Your textbook (Sipser) states, in Lemma 2.21, that any context-free grammar (CFG) can be converted into an equivalent pushdown automaton (PDA). The proof given there takes a CFG \(G\) and constructs a certain “3-state” PDA* \(M\), and gives intuition for why \(L(M) = L(G)\). The textbook stops short of giving a full formal proof, though. Here is a formal proof.

**Theorem:** For the PDA \(M\) constructed in the textbook (Figure 2.24), we have \(L(M) = L(G)\).

**Proof:** First, we introduce some notation. For \(y \in \Sigma^*\) and \(\gamma \in (V \cup \Sigma)^*\), we let \(M[y, \gamma]\) denote the statement “\(M\) can be in state \(q_{\text{loop}}\), having read the prefix \(y\) of the input string, and with \(\gamma\$\) on its stack.” Note that \(M[x, \varepsilon]\) iff \(M\) can make the transition to \(q_{\text{accept}}\) after reading \(x\), i.e., iff \(x \in L(M)\).

**Part 1:** \(L(G) \subseteq L(M)\): Suppose \(x \in L(G)\). Then \(S \Rightarrow x\) in \(n\) steps for some positive integer \(n\), via a leftmost derivation. Let \(S = s_0 \Rightarrow s_1 \Rightarrow s_2 \Rightarrow \cdots \Rightarrow s_n = x\) be such a leftmost derivation. Suppose

\[
\begin{align*}
\pi_i &= y_i A_i \gamma_i, \\
\text{where} \quad y_i &\in \Sigma^*, A_i \in V, \text{ and } \gamma_i \in (V \cup \Sigma)^*, \text{ for } 0 \leq i < n, \\
\text{and} \quad y_n &= x, A_n = \gamma_n = \varepsilon.
\end{align*}
\]

In other words, \(A_i\) denotes the leftmost variable in \(\pi_i\) (or \(\varepsilon\), in the case \(i = n\) when \(\pi_i\) has no variables). We claim that \(M[y_i, A_i \gamma_i]\) for all \(i, 0 \leq i \leq n\). In particular, this proves that \(M[x, \varepsilon]\), i.e., that \(x \in L(M)\). The proof of the claim is by induction on \(i\).

The base case is \(i = 0\). The transition out of \(q_{\text{start}}\) shows that \(M\) can be in state \(q_{\text{loop}}\) having read no input and with \(S\$\) on its stack, i.e., \(M[\varepsilon, S]\). Note that \(y_0 = \gamma_0 = \varepsilon\) and \(A_0 = S\); therefore \(M[y_0, A_0 \gamma_0]\).

For the induction step, suppose we have shown \(M[y_i, A_i \gamma_i]\), for some \(i\) with \(0 \leq i < n\). The derivation step \(\pi_i \Rightarrow \pi_{i+1}\) must expand the leftmost variable in \(\pi_i\), i.e., \(A_i\). Let \(\pi_{i+1} = A_i \alpha_i \gamma_i\) be the CFG rule used in this step. Then

\[
y_{i+1} A_{i+1} \gamma_{i+1} = \pi_{i+1} = y_i A_i \gamma_i.
\]

Since \(y_i\) is a prefix of \(y_{i+1}\), we can write \(A_i \gamma_i = z_i A_{i+1} \gamma_{i+1}\) for some \(z_i \in \Sigma^*\) (note, in particular, that this continues to hold even if \(i+1 = n\)). This implies \(y_{i+1} = y_i z_i\). Since \(M\) has a loop transition at state \(q_{\text{loop}}\) that can pop \(A_i\) and push \(\alpha_i\), we have \(M[y_i, \alpha_i \gamma_i]\), i.e., \(M[y_i, z_i A_{i+1} \gamma_{i+1}]\). Finally, since \(M\) has a loop transition at \(q_{\text{loop}}\) that can read any input character \(a \in \Sigma\) while popping \(a\) off the stack, and since \(y_i z_i = y_{i+1}\) is a prefix of the input \(x\), we have \(M[y_{i+1}, A_{i+1} \gamma_{i+1}]\), i.e., \(M[y_{i+1}, A_{i+1} \gamma_{i+1}]\). This completes the induction step and the proof of Part 1.

**Part 2:** \(L(M) \subseteq L(G)\): The proof of this is similar to the proof in Part 1. The details are left to you as an exercise. (It’s good practice; please try writing out the details.)

*In fact, the number of states could be much greater than 3, once we unroll the shorthand notation that allows us to push multiple symbols on the stack in a single move.*