From PDAs to CFGs

* Every PDA can be converted to one that has exactly one final state ($\{t\}$) & accepts after popping out the entire stack contents (Add ε-transitions from the final states to $\{t\}$)

* PDAs with a single final state to a PDA with a single state - store the state info on the stack.

  ↓ accepts by empty stack

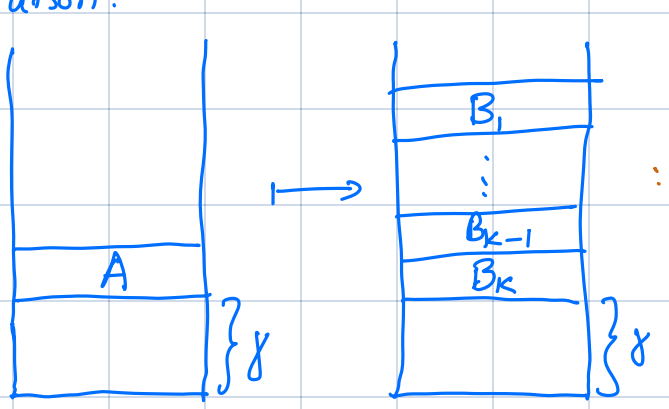
* PDAs with a single state $\longrightarrow$ CFG G.

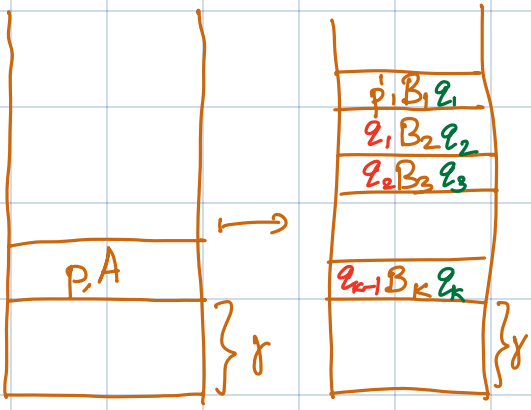
* Convert PDAs with one final state that empties the stack to PDA with a single state

$M = (Q, \Sigma, \Gamma, \delta, s, \perp, \{t\}) \longmapsto M' = (\{q\}, \Sigma, \Gamma', \delta', q, \perp', \phi)$

$(p, \sigma, A) \longmapsto (p', B_1 B_2 \dots B_k)$ ✓ $\quad \therefore \perp' = (s, \perp, t)$

Configuration:





$q_1, q_2, \dots q_{k-1}$ – guesses of the stak of the PDA M when the computation reaches that stage

$p', B_1, \rightsquigarrow$ pop (p', B_1).

(q_1, B_2)

Consistency check: In p', after processing B_1 and finding (q_1, B_2) on top.

Stack alphabet for M' $(\Gamma') = Q \times \Gamma \times Q$ (guess; guess for the state after popping the symbol.)

$((p, \sigma, A), (p', B_1 B_2 \dots B_k)) \in \delta$

In M': δ' $\quad \forall\, q_1, q_2, q_3, \dots, q_k \in Q$

add $(q, \sigma, (pAq_k')), (q, (p'B_1q_1)(q_1B_2q_2) \dots (q_{k-1}B_kq_k)) \in \delta'$

$(p, \sigma, A), (p', \varepsilon)$ → popping a symbol of the stack

$(Q \times \Sigma \times \Gamma) \times (Q \times \Gamma^*)$ add $((q, \sigma, (pAp')), (q, \varepsilon)) \in \delta'$

↳ δ

* Conversion of a single state PDA $\longrightarrow$ CFG G

Every transition of the PDA is of the form

$((q, \sigma, A), (q, B_1 B_2 \cdots B_k))$

$\downarrow$

$A \longrightarrow \sigma B_1 B_2 \cdots B_k$

$((q, \sigma, A), (q, \varepsilon)) \longmapsto A \to \sigma$

} Greibach Normal Form

∴ Decision problems for CFLs

① Given a CFG G, check if $L(G) = \phi$

$\exists ?\ w$ s.t $S \xrightarrow{*} w$

- Find all reachable non-terminals
  - ↳ Formulate as a graph problem
- Find non-terminals that don't generate any terminal symbols

} · Exercise
· not necessary for an efficient algorithm

② Given G, check if $L(G) = \Sigma^*$

- there does not exist an algorithm for this problem

Effective Computability

- what can be solved algorithmically?

Hilbert's Formalist Program

- Axiomatize mathematics
- Entscheidungsproblem - Given a mathematical statement, check if it is true (Algorithm for this)

Completeness theorem: FOL is axiomatizable (Gödel)

$\forall x \forall y \exists z \quad x < z < y \quad \ldots$

or $y < z < x$

or $x = y = z$