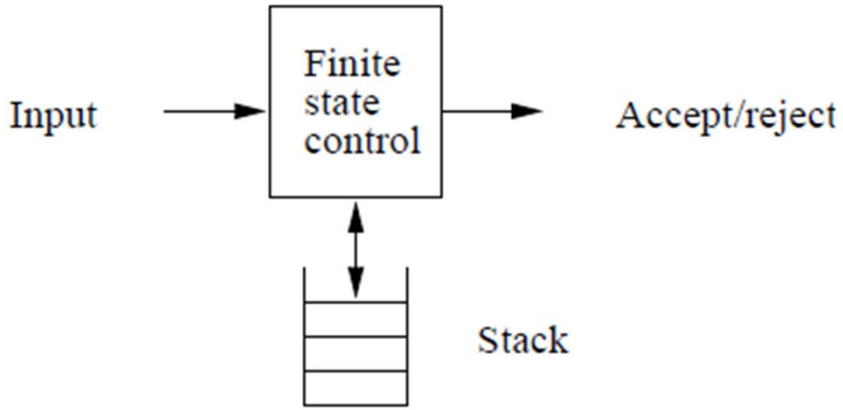
Pushdown Automata

Pushdown Automata

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



Pushdown Automata - Example

Example: Let's consider $L_{wwr} = \{ww^R : w \in \{0,1\}^*\}$,

with the grammar

$$P \to 0P0, \quad P \to 1P1, \quad P \to \varepsilon$$

- A PDA for L_{wwr} has tree states, and operates as follows:
  1. Guess that you are reading w. Stay in state 0, and push the input symbol onto the stack.
  2. Guess that you are in the middle of ww^R. Go spontaneously to state 1.
  3. You're now reading the head of w^R. Compare it to the top of the stack. If they match, pop the stack, and remain in state 1. If they don't match, go to sleep.
  4. If the stack is empty, go to state 2 and accept.

Pushdown Automata – Formal Definition

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

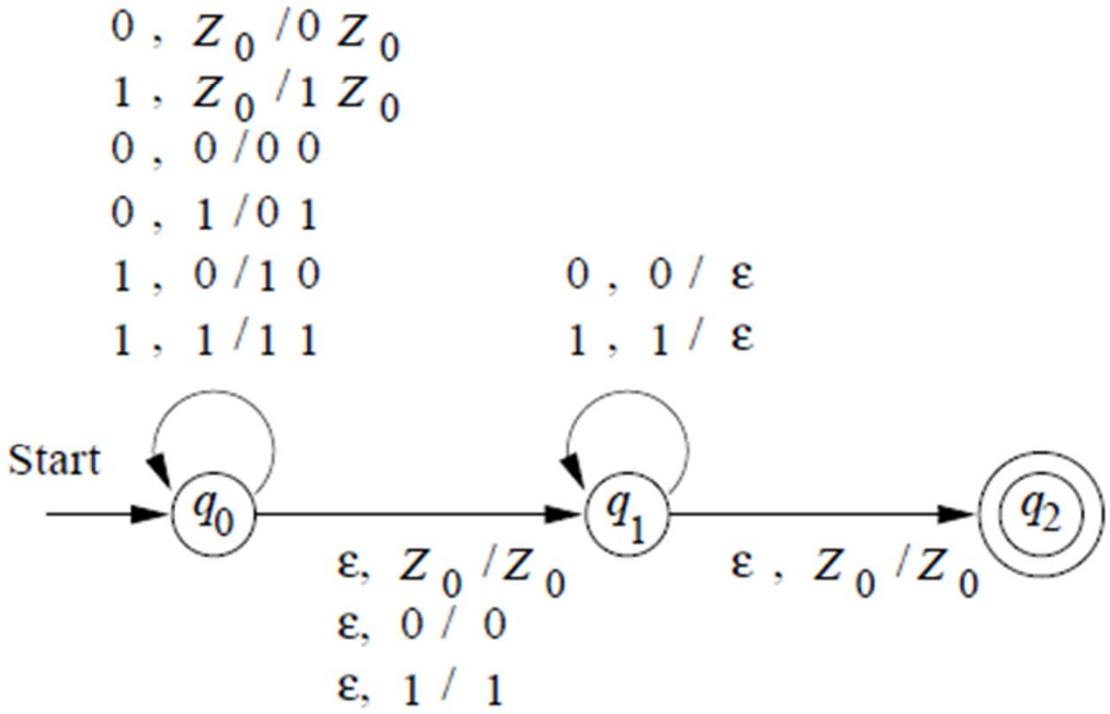
$$P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F),$$

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.

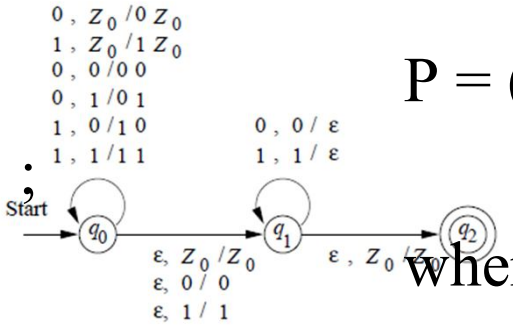
Pushdown Automata- Example

The PDA for L_{wwr} as a transition diagram:



Pushdown Automata- Example

- The PDA is 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 the transition function is the following table (set brackets missing)

	$0, Z_0$	$1, Z_0$	$0,0$	$0,1$	$1,0$	$1,1$	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		
$\star q_2$									

Instantaneous Descriptions 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" notation

- We need a notation that describes changes in the state, the input, and stack.
- The "turnstile" notation for connecting pairs of ID's that represent one or many mows of a PDA is used to show PDA moves.

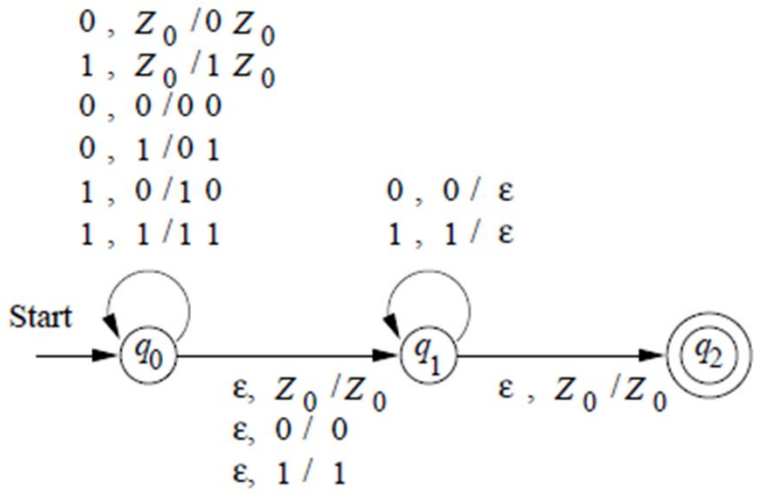
Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. Then
$\forall w \in \Sigma^*, \beta \in \Gamma^*$:

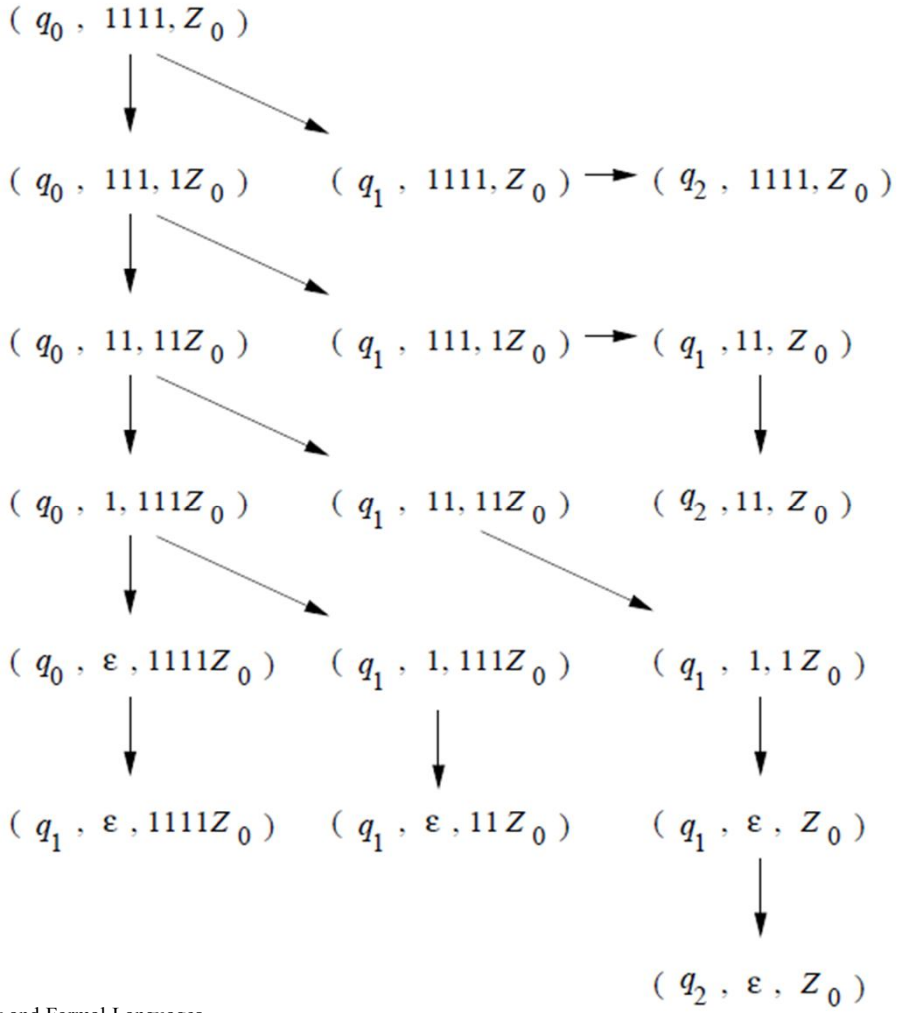
$$(q, aw, X\beta) \vdash (p, w, \alpha\beta) \qquad \text{if} \qquad (p, \alpha) \in \delta(q, a, X)$$

We define $\vdash^*$ to be the reflexive-transitive closure of $\vdash$.

PDA Move - Example

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





Three important principles about ID's

- The following properties hold:

1. If an ID sequence is a legal computation for a PDA, then so is the sequence obtained by adding an additional string at the end of component number two.

2. If an ID sequence is a legal computation for a PDA, then so is the sequence obtained by adding an additional string at the bottom of component number three.

3. If an ID sequence is a legal computation for a PDA, and some tail of the input is not consumed, then removing this tail from all ID's result in a legal computation sequence.

Three important principles about ID's

- We formalize points (1) and (2) in Th 6.5, and (3) in Th. 6.6.

Theorem 6.5: If $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA, and $(q, x, \alpha) \overset{*}{\underset{P}{\vdash}} (p, y, \beta)$, then for any strings w in Σ^* and γ in Γ^*, it is also true that

$$(q, xw, \alpha\gamma) \overset{*}{\underset{P}{\vdash}} (p, yw, \beta\gamma)$$

Note that if $\gamma = \epsilon$, then we have a formal statement of principle (1) above, and if $w = \epsilon$, then we have the second principle.

Theorem 6.6: If $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA, and

$$(q, xw, \alpha) \overset{*}{\underset{P}{\vdash}} (p, yw, \beta)$$

then it is also true that $(q, x, \alpha) \overset{*}{\underset{P}{\vdash}} (p, y, \beta)$. □

The Languages of a PDA

- 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 axe equivalent, in the sense that 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.

Acceptance by Final State

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. The *language accepted by P by final state* is

$$L(P) = \{w : (q_0, w, Z_0) \vdash^* (q, \epsilon, \alpha),\ q \in F\}.$$

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

Acceptance by Empty Stack

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. The *language accepted by P by empty stack* is

$$N(P) = \{w : (q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon)\}.$$

Note: q can be any state.

- That is, N(P) is the set of inputs w that P can consume and at the same time empty its stack.

From Empty Stack to Final State

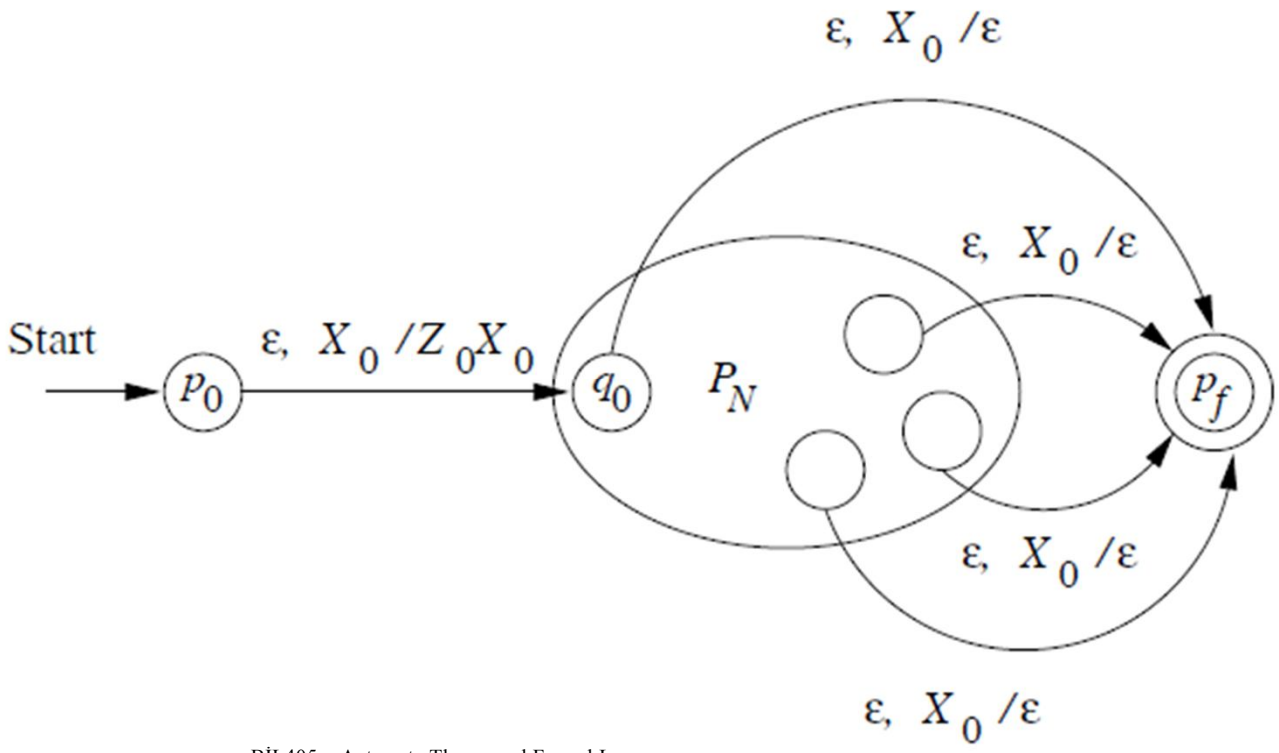
- If there is a PDA P_N that accepts a language L by empty stack then we can construct a PDA P_F that accepts L by final state.

Theorem 6.9: If $L = N(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, then $\exists$ PDA P_F, such that $L = L(P_F)$.

From Empty Stack to Final State

Proof: Let $P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$

where $\delta_F(p_0, \epsilon, X_0) = \{(q_0, Z_0 X_0)\}$, and for all $q \in Q, a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma : \delta_F(q, a, Y) = \delta_N(q, a, Y)$, and in addition $(p_f, \epsilon) \in \delta_F(q, \epsilon, X_0)$.



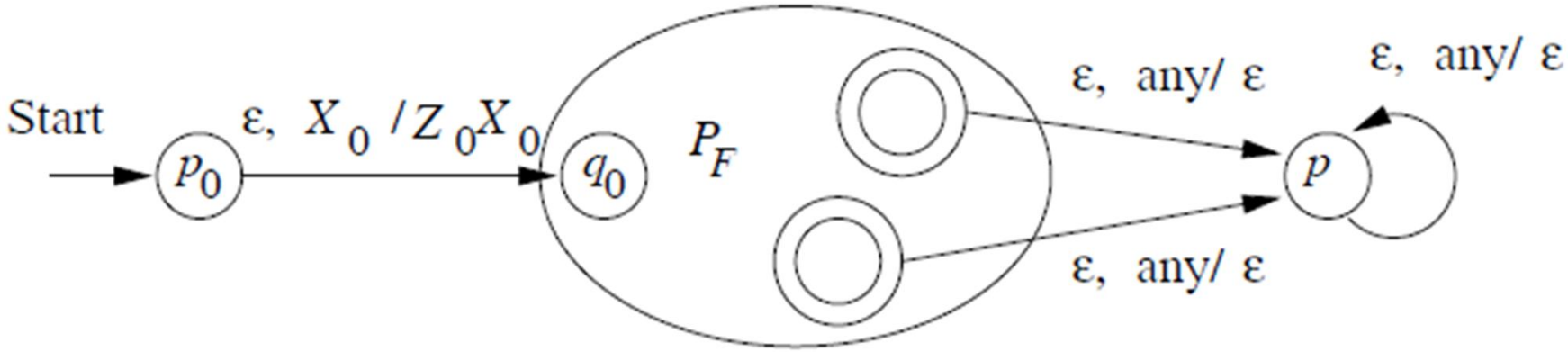
From Final State to Empty

Theorem 6.11: Let $L = L(P_F)$, for some PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$. Then $\exists$ PDA P_n, such that $L = N(P_N)$.

From Final State to Empty

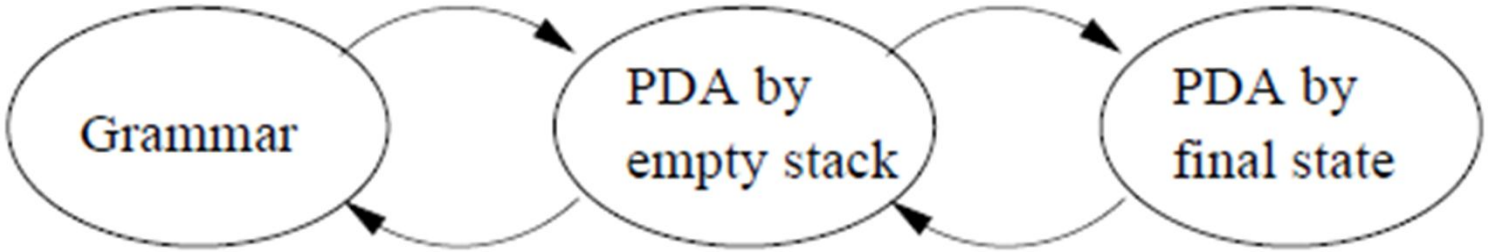
- **Proof:** Let $P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$

  where $\delta_N(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$, $\delta_N(p, \epsilon, Y) = \{(p, \epsilon)\}$, for $Y \in \Gamma \cup \{X_0\}$, and for all $q \in Q$, $a \in \Sigma \cup \{\epsilon\}, Y \in \Gamma : \delta_N(q, a, Y) = \delta_F(q, a, Y)$, and in addition $\forall q \in F$, and $Y \in \Gamma \cup \{X_0\}$: $(p, \epsilon) \in \delta_N(q, \epsilon, Y)$.



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 its left, 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 A α is ε.
- Let $\mathbf{xA\alpha} \Rightarrow_{\mathbf{lm}} \mathbf{x\beta\alpha}$
- This corresponds to the PDA rest having consumed x and having A on the stack, and then on ε it pops A and pushes β

.

From Grammars to Pushdown Automata - Formal

Let G = {V,T, Q, 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 5 is defined by:

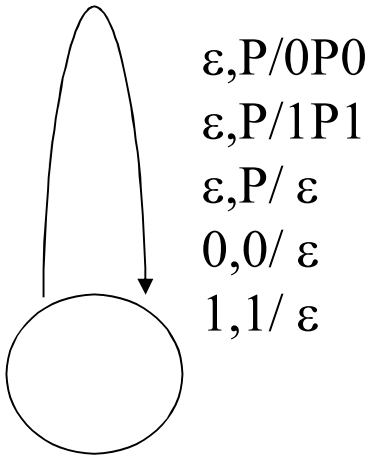
1. For each variable A,

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

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

From CFG to PDA - Example

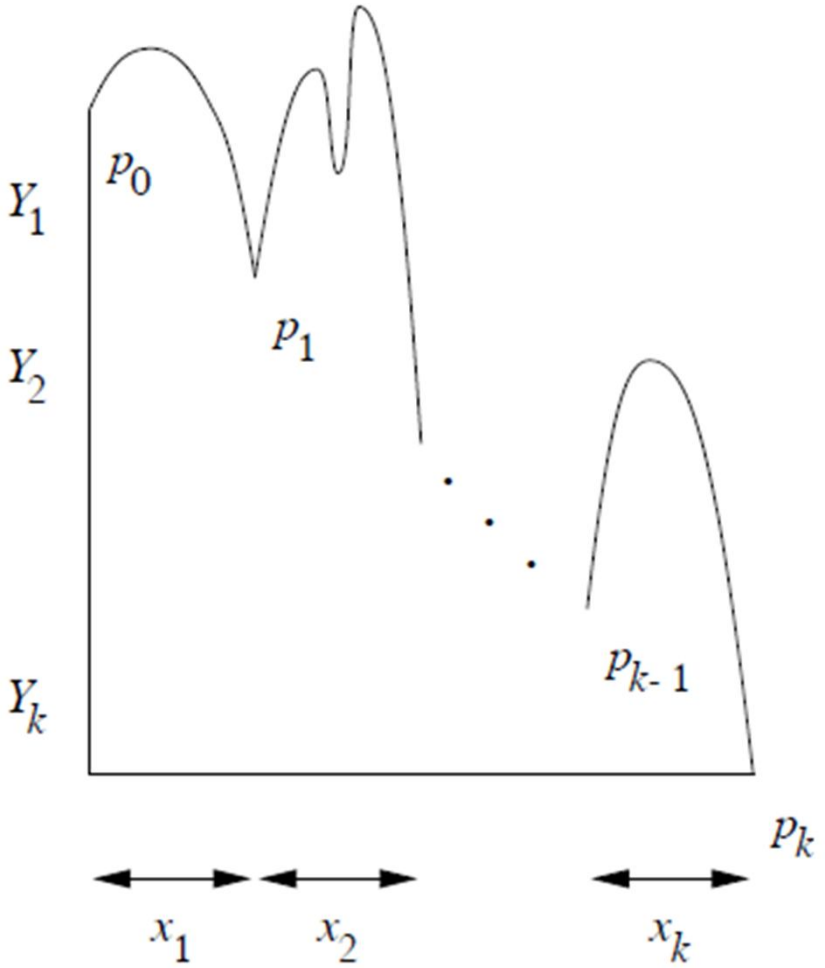
- $P \rightarrow 0P0, \quad P \rightarrow 1P1, \quad P \rightarrow \varepsilon$



From PDA with empty stack to CFG

Let's look at how a PDA can consume $x = x_1 x_2 \ldots x_k$ and empty the stack.

We shall define a grammar with variables of the form $[p_{i-1} Y_i p_i]$ representing going from p_{i-1} to p_i with net effect of popping Y_i.



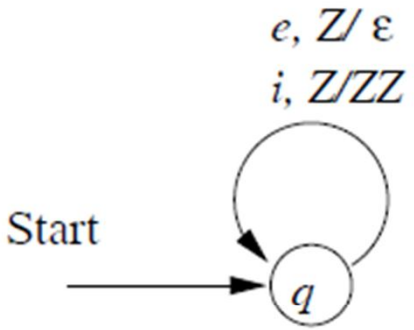
From PDA with empty stack to CFG

Formally, let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ be a PDA.
Define $G = (V, \Sigma, R, S)$, where

$$V = \{[pXq] : \{p, q\} \subseteq Q, X \in \Gamma\} \cup \{S\}$$

$$R = \{S \to [q_0 Z_0 p] : p \in Q\} \cup$$
$$\{[\mathbf{q}Xr_k] \to a[\mathbf{r}Y_1 r_1] \cdots [r_{k-1} Y_k r_k] :$$
$$a \in \Sigma \cup \{\epsilon\},$$
$$\{r_1, \ldots, r_k\} \subseteq Q,$$
$$(\mathbf{r}, Y_1 Y_2 \cdots Y_k) \in \delta(\mathbf{q}, a, X)\}$$

From PDA with empty stack to CFG - Example



$$P_N = (\{q\}, \{i, e\}, \{Z\}, \delta_N, q, Z),$$

where $\delta_N(q, i, Z) = \{(q, ZZ)\}$,
and $\delta_N(q, e, Z) = \{(q, \epsilon)\}$ to a grammar

$$G = (V, \{i, e\}, R, S),$$

where $V = \{[qZq], S\}$, and
$R = \{[qZq] \rightarrow i[qZq][qZq], [qZq] \rightarrow e\} \cup \{S \rightarrow qZq\}$

If we replace $[qZq]$ by A we get the productions
$S \rightarrow A$ and $A \rightarrow iAA|e$.

From PDA with empty stack to CFG – Example2

Example: Let $P = (\{p, q\}, \{0, 1\}, \{X, Z_0\}, \delta, q, Z_0)$, 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. $\delta(p, 0, Z_0) = \{(q, Z_0)\}$

From PDA with empty stack to CFG – Example2

We get $G = (V, \{0, 1\}, R, S)$, where $V = \{[pXp], [pXq], [pZ_0p], [pZ_0q], S\}$ and the productions in R are

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

From rule (1):

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

From rule (2):

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

From rule (3):

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

From rule (4):

$[qXq] \rightarrow \epsilon$

From rule (5):

$[pXp] \rightarrow 1$

From rule (6):

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

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.

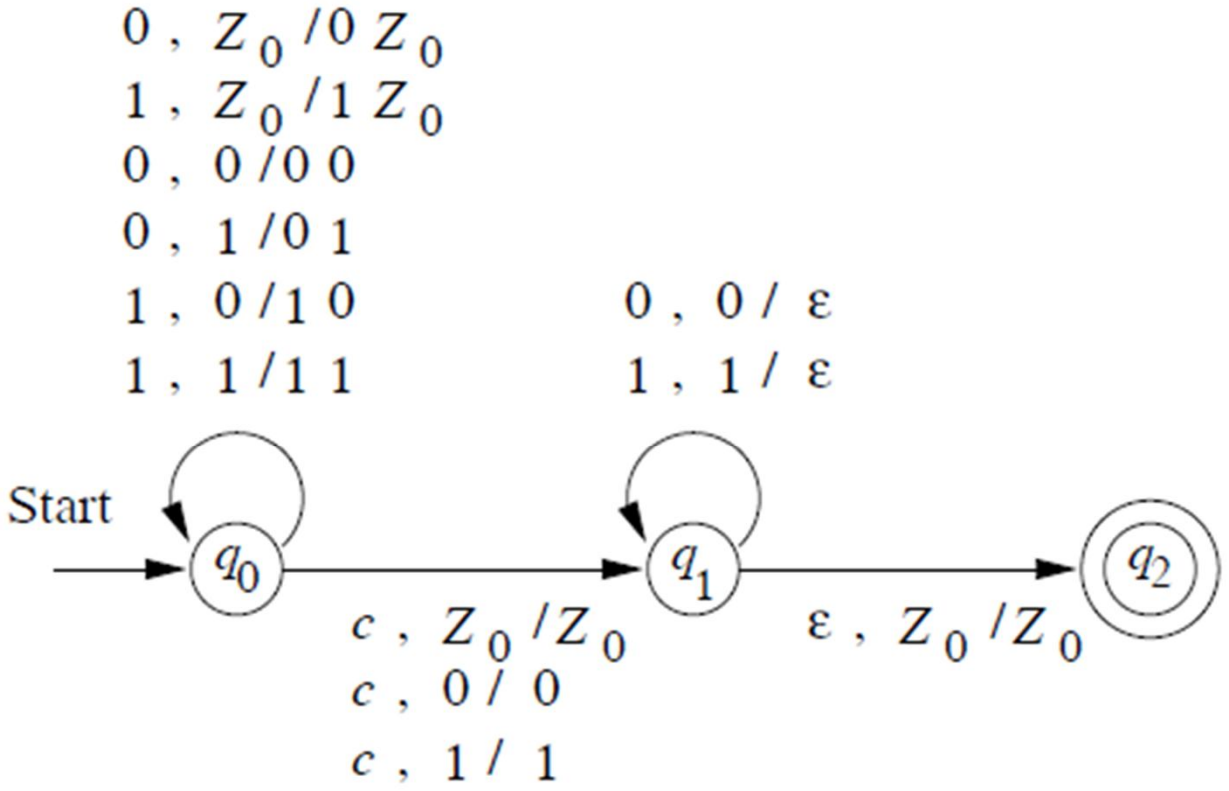
A PDA $P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ is deterministic iff

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

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

Then L_{wcwr} is recognized by the following DPDA



DPDA Properties

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

regular languages $\subset$ languages of DPDAs

L_{wcwr} is a member of languages of DPDAs but it is not regular.

languages of DPDAs $\subset$ context free languages

L_{wwr} is a contex free language
but it is not a member of languages of DPDAs.

DPDA Properties

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

  L_{wwr} has unambiguous grammar $S \rightarrow 0S0 \mid 1S1 \mid \varepsilon$
  but not is not L(DPDA).

- But we can always find an unambiguous grammar for the language of a given DPDA.