Theorem:
- PDA’s and CFG’s are equivalent

Proof of theorem (part 1):
- Suppose $G = (\Sigma, \mathcal{V}, S, \mathcal{R})$ is a CFG
- PDA for $L(G)$:
Example:

\[
E \rightarrow E + T \mid T \\
T \rightarrow T * F \mid F \\
F \rightarrow (E) \mid a \mid \cdots \mid z
\]

Leftmost derivation for \(a + b * c\):

\[
E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \\
\Rightarrow a + T * F \Rightarrow a + F * F \Rightarrow a + b * F \Rightarrow a + b * c
\]

<table>
<thead>
<tr>
<th>consumed input</th>
<th>stack</th>
<th>consumed input</th>
<th>stack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(E \ $)</td>
<td>(a +)</td>
<td>(T * F \ $)</td>
</tr>
<tr>
<td></td>
<td>(E + T \ $)</td>
<td>(a +)</td>
<td>(F * F \ $)</td>
</tr>
<tr>
<td></td>
<td>(T + T \ $)</td>
<td>(a +)</td>
<td>(b * F \ $)</td>
</tr>
<tr>
<td></td>
<td>(F + T \ $)</td>
<td>(a + b)</td>
<td>(* F \ $)</td>
</tr>
<tr>
<td></td>
<td>(a + T \ $)</td>
<td>(a + b *)</td>
<td>(F \ $)</td>
</tr>
<tr>
<td>(a)</td>
<td>+ (T \ $)</td>
<td>(a + b *)</td>
<td>(c \ $)</td>
</tr>
<tr>
<td>(a +)</td>
<td>(T \ $)</td>
<td>(a + b * c)</td>
<td>($)</td>
</tr>
</tbody>
</table>
Proof of theorem (part 2):

- Let $M$ be a PDA
- We want to construct an equivalent CFG
- Simplifying assumptions:
  1. $M$ has a single accept state $q_{\text{accept}}$
  2. $M$ empties its stack before accepting:
     \[
     \chi \in L(M) \iff (q_0, \chi, \varepsilon) \vdash^* (q_{\text{accept}}, \varepsilon, \varepsilon)
     \]
  3. Each transition either pushes or pops a single symbol (but not both)
Proof (cont’d):

- Idea: dynamic programming
- For all states $p, q$, define a variable $A_{pq}$ with the property that for all $x \in \Sigma^*$:
  \[ A_{pq} \Rightarrow^* x \iff (p, x, \epsilon) \vdash^* (q, \epsilon, \epsilon) \]
- Start symbol: $A_{q_0 q_{\text{accept}}}$
- Rules:
  - for each state $p$: $A_{pp} \to \epsilon$
  - for all states $p, q, r$: $A_{pq} \to A_{pr}A_{rq}$
  - for all transitions
    $[p \to r : \text{read } u, \text{push } t], [s \to q : \text{read } v, \text{pop } t]$, where $p, q, r, s$ are states, $u, v \in \Sigma^*$, and $t$ is a stack symbol, add the rule:
    \[ A_{pq} \to uA_{rs}v \]
Proof (cont’d):

- **Claim 1:** if $A_{pq} \Rightarrow^* x$, then $(p, x, \varepsilon) \vdash^* (q, \varepsilon, \varepsilon)$
  
  Easy induction on size of parse tree

- **Claim 2:** if $(p, x, \varepsilon) \vdash^* (q, \varepsilon, \varepsilon)$, then $A_{pq} \Rightarrow^* x$
  
  Induction on length $n$ of the computation:
  - $n = 0$: use the rule $A_{pp} \rightarrow \varepsilon$
  - $n > 0$ and the computation enters an intermediate configuration with an empty stack: use the appropriate rule $A_{pq} \rightarrow A_{pr}A_{rq}$
  - $n > 0$ and the stack never empties: use the appropriate rule $A_{pq} \rightarrow uA_{rs}v$
Definition:

- A language is *context free* if it is generated by some CFG (or equivalently, is recognized by some PDA)

Theorem (Pumping Lemma for CFL’s):

- Let $A$ be a context-free language
- $\exists p \in \mathbb{Z}_{>0}$ $\forall s \in A$ with $|s| \geq p$ $\exists u, v, w, x, y \in \Sigma^*$:
  0. $s = uvwxy$
  1. $|vwx| \leq p$
  2. $|vx| > 0$
  3. $uv^kwx^ky \in A$ for all $k \in \mathbb{Z}_{k \geq 0}$
Proof:

- Let \((\Sigma, \mathcal{V}, S, \mathcal{R})\) be a CFG for \(A\)
- Let \(B \geq 2\) be an upper bound on the length of the RHS of any rule
- Fact: for \(s \in A\), if \(T\) is a parse tree for \(s\) of height \(h\), then \(|s| \leq B^h\)
- Set \(p := B^{|\mathcal{V}|+1}\)
- Now let \(s \in A\) with \(|s| \geq p\)
- Choose a parse tree \(T\) for \(s\), and choose \(T\) so that there is no smaller parse tree for \(T\)
- Let \(h := \text{height of } T\)
Proof (cont’d):

- We have $B^{|\forall|+1} = p \leq |s| \leq B^h$,
- Thus, $h \geq |\forall| + 1$
- Consider a path of length $h$ in $T$
- This path contains $h + 1$ nodes, of which $h$ are variables
- As $h \geq |\forall| + 1$, some variable must repeat
- Following the path from leaf to root, let $A$ be the first variable that repeats
Proof (cont’d):

- We must have $v \neq \epsilon$ or $x \neq \epsilon$; otherwise, we would get a smaller parse tree for $s$, contradicting the minimality of $T$: 

```
A
A
A
```

```
A
```
Proof (cont’d)

- We must also have $|vwx| \leq p$
  
The height of the subtree for $vwx$ is $\leq |\mathcal{V}| + 1$, and so $|vwx| \leq B^{|\mathcal{V}|+1} = p$

- Finally, we can “pump down” or “pump up”, as desired:
Example:

- Let \( A = \{0^n \# 0^n \# 0^n : n \geq 0\} \)
- To apply pumping lemma, assume \( A \) is CF, and let \( p \) be the “pump length”
- Let \( s = 0^p \# 0^p \# 0^p \in A \)
- We have to show that no matter how we split \( s \) up as \( s = uvwxy \), with \( |vx| > 0 \) and \( |vwx| \leq p \), we can pump \( v \) and \( x \) to get a string outside of \( A \)
- The point is, since \( |vwx| \leq p \), the “pump handles” \( v \) and \( w \) can touch at most two of the three “0 regions”
- So by pumping, we will either throw the 0’s out of balance, or we’ll get the wrong number of \#’s
CF and regular languages:

- Every regular language is context free

Closure properties:

- If $A$ and $B$ are context free, then so are $A \cup B$, $AB$, and $A^*$ (the “regular operations”)
- CFL’s are not closed under intersection and complement

Example:

- Let $A := \{0^n \# 0^n \# 0^m : n, m \geq 0\}$
- Let $B := \{0^n \# 0^m \# 0^m : n, m \geq 0\}$
- $A$ and $B$ are CF, while $A \cap B = \{0^n \# 0^n \# 0^n : n \geq 0\}$ is not
Fact: if $A$ is context free, and $R$ is regular, then $A \cap R$ is context free

- Let $M$ be a PDA for $A$
- Let $D$ be a DFA for $R$
- Construct a PDA for $A \cap R$, whose state space is the Cartesian product of the state spaces of $M$ and $D$
Efficiently recognizing CFL’s:

- Let $G = (\Sigma, \mathcal{V}, S, \mathcal{R})$ be a CFG, and let $x \in \Sigma^*$
- We want to test if $x \in L(G)$
- Write $x = x[1..n]$, where $n = |x|$
- Assume $G$ has the following form:
  - for every $A \in \mathcal{V}$, there is exactly one rule, and it is of one of the following three forms:
    
    $A \rightarrow B_1 \mid B_2$ \hspace{1em} ($B_1, B_2 \in \mathcal{V}$)
    
    $A \rightarrow B_1B_2$ \hspace{1em} ($B_1, B_2 \in \mathcal{V}$)
    
    $A \rightarrow u$ \hspace{1em} ($u \in \Sigma^*$)

- Any grammar can be easily transformed into a grammar of this form, with only a constant blow-up in total size.
Recognizing CFL’s (cont’d):

- Construct and AND/OR graph with vertices
  \[(A, i, j) \quad A \in \mathcal{V}, \quad i = 1 \ldots n + 1, \quad j = i - 1 \ldots n\]

- Meaning: \((A, i, j) \iff A \Rightarrow^* x[i..j]\)

- \(A \to B_1 \mid B_2:\)
  \[A \Rightarrow^* x[i..j] \iff (B_1 \Rightarrow^* x[i..j]) \lor (B_2 \Rightarrow^* x[i..j])\]

So \((A, i, j)\) is an OR-node, with two successors, \((B_1, i, j)\) and \((B_2, i, j)\)
Recognizing CFL’s (cont’d):

- $A \rightarrow B_1B_2$:

\[
A \Rightarrow^* x[i..j] \iff \\
\exists k = i..j + 1 \left[ (B_1 \Rightarrow^* x[i..k - 1]) \land (B_2 \Rightarrow^* x[k..j]) \right]
\]

$(A, i, j)$ is an OR-node, whose successors are auxiliary nodes $(A, i, k, j)$, for $k = i..j + 1$

Each auxiliary node $(A, i, k, j)$ is an AND-node, with two successors, $(B_1, i, k - 1)$ and $(B_2, k, j)$
Recognizing CFL’s (cont’d):

- $A \rightarrow u$:
  
  $A \Rightarrow^* x[i..j] \iff x[i..j] = u$

  $(A, i, j)$ is a CONSTANT-node

- # of vertices = $O(|V|n^3)$
- # of edges = $O(|V|n^3)$

- Need to find a least fixed point

- Total time: $O(|G|n^3)$, where $|G|$ is the total size of the original grammar (without simplifications)