Equivalence of Pushdown Automata (PDA)

PDAs accepting by final state and empty stack are <u>equivalent</u>

- $P_F \rightarrow$ PDA accepting by final state
  - $P_F = (Q_F, \sum, \Gamma, \delta_F, q_0, Z_0, F)$
- $P_N \rightarrow$ PDA accepting by empty stack
  - $P_N = (Q_N, \sum, \Gamma, \delta_N, q_0, Z_0)$
- <u>Theorem:</u>
  - ($P_N \rightarrow P_F$) For every P_N, there exists a P_F s.t. $L(P_F)=L(P_N)$
  - ($P_F \rightarrow P_N$) For every P_F, there exists a P_N s.t. $L(P_F)=L(P_N)$

$P_N \rightarrow P_F$ Construction

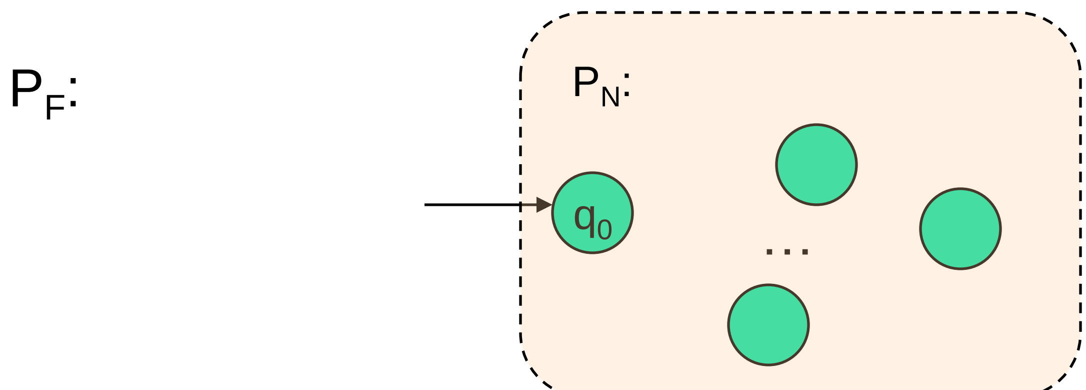
P_F:

$P_F = (Q_N \cup \{p_0,p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$

$P_N = (Q_N,\Sigma, \Gamma, \delta_N,q_0,Z_0)$

How to convert an empty stack PDA into a final state PDA?

$P_N \rightarrow P_F$ Construction

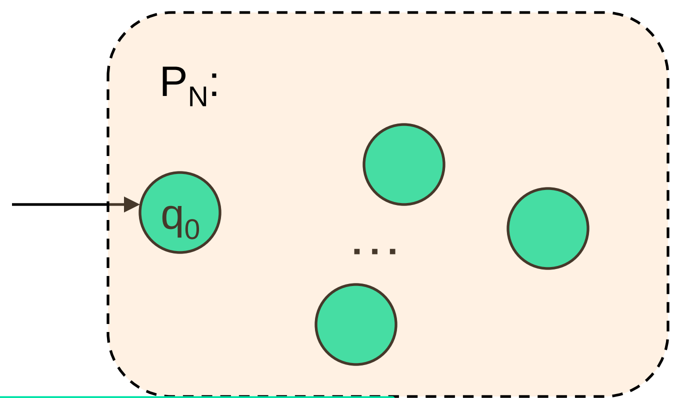


$\boxed{P_F = (Q_N \cup \{p_0,p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})}$

$P_N \rightarrow P_F$ Construction

- Whenever P_N's stack becomes empty, make P_F go to a final state without consuming any addition symbol

P_F:

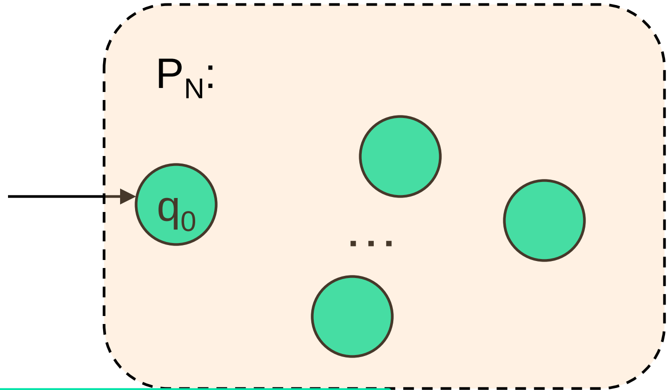


$P_F = (Q_N \cup \{p_0,p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$

$P_N \rightarrow P_F$ Construction

- Whenever P_N's stack becomes empty, make P_F go to a final state without consuming any addition symbol
- To detect empty stack in P_N: P_F pushes a new stack symbol X_0 (not in Γ of P_N) initially before simultating P_N

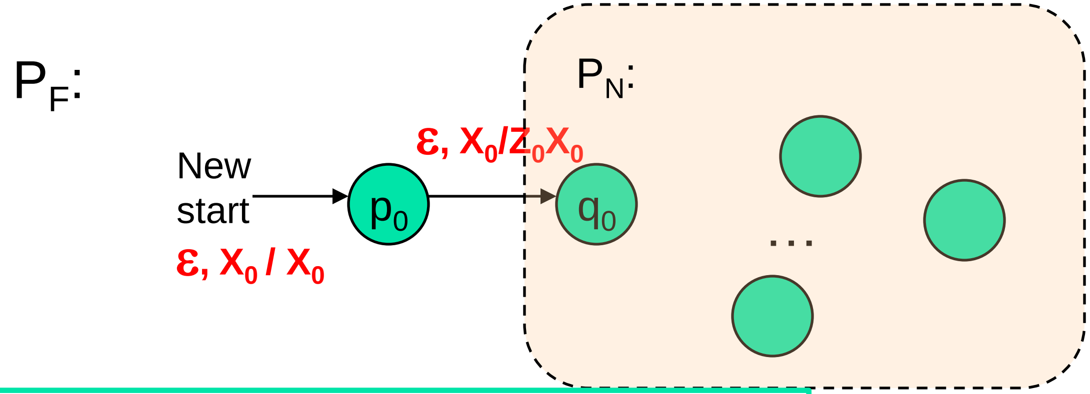
P_F:



$P_F = (Q_N \cup \{p_0,p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$

$P_N \rightarrow P_F$ Construction

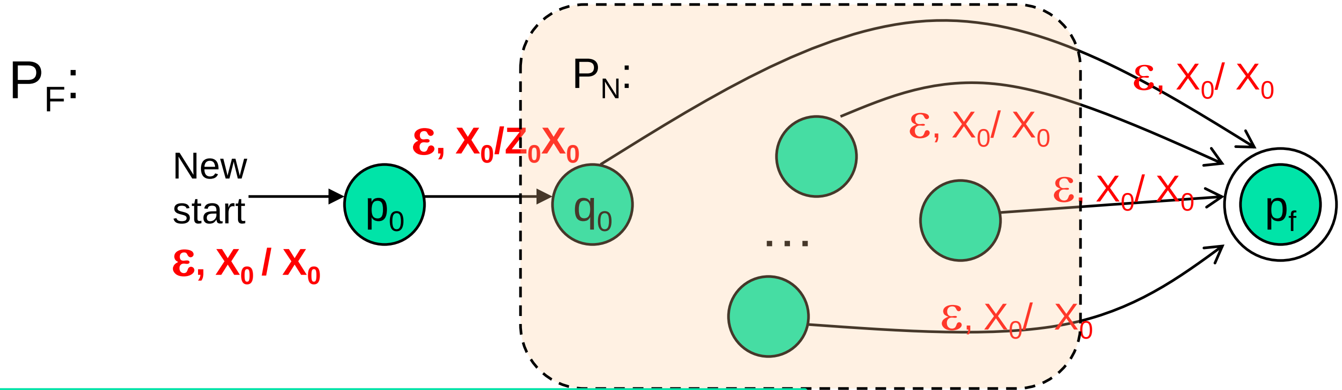
- Whenever P_N's stack becomes empty, make P_F go to a final state without consuming any addition symbol
- <u>To detect empty stack in P_N:</u> P_F pushes a new stack symbol X_0 (not in Γ of P_N) initially before simultating P_N



$P_F = (Q_N \cup \{p_0,p_f\}, \sum, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$

$P_N \rightarrow P_F$ Construction

- Whenever P_N's stack becomes empty, make P_F go to a final state without consuming any addition symbol
- <u>To detect empty stack in P_N:</u> P_F pushes a new stack symbol X_0 (not in Γ of P_N) initially before simulating P_N



$P_F = (Q_N \cup \{p_0,p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$

- δ_F is given by rules:

R_1 : $\delta_F (p_0, \varepsilon, x_0) = \{(q_0, z_0x_0)\}$

R_2 : $\delta_F (q, a, z_0) = \delta (q, a, z_0)$ for all q in Q, a in Σ or ε and z in Γ

R3 : $\delta_F(q, \varepsilon, x_0) = \{(p_f, x_0)\}$

$$(p_0, w, x_0) \vdash_F (q_0, w, z_0x_0)$$

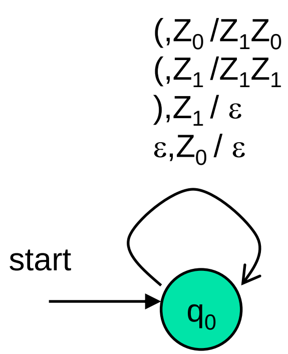
$$\vdash^*_F (q, \boldsymbol{\varepsilon}, x_0)$$

$$\vdash_F (p_f, \boldsymbol{\varepsilon}, x_0)$$

Example: Matching parenthesis "(" ")"

P_N: $(\{q_0\}, \{(,)\}, \{Z_0,Z_1\}, \delta_N, q_0, Z_0)$

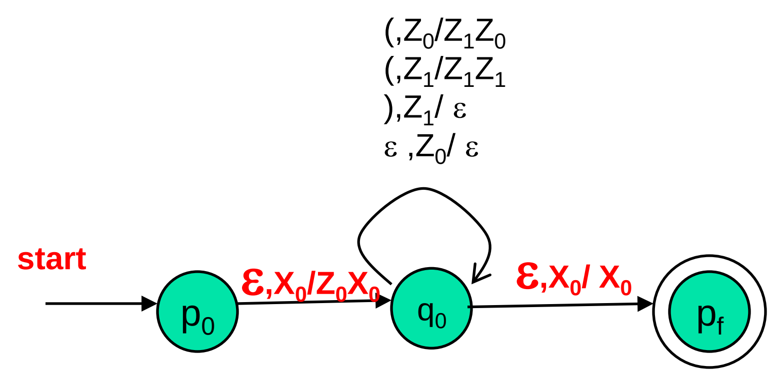
δ_N:
$\delta_N(q_0,(,Z_0) = \{ (q_0,Z_1Z_0) \}$
$\delta_N(q_0,(,Z_1) = \{ (q_0, Z_1Z_1) \}$
$\delta_N(q_0,),Z_1) = \{ (q_0, \varepsilon) \}$
$\delta_N(q_0, \varepsilon,Z_0) = \{ (q_0, \varepsilon) \}$



Accept by empty stack

P_f: $(\{p_0,q_0,p_f\}, \{(,)\}, \{X_0,Z_0,Z_1\}, \delta_f, p_0, X_0, p_f)$

δ_f:
$\delta_f(p_0, \varepsilon,X_0) = \{ (q_0,Z_0 X_0) \}$
$\delta_f(q_0,(,Z_0) = \{ (q_0,Z_1 Z_0) \}$
$\delta_f(q_0,(,Z_1) = \{ (q_0, Z_1Z_1) \}$
$\delta_f(q_0,),Z_1) = \{ (q_0, \varepsilon) \}$
$\delta_f(q_0, \varepsilon,Z_0) = \{ (q_0, \varepsilon) \}$
$\delta_f(p_0, \varepsilon,X_0) = \{ (p_f, X_0) \}$



Accept by final state

$P_F = (Q_F, \sum, \Gamma, \delta_F, q_0, Z_0, F)$

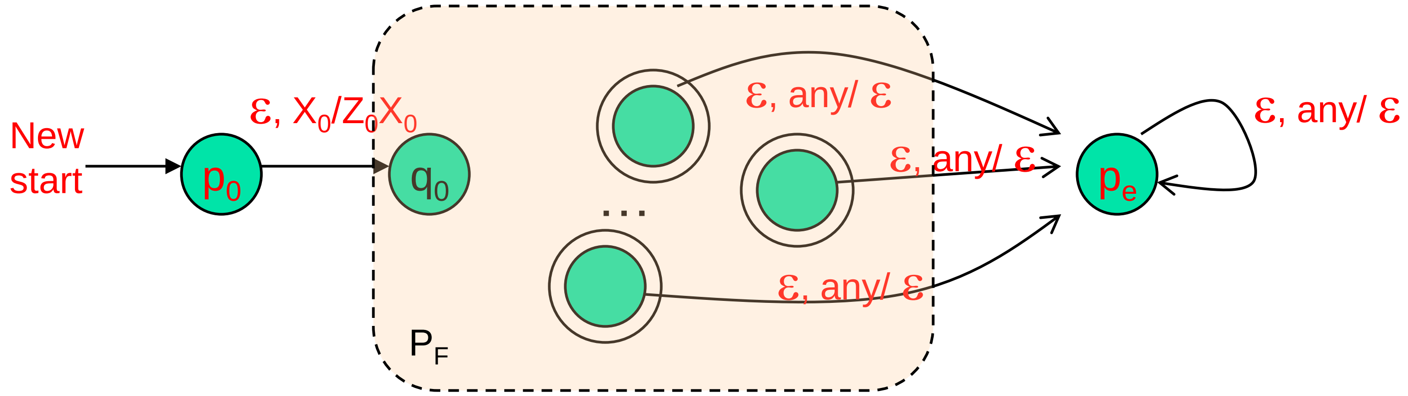
$P_F \rightarrow P_N$ construction

- <u>Main idea:</u>
  - Whenever P_F reaches a final state, just make an ε-transition into a new end state, clear out the stack and accept
  - Danger: What if P_F design is such that it clears the stack midway *without* entering a final state?

    ➔ to address this, add a new start symbol X_0 (not in Γ of P_F)

$P_N = (Q \cup \{p_0, p_e\}, \sum, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0, \{p_e\})$

P_N:



- δ_N is defined by R_1, R_2, R_3 and R_4 as:

R_1 : $\delta_N(p_0, \varepsilon, x_0) = \{(q_0, z_0x_0)\}$

R_2 : $\delta_N(p, \varepsilon, z) = \{(p, \varepsilon)\}$

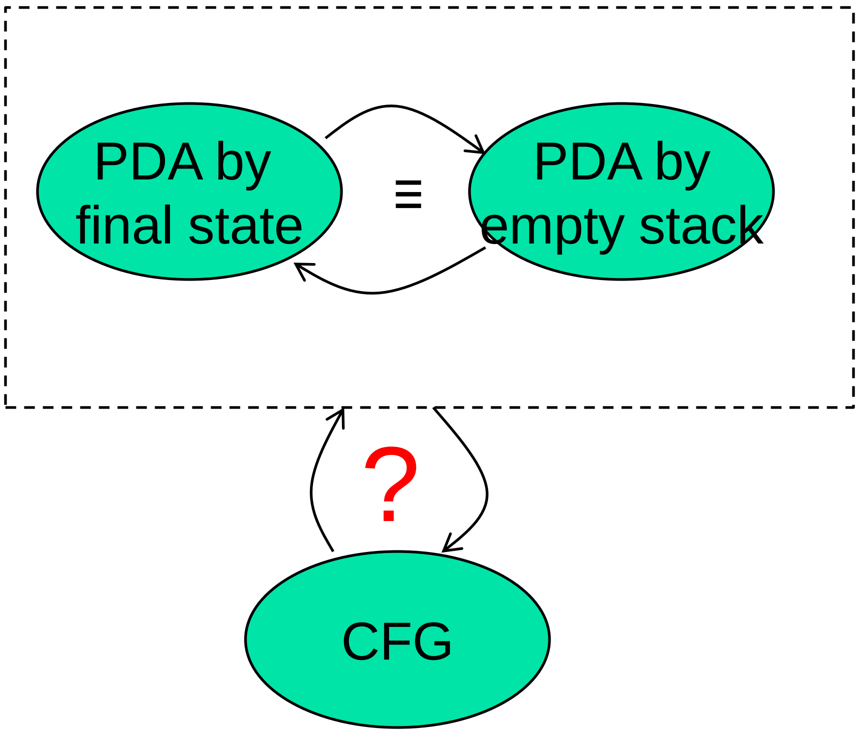
for all $z \in \Gamma \cup \{x_0\}$

R_3 : $\delta_N(q, a, z) = \delta(q, a, z)$ for all $a \in z$, $q \in Q$, $z \in \Gamma$.

R_4 : $\delta_N(q, \varepsilon, z) = \delta(q, \varepsilon, z) \cup \{(p, \varepsilon)\}$ for all $z \in \Gamma \cup \{x_0\}$ and $q \in F$.

Equivalence of PDAs and CFGs

CFGs ↔ PDAs ==> CFLs



Formal construction of PDA from CFG

- **Theorem**

  For any context free language L, there exists an PDA M such that L = L(M)

- **Proof**

  Let G = (V, T, P, S) be a grammar. There exists a Greibach Normal Form then we can construct PDA which simulates left most derivations in this grammar.

$M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$, where

$Q = \{q_0, q_1, q_f\}$ → set of states

Σ = terminals of grammar G

$\Gamma = V \cup \{z\}$ where V is the variables in grammar G

$F = \{q_f\}$ → final state.

- The transition function will include

1. $\delta(q0, \lambda, z) = \{(q1, Sz)\}$

   S → start symbol

2. $\delta(q_1, \varepsilon, A) = \{(q, \alpha)\}$

   for each $A \rightarrow \alpha$ in P

3. $\delta(q, a, a) = \{(q, \varepsilon)\}$ for each $a \in \Sigma$