Theorem 4.6. *Let Σ be an alphabet, and let L be a language over L. Then L is context-free if and only if there is a pushdown automaton whose input alphabet is Σ such that $L = L(M)$.*

- construction of a pushdown automaton for *L*

Suppose that L is a context-free language over an alphabet Σ. Let $G = (V, \Sigma, P, S)$ be a context-free grammar for L. Then we can construct a pushdown automaton M that accepts L. In fact, we can take $M = (Q, \Sigma, \Lambda, q_0, \partial, F)$ where $Q = \{q_0, q_1\}$, $\Lambda = \Sigma \cup V$, $F = \{q_1\}$, and ∂ contains transitions of the forms

1. $((q_0, \varepsilon, \varepsilon), (q_1, S))$; — push the start symbol on the stack
2. $((q_1, \sigma, \sigma), (q_1, \varepsilon))$, for $\sigma \in \Sigma$; and — read a terminal symbol, matching it with the top of the stack
3. $((q_1, \varepsilon, A), (q_1, x))$, for each production $A \longrightarrow x$ in G. — replace a non-terminal on the top of the stack with the right side of the production

1. $((q_0, \varepsilon, \varepsilon), (q_1, S))$; — push the start symbol on the stack
2. $((q_1, \sigma, \sigma), (q_1, \varepsilon))$, for $\sigma \in \Sigma$; and — read a terminal symbol, matching it with the top of the stack
3. $((q_1, \varepsilon, A), (q_1, x))$, for each production $A \longrightarrow x$ in G. — replace a non-terminal on the top of the stack with the right side of the production

$S \longrightarrow AB$
$A \longrightarrow aAb$
$A \longrightarrow \varepsilon$
$B \longrightarrow bB$
$B \longrightarrow b$

$((q_0, \varepsilon, \varepsilon), (q_1, S))$
$((q_1, a, a), (q_1, \varepsilon))$
$((q_1, b, b), (q_1, \varepsilon))$
$((q_1, \varepsilon, S), (q_1, AB))$
$((q_1, \varepsilon, A), (q_1, aAb))$
$((q_1, \varepsilon, A), (q_1, \varepsilon))$
$((q_1, \varepsilon, B), (q_1, bB))$
$((q_1, \varepsilon, B), (q_1, b))$

input: aabbbb

Transition	Input Consumed	Stack (← top)	Derivation
$((q_0, \varepsilon, \varepsilon), (q_1, S))$		S	
$((q_1, \varepsilon, S), (q_1, AB))$		AB	$S \Longrightarrow AB$
$((q_1, \varepsilon, A), (q_1, aAb))$		$aAbB$	$\Longrightarrow aAbB$
$((q_1, a, a), (q_1, \varepsilon))$	a	AbB	
$((q_1, \varepsilon, A), (q_1, aAb))$	a	$aAbbB$	$\Longrightarrow aaAbbB$
$((q_1, a, a), (q_1, \varepsilon))$	aa	$AbbB$	
$((q_1, \varepsilon, A), (q_1, \varepsilon))$	aa	bbB	$\Longrightarrow aabbB$
$((q_1, b, b), (q_1, \varepsilon))$	aab	bB	
$((q_1, b, b), (q_1, \varepsilon))$	$aabb$	B	
$((q_1, \varepsilon, B), (q_1, bB))$	$aabb$	bB	$\Longrightarrow aabbbB$
$((q_1, b), (q_1, b, \varepsilon))$	$aabbb$	B	
$((q_1, \varepsilon, B), (q_1, b))$	$aabbb$	b	$\Longrightarrow aabbbb$
$((q_1, b, b), (q_1, \varepsilon))$	$aabbbb$		

7. Let $M = (Q, \Sigma, \Lambda, q_0, \partial, F)$ be a pushdown automaton. Define $L'(M)$ to be the language $L'(M) = \{w \in \Sigma^* \mid$ it is possible for M to start in state q_0, read all of w, and end in an accepting state$\}$. $L'(M)$ differs from $L(M)$ in that for $w \in L'(M)$, we do not require that the stack be empty at the end of the computation.

a) Show that there is a pushdown automaton M' such that $L(M') = L'(M)$.

b) Show that a language L is context-free if and only if there is a pushdown automaton M such that $L = L'(M)$.

c) Identify the language $L'(M)$ for each of the automata in Exercise 1.

8. Let L be a regular language over an alphabet Σ, and let K be a context-free language over the same alphabet. Let $M = (Q, \Sigma, q_0, \delta, F)$ be a DFA that accepts L, and let $N = (P, \Sigma, \Lambda, p_0, \partial, E))$ be a pushdown automaton that accepts K. Show that the language $L \cap K$ is context-free by constructing a pushdown automaton that accepts $L \cap K$. The pushdown automaton can be constructed as a "cross product" of M and N in which the set of states is $Q \times P$. The construction is analogous to the proof that the intersection of two regular languages is regular, as outlined in Exercise 3.6.7.

Non-Context-Free Languages

- just like there are languages that aren't regular, there are languages that aren't context-free
- there is a pumping lemma for context-free languages similar to the one for regular languages
  - idea for regular languages
    - a DFA/NFA only has a finite number of states, so for any sufficiently long string, a state must be repeated – that provides a loop which can be "pumped" to generate a set of strings, all of which are in the language
  - idea for context-free languages
    - a context-free grammar only has a finite number of non-terminals, so for any sufficiently long string, there's a root-to-leaf path in the parse tree where a non-terminal is repeated – that provides a loop which can be "pumped" to generate a set of strings, all of which are in the language
- the intersection of two context-free languages is not necessarily context-free