

Theorem. If $L = L(M)$ for some PDA M then there (effectively) exists a PDA M' such that $L = N(M')$.

Proof.

Theorem. If $L = L(M)$ for some PDA M then there (effectively) exists a PDA M' such that $L = N(M')$.

Proof. Idea: M accepts L by final state. We construct a PDA M' that simulates M , with the additional option that when M enters a final state, M' may enter a special **erase the stack** -state q_e and remove all symbols from the stack. Then M' accepts w by empty stack if M entered a final state.

Theorem. If $L = L(M)$ for some PDA M then there (effectively) exists a PDA M' such that $L = N(M')$.

Proof. Idea: M accepts L by final state. We construct a PDA M' that simulates M , with the additional option that when M enters a final state, M' may enter a special **erase the stack** -state q_e and remove all symbols from the stack. Then M' accepts w by empty stack if M entered a final state.

But: We have to be careful to make sure that M' does not accept a word accidentally if M empties the stack without entering an accepting state. We prevent this by inserting a **new bottom of the stack symbol** X_0 below the old start symbol Z_0 . Even if M empties the stack, M' will have the symbol X_0 in the stack. We have a new initial state q'_0 that simply places Z_0 above X_0 and starts the simulation of M .

If M empties the stack without entering the final state then it is stuck. Corresponding computation in M' gets stuck with X_0 on the stack.

Theorem. If $L = N(M)$ for some PDA M then there (effectively) exists a PDA M' such that $L = L(M')$.

Proof.

Theorem. If $L = N(M)$ for some PDA M then there (effectively) exists a PDA M' such that $L = L(M')$.

Proof. Idea: Now M recognizes L by empty stack. We construct a PDA M' that simulates M and detects when the stack is empty. When that happens the PDA enters a final state. In order to be able to detect the empty stack and change the state, we again introduce a new **bottom of the stack symbol** X_0 . As soon as X_0 is revealed the new PDA enters a new final state q_f .

Example. Let us modify our first sample PDA so that it recognizes the language

$$L = \{a^n b^n \mid n \geq 1\}$$

using final states instead of the empty stack.

Next we prove that **PDA recognize all context-free languages**. In fact, all context-free languages are recognized by PDA having only **one state**. We prove this by showing how any context-free grammar can be simulated using only the stack.

Example. Consider the grammar $G = (V, T, P, S)$ where $V = \{S\}$, $T = \{a, b\}$ and P contains productions

$$S \longrightarrow aSbS \mid bSaS \mid \varepsilon.$$

Let us construct a PDA M that recognizes $L(G)$ using empty stack.

Example. Consider the grammar $G = (V, T, P, S)$ where $V = \{S\}$, $T = \{a, b\}$ and P contains productions

$$S \longrightarrow aSbS \mid bSaS \mid \varepsilon.$$

Let us construct a PDA M that recognizes $L(G)$ using empty stack.

The stack symbols of M are the terminals and variables of G . At all times, the content of the stack is a suffix of a sentential form by G , from which the remaining input can be generated.

Initially the stack contains the start symbol of the grammar.

- If the topmost symbol of the stack is a terminal symbol it is compared against next input letter. If they are identical, the symbol is popped from the stack and the PDA moves to the next input letter. If they are different the simulation halts.
- If the topmost symbol of the stack is a variable the PDA replaces it by a right-hand-side of a production using an ε -move.

Theorem. If $L = L(G)$ for some context-free grammar G then there (effectively) exists a PDA M such that $L = N(M)$.

Proof.