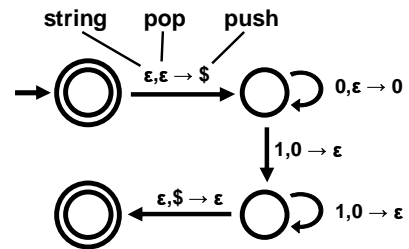


**PDA_s ARE EQUIVALENT TO CFG_s
AND
THE CHOMSLY NORMAL FORM**

THURSDAY SEP 15



CONTEXT-FREE GRAMMARS

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

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

A Language is generated by a CFG

\Leftrightarrow

It is recognized by a PDA

A Language is generated by a CFG

⇒

It is recognized by a PDA

Suppose L is generated by a CFG $G = (V, \Sigma, R, S)$

Construct $P = (Q, \Sigma, \Gamma, \delta, q, F)$ that recognizes L

Suppose L is generated by a CFG $G = (V, \Sigma, R, S)$

Construct $P = (Q, \Sigma, \Gamma, \delta, q, F)$ that recognizes L

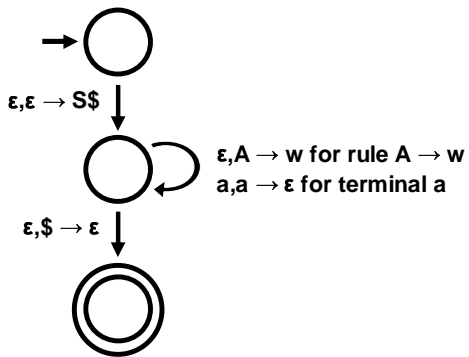
(1) Place the marker symbol \$ and the start variable on the stack

(2) Repeat the following steps forever:

(a) If top of stack is variable, non-deterministically select rule that matches the variable and push result into the stack

(b) If top of stack is terminal, read next symbol from input and compare it to terminal. If different, reject.

(c) If top of stack is \$, then accept



$S \rightarrow aTb$
 $T \rightarrow Ta \mid \epsilon$

A Language is generated by a CFG
 \Leftarrow
 It is recognized by a PDA

A Language is generated by a CFG
 \Leftarrow
 It is recognized by a PDA

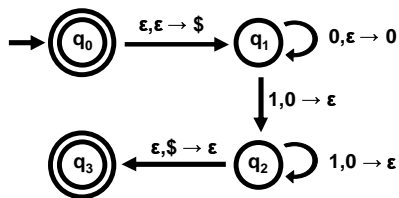
Given PDA $P = (Q, \Sigma, \Gamma, \delta, q, F)$

Construct a CFG $G = (V, \Sigma, R, S)$ such that $L(G)=L(P)$

First, simplify P to have the following form:

- (1) It has a single accept state, q_{accept}
- (2) It empties the stack before accepting
- (3) Each transition either pushes a symbol or pops a symbol, but not both at the same time

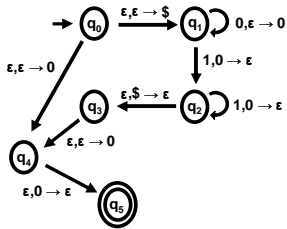
SIMPLIFY



Idea: for each pair of states p and q in P , the grammar will have a variable A_{pq} that generates all strings that can take P from p with an empty stack to q with an empty stack

$$V = \{A_{pq} \mid p, q \in Q\}$$

$$S = A_{q_0 q_{\text{accept}}}$$



What strings does A_{q_0, q_1} generate?
 What strings does A_{q_1, q_2} generate?
 What strings does A_{q_1, q_3} generate?

A_{pq} generates all strings that take p with an empty stack to q with an empty stack

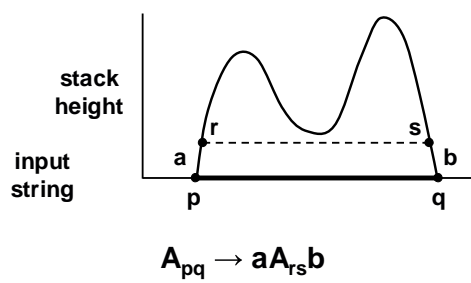
Let x be such a string

- P's first move on x must be a push
- P's last move on x must be a pop

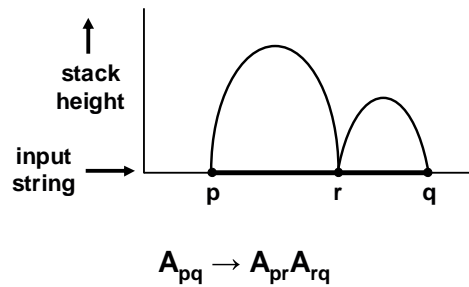
Two possibilities:

1. The symbol popped at the end is the one pushed at the beginning
2. The symbol popped at the end is not the one pushed at the beginning

1. The symbol popped at the end is the one pushed at the beginning



2. The symbol popped at the end is not the one pushed at the beginning



Formally:

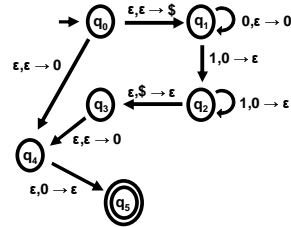
$$V = \{A_{pq} \mid p, q \in Q\}$$

$$S = A_{q_0 q_{\text{accept}}}$$

For each $p, q, r, s \in Q$, $t \in \Gamma$ and $a, b \in \Sigma_\epsilon$
 If $(r, t) \in \delta(p, a, \epsilon)$ and $(q, \epsilon) \in \delta(s, b, t)$
 Then add the rule $A_{pq} \rightarrow aA_{rs}b$

For each $p, q, r \in Q$,
 add the rule $A_{pq} \rightarrow A_{pr}A_{rq}$

For each $p \in Q$,
 add the rule $A_{pp} \rightarrow \epsilon$



What strings does $A_{q_0 q_1}$ generate?
 What strings does $A_{q_1 q_2}$ generate?
 What strings does $A_{q_1 q_3}$ generate?

A_{pq} generates x

\Rightarrow

x can bring P from p with an empty stack to q with an empty stack

Proof (by induction on the number of steps in the derivation of x from A_{pq}):

Inductive Step:

Assume true for derivations of length at most k and prove true for derivations of length $k+1$

$A_{pq} \Rightarrow x$ in $k+1$ steps

$$A_{pq} \rightarrow A_{pr}A_{rq}$$

$$A_{pq} \rightarrow aA_{rs}b$$

$$(r, t) \in \delta(p, a, \epsilon) \text{ and } (q, \epsilon) \in \delta(s, b, t)$$

state push state alphabet pop

A_{pq} generates x

\Leftarrow

x can bring P from p with an empty stack to q with an empty stack

Proof (by induction on the number of steps in the computation of P from p to q with empty stacks on input x):

Base Case: The computation has 0 steps

We must show that $A_{pp} \Rightarrow x$

But it must be that $x = \epsilon$, so we are done

Inductive Step:

Assume true for computations of length at most k and prove true for computations of length $k+1$

Suppose that P has a computation wherein x brings p to q with empty stacks in $k+1$ steps

Two cases:

1. The stack is empty only at the beginning and the end of this computation
2. The stack is empty somewhere in the middle of the computation

A Language is generated by a CFG

\Leftrightarrow

It is recognized by a PDA

Corollary: Every regular language is context-free

THE CHOMSKY NORMAL FORM

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

$A \rightarrow BC$ B and C are not start variable

$A \rightarrow a$ a is a terminal

$S \rightarrow \epsilon$ S is the start variable

$S \rightarrow 0S1$

$S \rightarrow TT$

$T \rightarrow \epsilon$

$S_0 \rightarrow S$

$T \rightarrow 0$

$U \rightarrow SV$

$S \rightarrow TU \mid \#$

$V \rightarrow 1$

Theorem: Any context-free language can be generated by a context-free grammar in Chomsky normal form

“Can transform any CFG into Chomsky normal form”

$S \rightarrow 0S1$
 $S \rightarrow T\#T$
 $S \rightarrow T$
 $T \rightarrow \epsilon$

1. Add a new start variable S_0 and add the rule $S_0 \rightarrow S$

$S_0 \rightarrow S$
 $S \rightarrow 0S1$
 $S \rightarrow T\#T$
 $S \rightarrow T$
 $T \rightarrow \epsilon$

2. Remove all $A \rightarrow \epsilon$ rules (where A is not S_0)

For each occurrence of A on right hand side of a rule, add a new rule with the occurrence deleted

If we have the rule $B \rightarrow A$, add $B \rightarrow \epsilon$, unless we have previously removed $B \rightarrow \epsilon$

3. Remove unit rules $A \rightarrow B$

Whenever $B \rightarrow w$ appears, add the rule $A \rightarrow w$ unless this was a unit rule previously removed

$S_0 \rightarrow S$
 $S \rightarrow 0S1$
 $S \rightarrow T\#T$
 $S \rightarrow T$
 $T \rightarrow \epsilon$

4. Convert all remaining rules into the proper form

$$S_0 \rightarrow 0S1$$

$$S_0 \rightarrow A_1A_2$$

$$A_1 \rightarrow 0$$

$$A_2 \rightarrow SA_3$$

$$A_3 \rightarrow 1$$

$$S_0 \rightarrow T\#$$

$$S_0 \rightarrow TA_4$$

$$A_4 \rightarrow \#$$

$$S_0 \rightarrow \epsilon$$

$$S_0 \rightarrow 0S1$$

$$S_0 \rightarrow T\#T$$

$$S_0 \rightarrow T\#$$

$$S_0 \rightarrow \#T$$

$$S_0 \rightarrow \#$$

$$S_0 \rightarrow 01$$

$$S \rightarrow 0S1$$

$$S \rightarrow T\#T$$

$$S \rightarrow T\#$$

$$S \rightarrow \#T$$

$$S \rightarrow \#$$

$$S \rightarrow 01$$

Convert the following into Chomsky normal form:

$$A \rightarrow BAB \mid B \mid \epsilon$$

$$B \rightarrow 00 \mid \epsilon$$

THE CHOMSKY NORMAL FORM

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

$$A \rightarrow BC \quad B \text{ and } C \text{ are not start variable}$$

$$A \rightarrow a \quad a \text{ is a terminal}$$

$$S \rightarrow \epsilon \quad S \text{ is the start variable}$$

Any variable A that is not the start variable can only generate strings of length > 0

Theorem: If G is in CNF, $w \in L(G)$ and $|w| > 0$, then any derivation of w in G has length $2|w| - 1$

Proof (by induction on $|w|$):

Base Case: If $|w| = 1$, then any derivation of w must have length 1

Inductive Step: Assume true for any string of length at most $k \geq 1$, and let $|w| = k+1$

Since $|w| > 1$, derivation starts with $S \rightarrow AB$

So $w = xy$ where $A \Rightarrow x$, $|x| > 0$ and $B \Rightarrow y$, $|y| > 0$

By the inductive hypothesis, the length of any derivation of w must be

$$1 + (2|x| - 1) + (2|y| - 1) = 2(|x| + |y|) - 1$$

WWW.FLAC.WS