Teoria da Computação

Autômatos de Pilha e Gramáticas Livres de Contexto

Professor Thiago Alves

Introdução

- Autômatos de pilha e gramáticas livres de contexto são capazes de descrever exatamente a classe das linguagem livres de contexto
- Vamos mostrar como converter uma GLC em um autômato de pilha e vice-versa

Lema

- Se A é uma linguagem livre de contexto então existe um autômato de pilha P tal que L(P) = A.

Lema

- Se A é uma linguagem livre de contexto então existe um autômato de pilha P tal que L(P) = A.
  - Se A é uma linguagem livre de contexto então existe um GLC G tal que L(G) = A
  - Vamos converter G em um autômato de pilha P

GLC → AP

- Temos que descrever um autômato de pilha P que aceita sua entrada w, se G gera w
- A pilha do autômato vai servir para guardar a string de variáveis e terminais
- A cada passo, o autômato deve escolher de forma não-determinística uma das regras para aplicar

GLC → AP

- O autômato de pilha inicia colocando a variável inicial de G na pilha
- Pára quando chegar em uma string apenas com terminais
- Aceita se a string na pilha for igual a string de entrada

GLC → AP

- Se G for

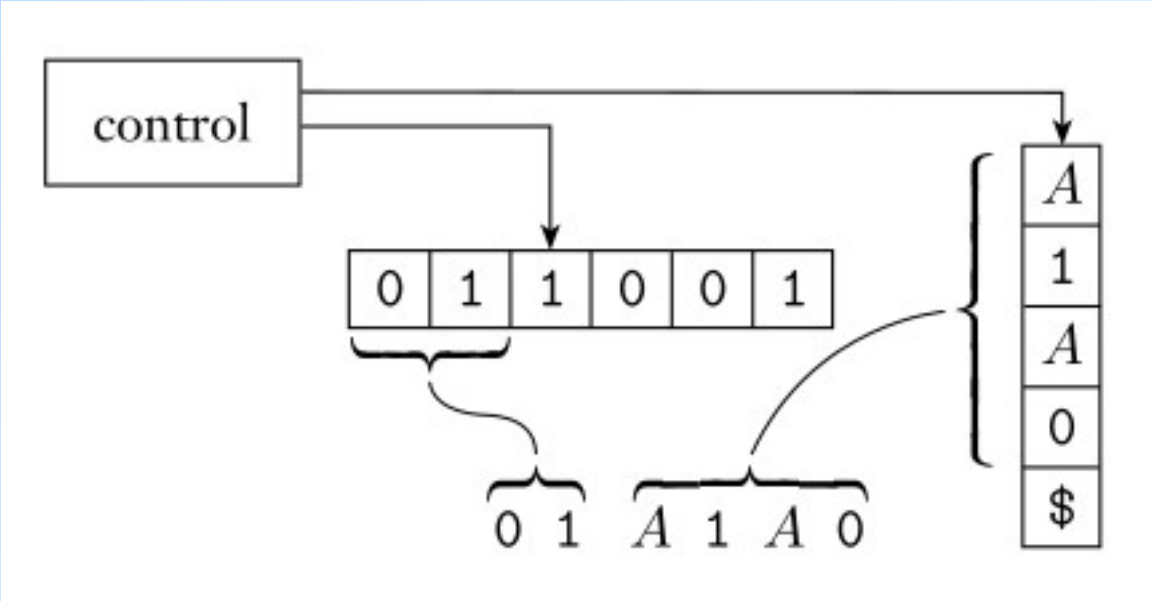
  S → 01A1

  A → 0A | ε

- Como usar a pilha do autômato?
  - Só pode acessar o símbolo do topo da pilha

GLC → AP

- Manter apenas parte da string
  - Símbolos iniciando com a primeira variável



GLC → AP

- Vamos chamar o estado inicial de q_{start}
- $\delta(q_{start},\varepsilon,\varepsilon) = \{(q_1,\$)\}$
- $\delta(q_1,\varepsilon,\varepsilon) = \{(q_{loop},S)\}$

GLC → AP

- Seja V uma variável em G e suas regras:
  - $V \rightarrow a_{11}...a_{1k1}$
  - ...
  - $V \rightarrow a_{n1}...a_{nkn}$
- Fazemos:
- $\delta(q_{loop},\varepsilon,V) = \{(q_{V11},a_{1k1}),...,(q_{Vn1},a_{nkn})\}$

GLC → AP

- Para cada regra $V \rightarrow a_{i1}...a_{iki}$ em G:
- $(q_{Vi1},a_{iki}) \in \delta(q_{loop},\varepsilon,V)$ pela anterior
- Fazemos:
- $\delta(q_{Vi1},\varepsilon,\varepsilon) = \{(q_{Vi2},a_{i(ki-1)})\}$
- $\delta(q_{Vi2},\varepsilon,\varepsilon) = \{(q_{Vi3},a_{i(ki-2)})\}$
- ...
- $\delta(q_{Vi(ki-1)},\varepsilon,\varepsilon) = \{(q_{loop},a_{i1})\}$

GLC → AP

- Se no topo da pilha tiver um terminal, devemos verificar se é igual a entrada:
- $\delta(q_{loop},a,a) = \{(q_{loop},\varepsilon)\}$ em que a é um terminal
- Caso o topo seja $, a string gerada por G é a mesma da entrada do autômato:
- $\delta(q_{loop},\varepsilon,\$) = \{(q_{accept},\varepsilon)\}$

GLC → AP

- Para finalizar, a definição do autômato de pilha P é:
- P = $(Q,\Sigma,\Gamma,\delta,q_{start},F)$ em que:
  - Q é o conjunto de estados utilizados
  - Σ é o conjunto de terminais de G
  - Γ é o conjunto de variáveis e de terminais de G
  - δ foi definido anteriormente
  - F = $\{q_{accept}\}$

Exemplo

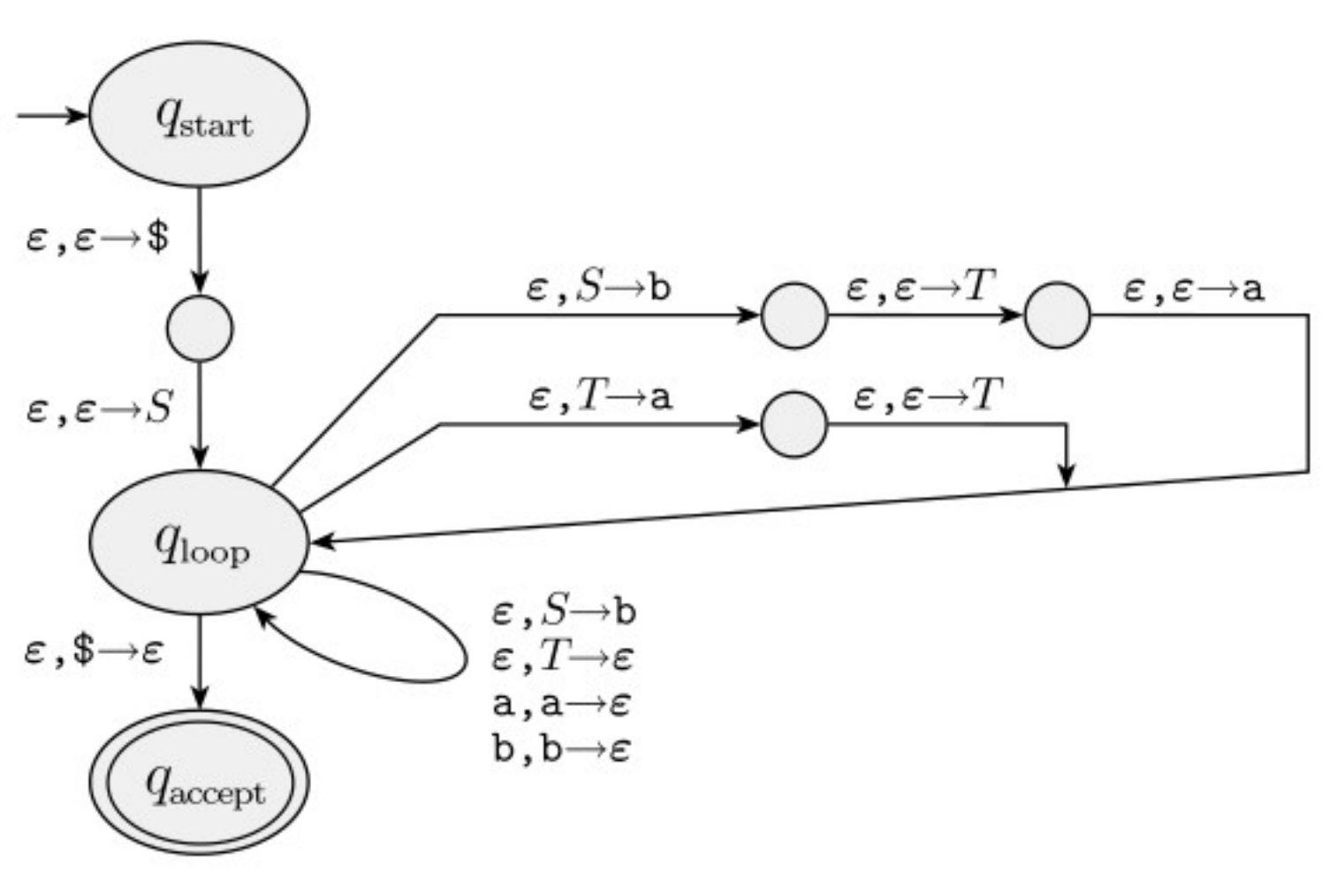
- Seja a gramática livre de contexto G:

  $S \rightarrow aTb \mid b$

  $T \rightarrow Ta \mid \varepsilon$

- Vamos construir o autômato de pilha P equivalente:

Exemplo


q_{start}
$\varepsilon, \varepsilon \to \$$
$\varepsilon, \varepsilon \to S$
$\varepsilon, S \to \mathtt{b}$
$\varepsilon, \varepsilon \to T$
$\varepsilon, \varepsilon \to \mathtt{a}$
$\varepsilon, T \to \mathtt{a}$
$\varepsilon, \varepsilon \to T$
q_{loop}
$\varepsilon, S \to \mathtt{b}$
$\varepsilon, T \to \varepsilon$
$\mathtt{a}, \mathtt{a} \to \varepsilon$
$\mathtt{b}, \mathtt{b} \to \varepsilon$
$\varepsilon, \$ \to \varepsilon$
q_{accept}

Lema

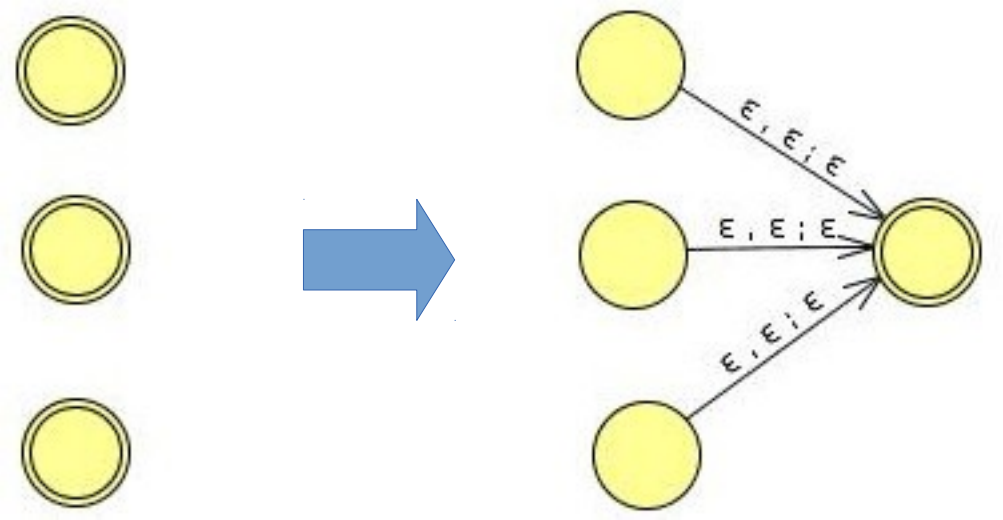
- Se A é uma linguagem e P é um autômato de pilha tal que L(P) = A então A é livre de contexto.

Lema

- Se A é uma linguagem e P é um autômato de pilha tal que L(P) = A então A é livre de contexto.
  - Temos um autômato P e queremos construir uma gramática G que gera as strings que P aceita
  - Vamos simplificar o autômato de pilha P para facilitar

AP → GLC

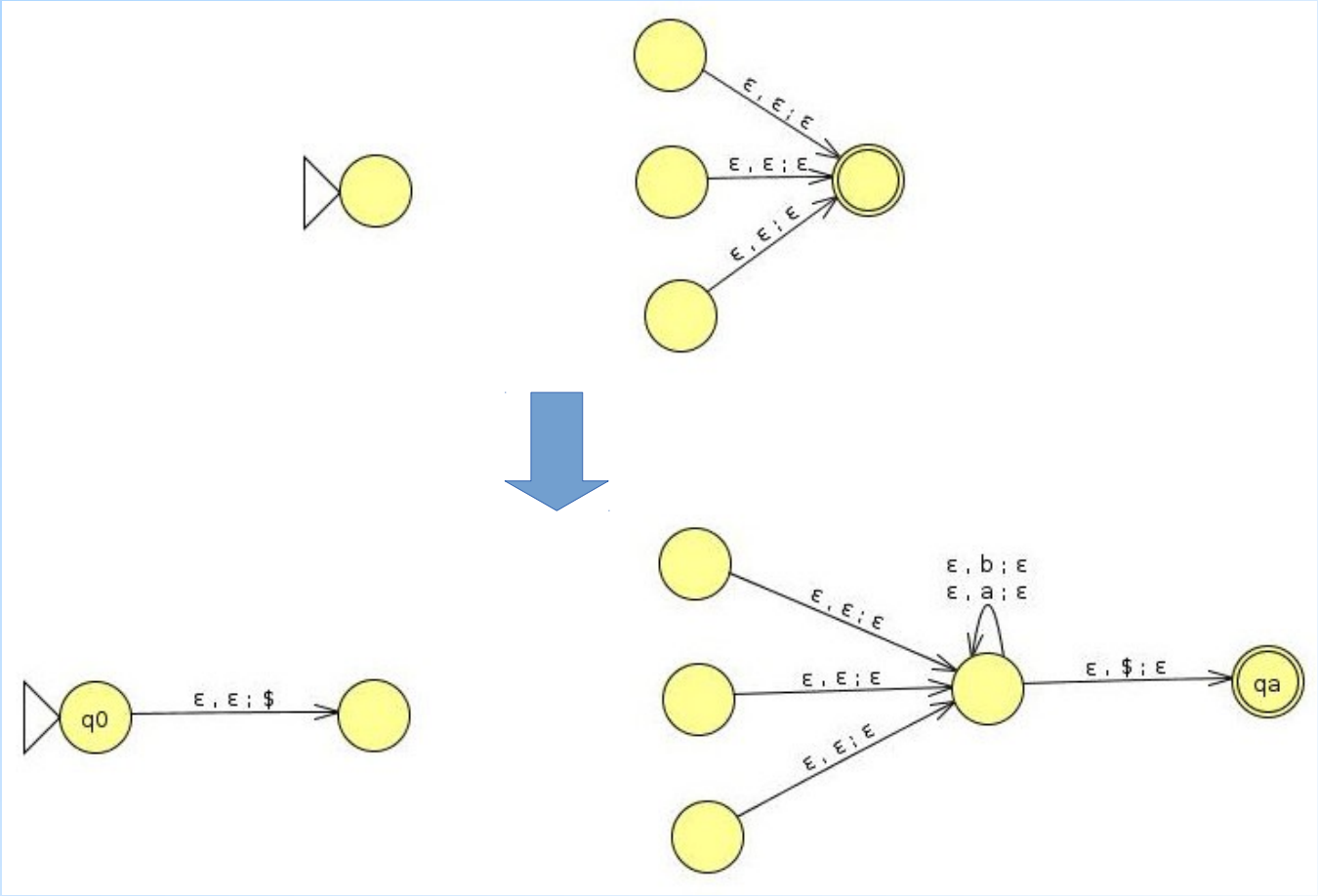
- O autômato de pilha P possui um único estado final q_a
  - Basta criar transições ε dos estados que eram finais para q_a



AP → GLC

- Esvazia a pilha antes de aceitar
  - O estado que era final fica responsável por esvaziar a pilha e depois vai para q_a

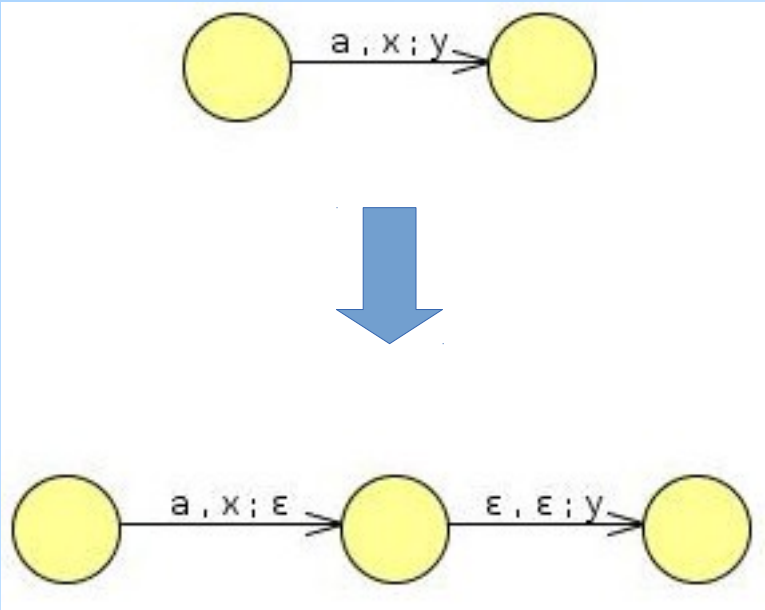
AP → GLC

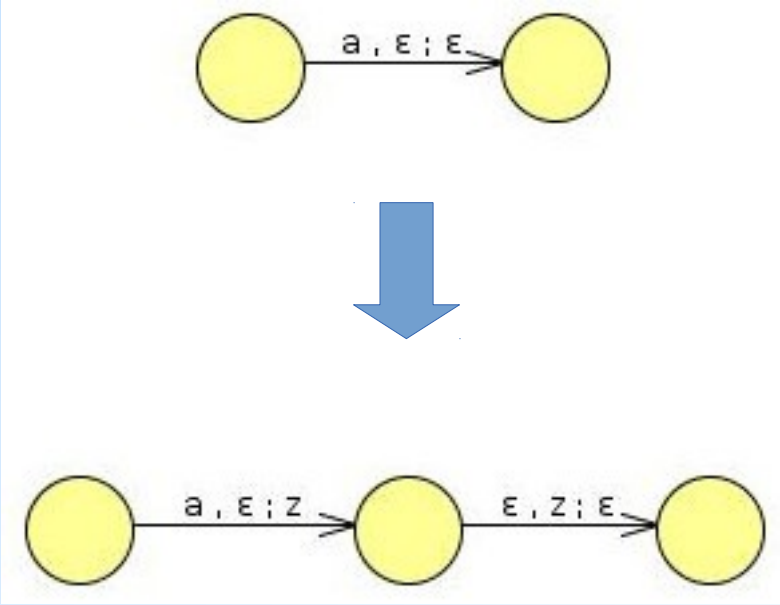


AP → GLC

- Cada transição coloca um símbolo na pilha ou tira um símbolo da pilha mas não ambos
  - Uma transição que faz os dois ao mesmo tempo pode ser dividida em uma que coloca e outra que tira
  - Uma transição que não faz nenhum, pode ser adaptada em uma que coloca um símbolo qualquer e outra que tira esse símbolo

AP → GLC





AP → GLC

- Seja q_s o estado inicial de P
  - Vamos chamar a variável inicial de G por A_{sa}
  - A_{sa} gera as strings que saem de q_s com pilha vazia e chegam em q_a com pilha vazia
  - O primeiro movimento é colocar um símbolo na pilha pois não pode tirar da pilha vazia
  - O último movimento é tirar um símbolo da pilha

AP → GLC

- Se o símbolo que for colocado no início for o mesmo tirado no final
  - A pilha fica vazia apenas no início e no final
- Adicionar regra $A_{sa} \rightarrow aA_{ij}b$
  - Em que a é o símbolo lido no primeiro movimento, b no último, q_i é o estado depois de q_s e q_j o estado antes de q_a

AP → GLC

- Caso contrário,
  - o símbolo colocado no início foi retirado em algum momento antes do final
  - A pilha ficou vazia antes do final
- Adicionamos $A_{sa} \rightarrow A_{si}A_{ia}$
  - Em que q_i é o estado corrente quando a pilha fica vazia

AP → GLC

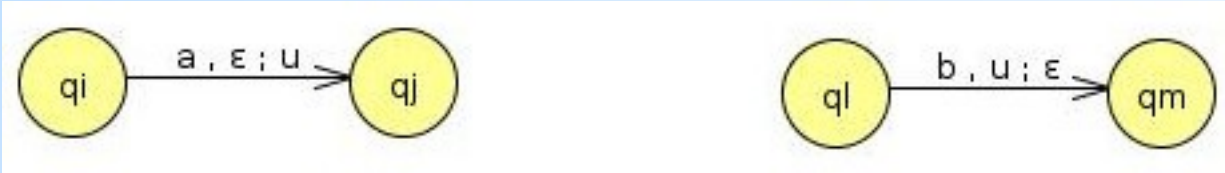
- A mesma ideia pode ser aplicada para qualquer par de estados $q_i, q_j \in Q$
- A_{ij} gera as strings que saem do estado qi e chegam no estado qj sem modificar o que já estava na pilha
  - A pilha pode ser adicionada de elementos no caminho
  - Mas os elementos que já estavam não são alterados

AP → GLC

- Seja $P = (Q,\Sigma,\Gamma,\delta,q_s,\{q_a\})$
- Vamos construir $G = (V,\Sigma,R,A_{sa})$ com $V=\{A_{ij} \mid q_i, q_j \in Q\}$
- Temos que descrever o conjunto de regras R

AP → GLC

- Temos que descrever o conjunto de regras R
  - Se $(q_j, u) \in \delta(q_i, a, \varepsilon)$ e $(q_m, \varepsilon) \in \delta(q_l, b, u)$, colocamos a regra $A_{im} \rightarrow aA_{jl}b$



AP → GLC

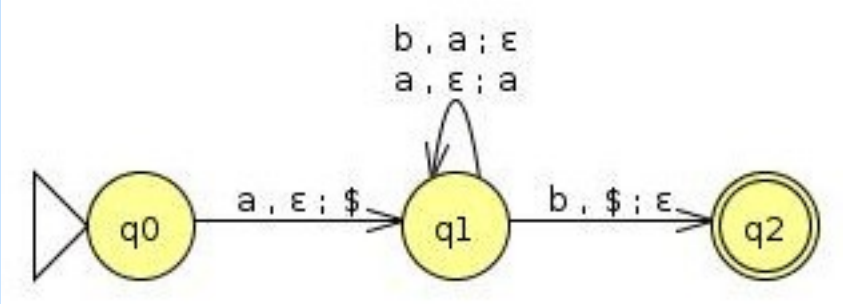
- Temos que descrever o conjunto de regras R
  - Para $q_i, q_j, q_m \in Q$, colocamos $A_{im} \rightarrow A_{ij}A_{jm}$

AP → GLC

- Temos que descrever o conjunto de regras R
  - Para cada $q_i \in Q$, colocamos $A_{ii} \rightarrow \varepsilon$

Exemplo

- Vamos converter o autômato de pilha abaixo em uma gramática livre de contexto



Exemplo

- $A_{02} \rightarrow aA_{11}b$
  - pois $(q_1, \$) \in \delta(q_0, a, \varepsilon)$ e $(q_2, \varepsilon) \in \delta(q_1, b, \$)$
- $A_{11} \rightarrow aA_{11}b$
  - pois $(q_1, a) \in \delta(q_1, a, \varepsilon)$ e $(q_1, \varepsilon) \in \delta(q_1, b, a)$
- $A_{00} \rightarrow A_{00}A_{00} \mid A_{01}A_{10} \mid A_{02}A_{20} \mid \varepsilon$
- $A_{01} \rightarrow A_{00}A_{01} \mid A_{01}A_{11} \mid A_{02}A_{21}$
- ...
- $A_{11} \rightarrow A_{10}A_{01} \mid A_{11}A_{11} \mid A_{12}A_{21} \mid \varepsilon$
- ...

Teorema

- Uma linguagem B é livre de contexto se e somente se existe uma autômato de pilha P tal que L(P) = B.

Linguagens Regulares

- Faça um autômato de pilha para reconhecer a linguagem $\{w \in \{0,1\}^* \mid w \text{ termina em } 1\}$

Teorema

- Se B é uma linguagem regular então B é livre de contexto.

Teorema

- Se B é uma linguagem regular então B é livre de contexto.
  - Se uma linguagem B é regular então existe um AFN A tal que L(A) = B.
  - A partir de A podemos construir um autômato de pilha P tal que L(P) = B.
  - Pelo Teorema anterior, B é livre de contexto.