Example

NFA whose language is 4 last characters are 1010

Give state diagram:

Give grammar rules:
PDA ⇒ Context-Free Grammar: Idea

- How do we expand our proof to account for a PDA?
- How does a PDA differ from an NFA?
- Change our definition of $A_{pq}$

$A_{pq} \rightarrow A_{pr} A_{rq}$ for every $r$

Grammar needs base cases

A Few More Details
Simplified PDAs

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

- Has a single accept state
- Empties the stack before accepting
- Has a single accept state

Can any PDA be converted into this simplified form?

- Each transition either pushes a symbol onto stack or pops one off stack
- If transition doesn't push or pop, split into two, in which first pushes and second pops some character
- Transition doesn't push or pop, split into two, in which first pushes and second pops some character
- Transition pushes and pops, split into two transitions, with a new state
- Do every thing off until we get to $\epsilon$
- Add extra states so that we start by pushing $\epsilon$ onto the stack, and after accepting pop everything off the stack
- If multiple acceptors, add a new one, and transition to it with an epsilon read, no pop of the stack

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

PDA $\Rightarrow$ 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 $a$ and $b$ in $P$, create a new production $S_{ab} \rightarrow \epsilon$
- So can be used for starting and ending with the same stack
- Start with an empty stack and ending an empty stack
- Grammar will generate all possible strings in some form from top to bottom

$P_{real} \Rightarrow$ PDA

$P_{real} \Rightarrow$ Context-Free Grammar
More Formally

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

- Construct $G$ as follows
  - The variables are $\{ A_{pq} \mid p, q \in Q \}$
  - For each $p, q, r, s \in Q$, $t \in \Gamma$, and $a, b \in \Sigma$ $\epsilon^+$ 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 $p, q, r \in Q$, add the rule $A_{pq} \rightarrow A_{pr}A_{qr}$
  - For each $p \in Q$, add the rule $A_{pp} \rightarrow \epsilon$

Construct $G$ as follows

$(\{p_{\text{accept}}\} \cup \{ q \mid q \in Q \} = d$ =

Designing the Grammar

- How do we end? $A_{pp} \rightarrow \epsilon$
  - Start variable: $A_{q_0q_{\text{accept}}}$
  - How do we begin? $\epsilon$ $\rightarrow^{d_{dV}} V$

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

- Case 1: if the initial push is popped at the very end
  - The initial push is popped at the very end

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

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

Cass 2: If the initial push is popped part-way through

- Cass 2: if the initial push is popped part-way through
  - The initial push is popped part-way through

So if $C$ generates all strings that can take $b$ to $d$, then $A_{pq} \rightarrow^*$ $\epsilon$
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 $x$.

  **Basis:** derivations has 1 step $b_{td}$

  We prove this claim by induction on the number of steps in the derivation of $x$.

  - Claim: if Grammar generates $x$ so does PDA

**Proof**

- Must show
- If PDA accepts $x$, Grammar can generate $x$
- If Grammar generates $x$, PDA accepts $x$
- Claim: if Grammar generates $x$, PDA accepts $x$
- Claim: if Grammar generates $x$, PDA accepts $x$
- Corollary: if Grammar generates $x$, PDA accepts $x$
- Corollary: if Grammar generates $x$, PDA accepts $x$
- Corollary: if Grammar generates $x$, PDA accepts $x$
- Corollary: if Grammar generates $x$, PDA accepts $x$

- Must show
Continued: if Grammar generates $x$ so does PDA

- Case 2: First step in derivation is $A_{pq} \Rightarrow A_{pr}A_r$.
  - Say $A_{pr} \Rightarrow y$ and $A_r \Rightarrow z$.
  - 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 $P$ can generate $z$ going from $r$ to $q$ starting/ending with an empty stack.

- So, $P$ can generate $yz = x$ going from $p$ to $q$.

- 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 deriv. 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 $x = ayb$.
      - So $P$ can go from $r$ to $s$ starting/ending with an empty stack and generate $y$.
      - So if $P$ starts at $p$ with empty stack.
        - Then after reading $a$ it can go to state $r$ and push $t$ on the stack.
        - Then reading string $y$ can bring it to state $s$ leaving $t$ on the stack.
        - Then after reading $b$ it can go to state $q$ 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 $A_{pq} \Rightarrow A_{pr}A_r$.
  - Say $A_{pr} \Rightarrow y$ and $A_r \Rightarrow z$.
  - 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 $P$ can generate $z$ going from $r$ to $q$ starting/ending with an empty stack.

- So, $P$ can generate $yz = x$ going from $p$ to $q$. 
Claim: If PDA accepts $x$ so does Grammar

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
  - 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 $y$ without touching symbol $t$, and so can $G$
  - Let $y$ be such that $x = ayb$ can go from $r$ to $s$ by reading $y$ with an empty stack at begin and end
  - So, rule $A_{pq} \rightarrow aA_{rs}b$ is in $G$
  - By induction, $A_{rs} \ast \Rightarrow y$
  - Hence $A_{pq} \ast \Rightarrow ayb = x$

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

\[ x = \phi \Rightarrow \phi \Rightarrow^{bd} \phi \Rightarrow^{+} \]

- Basis: computation has 0 steps from $P$
  - $b$ comes in, $d$ goes out, and not touched the stack
  - With empty stack, $x$ 

Claim 2.31: If string $x$ can bring $P$ from $y$ to $z$ in $k$ steps, then $G$ can bring $x$ to $z$ in $k$ steps, where $y \leq 0$

- We use the rule $x \Rightarrow^{+}$ or generates $x$, as required
  - If in steps, we can just stay in the same state $b$, and we will have read $b$ from $d$
  - Proof by induction on number of steps in computation of $G$
    - With empty stack, $x$ generates $x$ in $G$

Claim: If PDA accepts $x$ so does Grammar
Relationship to Regular Languages

- So every regular language is context-free.
- So a PDA can recognize L.
- Any FA is also a PDA, just with ε pops and pushes of stack.
- So if there is an FA D such that L(D) = L, let D be a regular language.

Continued: if PDA accepts x so does Grammar

- Case 2: Stack is empty somewhere in middle of computation.
  - Let S be the state where the stack is empty.
  - Then the portion of the computation from S to S′ has zero steps.
  - Say y is read from during first part (S to S′) and z is read from S′ to t.
  - Induction tells us that A_{S} \rightarrow y and A_{S′} \rightarrow z.
  - We have rule A_{S} \rightarrow yA_{S′} in G, so A_{S} \rightarrow yz = x.

\[ x = zfi \xleftarrow{bd} \text{ in } C, \text{ so } A_{S} \xrightarrow{bd} A_{S′} \xleftarrow{def} \text{ in } G. \]