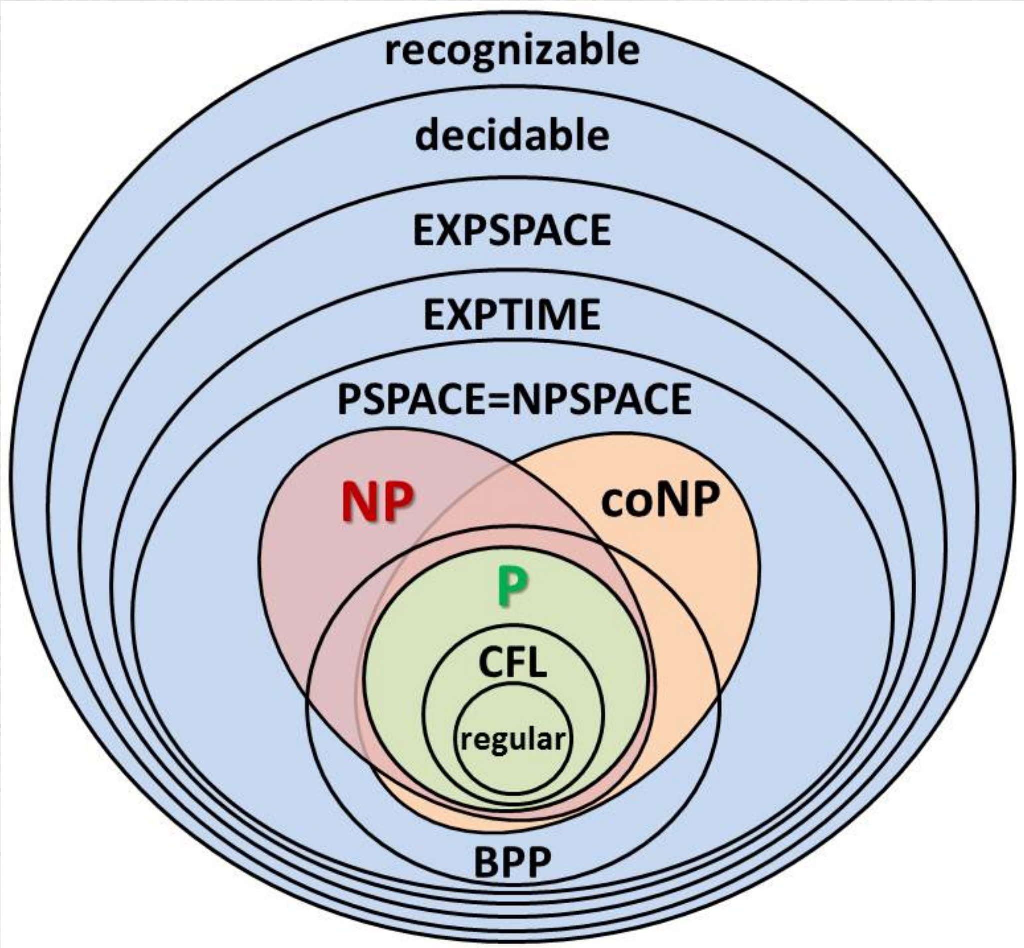
CS311: Computational Theory

Lecture 8: CONTEXT-FREE GRAMMARS – Ch 2 (Cont'd)

Lecture Learning Objectives

1. *Convert among equivalently powerful notations for a language, including among PDAs and CFGs.*

Abstract View of Objectives

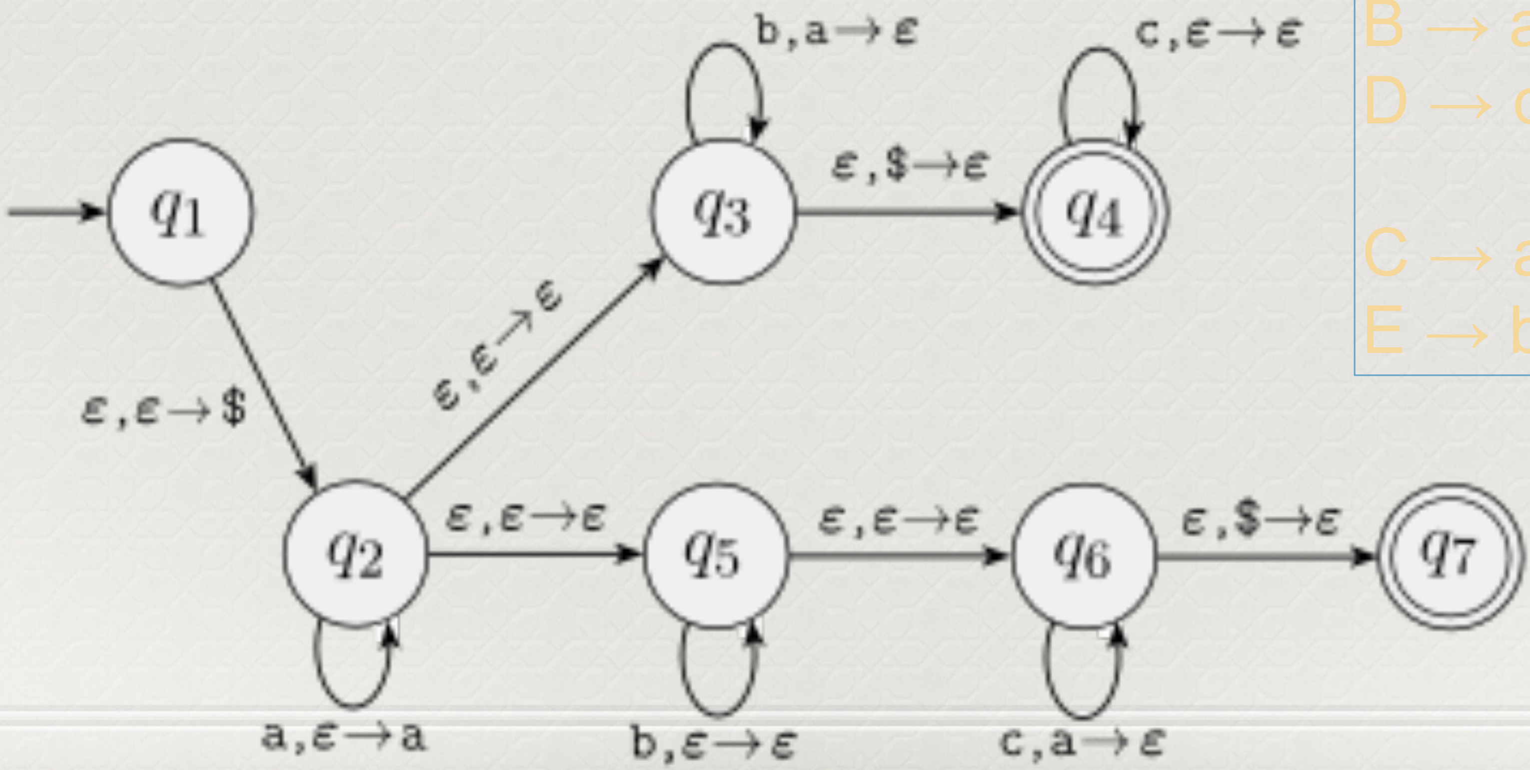


PDA: Example 4

Context Free Language:

$\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$.

Push Down Automaton:



Context Free Grammar:

S → BC
B → aBbD | ε
D → cD |ε

C → aCc | bE |ε
E → bE |ε

Chomsky Normal Form

- *A context-free grammar is in **Chomsky normal form** if every rule is of the form*

  $$A \to BC$$

  $$A \to a$$

- *where a is any terminal and A, B, and C are any variables—except that B and C may not be the start variable. In addition, we permit the rule $S \to \varepsilon$, where S is the start variable.*

Conversion to Chomsky Normal Form

- *First, add a new start variable.*
- *Then, eliminate all ε-rules of the form $A \rightarrow \varepsilon$.*
- *Also eliminate all unit rules of the form $A \rightarrow B$.*
- *In both cases, patch up the grammar to be sure that it still generates the same language.*
- *Finally, convert the remaining rules into the proper form.*

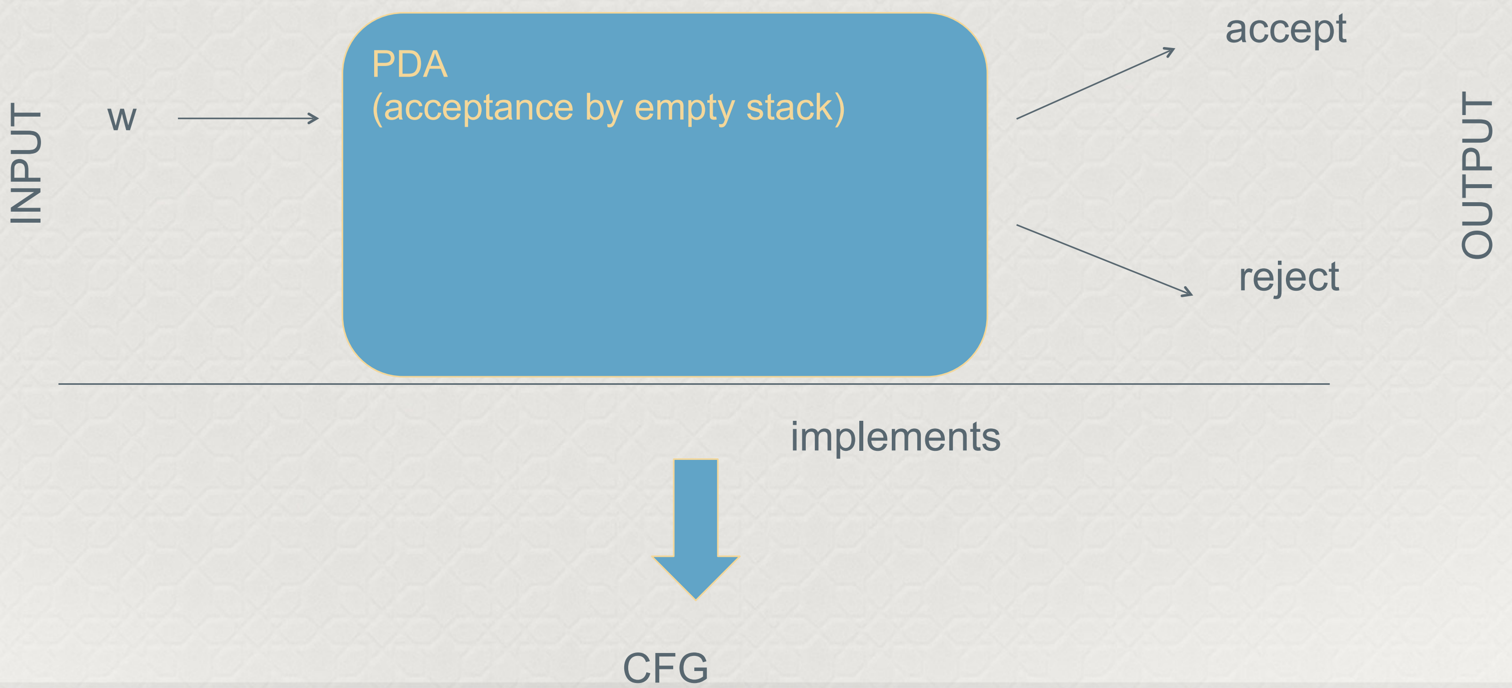
THEOREM

- *A language is context free if and only if some pushdown automaton recognizes it.*

This is same as: “implementing a CFG using a PDA”

Converting CFG to PDA

Main idea: *The PDA simulates the leftmost derivation on a given w, and upon consuming it fully it either arrives at acceptance (by empty stack) or non-acceptance.*



Convert CFG to PDA

- *Design the PDA to determine whether some series of substitutions using the rules of CFG can lead from the start variable to the input string w.*
- *It is difficult to non-deterministically choose which substitution to use to substitute for a variable:*
  - *Begin by writing the start variable on the stack.*
  - *Then make a series of intermediate strings.*
  - *Eventually the PDA may arrive at a string that contains only terminal symbols matching the input w and accept, meaning that it has used the grammar to derive a string.*

Convert CFG to PDA – Cont'd

- *Use the stack of the PDA to simulate the derivation of a string in the grammar.*
  - *Push S (start variable of G) on the stack*
  - *From this point on, there are two moves the PDA can make:*
    - *If a variable A is on the top of the stack, pop it and push the right-hand side of a production $A \rightarrow \beta$ from G.*
    - *If a terminal, a is on the top of the stack, pop it and match it with whatever symbol is being read from the tape.*

Convert CFG to PDA – Steps

1. *Place the marker symbol $ and the start variable on the stack.*
2. *Repeat the following steps forever.*
   a. *If the top of stack is a variable symbol A, non-deterministically select one of the rules for A and substitute A by the string on the right-hand side of the rule.*
   b. *If the top of stack is a terminal symbol a, read the next symbol from the input and compare it to a. If they match, repeat. If they do not match, reject on this branch of the non-determinism.*
   c. *If the top of stack is the symbol $, enter the accept state. Doing so accepts the input if it has all been read.*

1. PDA Construction Informal Steps

- $P = (Q, \Sigma, \Gamma, \delta, q_{start}, F)$, $q, r \in Q$, $a \in \Sigma$, $S \in \Gamma$
- PDA to go from q to r when it reads a and pops S
- It pushes the entire string $u = u_1 \cdots u_l$ on the stack at the same time, then introduce new states $q_1, \ldots, q_{l-1}$
- Set the transition function as follows:

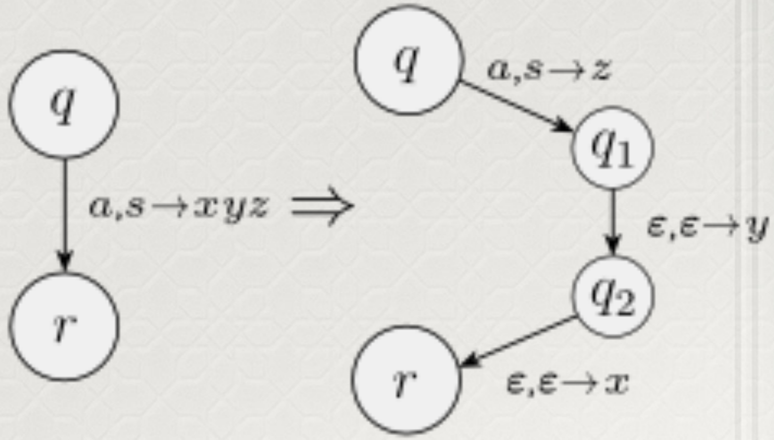
$\delta(q, a, s)$ to contain (q_1, u_l),

$\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, u_{l-1})\}$,

$\delta(q_2, \varepsilon, \varepsilon) = \{(q_3, u_{l-2})\}$,

…

$\delta(q_{l-1}, \varepsilon, \varepsilon) = \{(r, u_1)\}$.



2. PDA Construction Formal Steps

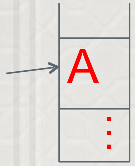
- *Generally, $Q = \{q_{start}, q_{loop}, q_{accept}\} \cup E$, where E is the set of states we need for implementing the grammar.*
- *The transition function is defined as follows.*
  - *Begin by initializing the stack to contain the symbols \$ and S: $\delta(q_{start}, \varepsilon, \varepsilon) = \{(q_{loop}, S\$)\}$.*
  - *Then put in transitions for the main loop of step 2.*
    - ➢ *First, case (a) wherein the top of the stack contains a variable. Let $\delta(q_{loop}, \varepsilon, A) = \{(q_{loop}, w) \mid \text{where } A \rightarrow w \text{ is a rule in } R\}$.*
    - ➢ *Second, case (b) wherein the top of the stack contains a terminal. Let $\delta(q_{loop}, a, a) = \{(q_{loop}, \varepsilon)\}$.*
    - ➢ *Finally, case (c) wherein the empty stack marker \$ is on the top of the stack. Let $\delta(q_{loop}, \varepsilon, \$) = \{(q_{accept}, \varepsilon)\}$.*

Formal construction of PDA from CFG

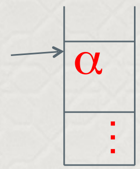
Note: Initial stack symbol (S) same as the start variable in the grammar

- *Given:* $G= (V,T,P,S)$
- *Output:* $P_N = (\{q\}, T, V \cup T, \delta, q, S)$
- δ:
  - For all $A \in V$, add the following transition(s) in the PDA:
    - ✓ $\delta(q, \varepsilon ,A) = \{ (q, \alpha) \mid \text{“}A ==>\alpha\text{”} \in P\}$
  - For all $a \in T$, add the following transition(s) in the PDA:
    - ✓ $\delta(q,a,a)= \{ (q, \varepsilon) \}$

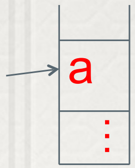
Before:



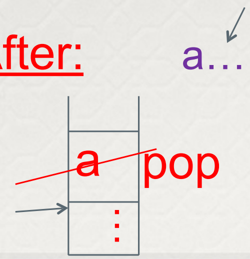
After:



Before:



After:



There are two types of PDAs that one can design: those that accept by final state or by empty stack

Acceptance by…

- ***PDAs that accept by final state:***
  - *For a PDA P, the language accepted by P, denoted by L(P) by final state, is:*
    - $\{w \mid (q_0,w,Z_0) \mid\text{---}^* (q,\varepsilon, A) \}$, s.t., $q \in F$

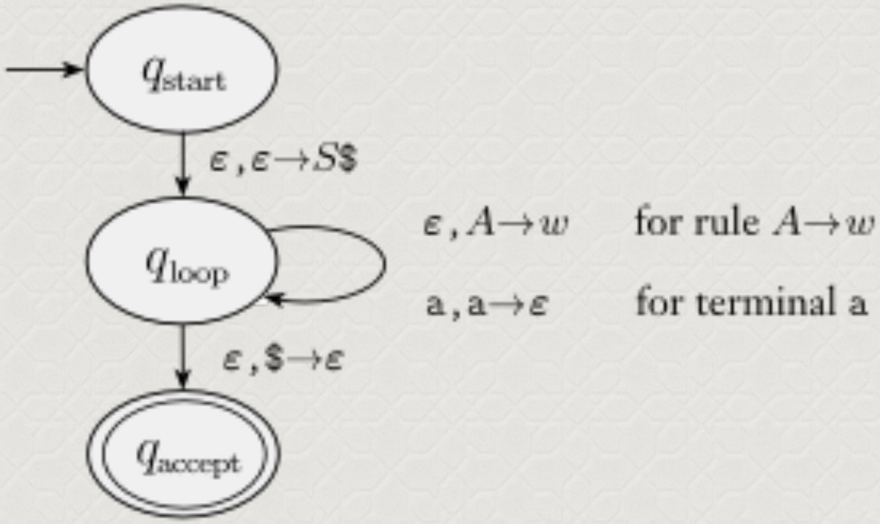
Checklist:
- input exhausted?
- in a final state?

- ***PDAs that accept by empty stack:***
  - *For a PDA P, the language accepted by P, denoted by N(P) by empty stack, is:*
    - $\{w \mid (q_0,w,Z_0) \mid\text{---}^* (q, \varepsilon, \varepsilon) \}$, for any $q \in Q$.

Checklist:
- input exhausted?
- is the stack empty?

Q) Does a PDA that accepts by empty stack need any final state specified in the design?

PDA Construction Formal Steps



CFG to PDA Example 1

- *Given the Grammar G = (V, Σ, S, P)*

V = {S,X,F}, Set of Variables

Σ = {a,+,,(,)}, Set of Terminals*

Start variable is S

Production Rules P = {   S -> S+X | X

*X -> X*F | F*

F -> (S) | a   }

- *This grammar generates a subset of all legal arithmetic expressions.*

CFG to PDA Example 1 – Cont'd

- $PDA = (Q, q_{start}, A, \Sigma, \Gamma, Z, \delta)$
- Set of States $Q = \{q_{start}, q_{loop}, q_{accept}\}$,
- A is the accepting state (q_{accept}),
- Σ is the language alphabet (terminals)
- Γ is the stack variables ($V \cup \Sigma \cup Z$)
- Choose an initial stack symbol Z not in S or V.

$q_{start}, \varepsilon, Z \rightarrow q_{loop}, SZ$

$q_{loop}, \varepsilon, S \rightarrow q_{loop}, S{+}X$

$q_{loop}, \varepsilon, X \rightarrow q_{loop}, X{*}F$

$q_{loop}, \varepsilon, F \rightarrow q_{loop}, (S)$

$q_{loop}, a, a \rightarrow q_{loop}, \varepsilon$

$q_{loop}, *, * \rightarrow q_{loop}, \varepsilon$

$q_{loop},),) \rightarrow q_{loop}, \varepsilon$

$q_{loop}, \varepsilon, Z \rightarrow q_{accept}, Z$

$q_{loop}, \varepsilon, S \rightarrow q_{loop}, X$

$q_{loop}, \varepsilon, X \rightarrow q_{loop}, F$

$q_{loop}, \varepsilon, F \rightarrow q_{loop}, a$

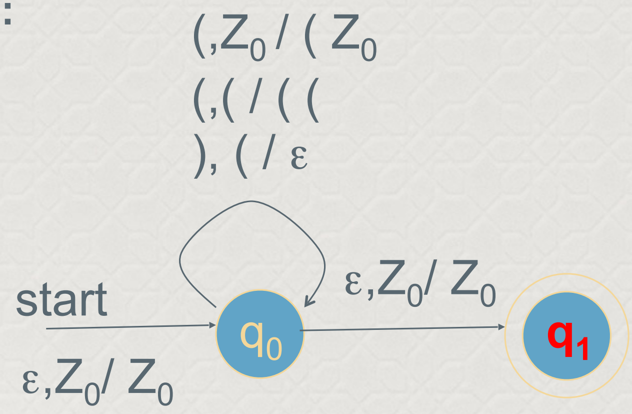
$q_{loop}, +, + \rightarrow q_{loop}, \varepsilon$

$q_{loop}, (, (\rightarrow q_{loop}, \varepsilon$

Example 2: L of balanced parenthesis

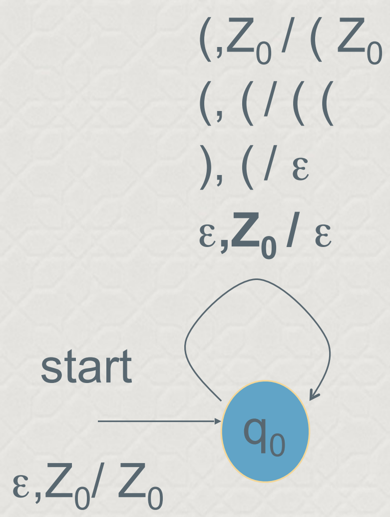
PDA that accepts by final state

P_F:



An equivalent PDA that accepts by empty stack

P_N:



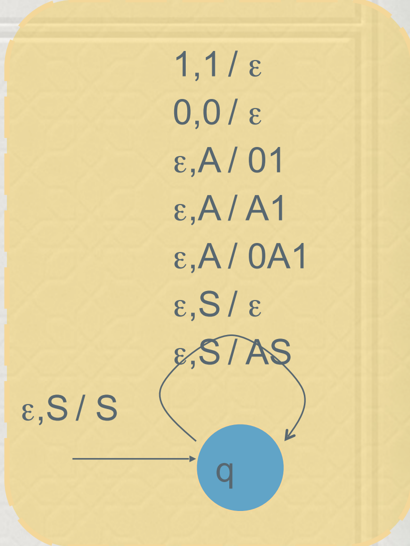
How will these two PDAs work on the input: ((( )) ( )) ( )

PDAs accepting by final state and empty stack are <u>equivalent</u>

- P_F <= PDA accepting by final state
  - $P_F = (Q_F, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$
- P_N <= PDA accepting by empty stack
  - $P_N = (Q_N, \Sigma, \Gamma, \delta_N, q_0, Z_0)$
- *<u>Theorem:</u>*
  - (P_N==> P_F) For every P_N, there exists a P_F s.t. $L(P_F)=L(P_N)$
  - (P_F==> P_N) For every P_F, there exists a P_N s.t. $L(P_F)=L(P_N)$

Example 3: CFG to PDA

- $G = (\{S,A\}, \{0,1\}, P, S)$
- P:
  - $S \rightarrow AS \mid \varepsilon$
  - $A \rightarrow 0A1 \mid A1 \mid 01$
- $PDA = (\{q\}, \{0,1\}, \{0,1,A,S\}, \delta, q, S)$
- δ:
  - *All Variables:*
    - ➢ $\delta(q, \varepsilon, S) = \{ (q, AS), (q, \varepsilon) \}$
    - ➢ $\delta(q, \varepsilon, A) = \{ (q,0A1), (q,A1), (q,01) \}$
  - *Then All Terminals:*
    - ➢ $\delta(q, 0, 0) = \{ (q, \varepsilon) \}$
    - ➢ $\delta(q, 1, 1) = \{ (q, \varepsilon) \}$



How will this new PDA work?

Lets simulate string <u>0011</u>

Simulating string 0011 on the new PDA …

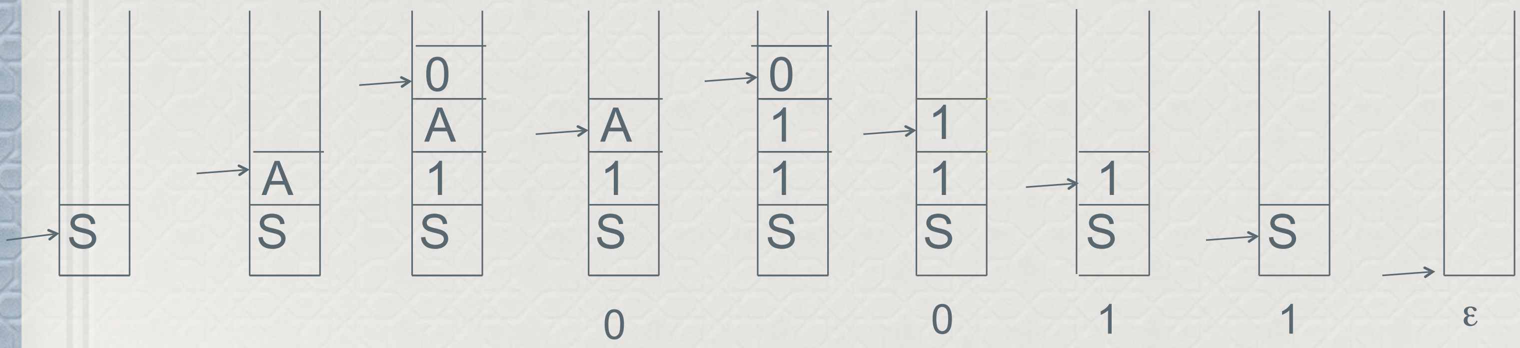
PDA (δ):

$\delta(q, \varepsilon, S) \to \{ (q, AS), (q, \varepsilon) \}$
$\delta(q, \varepsilon, A) \to \{ (q,0A1), (q,A1), (q,01) \}$
$\delta(q, 0, 0) \to \{ (q, \varepsilon) \}$
$\delta(q, 1, 1) \to \{ (q, \varepsilon) \}$

Leftmost deriv.:

$S \to AS$
$\to 0A1S$
$\to 0011S$
$\to 0011$

Stack moves (shows only the successful path):



Accept by empty stack

$S \to AS \to 0A1S \to 0011S \to 0011$

Converting a PDA into a CFG - 1

1. *Transform the PDA such that:*
   - *Only one character must be popped from the stack at a time.*
   - *For every transition that does not inspect the stack (i.e., the pop character is Ɛ), add one transition that pops a single character and pushes it back again, for each letter in the stack alphabet.*

Converting a PDA into a CFG - 2

2. ***Reverse engineer the productions from transitions***
   - *Add a rule S → <sƐf> for the start state, s, and each final state, f.*
   - *Add a rule <qƐq> → Ɛ for each state q.*
   - *For every transition that pushes a terminal a (that can be Ɛ and disappears) into the stack:*

   $\delta(q,a,Z) \rightarrow (p, Y_1 Y_2 Y_3 \ldots Y_k)$:
     - ✓ *State is changed from q to p;*
     - ✓ *Terminal a is consumed;*
     - ✓ *Stack top symbol Z is popped and replaced with a sequence of k variables.*
   - ***Action:*** *Create a grammar variable called "[qZp]" which includes the following production:*

   [qZp] → a[pY1q1] [q1Y2q2] [q2Y3q3]… [qk-1Ykqk]

Example 1: Bracket matching

- *To avoid confusion, we will use b="(" and e=")":*
- *P_N: ($\{q_0\}$, $\{b,e\}$, $\{Z_0, Z_1\}$, δ, q_0, Z_0)*

1. $\delta(q_0,b,Z_0) \rightarrow \{ (q_0,Z_1Z_0) \}$
2. $\delta(q_0,b,Z_1) \rightarrow \{ (q_0,Z_1Z_1) \}$
3. $\delta(q_0,e,Z_1) \rightarrow \{ (q_0, \varepsilon) \}$
4. $\delta(q_0, \varepsilon ,Z_0) \rightarrow \{ (q_0, \varepsilon) \}$

0. $S \rightarrow [q_0Z_0q_0]$
1. $[q_0Z_0q_0] \rightarrow b\ [q_0Z_1q_0]\ [q_0Z_0q_0]$
2. $[q_0Z_1q_0] \rightarrow b\ [q_0Z_1q_0]\ [q_0Z_1q_0]$
- $[q_0Z_1q_0] \rightarrow e$
1. $[q_0Z_0q_0] \rightarrow \varepsilon$

Let $A=[q_0Z_0q_0]$
Let $B=[q_0Z_1q_0]$

0. $S \rightarrow A$
1. $A \rightarrow b\ B\ A$
- $B \rightarrow b\ B\ B$
- $B \rightarrow e$
- $A \rightarrow \varepsilon$

Simplifying,

0. $S \rightarrow b\ B\ S \mid \varepsilon$
1. $B \rightarrow b\ B\ B \mid e$

If you were to directly write a CFG:

$S \rightarrow b\ S\ e\ S \mid \varepsilon$

Example 2: $\{0^n1^n\}$

- PDA = $(Q,\Sigma,\Gamma,\delta,q,Z)$, $Q = \{q,r\}$, $\Sigma = \{0,1\}$, $\Gamma = \{Z,X\}$, δ is:

1. $\delta(q,0,Z) \rightarrow \{ (q,XZ) \}$
2. $\delta(q,0,X) \rightarrow \{ (q,XX) \}$
3. $\delta(q,1,X) \rightarrow \{ (r,\varepsilon) \}$
4. $\delta(r,1,X) \rightarrow \{ (r,\varepsilon) \}$
5. $\delta(r,\varepsilon,Z) \rightarrow \{ (r,\varepsilon) \}$

0. S $\rightarrow$ [qZq] [qZr]
1. [qZq] $\rightarrow$ 0 [qXq] [qZq] | 0[qXr][rZq]
2. [qZr] $\rightarrow$ 0[qXq][qZr] | 0[qXr][rZr]
3. [qXq] $\rightarrow$ 0 [qXq] [qXq] | 0[qXr][rXq]
4. [qXr] $\rightarrow$ 1 | 0[qXq][qXr] | 0[qXr][rXr]
5. [rZq] $\rightarrow$
6. [rXr] $\rightarrow$ 1
7. [rZr] $\rightarrow$ ε
8. [rXq] $\rightarrow$

Example 2: $\{0^n1^n\}$

- PDA = $(Q,\Sigma,\Gamma,\delta,q,Z)$, $Q = \{q,r\}$, $\Sigma = \{0,1\}$, $\Gamma = \{Z,X\}$, δ is:

1. $\delta(q,0,Z) \to \{ (q,XZ) \}$
2. $\delta(q,0,X) \to \{ (q,XX) \}$
3. $\delta(q,1,X) \to \{ (r, \varepsilon) \}$
4. $\delta(r, 1, X) \to \{ (r, \varepsilon) \}$
5. $\delta(r, \varepsilon, Z) \to \{ (r, \varepsilon) \}$

0. S $\to$ [qZq] [qZr]
1. [qZq] $\to$ 0 [qXq] [qZq] | 0[qXr][rZq]
2. [qZr] $\to$ 0[qXq][qZr] | 0[qXr][rZr]
3. [qXq] $\to$ 0 [qXq] [qXq] | 0[qXr][rXq]
4. [qXr] $\to$ 1 | 0[qXq][qXr] | 0[qXr][rXr]
5. [rZq] $\to$
6. [rXr] $\to$ 1
7. [rZr] $\to$ ε
8. [rXq] $\to$

0. S $\to$ [qZq] [qZr]
1. [qZq] $\to$ 0 [qXq] [qZq]
2. [qZr] $\to$ 0[qXq][qZr] | 0[qXr][rZr]
3. [qXq] $\to$ 0 [qXq] [qXq]
4. [qXr] $\to$ 1 | 0[qXq][qXr] | 0[qXr][rXr]
5. [rXr] $\to$ 1
6. [rZr] $\to$ ε

Example 2: $\{0^n1^n\}$

- *PDA = (Q,Σ,Γ,δ,q,Z) , Q = {q,r}, Σ = {0,1}, Γ = {Z,X}, δ is:*

1. δ(q,0,Z) → { (q,XZ) }
2. δ(q,0,X) → { (q,XX) }
3. δ(q,1,X) → { (r, ε) }
4. δ(r, 1 ,X) → { (r, ε) }
5. δ(r, ε ,Z) → { (r, ε) }

0. S → [qZq] [qZr]
1. [qZq] → 0 [qXq] [qZq] | 0[qXr][rZq]
2. [qZr] → 0[qXq][qZr] | 0[qXr][rZr]
3. [qXq] → 0 [qXq] [qXq] | 0[qXr][rXq]
4. [qXr] → 1 | 0[qXq][qXr] | 0[qXr][rXr]
5. [rZq] →
6. [rXr] → 1
7. [rZr] → ε
8. [rXq] →

0. S → [qZq] [qZr]
1. [qZq] → 0 [qXq] [qZq]
2. [qZr] → 0[qXq][qZr] | 0[qXr][rZr]
3. [qXq] → 0 [qXq] [qXq]
- [qXr] → 1 | 0[qXq][qXr] | 0[qXr][rXr]
1. [rXr] → 1
2. [rZr] → ε

0. S → [qZr]
1. [qZr] → 0[qXr][rZr]
2. [qXr] → 1 | 0[qXq][qXr]
3. [rXr] → 1
4. [rZr] → ε

Example 2: $\{0^n1^n\}$

- *PDA = (Q,Σ,Γ,δ,q,Z) , Q = {q,r}, Σ = {0,1}, Γ = {Z,X}, δ is:*

1. $\delta(q,0,Z) \to \{ (q,XZ) \}$
2. $\delta(q,0,X) \to \{ (q,XX) \}$
3. $\delta(q,1,X) \to \{ (r, \varepsilon) \}$
4. $\delta(r, 1 ,X) \to \{ (r, \varepsilon) \}$
5. $\delta(r, \varepsilon ,Z) \to \{ (r, \varepsilon) \}$

0. S → [qZq] [qZr]
1. [qZq] → 0 [qXq] [qZq] | 0[qXr][rZq]
2. [qZr] → 0[qXq][qZr] | 0[qXr][rZr]
3. [qXq] → 0 [qXq] [qXq] | 0[qXr][rXq]
4. [qXr] → 1 | 0[qXq][qXr] | 0[qXr][rXr]
5. [rZq] →
6. [rXr] → 1
7. [rZr] → ε
8. [rXq] →

0. S → [qZq] [qZr]
1. [qZq] → 0 [qXq] [qZq]
2. [qZr] → 0[qXq][qZr] | 0[qXr][rZr]
3. [qXq] → 0 [qXq] [qXq]
- [qXr] → 1 | 0[qXq][qXr] | 0[qXr][rXr]
1. [rXr] → 1
2. [rZr] → ε

0. S → [qZr]
1. [qZr] → 0[qXr][rZr]
2. [qXr] → 1 | 0[qXq][qXr]
3. [rXr] → 1
4. [rZr] → ε

0. S → [qZr]
1. [qZr] → 0[qXr]
2. [qXr] → 1 | 0[qXq]1

Example 2: $\{0^n1^n\}$

- *PDA = (Q,Σ,Γ,δ,q,Z) , Q = {q,r}, Σ = {0,1}, Γ = {Z,X}, δ is:*

1. δ(q,0,Z) → { (q,XZ) }
2. δ(q,0,X) → { (q,XX) }
3. δ(q,1,X) → { (r, ε) }
4. δ(r, 1 ,X) → { (r, ε) }
5. δ(r, ε ,Z) → { (r, ε) }

0. S → [qZq] [qZr]
1. [qZq] → 0 [qXq] [qZq] | 0[qXr][rZq]
2. [qZr] → 0[qXq][qZr] | 0[qXr][rZr]
3. [qXq] → 0 [qXq] [qXq] | 0[qXr][rXq]
4. [qXr] → 1 | 0[qXq][qXr] | 0[qXr][rXr]
5. [rZq] →
6. [rXr] → 1
7. [rZr] → ε
8. [rXq] →

0. S → [qZq] [qZr]
1. [qZq] → 0 [qXq] [qZq]
2. [qZr] → 0[qXq][qZr] | 0[qXr][rZr]
3. [qXq] → 0 [qXq] [qXq]
- [qXr] → 1 | 0[qXq][qXr] | 0[qXr][rXr]
1. [rXr] → 1
2. [rZr] → ε

0. S → [qZr]
1. [qZr] → 0[qXr][rZr]
2. [qXr] → 1 | 0[qXq][qXr]
3. [rXr] → 1
4. [rZr] → ε

0. S → [qZr]
1. [qZr] → 0[qXr]
2. [qXr] → 1 | 0[qXq]1

Let A=[qZr]
Let B=[qXr]
0. S → A
1. A → 0 B
2. B → 1|0 B 1

Example 2: $\{0^n1^n\}$

- *PDA = (Q,Σ,Γ,δ,q,Z) , Q = {q,r}, Σ = {0,1}, Γ = {Z,X}, δ is:*

1. δ(q,0,Z) → { (q,XZ) }
2. δ(q,0,X) → { (q,XX) }
3. δ(q,1,X) → { (r, ε) }
4. δ(r, 1 ,X) → { (r, ε) }
5. δ(r, ε ,Z) → { (r, ε) }

0. S → [qZq] [qZr]
1. [qZq] → 0 [qXq] [qZq] | 0[qXr][rZq]
2. [qZr] → 0[qXq][qZr] | 0[qXr][rZr]
3. [qXq] → 0 [qXq] [qXq] | 0[qXr][rXq]
4. [qXr] → 1 | 0[qXq][qXr] | 0[qXr][rXr]
5. [rZq] →
6. [rXr] → 1
7. [rZr] → ε
8. [rXq] →

0. S → [qZq] [qZr]
1. [qZq] → 0 [qXq] [qZq]
2. [qZr] → 0[qXq][qZr] | 0[qXr][rZr]
3. [qXq] → 0 [qXq] [qXq]
- [qXr] → 1 | 0[qXq][qXr] | 0[qXr][rXr]
1. [rXr] → 1
2. [rZr] → ε

0. S → [qZr]
1. [qZr] → 0[qXr][rZr]
2. [qXr] → 1 | 0[qXq][qXr]
3. [rXr] → 1
4. [rZr] → ε

0. S → [qZr]
1. [qZr] → 0[qXr]
2. [qXr] → 1 | 0[qXq]1

Let A=[qZr]
Let B=[qXr]
0. S → A
1. A → 0 B
2. B → 1|0 B 1

Simplifying,

S → 0 S 1 | ε

Two ways to build a CFG

Build a PDA → Construct CFG from PDA (indirect)

Derive CFG directly (direct)

Similarly… Two ways to build a PDA

Derive a CFG → Construct PDA from CFG (indirect)

Design a PDA directly (direct)