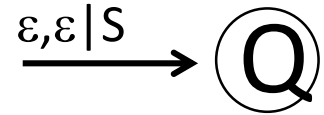


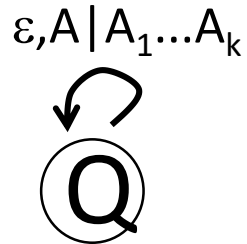
Equivalence of PDAs and Grammars

Theorem: Every language described by a context-free grammar is accepted by a PDA.

Construction: Start with a grammar for the language, where S is the start symbol. Make a start state Q for the DFA and begin

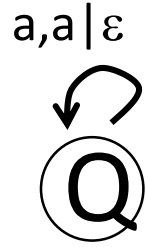


For each grammar rule $A \Rightarrow A_1 \dots A_k$ add transition



i.e. push A_k , then A_{k-1} , etc., finally pushing A_1 .

Construction continued: For each terminal symbol a in Σ add the transition



This completes the construction. Note that the DFA has only one state. It accepts by empty stack.

Example:

$E \Rightarrow E+T \mid T$

$T \Rightarrow T^*F \mid F$

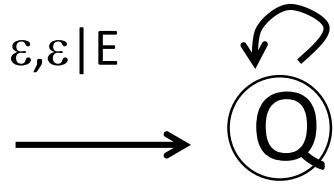
$F \Rightarrow F \text{ digit} \mid \text{digit}$

$+, + \mid \epsilon \text{ etc.}$

$0, 0 \mid \epsilon$

$\epsilon, E \mid T \text{ etc.}$

$\epsilon, E \mid E+T$



Following is a configuration analysis that shows this DFA accepts $3+4^*5$

$(Q, 3+4*5, E) \Rightarrow (Q, 3+4*5, E+T)$
 $\Rightarrow (Q, 3+4*5, T+T)$
 $\Rightarrow (Q, 3+4*5, F+T)$
 $\Rightarrow (Q, 3+4*5, 3+T)$
 $\Rightarrow (Q, +4*5, +T)$
 $\Rightarrow (Q, 4*5, T)$
 $\Rightarrow (Q, 4*5, T*F)$
 $\Rightarrow (Q, 4*5, F*F)$
 $\Rightarrow (Q, 4*5, 4*F)$
 $\Rightarrow (Q, *5, *F)$
 $\Rightarrow (Q, 5, F)$
 $\Rightarrow (Q, 5, 5)$
 $\Rightarrow (Q, \varepsilon, \varepsilon) \text{ accept}$

Now, how do we know this PDA accepts the language generated by the grammar?

Suppose string w is generated by the grammar. Then there is a derivation of w that always expands the left-most nonterminal symbol:

$$E \Rightarrow \underline{E}+T$$

$$\Rightarrow \underline{T}+T$$

$$\Rightarrow \underline{F}+T$$

etc.

At each step i let A_i be the left-most nonterminal, α_i everything to its left, and β_i everything to its right so the phrase that has been derived is $\alpha_i A_i \beta_i$ and all of the symbols in α_i are terminal.

The automaton has been constructed so that at step i of the automaton computation the stack will be $A_i\beta_i$ and the α_i symbols of the input will have been consumed. In other words, an easy induction shows that

$$(Q, w, S) \xRightarrow{*} (Q, w - \alpha_i, A_i\beta_i)$$

So eventually $(Q, w, S) \xRightarrow{*} (Q, \varepsilon, \varepsilon)$ and the automaton accepts w .

On the other hand, suppose that for a nonterminal symbol A

$(Q, w, A) \xRightarrow{*} (Q, \varepsilon, \varepsilon)$. We will show by induction that there is a grammar derivation of w from symbol A . The induction is on the number of moves made by the automaton.

Base case: There must be a grammar rule $A \Rightarrow a$ and $w = a$.

Inductive case: Suppose this is true for all strings accepted by the PDA in n moves and the PDA accepts w in $n+1$ moves.

Since the configuration (Q, w, A) starts with a nonterminal at the top of the stack the first move must be using a rule $A \Rightarrow X_1..X_k$. For each i let w_i be the string of input needed to remove X_i from the stack, i.e.,

$$(Q, w_i, X_i) \xRightarrow{*} (Q, \varepsilon, \varepsilon)$$

By induction $X_i \xRightarrow{*} w_i$.

Altogether $A \Rightarrow X_1..X_k \stackrel{*}{\Rightarrow} w_1..w_k = w$. So if the automaton accepts w the grammar derives w .

Theorem (Chomsky): Given a PDA that accepts by empty stack, we can find a context free grammar that generates the set of strings accepted by the PDA.

Construction: This builds a huge grammar whose derivations mimic the configurations of the PDA.

Step 1. The nonterminal symbols of the grammar are a new start symbol S and all symbols of the form $[pXq]$ where p and q are states of the PDA and X is any one stack symbol

$[pXq]$ will generate all strings w so that $(p, w, X) \xRightarrow{*} (q, \varepsilon, \varepsilon)$

i.e., all strings w that take the PDA from state p to state q while popping X off the stack.

Step 2. Grammar rules

Rule 1: If Q is the start state of the PDA and Z_0 is the stack bottom symbol then for every state p add the grammar rule

$$S \Rightarrow [QZ_0p]$$

i.e., S will generate all strings that take the PDA from Q to any other state while emptying the stack.

Rule 2: Suppose the PDA has transition

$$\textcircled{q} \xrightarrow{a, X | Y_1 \dots Y_k} \textcircled{r}$$

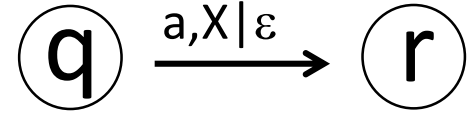
Then for every sequence of k states $r_1 \dots r_k$ add the rule

$$[qXr_k] \Rightarrow a[rY_1r_1][r_1Y_2r_2] \dots [r_{k-1}Y_kr_k]$$

i.e., the strings that take the PDA from q to r_k while removing X from the stack include those that

1. first consume a and move from q to r
2. then consume anything generated by $[rY_1r_1]$
3. then consume anything generated by $[r_1Y_2r_2]$
4. etc.

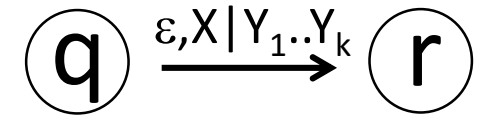
Rule 3: If there is a transition



then add the rule

$$[qXr] \Rightarrow a$$

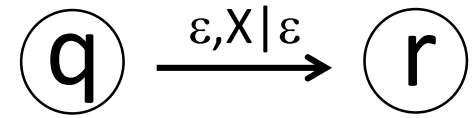
Rule 4: If there is a transition



then for any sequence of states $r_1 \dots r_k$ add the rule

$$[qXr_k] \Rightarrow [rY_1r_1][r_1Y_2r_2] \dots [r_{k-1}Y_kr_k]$$

Rule 5: If there is a transition

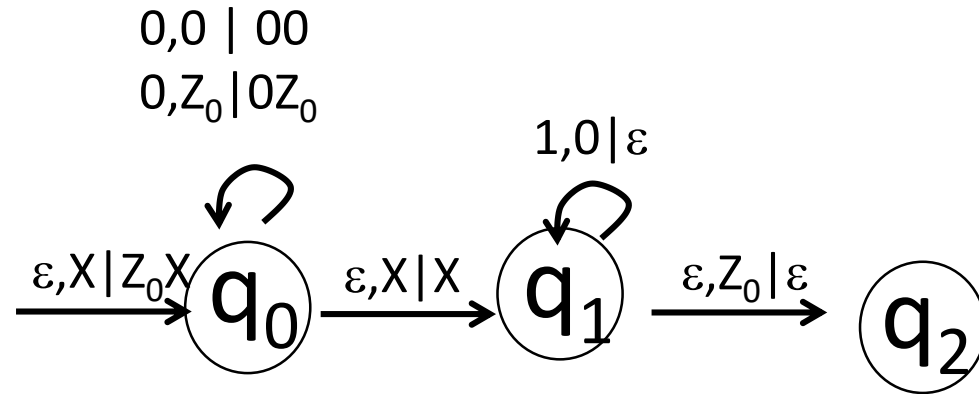


then add the rule

$$[qXr] \Rightarrow \epsilon$$

This is the complete construction.

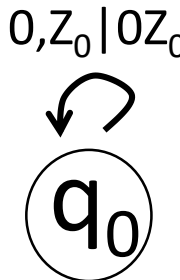
Example: The following automaton accepts $\{0^n 1^n \mid n \geq 0\}$ by empty stack



Here is a derivation of 0011 with the constructed grammar:

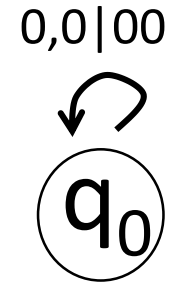
$S \Rightarrow \underline{[q_0 Z_0 q_2]}$ Rule 1 with $p=q_2$ since Z_0 is popped at q_2 .

$\Rightarrow 0 \underline{[q_0 0 q_1]} [q_1 Z_0 q_2]$ Rule 2 with $q, r=q_0, r_1=q_1, r_2=q_2$



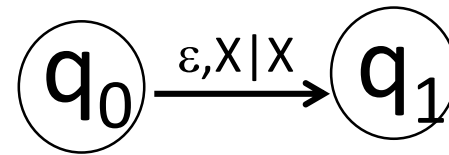
$\Rightarrow 00[\underline{q_0}0\underline{q_1}][q_10q_1][q_1Z_0q_2]$

Rule 2 with
 $q, r = q_0,$
 $r_1 = r_2 = q_1$

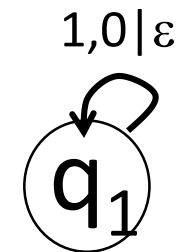


$\Rightarrow 00[\underline{q_1}0\underline{q_1}][\underline{q_1}0\underline{q_1}][q_1Z_0q_2]$

Rule 4 with
 $r = q_1 = r_1$



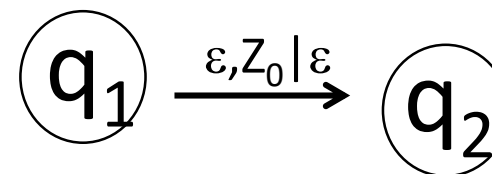
$\Rightarrow 0011[q_1 Z_0 q_2]$



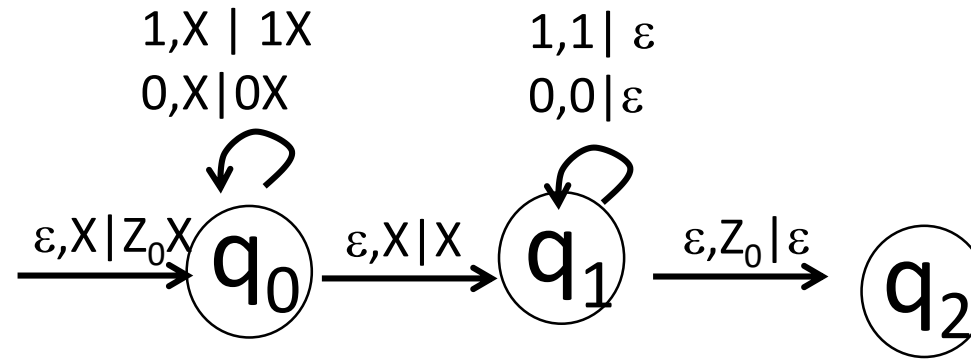
Rule 3 twice with

$\Rightarrow 0011$

Rule 5 with



Another example



This accepts by empty stack $\{ww^{\text{rev}} \mid w \in (0+1)^*\}$

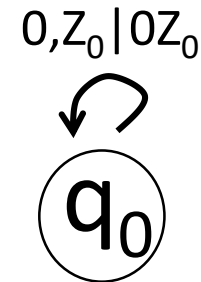
We will derive 0110 from the constructed grammar.

$$S \Rightarrow \underline{[q_0 Z_0 q_2]} \quad \text{Rule 1}$$

$$\Rightarrow 0 \underline{[q_0 0 q_1]} [q_1 Z_0 q_2]$$

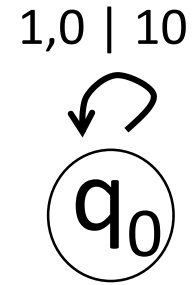
Rule 2 with

$$r = q_0, r_1 = q_1, r_2 = q_2$$



$\Rightarrow 01[\underline{q_0}1\underline{q_1}][q_10q_1][q_1Z_0q_2]$

Rule 2 with
 $r=q_0, r_1=q_1, r_2=q_1$



$\Rightarrow 01[\underline{q_1}1\underline{q_1}][\underline{q_1}0\underline{q_1}][q_1Z_0q_2]$

Rule 4 with
 $r=q_1, r_1=q_1$

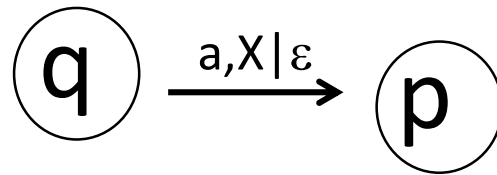
$\Rightarrow 0110[\underline{q_1}Z_0\underline{q_2}]$ Rule 3 twice

$\Rightarrow 0110$ Rule 5

Lemma 1: If string w can take the PDA from state q to state p while popping X off the stack then $[qXp] \stackrel{*}{\Rightarrow} w$. As a consequence, if w is accepted by the PDA it is generated by the grammar.

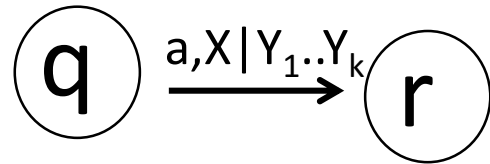
Proof of Lemma 1: Induction on the number of steps the PDA takes to transform configuration (q, w, X) to $(p, \varepsilon, \varepsilon)$

Base case: 1 step. The step must be $(q, w, X) \Rightarrow (p, \varepsilon, \varepsilon)$ so the PDA must have a transition



This means the grammar has a rule $[qXp] \Rightarrow a$ (Rule 3)

Inductive case: Suppose the lemma is true for all strings w that take n or fewer steps in the configuration computation, and w takes $n+1$ steps. The first step must use a transition of the form



By Rule 2 the grammar will have a rule of the form

(*) $[qXp] \Rightarrow a[rY_1r_1][r_1Y_2r_2] \dots [r_{k-1}Y_kp]$ for any sequence (r_i) of states.

Let w_i be the input that pops Y_i off the stack; let r_i be the state where this is completed.

By the inductive hypothesis we must have (**) $[r_{i-1}Y_i r_i] \stackrel{*}{\Rightarrow} w_i$

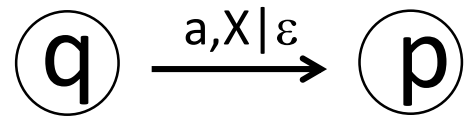
Putting (*) and (**) together we have

$$[qXp] \stackrel{*}{\Rightarrow} aw_1w_2\dots w_k=w$$

Lemma 2: If $[qXp] \xRightarrow{*} w$ then $(q, w, X) \Rightarrow^* (p, \varepsilon, \varepsilon)$. As a consequence, if a string is generated by the grammar it is accepted by the PDA.

Proof of Lemma 2: We do induction on the number of steps in the grammar derivation $[qXp] \xRightarrow{*} w$.

Base case: 1 step. There must be a rule $[qXp] \Rightarrow a$, so it must come from a transition



So $(q, a, X) \Rightarrow^* (p, \varepsilon, \varepsilon)$

Inductive case: Suppose this is true of all derivations of n or fewer steps and we have one with $n+1$ steps.

The first step must have the form $[qXp] \Rightarrow a[r_1 y_1 r_1][r_1 Y_2 r_2] \dots [r_{k-1} Y_k p]$

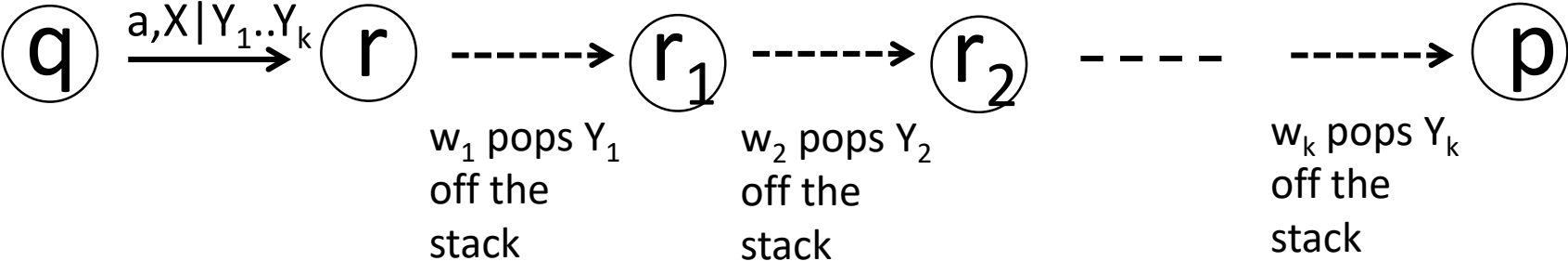
For this to be a grammar rule the PDA must have a transition

$$\textcircled{q} \xrightarrow{a, X | Y_1 \dots Y_k} \textcircled{p}$$

Each $[r_{i-1} Y_i r_i]$ symbol must generate a string of terminal symbols; call this string w_i .

By induction $(r_{i-1}, w_i, y_i) \stackrel{*}{\Rightarrow} (r_i, \varepsilon, \varepsilon)$

In other words the automaton goes through a series of transitions:



i.e., $aw_1w_2\dots w_k$ takes the automaton from q to p while popping X off the stack.