A Few More Details

- Other Rules:
  - Add $A_{pq} \rightarrow A_{pr}A_{rq}$ for every $r$
- Grammar needs base cases

• Start variable?
Example

• Base Cases
  • $A_{13} \rightarrow xA_{r,s}y$
    - via transition
      $A_{2,2} \rightarrow a$
      $A_{1,3} \rightarrow \epsilon$
      $A_{3,1} \rightarrow a$
      $A_{1,2} \rightarrow b$
      $A_{2,3} \rightarrow a$
      $A_{2,3} \rightarrow b$
    - no transition
      $A_{1,3} \rightarrow bA_{2,1}$
      $A_{1,3} \rightarrow A_{3,1}$
      $A_{1,3} \rightarrow bA_{2,2}a$
      $A_{1,3} \rightarrow A_{3,2}a$
      $A_{1,3} \rightarrow bA_{2,2}b$
      $A_{1,3} \rightarrow A_{3,2}b$

• Similarly for
  - $A_{1,1}$, $A_{1,2}$, $A_{2,1}$, $A_{2,2}$, ...

• Rules of form
  $A_{x,z} \rightarrow A_{x,y}A_{y,z}$
  - 27 rules of this form

PDA $\Rightarrow$ Context-Free Grammar: Idea

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

• Change our definition of $A_{pq}$
  - Generate all possible strings that can take $P$ from $p$ to $q$
    + Starting with an empty stack and ending at an empty stack
  - So can be used for starting and ending with the same stack
PDA ⇒ Context-Free Grammar

• 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 at an empty stack
    + So can be used for starting and ending with the same stack

Simplified PDAs

• To simplify the proof, let’s use simpler version of PDAs
  - Has a single accept state \( q_{\text{accept}} \)
  - Empties its stack before accepting
  - Each transition either pushes symbol onto stack or pops one off stack

• Can any PDA be converting into this simplified form?
  + If multiple accepts, add a new one, and transition to it with an epsilon read, no pop off of the stack
  + Add extra states so that we start by pushing \( \$ \) onto the stack, and after accept, pop everything off until we get to \( \$ \)
  + If transition pushes and pops, split into two transitions, with a new state inbetween
  + If transition doesn’t push or pop, split into two, in which first pushes and second pops some character
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 1: the initial push is popped at the very end
  - $A_{pq} \rightarrow a A_{rs} b$ 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)
    i.e., for each $r, q \in Q$ and $a, b \in \Sigma$ and each $c \in \Gamma$, s.t. $(r, c) \in \delta(p, a, \epsilon)$ and $(q, c) \in \delta(s, b, \epsilon)$
- Case 2: the initial push is popped part-way through
  - $A_{pq} \rightarrow A_{pr} A_{rq}$ where $r$ is the state where the stack becomes empty
- Start variable: $A_{q_0, q_{accept}}$
- How do we end? $A_{pp} \rightarrow \epsilon$

More Formally

Say that $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{accept}\})$

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

- Must show
  - if grammar $G$ generates $x$, PDA $P$ accepts $x$
    + Claim: if $A_{pq}$ generates $x$, $P$ can go from $p$ to $q$ starting/ending with empty stack on input $x$
    + Corollary: if $A_{pq}$, $q_{accept}$ generates $x$, $P$ accepts $x$
  - if PDA accepts $x$, grammar can generate $x$
    + Claim: If string $x$ can bring $P$ from $p$ to $q$ starting/ending with empty stack, $A_{pq}$ generates $x$
    + Corollary: if $x$ can bring $P$ from start to accept state starting/ending with empty stack, $A_{pq}$, $q_{accept}$ generates $x$

Claim: if Grammar generates $x$ so does PDA

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

- We prove this claim by induction on the number of steps in the derivation of $x$ from $A_{pq}$
- Basis: derivation has 1 step
  - Must use a rule whose right hand side contains no variables
  - Must use $A_{pp} \rightarrow \epsilon$
  - Only thing by grammar is $\epsilon$
  - $\epsilon$ can take $P$ from $p$ to $p$ with empty stack to empty stack
Induction: 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
- Case 1: 1st step in der. is $A_{pq} \Rightarrow A_{rs}b$ where $a, b \in \Sigma$ and $r, s \in Q$
  - Say that $A_{rs} \Rightarrow y$ in the derivation, so $x = ayb$
    - $A_{rs}$ generates $y$ in $k$ steps
    - So $P$ can go from $r$ to $s$ starting/ending with empty stack and generate $y$
  - Since $A_{pq} \Rightarrow A_{rs}b$ is in grammar
    - $(r, t) \in \delta(p, a, \epsilon)$ for some $t \in \Gamma$ and $(q, \epsilon) \in \delta(s, b, t)$
    - 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 reading string $y$ can bring it to $s$ leaving $t$ 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 $ayb = x$

Case 2: First step in derivation is $A_{pq} \Rightarrow A_{pr}A_{rq}$

- Say $A_{pr} \Rightarrow y$ and $A_{rq} \Rightarrow z$
- Each does in less than $k + 1$ steps
  - So $P$ can generate $y$ going from $p$ to $r$ starting/ending with an empty stack
  - and can generate $z$ going from $r$ to $q$ starting/ending with an empty stack
- So, $P$ can generate $yz = x$ going from $p$ to $q$ starting/ending with empty stack
Continued: if PDA accepts $x$ so does Grammar

- Case 1: stack is empty only at beginning and end
  - Symbol pushed at beginning must be same as symbol popped at end, say $t$
  - Let $a$ be the input read in the first move, and $b$ be the input read in the last
  - Let $r$ be the state after the first read, and $s$ be the state before the last read
  - So, $(r, t) \in \delta(p, a, \epsilon)$ and $(q, \epsilon) \in \delta(s, b, t)$
  - So, rule $A_{pq} \rightarrow aA_{rs}b$ is 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 $P$ can go from $r$ to $s$ by reading $y$ with an empty stack at begin and end
    - This has $k - 1$ steps in the computation
    - By induction, $A_{rs} \Rightarrow y$
    - Hence $A_{pq} \Rightarrow ayb = x$
Claim: if PDA accepts $x$ so does Grammar

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 $x = \epsilon$ and not touched the 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 $p$ to $q$ with empty stacks in $k + 1$ steps

Relationship to Regular Languages

- Let $L$ be a regular language
- So, it there is an FA $D$ such that $L(D)$
- Any FA is also a PDA, just with $\epsilon$ pops and pushes of stack
- So, a PDA can recognize $L$
- So, every regular language is context free
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 that $A_{pr} \Rightarrow y$ and $A_{rq} \Rightarrow z$
  - We have rule $A_{pq} \rightarrow A_{pr}A_{rq}$ in $G$, so $A_{pq} \Rightarrow yz = x$