We began to show CFL = PDA

**Theorem 1.** Every context-free language is accepted by some PDA.

**Theorem 2.** For every PDA M, the language L(M) is context-free.

We showed how a PDA could be constructed from a CFL. Given a CFG \( G = (V, T, P, S) \), we define a PDA \( M = (\{q\}, T, T \cup V, \delta, q, S) \), with \( \delta \) given by

- If \( A \in V \), then \( \delta(q, \Lambda, A) = \{ (q, \alpha) \mid A \rightarrow \alpha \text{ is in } P \} \)
- If \( a \in T \), then \( \delta(q, a, a) = \{ (q, \Lambda) \} \)

1. The stack symbols of the new PDA contain all the terminal and non-terminals of the CFG
2. There is only 1 state in the new PDA
3. Add transitions on \( \Lambda \), one for each production
4. Add transitions on \( a \in T \), one for each terminal.
Transitions simulate left-most derivation

\[ S \Rightarrow SS \Rightarrow (S)S \Rightarrow ((S))S \Rightarrow (((S))(S) \Rightarrow (((S))(S)) \Rightarrow (((S))(S)) \]

\[ \begin{align*}
1. \quad & \delta(q, \Lambda, S) = (q, SS) & S \rightarrow SS \\
2. \quad & \delta(q, \Lambda, S) = (q, (S)) & S \rightarrow (S) \\
3. \quad & \delta(q, \Lambda, S) = (q, \Lambda) & S \rightarrow \Lambda \\
4. \quad & \delta(q, (, ( ) = (q, \Lambda) \\
5. \quad & \delta(q, ), ) ) = (q, \Lambda)
\end{align*} \]

Note there is an entry in \( \delta \) for each terminal and non-terminal symbol. The stack operations mimic a top down parse, replacing non-terminals with the rhs of a production.
To prove that every string of \(L(G)\) is accepted by the PDA \(M\), prove the following more general fact:

If \(S \Rightarrow_{\text{left-most}} \alpha\) then \((q,uv,S) \vdash^* (q,v,\beta)\)

where \(\alpha = u\beta\) is the “leftmost factorization” of \(\alpha\) (\(u\) is the longest prefix of \(\alpha\) that belongs to \(T^*\), i.e. all terminals).

For example: if \(\alpha = abcWdXa\) then \(u = abc\), and \(\beta = WdXa\), since the next symbol after \(abc\) is \(W \in V\) (a non-terminal or \(\Lambda\))

\(S \Rightarrow_{\text{lm}} \alpha \vdash^* (q, abcV,S) \vdash^* (q,V, W...)\)

The proof is by induction on the length of the derivation of \(\alpha\).
We also need to prove that every string accepted by M belongs to L(G). Again, to make induction work, we need to prove a slightly more general fact:

If \((q,w,A) \vdash^* (q, \Lambda, \Lambda)\), then \(A \Rightarrow^* w\)

For all Stacks A, letting A = Start we have our proof.

This time we induct on the length of execution of M that leads from the ID \((q,w,A)\) to \((q, \Lambda, \Lambda)\).
A Grammar from a PDA

Assume the $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ is given, and that it accepts by empty stack. Consider execution of $M$ on an accepted input string.

If at some point of the execution of $M$ the stack is $Z \zeta$ (Z is on top, $\zeta$ is the rest of stack)

In terms of instantaneous descriptions

$(\text{state}_i, \text{input}, Z \zeta) \vdash \ldots$

Then we know that eventually the stack will be $\zeta$.

Why? Because we assume the input is accepted, and $M$ accepts by empty stack, so eventually $Z$ must be removed from the stack.
\[(\text{state}_i, \alpha X, Z\zeta) \vdash^* (\text{state}_j, X, \zeta)\]

The sequence of moves between these two instants is the “net popping” of Z from the stack.

During this sequence of moves, the stack may grow and shrink several times, some input will be consumed (the \(\alpha\)), and M will pass through a sequence of states, from state\(_i\) to state\(_j\).
Net Popping

Net popping is fundamental for the construction of a CFG G equivalent to M.

We will have a variable (Non-terminal) $[qZp]$ in the CFG G for every triple in $(q,Z,p) \in Q \times \Gamma \times Q$ from the PDA. Recall

1. Q is the set of states
2. $\Gamma$ is the set of stack symbols

We want the rhs of a production whose lhs is $[qZp]$ to generate precisely those strings $w \in \Sigma^*$ such that M can move from q to p while reading the input w and doing the net popping of Z. A production like $[qZp] \rightarrow ?$

This can be also expressed as $(q,w,Z) \xrightarrow{*} (p, \Lambda, \Lambda)$

Productions of G correspond to transitions of M.
If \((p, \zeta) \in \delta(q, a, Z)\), then there is one or more corresponding productions, depending on complexity of \(\zeta\).

1. If \(\zeta = \Lambda\), we have \([qZp] \rightarrow a\)
2. If \(\zeta = Y\), we have \([qZr] \rightarrow a[pYr]\) for every state \(r\)
3. If \(\zeta = YY'\) we have \([qZs] \rightarrow a[pYr][rY's]\), for every pair of states \(r\) and \(s\).
4. You can guess the rule for longer \(\zeta\).
Example

Q = \{0,1\}
S = \{a,b\}
\Gamma = \{X\}
\delta(0,a,X) = \{(0,X)\}
\delta(0,\Lambda,X) = \{(1,\Lambda)\}
\delta(1,b,X) = \{(1,\Lambda)\}
Q_0 = 0
Z_0 = X
F = \{\}, accepts by empty stack

Non-terminals
(q,Z,p) \in Q \times \Gamma \times Q
(0,'X',0)
(0,'X',1)
(1,'X',0)
(1,'X',1)

Productions, At least one from each element in delta
(p,z) \in \delta(q,a,Z)

(0,a,X,0,X)
(1,b,X,1,\Lambda)
(0,\Lambda,X,1,\Lambda)

0X0 \rightarrow a 0X0
0X1 \rightarrow a 0X1
1X1 \rightarrow b
0X1 \rightarrow \Lambda