Learning Outcomes

By the end of this session you should be able to

• convert CFG into a PDA

• convert PDA into a CFG
Recall

- **Pushdown automata**
  - finite automata with an unbounded stack;
  - can construct one to accept \( \{a^n b^n \mid n \geq 0 \} \);
  - accept by final state or empty stack

- **Context free grammars**
  - nonterminals, terminals, derivations;
  - can come up with one to derive \( \{a^n b^n \mid n \geq 0 \} \);
  - can put one in Chomsky Normal Form (CNF) or Greibach Normal Form (GNF) (losing \( \epsilon \)).
Today: Hacking PDAs

• In this lecture we will show how to go from:
  1. grammar in GNF to a PDA;
  2. one state PDA to a grammar;
  3. PDA to a one state PDA.

• we can use this data to convert

  \[ \text{CFG} \iff \text{GNF} \iff \text{PDA} \]

and

  \[ \text{PDA} \iff \text{one state PDA} \iff \text{CFG}. \]

So PDAs and CFGs have the same power!
From CFG to PDA

• Suppose that \( G = (N, \Sigma, P, S) \) is a CFG. We will construct a PDA \( M \) such that \( L(M) = L(G) \).

  ○ without loss of generality, assume that \( G \) is in Greibach normal form, i.e. all productions look like:

    \[ A \rightarrow \sigma B_1 \ldots B_k \]

  ○ the PDA will have only one state \( * \) and will accept by empty stack, i.e. \( x \) will be accepted when there exists a computation

    \[ (*, x, \perp) \Rightarrow (*, \epsilon, \epsilon) \]
The construction

• Let $G = (N, \Sigma, P, S)$. The required PDA is $(\{\ast\}, \Sigma, N, \delta, \ast, S, \emptyset)$:
  
  ○ $\ast$ is the only state;
  
  ○ $N$, the set nonterminals of $G$, is the stack alphabet;
  
  ○ $S$, the start production, is the initial stack symbol;
  
  ○ we accept by empty stack, so no need for final states;
  
  ○ $\delta$ is constructed as follows:

  for each $A \rightarrow \sigma B_1 \ldots B_k$ in $P$, $((\ast, \sigma, A), (\ast, B_1 \ldots B_k)) \in \delta$
Example

- Consider the grammar:

  \[ S \rightarrow aSB \mid aB, \quad B \rightarrow b \]

- The corresponding PDA has one state, initial stack symbol \( S \) and the following productions:
Derivations and computations

- **leftmost** derivation def productions are applied only to the leftmost nonterminal;
- leftmost derivations of a grammar in GNF = accepting runs in the corresponding PDA.
- **Example:**

\[
S \Rightarrow aSB \\
\Rightarrow aaBB \\
\Rightarrow aabB \\
\Rightarrow aabb
\]

\[
(\ast, aabb, S) \rightarrow (\ast, abb, SB) \\
\rightarrow (\ast, bb, BB) \\
\rightarrow (\ast, b, B) \\
\rightarrow (\ast, \epsilon, \epsilon)
\]
PDA with one state to CFG

• the construction is reversible! Suppose that we have a PDA $\langle \{q\}, \Sigma, \Gamma, \delta, q, \bot, \emptyset \rangle$
  
  ○ Let $G = (\Gamma, \Sigma, P, \bot)$
  
  ○ $P$ is constructed as follows:

    for each $\langle (q, c, A), (q, B_1 \ldots B_k) \rangle \in \delta$
    
    let $A \rightarrow cB_1 \ldots B_k$ be in $P$.

• correctness: because of the correspondence (leftmost derivations) $\Leftrightarrow$ (computations), the languages defined by the grammar and the pushdown automaton coincide.
From PDA to one state PDA 1

- **Idea:** all state info is maintained on the stack
- **First step:** transform a PDA $M = (Q, \Sigma, \Gamma, \delta, s, \bot, F)$ to a PDA $M'$ with a single final state $t$ so that $M'$ can clear its stack after getting to $t$.

- Clearly $L(M) = L(M')$ and $M'$ can clear its stack once in $t$. 
From PDA to one state PDA 2

- Can assume that PDA has one accept state $t$ and can clear its stack once it reaches the final state:

$$M = (Q, \Sigma, \Gamma, \delta, s, \bot, \{t\})$$

- Let $\Gamma' \overset{\text{def}}{=} Q \times \Gamma \times Q$. So a symbol in $\Gamma'$ is a triple $(p, A, q)$ where $p, q \in Q$ and $A \in \Gamma$. We will write $\langle pAq \rangle$ to denote such an element of $\Gamma'$.

- Let $M' \overset{\text{def}}{=} (\{*\}, \Sigma, \Gamma', \delta', *, \langle s\bot t \rangle, \emptyset)$ where $\delta'$ is defined on the next slide.
• $\delta'$ is defined as follows:

  for each transition $((p, \sigma, A), (q_0, B_1 B_2 \ldots B_k)) \in \delta$

  include $((*, \sigma, \langle p A q_0 \rangle), (*, \langle q_0 B_1 q_1 \rangle \langle q_1 B_2 q_2 \rangle \ldots \langle q_{k-1} B_k q_k \rangle))$ in $\delta'$

  for all possible choices of $q_1, q_2, \ldots, q_k \in Q$

• Note that when $k = 0$, this reduces to:

  for each transition $((p, \sigma, A), (q_0, \epsilon)) \in \delta$

  include $((*, \sigma, \langle p A q_0 \rangle), (*, \epsilon))$ in $\delta'$

• Idea: at each stage, we nondeterministically guess the remainder of the computation of $M$ and then verify it.
Example

- consider the PDA: $\{\{0, 1, 2\}, \{a, b\}, \{a, \bot\}, \delta, 0, \bot, \{2\}\}$

$\begin{align*}
(0, aabb, \bot) & \rightarrow (0, abb, a\bot) & \rightarrow (\ast, abb, \langle 0a1 \rangle \langle 1\bot 2 \rangle) \\
& \rightarrow (0, bb, aa\bot) & \rightarrow (\ast, bb, \langle 0a1 \rangle \langle 1a1 \rangle \langle 1\bot 2 \rangle) \\
& \rightarrow (1, b, a\bot) & \rightarrow (\ast, b, \langle 1a1 \rangle \langle 1\bot 2 \rangle) \\
& \rightarrow (1, \epsilon, \bot) & \rightarrow (\ast, \epsilon, \langle 1\bot 2 \rangle) \\
& \rightarrow (2, \epsilon, \epsilon) & \rightarrow (\ast, \epsilon, \epsilon)
\end{align*}$
Learning Outcomes

You should be able to . . .

• . . . convert CFG into a PDA

• . . . convert PDA into a CFG