A Few More Details

Grammar needs base cases
- Add $A_{pq} \rightarrow b$ for every $r$

Other Rules:

- Start variable?

Grammar needs base cases
- $A_{pq} \rightarrow aA_{rs}b$ for every $r, q \in Q$

A more convoluted way:

- Define $A_{pq}$ for each pair of states $p$ and $q$

Repeat: Converting a NFA $N$ to a CFG
Proof Idea:

We have a PDA $P$. We want to construct a CFG $G$ that generates $L(P)$.

For each pair of states $p$ and $q$ in $P$, create $A_{pq}$ that will generate all possible strings in going from $p$ to $q$.

- Starting with an empty stack and ending with an empty stack
- PDA will generate all possible strings in going from $p$ to $q$.

This can be used for starting and ending with the same stack.

How do we expand our proof to account for a PDA $P$? How does a PDA differ from an NFA?

Change our definition of $A_{pq}$.
How do we end?

Start variable: \( A \)

- \( \text{How do we end?} \)

Designing the Grammar

- How do we define \( A_{pq} \) so it generates all strings that can take \( P \) from \( p \) to \( q \) starting and ending with empty stack?

- \( P \) must first push, as stack is empty, and must end by popping

Case 2: the initial push is popped part-way through

<table>
<thead>
<tr>
<th>( (r, c) )</th>
<th>( \delta(p, a, \epsilon) )</th>
<th>( (q, \epsilon) )</th>
<th>( \delta(s, b, c) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>( V )</td>
<td>( V )</td>
<td>( V )</td>
<td></td>
</tr>
</tbody>
</table>

Case 1: the initial push is popped at the very end

- \( A_{pq} \rightarrow aA_{rs}b \) where there is a transition from \( p \) to \( r \) on input \( a \) and pushing \( c \) (with no pop)
- + from \( s \) to \( q \) on input \( b \) and popping \( c \) (with no push)
- + from \( d \) to \( e \) on input \( c \) and popping \( e \) (with no push)

\( \Rightarrow \text{So if \( \epsilon \) occurs on all strings that can take \( V \) from \( d \) to \( e \), and ending with empty stack,} \)

- How do we define \( V \)

Simplified PDAs

- To simplify the proof, let's use simpler version of PDAs

- Has a single accept state
- Empties its stack before accepting
- Each transition either pushes symbol onto stack or pops one of stack

- Can any PDA be converting into this simplified form?

To simplify the proof, let's use simpler version of PDAs
Proof

Claim: if $A_{pq}$ generates $x$, $P$ can go from $p$ to $q$ starting/ending with empty stack.

Corollary: if $A_{pq}$ accepts $x$, $P$ accepts $x$.

Claim: if string $x$ can bring $P$ from $p$ to $q$ starting/ending with empty stack,

Corollary: if $P$ accepts $x$, grammar can generate $x$.

More formally, say that $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{\text{accept}}\})$.

- Construct $G$ as follows:
  - The variables are $\{A_{pq} | p, q \in Q\}$.
  - For each $p, q, r, s \in Q$, $t \in \Gamma$, and $a, b \in \Sigma$, if $r, t \in \delta(p, a, \epsilon)$ and $q, \epsilon \in \delta(s, b, t)$, add rule $A_{pq} \rightarrow aA_{rs}b$.
  - For each $p, q \in Q$, add the rule $A_{pq} \rightarrow A_{pr}A_{rq}$.
  - For each $p \in Q$, add the rule $A_{pp} \rightarrow \epsilon$.

- For each $x \in \mathcal{L}(G)$, and the rule $A_{pq} \rightarrow \epsilon$, add the rule $A_{pq} \rightarrow \delta^{\epsilon}(p, a, \epsilon) \cap \{q \} \subseteq \{q \} \in \mathcal{L}(G)$.

- Construct $C$ as follows:
  - Say that $P = (\{p_{\text{accept}}\}, \{q_0\}, \{q\}, \{q_{\text{accept}}\}) = P$.

More formally,
Claim: if Grammar generates $x$ so does PDA

• Assume true for derivations of length at most $k$, where $k \geq 1$.

• Prove for deviation of length $k + 1$.

  • Assume $A_{pq} \Rightarrow x$ with $k + 1$ steps
  • Assume $A_{pq} \Rightarrow y$ in the derivation, so $x = ayb$
    - So, if $P$ starts at $p$ with empty stack
    - $A_{pq} \Rightarrow ayb$ where $a, b \in \Sigma$ and $r, s \in Q$

  - Case 1: 1st step in derivation is $A_{pq} \Rightarrow aA_{rs}b$ where $a, b \in \Sigma$ and $r, s \in Q$
    - Say that $A_{rs} \Rightarrow y$ in the derivation, so $y \in \Sigma^*$
    - Since $A_{pq} \Rightarrow aA_{rs}b$ is in grammar
    - $\delta(p, a, \epsilon) = (r, t)$ for some $t \in \Gamma$
    - $\delta(s, b, t) = (q, \epsilon)$

    - So if $P$ starts at $p$ with empty stack
      - After reading $a$ it can go to state $r$ and push $t$ on the stack
      - Then after reading $b$ it can go to $q$ and pop $t$ off the stack

    - So, $P$ can go from $p$ to $q$ with empty stack, reading $a\epsilon b = x$

We prove this claim by induction on the number of steps in the derivation of $x$ from $A_{pq}$

Claim 2.30: If $A_{pq}$ generates string $x$, then $x$ can bring $P$ from $p$ with empty stack to $q$ with empty stack.

• Basis: derivation has 1 step
  - Must use $A_{pq} \Rightarrow \epsilon$
    - Only thing by grammar is $\epsilon$
    - $\epsilon$ can take $P$ from $p$ to $p$ with empty stack to empty stack
Claim: if PDA accepts $x$ so does Grammar $A$

Claim 2.31: If string $x$ can bring $P$ from $p$ to $q$ starting/ending with empty stack, $A_{pq}$ generates $x$

Proof by induction on number of steps in computation of $P$

- Basis: computation has $0$ steps from $p$ to $q$
  - In $0$ steps, we can just stay at the same state $q$, and we will have read $b$ from an empty stack
  - We have the rule $A_{pp} \rightarrow \epsilon$, which generates $\epsilon$ as required

- Induction:
  - Assume true for computations of length at most $k$, where $k \geq 0$
  - Suppose $P$ has a computation wherein $x$ brings $P$ from $p$ to $q$ with empty stack in $k+1$ steps
  - We have the rule $d \rightarrow d'$, which generates $\epsilon$, as required
  - In $k + 1$ steps, we can just stay at the same state $q$, and we will have read $b$ from $d$ and not touched the stack
  - Each does in less than $k + 1$ steps
  - $x = yz$, so $P$ can generate $y$ going from $p$ to $r$ starting/ending with empty stack
  - and can generate $z$ going from $r$ to $q$ starting/ending with empty stack
  - Each does in less than $k + 1$ steps
  - $z \leftarrow b^y d^z$ and $b \leftarrow d^y b^z$ and $d \leftarrow b^y d^z$

Case 2: First step in derivation is $d$. Generates $x$ so does PDA
continued: if PDA accepts $x$ so does Grammar

- Case 2: stack is empty somewhere in middle of computation

- Let $r$ be the state where the stack is empty

- Then the portions of the computation from $p$ to $r$ and from $r$ to $q$ each contain at most $k$ steps

- Say $y$ is read from during first part ($p$ to $r$) and $z$ is read from $r$ to $q$

- Induction tells us $A_p r^* \Rightarrow y$ and $A_q r^* \Rightarrow z$

- We have rule $A_p q r \rightarrow A_p y A_q z$ in $G$

- Let $y$ be such that $x = ayb$ +

- P can go from $r$ to $s$ by reading $y$ without touching symbol $t$, and so can P from $r$ to $s$ with an empty stack at begin and end +

- This has $k-1$ steps in the computation +

- So rule $q r^* \rightarrow b d r^* \leftarrow \text{in } G$

- So, rule $q r^* \rightarrow b d r^*$ +

- Let $a$ be the state after the first read and $b$ be the state before the last read -

- Let $a$ be the input read in the first move, and $b$ be the input read in the last -

- Symbol popped at beginning must be same as symbol popped at end, say $c$

- Case 1: stack is empty only at beginning and end

- Hence $x = q r d \Rightarrow b d r^*$ +

- By induction, rules in the computation +

- So $q r^* \rightarrow b d r^* \leftarrow \text{in } G$

- So rule $q r^* \rightarrow b d r^*$ +

- Case 1: stack is empty only at beginning and end

continued: if PDA accepts $x$ so does Grammar
Relationship to Regular Languages

• Let $L$ be a regular language
  • So, there is an FA $D$ such that $L(D)$

Any FA is also a PDA, just with $\epsilon$ pops and pushes on stack.

• So, if there is an FA $D$ such that $L(D)$
  • Let $L$ be a regular language