initial ID with w waiting on the input, P consumes w from the input and enters an accepting state.
  - The contents of the stack at that time is irrelevant.

The Languages of a PDA

Acceptance By Empty Stack

Let $P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0)$ be a PDA. Then L(P), the language of P which accepts by empty stack, is:

$$L(P) = \{ w : (q_0,w,Z_0) \vdash^* (q,\varepsilon,\varepsilon) \}$$

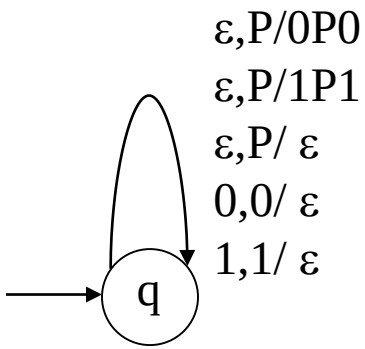
for any state q.

- Since *final states are irrelevant for PDAs that accept by empty stack*, we do not define final states for those PDAs.
  - That is, a PDA that accepts by empty stack is 6 tuple (not 7 tuple).

- L(P) is the set of inputs w that P can consume and at the same time empty its stack.
  - Final states are irrelevant.

A PDA Accepts By Empty Stack

- A PDA $P_N = (\{q\},\{0,1\},\{P,0,1\},\delta,q,P)$ which **accepts by empty stack**.
- The language of this PDA is $\{ ww^r : w \in \{0,1\}^* \}$.



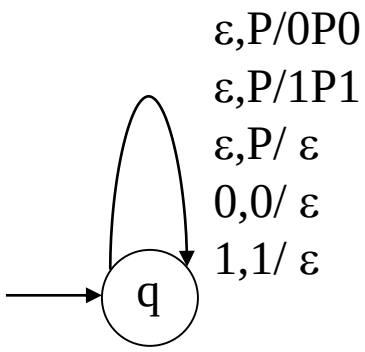
- A computation sequence for 0110:

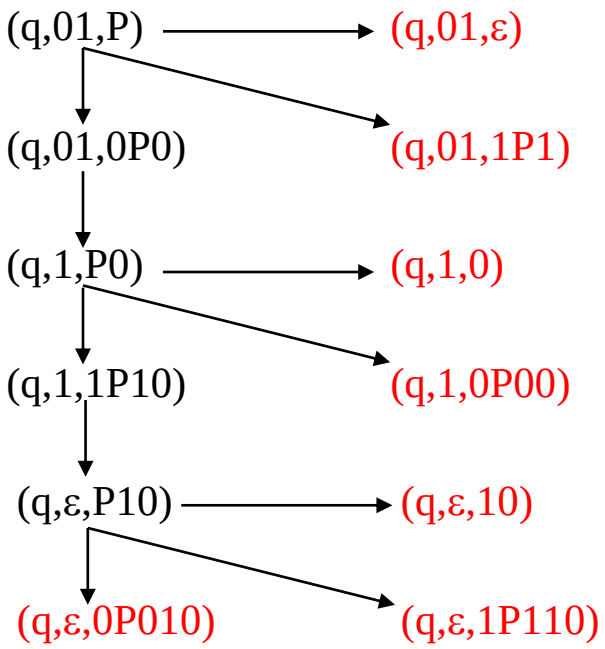
$$(q,0110,P) \vdash (q,0110,0P0) \vdash (q,110,P0) \vdash (q,110,1P10) \vdash (q,10,P10) \vdash (q,10,10)$$
$$\vdash (q,0,0) \vdash (q,\varepsilon,\varepsilon)$$

- Since there is a computation starts with initial ID and ends with empty stack and empty string for the string 0110, the string 0110 is recognized by this PDA.

A PDA Accepts By Empty Stack

- A PDA $P_N = (\{q\},\{0,1\},\{P,0,1\},\delta,q,P)$ which **accepts by empty stack**.
- The language of this PDA is $\{ ww^r : w \in \{0,1\}^* \}$.





- Since all computations do NOT end with an ID containing empty stack and empty string, the string 01 is NOT accepted by this PDA.

Equivalence of Language Definitions

Theorem:

- **If $L=L(P_F)$ for some PDA $P_F = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ which accepts by final state, then there exists another PDA P_N which accepts by empty stack such that $L=L(P_N)$.**

- **If $L=L(P_N)$ for some PDA $P_N = (Q,\Sigma,\Gamma,\delta,q_0,Z_0)$ which accepts by empty stack, then there exists another PDA P_F which accepts by final state such that $L=L(P_F)$.**

Creating an Equivalent PDA Accepting by Empty Stack from a PDA Accepting by Final State

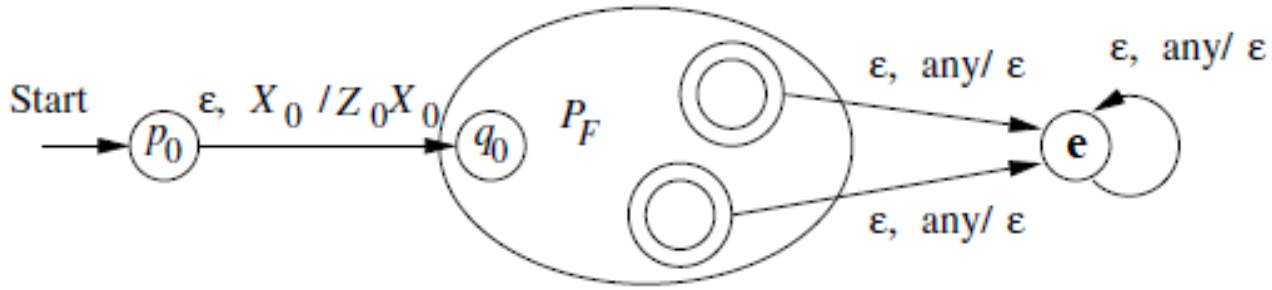
Proof: **We will construct a PDA P_N which accepts by empty stack from a given PDA $P_F = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ which accepts by final state.**

- **Let** $P_N = (Q \cup \{p_0,e\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$
  - The states of P_N will contain two new extra states p_0 and **e**.
    - The state p_0 is the start state of P_N.
    - The state **e** is an **erase state** which will empty the stack.
  - P_N will have a new stack symbol X_0, and X_0 will be the initial stack symbol of P_N.
    - The new stack symbol X_0 is to guard the stack bottom against accidental emptying.
  - The transition function δ_N will contain everything in the transition function δ of P_F and the following new transitions:
    - $\delta_N(p_0,\varepsilon,X_0)=\{(q_0,Z_0X_0)\}$ i.e. The new start state p_0 pushes the initial stack symbol Z_0 of P_F into the stack and moves to the state q_0 which is the start state of P_F.
    - For any final state **f** of P_F, $\delta_N(f,\varepsilon,Y)=\{(e,\varepsilon)\}$ for any stack symbol **Y**. i.e. A final state of P_F can move to the new erase **e** state by erasing the top of stack symbol.
    - $\delta_N(e,\varepsilon,Y)=\{(e,\varepsilon)\}$ i.e. The new erase state **e** erases all symbols from the stack.

Creating an Equivalent PDA Accepting by Empty Stack from a PDA Accepting by Final State

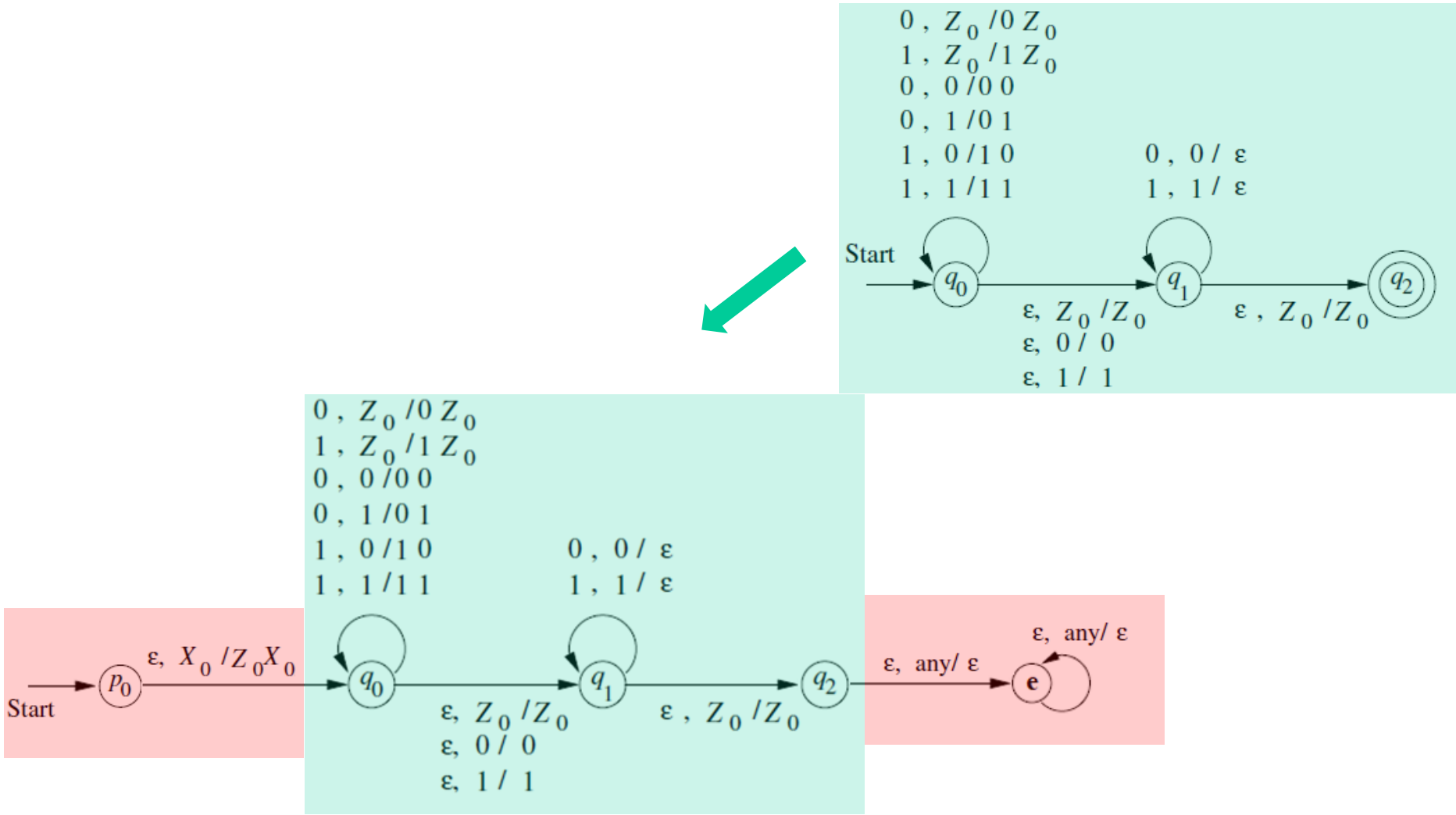
Proof (continued):

- From $P_F = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ construct $\mathbf{P_N = (Q \cup \{p_0,e\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)}$ as follows:



- Now, we must prove that $L(P_N)=L(P_F)$.
  - If $w \in L(P_F)$ then $w \in L(P_N)$. i.e. if $\mathbf{(q_0,w,Z_0) \vdash^*_{P_F} (f,\varepsilon,\alpha)}$ then $\mathbf{(p_0,w,X_0) \vdash^*_{P_N} (e,\varepsilon,\varepsilon)}$
  - If $w \in L(P_N)$ then $w \in L(P_F)$. i.e. if $\mathbf{(p_0,w,X_0) \vdash^*_{P_N} (e,\varepsilon,\varepsilon)}$ then $\mathbf{(q_0,w,Z_0) \vdash^*_{P_F} (f,\varepsilon,\alpha)}$

From a PDA Accepting by Final State to a PDA Accepting by Empty Stack : Example



Creating an Equivalent PDA Accepting by Final State from a PDA Accepting by Empty Stack

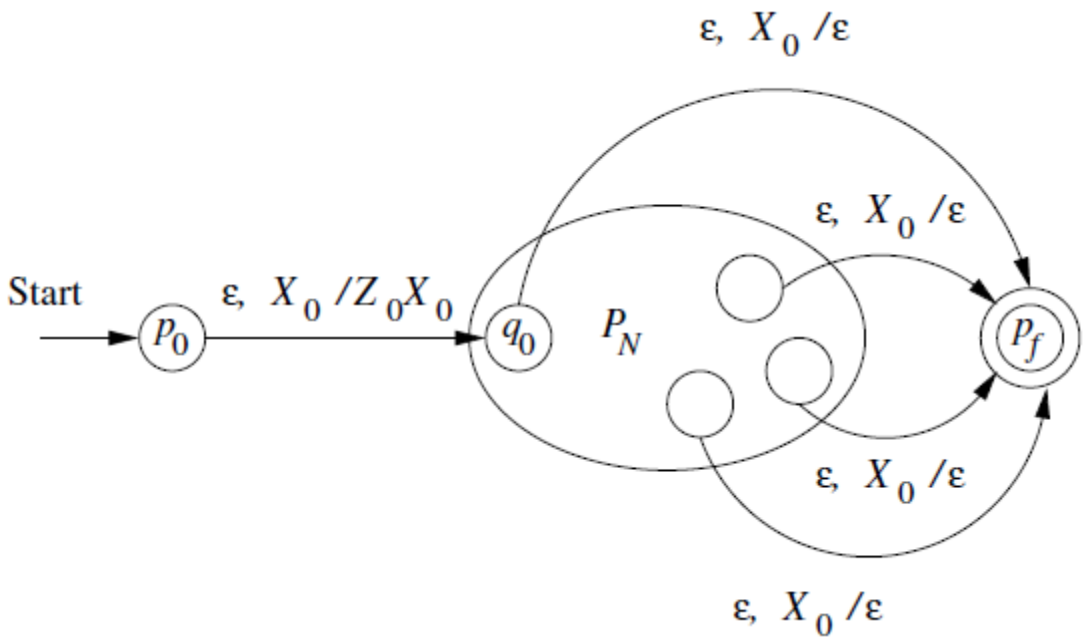
Proof: We will construct a PDA P_F which accepts by final state from a given PDA $P_N = (Q,\Sigma,\Gamma,\delta,q_0,Z_0)$ which accepts by empty stack.

- Let $\mathbf{P_F = (Q \cup \{p_0,p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})}$
  - The states of $\mathbf{P_F}$ will contain two new extra states $\mathbf{p_0}$ and $\mathbf{p_f}$.
    - The state $\mathbf{p_0}$ is the start state of $\mathbf{P_F}$.
    - The state $\mathbf{p_f}$ is only final state of $\mathbf{P_F}$.
  - $\mathbf{P_F}$ will have a new stack symbol $\mathbf{X_0}$, and $\mathbf{X_0}$ will be the initial stack symbol of $\mathbf{P_F}$.
    - The new stack symbol $\mathbf{X_0}$ is to guard the stack bottom against accidental emptying.
  - The transition function δ_F will contain everything in the transition function δ of $\mathbf{P_N}$ and the following new transitions:
    - $\mathbf{\delta_F(p_0,\varepsilon,X_0)=\{(q_0,Z_0X_0)\}}$ i.e. The new start state $\mathbf{p_0}$ pushes the initial stack symbol $\mathbf{Z_0}$ of P_N into the stack and moves to the state $\mathbf{q_0}$ which is the start state of $\mathbf{P_N}$.
    - For any state $\mathbf{q}$ of $\mathbf{P_N}$, $\mathbf{\delta_F(q,\varepsilon,X_0)=\{(p_f,\varepsilon)\}}$. i.e. Every state $\mathbf{q}$ of $\mathbf{P_N}$ can move to the new final state $\mathbf{p_f}$ when the stack symbol is $\mathbf{X_0}$ and erases the top of stack symbol.

Creating an Equivalent PDA Accepting by Final State from a PDA Accepting by Empty Stack

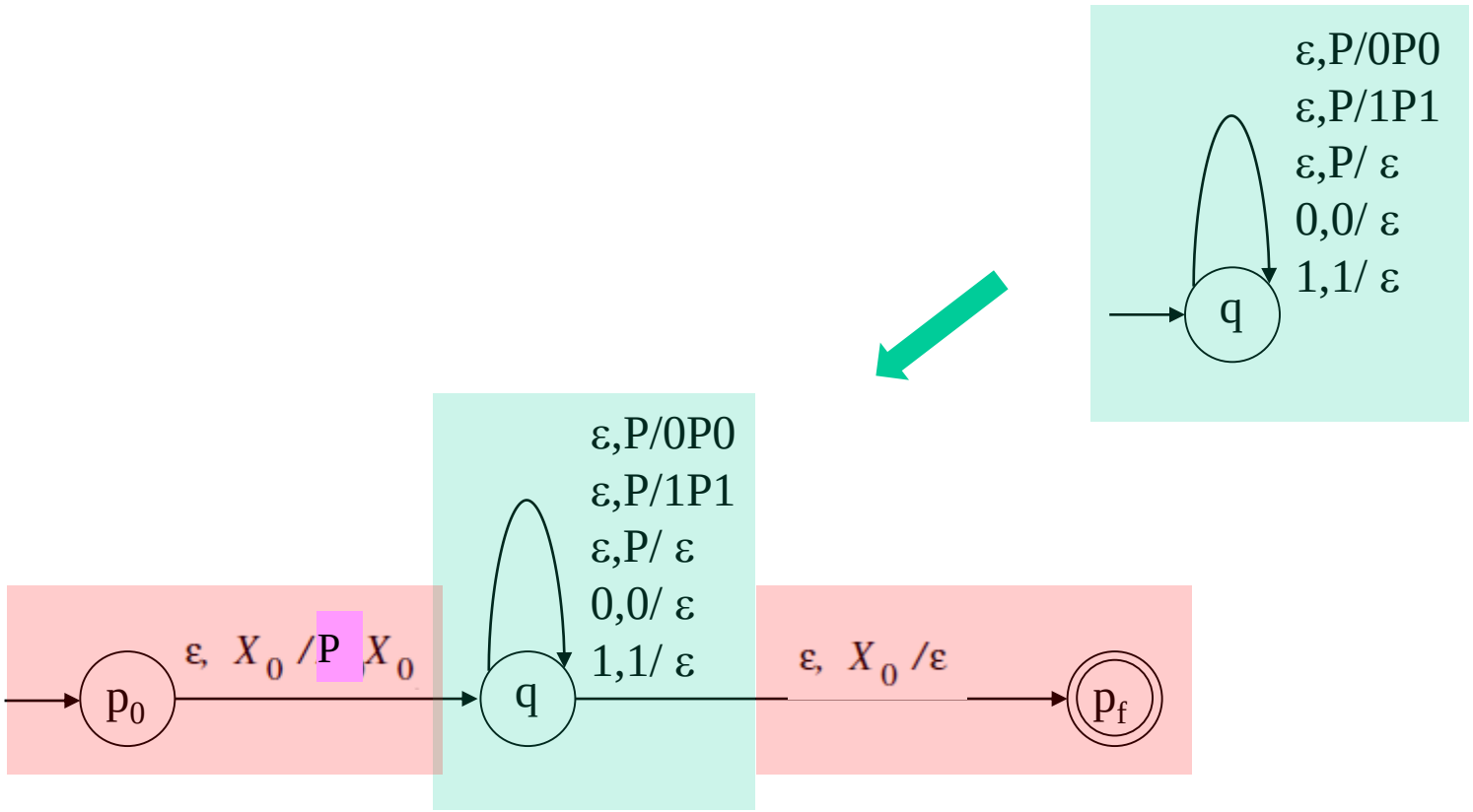
Proof (continued):

From $P_N=(Q,\Sigma,\Gamma,\delta,q_0,Z_0)$ construct $\mathbf{P_F=(Q\cup\{p_0,p_f\},\Sigma,\Gamma\cup\{X_0\},\delta_F,p_0,X_0,\{p_f\})}$ as follows:



- Now, we must prove that $L(P_F)=L(P_N)$.
  - If $w\in L(P_F)$ then $w\in L(P_N)$. i.e. if $\mathbf{(p_0,w,X_0)} \vdash^*_{P_F} \mathbf{(p_f,\varepsilon,\varepsilon)}$ then $\mathbf{(q_0,w,Z_0)} \vdash^*_{P_N} \mathbf{(q,\varepsilon,\varepsilon)}$ for any **q**
  - If $w\in L(P_N)$ then $w\in L(P_F)$. i.e. if $\mathbf{(q_0,w,Z_0)} \vdash^*_{P_N} \mathbf{(q,\varepsilon,\varepsilon)}$ for any **q** then $\mathbf{(p_0,w,X_0)} \vdash^*_{P_F} \mathbf{(p_f,\varepsilon,\varepsilon)}$

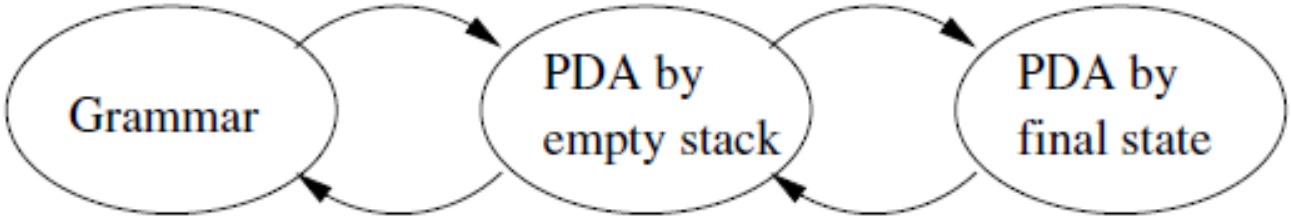
From a PDA Accepting by Empty Stack to a PDA Accepting by Final State : Example



Equivalence of PDA's and CFG's

Equivalence of PDA's and CFG's

- The following three classes of languages are all the same class.
  1. The context-free- languages, i.e.. the languages defined by CFG's.
  2. The languages that are accepted by final state by some PDA.
  3. The languages that are accepted by empty stack by some PDA.



- We have already shown that (2) and (3) are the same.
- It turns out to be easiest next to show that (1) and (3) are the same, thus Implying the equivalence of all three.

From Grammars to Pushdown Automata

- Given a CFG G, **we construct a PDA that simulates the leftmost derivations of G**.
- Any left-sentential form that is not a terminal string can be written as **xAα**, where
  - **A** is the leftmost variable,
  - **x** is whatever terminals appear to the left of **A**, and
  - **α** is the string of terminals and variables that appear to the right of **A**.
  - If a left-sentential form consists of terminals only, then **Aα** is **ε**.

- Let $\mathbf{xA\alpha \Rightarrow_{lm} x\beta\alpha}$ be a derivation step from the left-sentential form **xAα.**
  - This derivation step corresponds to the PDA having consumed **x** and having **A** on the top of the stack, and then the PDA on **ε** it pops **A** and pushes **β** into the stack.

From Grammars to Pushdown Automata

PDA Construction from CFG

PDA Construction from CFG:

- **Let G = (V, T, R, S) be a CFG.**
- **Construct the PDA P that accepts L(G) by empty stack as follows:**

$$P = (\{q\}, T, V \cup T, \delta, q, S)$$

where transition function δ is defined by:

1. For each variable A,

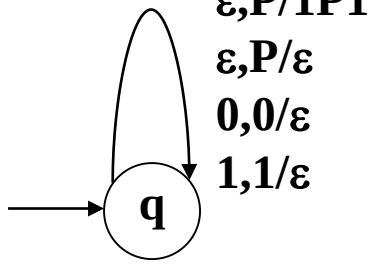
$$\delta(q,\varepsilon,A) = \{ (q, \beta) \mid A \rightarrow \beta \text{ is a production of } G \}$$

2. For each terminal a, $\delta(q,a,a) = \{(q,\varepsilon)\}$

From CFG to PDA - Example

- Let CFG G = ({P}, {0,1}, {P $\rightarrow$ 0P0, P $\rightarrow$ 1P1, P $\rightarrow\varepsilon$}, P)
- L(G) is { ww^r : w$\in${0,1}* }

- **We can construct PDA A = ({q}, {0,1}, {0,1,P}, δ, q, P) with following transitions.**



δ(q,ε,P) = { (q,0P0), (q,1P1), (q,ε) }

δ(q,0,0) = { (q,ε) }

δ(q,1,1) = { (q,ε) }

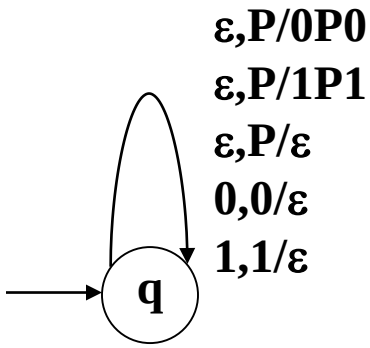
From CFG to PDA – Example

PDA Moves vs Leftmost Derivations

$P \rightarrow 0P0$

$P \rightarrow 1P1$

$P \rightarrow \varepsilon$



- A computation sequence for 0110:

  $(q,0110,P) \vdash (q,0110,0P0) \vdash (q,110,P0)$

- A leftmost derivation sequence of 0111:

  $P \Rightarrow 0P0$

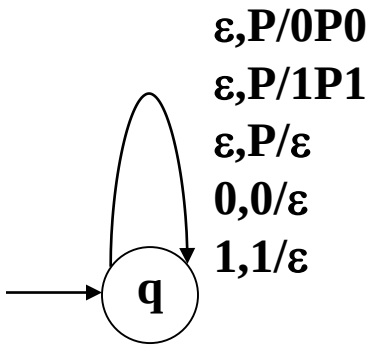
From CFG to PDA – Example

PDA Moves vs Leftmost Derivations

$P \rightarrow 0P0$

$P \rightarrow 1P1$

$P \rightarrow \varepsilon$



- A computation sequence for 0110:

  $(q,0110,P) \vdash (q,0110,0P0) \vdash (q,110,P0) \vdash (q,110,1P10) \vdash (q,10,P10)$

- A leftmost derivation sequence of 0111:

  $P \Rightarrow 0P0 \Rightarrow 01P10$

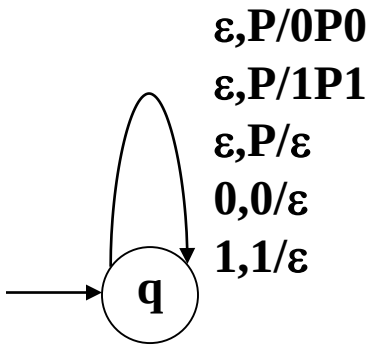
From CFG to PDA – Example

PDA Moves vs Leftmost Derivations

$P \rightarrow 0P0$

$P \rightarrow 1P1$

$P \rightarrow \varepsilon$



- A computation sequence for 0110:

  $(q,0110,P) \vdash (q,0110,0P0) \vdash (q,110,P0) \vdash (q,110,1P10) \vdash (q,10,P10)$

  $\vdash (q,10,10) \vdash (q,0,0) \vdash (q,\varepsilon,\varepsilon)$

- A leftmost derivation sequence of 0111:

  $P \Rightarrow 0P0 \Rightarrow 01P10 \Rightarrow 0110$

From Grammars to Pushdown Automata

PDA Construction from CFG

Theorem: **If PDA P is constructed from CFG G by the construction algorithm then L(P) = L(G).**

Proof:

- **We have to show that $(q,w,S) \vdash^* (q,\varepsilon,\varepsilon)$ if and only if $S \Rightarrow^* w$**
- **We need to prove something more general:**
  - **We need to show that $(q,wx,S) \vdash^* (q,x,\alpha)$ for any x if and only if $S \Rightarrow^*_{lm} w\alpha$**
  - **After this proof, we can let $x=\alpha=\varepsilon$**
    - **Then $(q,w,S) \vdash^* (q,\varepsilon,\varepsilon)$ if and only if $S \Rightarrow^* w$**
    - **That is, w is in L(P) if and only if w is in L(G).**

From Grammars to Pushdown Automata

PDA Construction from CFG

Proof (Only-If Part):

- Suppose $(q,wx,S) \vdash^* (q,x,\alpha)$ for any x.
- We have to show that $S \Rightarrow^*_{lm} w\alpha$
- Proof is an induction on the number steps made by P

Basis: 0 steps

- Then $\alpha=S$, $w=\varepsilon$, and $S \Rightarrow^*_{lm} S$ is surely true.

Induction:

- Consider **n** moves of **P**: $(q,wx,S) \vdash^* (q,x,\alpha)$ and assume the IH for sequences of **n-1** moves.
- There are two cases, depending on whether the last move uses a terminal or a variable on the top of stack.

From Grammars to Pushdown Automata

PDA Construction from CFG

Induction (cont.):

Case 1: **The move sequence must be of the form $(q,yax,S) \vdash^* (q,ax,a\alpha) \vdash (q,x,\alpha)$, where $ya = w$.**

- By the IH applied to the first n-1 steps, $S \Rightarrow^*_{lm} ya\alpha$.
- But $ya = w$, so $S \Rightarrow^*_{lm} w\alpha$.

Case 2: **The move sequence must be of the form $(q,wx,S) \vdash^* (q,x,A\beta) \vdash (q,x,\gamma\beta)$, where $A \rightarrow \gamma$ is a production and $\alpha = \gamma\beta$.**

- By the IH applied to the first n-1 steps, $S \Rightarrow^*_{lm} wA\beta$.
- Thus, $S \Rightarrow^*_{lm} w\gamma\beta = w\alpha$.

From Grammars to Pushdown Automata

PDA Construction from CFG

Proof (If Part):

- Suppose $S \Rightarrow^*_{lm} w\alpha$
- We have to show that $(q,wx,S) \vdash^* (q,x,\alpha)$ for any x.
- Proof is an induction on the length of the leftmost derivation.

- The proof is similar to Only-If part proof and it is in the book.

From a PDA Accepting with Empty Stack to a CFG

- Assume we have a PDA $P=(Q,\Sigma,\Gamma,\delta,q_0,Z_0)$ which accepts with empty stack.
- We'll construct a CFG G such that L(P) = L(G).

- Intuition:
  - G will have variables generating exactly the inputs that cause P to have the net effect of popping a stack symbol X while going from state p to state q.
  - P never gets below this X while doing so.

From a PDA to a CFG

Construct a CFG G=(V,T,R,S) from PDA $P=(Q,\Sigma,\Gamma,\delta,q_0,Z_0)$ such that L(P) = L(G).

Variables of G:

- G's variables are of the form **[pXq]** where **p∈Q, q∈Q, X∈Γ**,
  - The variable **[pXq]** generates all and only the strings **w** such that **$(p,w,X) \vdash^* (q,\varepsilon,\varepsilon)$.**
- In addition, G will also have a start symbol S.
- Thus, **V = { [pXq] : p∈Q, q∈Q, X∈Γ} ∪ {S}**

From a PDA to a CFG

Construct a CFG G=(V,T,R,S) from PDA P=(Q,Σ,Γ,δ,q_0,Z_0) such that L(P) = L(G).

Productions of G:

- Each production for **[pXq]** comes from a move of **P** in state **p** with stack symbol **X**.

CASE 1: **δ(p,a,X)** contains **(q,ε)**.

- G has the production **[pXq] → a** where **a∈Σ∪{ε}**.
  - **[pXq]** generates **a**, because reading **a** is one way to pop **X** and go from **p** to **q**.

CASE 2: **δ(p,a,X)** contains **(r,Y)** for some state **r** and a stack symbol **Y**.

- For each state **q∈Q**, G has the production **[pXq] → a[rYq]** where **a∈Σ∪{ε}**.
  - We can erase **X** and go from **p** to **q** by reading **a** (entering state **r** and replacing **X** by **Y**) and then reading some **w** that gets **P** from **r** to **q** while erasing **Y**.
  - Note: **[pXq] ⇒* aw** whenever **[rYq] ⇒* w**.

From a PDA to a CFG

Productions of G:

GENERAL CASE: **$\delta(p,a,X)$** contains **$(r,Y_1 \ldots Y_k)$** for some state **r** and $k \geq 2$.

- Generate a family of productions (For all states **q, s_1, …, s_{k-1}**)

  $$[pXq] \rightarrow a\,[rY_1s_1]\,[s_1Y_2s_2] \ldots [s_{k-2}Y_{k-1}s_{k-2}]\,[s_{k-1}Y_kq]$$

When k=2: **$\delta(p,a,X)$** contains **(r,YZ)** for some state **r**

- Now, **P** has replaced **X** by **YZ**.
- To have the net effect of erasing **X**, **P** must erase **Y**, going from state **r** to some state **s**, and then erase **Z**, going from **s** to **q**.
- Since we do not know state **s**, we must generate a family of productions:

  $$[pXq] \rightarrow a[rYs][sZq]$$ for all states **s**.

- Note: $[pXq] \Rightarrow^* awx$ whenever $[rYs] \Rightarrow^* w$ and $[sZq] \Rightarrow^* x$.

From a PDA to a CFG

Completion of the Construction:

- We can prove that $\mathbf{(q_0,w,Z_0) \vdash^* (p,\varepsilon,\varepsilon)}$ if and only if $\mathbf{[q_0Z_0p] \Rightarrow^* w}$.
- But state **p** can be anything.
- Thus, add productions $\mathbf{S \rightarrow [q_0Z_0p]}$ for each state **p**.

From a PDA to a CFG: Example

- Convert the PDA P=({p,q},{0,1},{X,Z_0},δ,q,Z_0) where δ is given by:

1. $\delta(q, 1, Z_0) = \{(q, XZ_0)\}$
2. $\delta(q, 1, X) = \{(q, XX)\}$
3. $\delta(q, 0, X) = \{(p, X)\}$
4. $\delta(q, \epsilon, X) = \{(q, \epsilon)\}$
5. $\delta(p, 1, X) = \{(p, \epsilon)\}$
6. $\delta(p, 0, Z_0) = \{(q, Z_0)\}$

- We get G=({[pXp],[pXq],[pZ_0p],[pZ_0q], [qXp],[qXq],[qZ_0p],[qZ_0q],S}, {0,1}, R, S)

From a PDA to a CFG: Example

G=({[pXp],[pXq],[pZ$_0$p],[pZ$_0$q], [qXp],[qXq],[qZ$_0$p],[qZ$_0$q], S}, {0,1}, R, S)

and productions in R are:

$$S \rightarrow [qZ_0q]|[qZ_0p]$$

From 1. $\delta(q,1,Z_0) = \{(q,XZ_0)\}$

$$[qZ_0q] \rightarrow 1[qXq][qZ_0q]$$
$$[qZ_0q] \rightarrow 1[qXp][pZ_0q]$$
$$[qZ_0p] \rightarrow 1[qXq][qZ_0p]$$
$$[qZ_0p] \rightarrow 1[qXp][pZ_0p]$$

From 2. $\delta(q,1,X) = \{(q,XX)\}$

$$[qXq] \rightarrow 1[qXq][qXq]$$
$$[qXq] \rightarrow 1[qXp][pXq]$$
$$[qXp] \rightarrow 1[qXq][qXp]$$
$$[qXp] \rightarrow 1[qXp][pXp]$$

From 3. $\delta(q,0,X) = \{(p,X)\}$

$$[qXq] \rightarrow 0[pXq]$$
$$[qXp] \rightarrow 0[pXp]$$

From 4. $\delta(q,\epsilon,X) = \{(q,\epsilon)\}$

$$[qXq] \rightarrow \epsilon$$

From 5. $\delta(p,1,X) = \{(p,\epsilon)\}$

$$[pXp] \rightarrow 1$$

From 6. $\delta(p,0,Z_0) = \{(q,Z_0)\}$

$$[pZ_0q] \rightarrow 0[qZ_0q]$$
$$[pZ_0p] \rightarrow 0[qZ_0p]$$

Deterministic Pushdown Automata (DPDA)

Deterministic Pushdown Automata (DPDA)

- While PDA's are by definition allowed to be nondeterministic, the deterministic subcase is quite important.
- In particular, parsers generally behave like deterministic PDA's, so the class of languages that can be accepted by these automata is interesting for the insights it gives us into what constructs are suitable for use in programming languages.

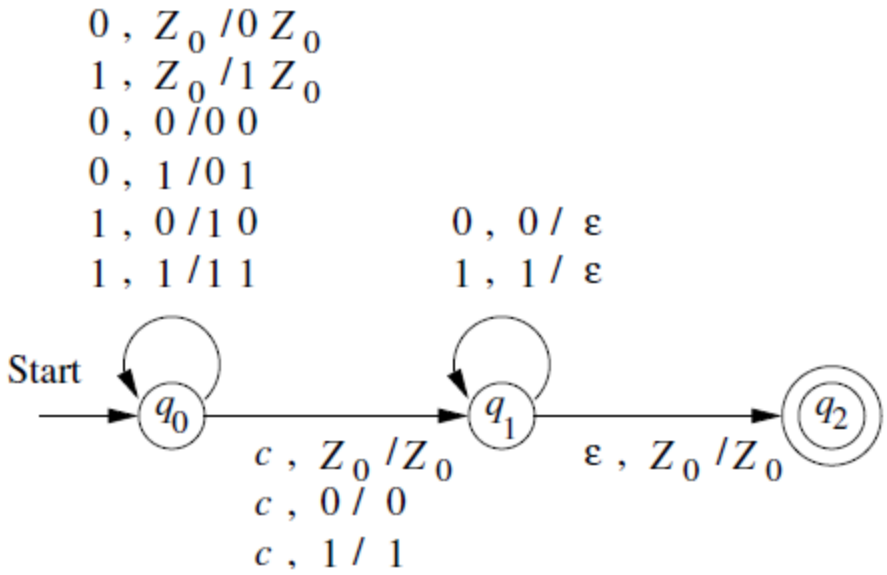
A PDA $P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ is deterministic iff

1. **(q,a,X) is always empty or a singleton where $a \in \Sigma$, or a is ε.**
2. **If (q,a,X) is nonempty where $a \in \Sigma$, then (q,ε,X) must be empty.**

DPDA - Example

$L = \{ wcw^R : w \in \{0,1\}^* \}$

- L is recognized by the following DPDA:



DPDA Properties

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

- **regular languages $\subset$ languages of DPDAs**

There are languages that are not regular, but there are DPDAs for them.

- $\{ wcw^R : w \in \{0,1\}^* \}$ is a member of languages of DPDAs but it is not regular.

There are context free languages which are NOT members of languages of DPDAs.

- $\{ ww^R : w \in \{0,1\}^* \}$ is NOT a member of languages of DPDAs. i.e. there is NO DPDA for the language $\{ ww^R : w \in \{0,1\}^* \}$.
- **languages of DPDAs $\subset$ context free languages**

DPDA Properties

There are NO DPDAs for inherently ambiguous CFLs.

For a given unambiguous grammar we may NOT find a DPDA.

- $\{ ww^R : w \in \{0,1\}^* \}$ has unambiguous grammar $S \rightarrow 0S0 \mid 1S1 \mid \varepsilon$
- But $\{ ww^R : w \in \{0,1\}^* \}$ is NOT a member of languages of DPDAs.

We can always find an unambiguous grammar for the language of a given DPDA.

- The **languages of DPDAs** are equal to **languages of deterministic context free grammars**.
- Each **LR(k) grammar** is an **unambiguous grammar**, but **not all unambiguous grammars are LR(k) grammars.**
- The **languages of LR(k) grammars (k≥1)** are equal to languages of **deterministic context free grammars (languages of DPDAs)**.
- LR(k) grammars are widely used in the parsing of programming languages (LR parsers).

Formal Languages -- So far