Autômatos com Pilha

Eduardo Furlan Miranda

Baseado em: GARCIA, A. de V.; HAEUSLER, E. H.
Linguagens Formais e Autômatos. Londrina: EDA, 2017.

Hierarquia de Chomsky

R	$a^n b$	ε, b, ab, aab	AF
LC	$a^n b^n$	ab, aabb	AFP
SC	$a^n b^n c^n$	abc, aabbcc	ALI
I	$a^{2^\wedge n}$	a, aa, aaaa	MT

apenas estas

Gramáticas	Regras	Ex. ling. geradas/Reconhecedor
GR (tipo 3) Regulares	A→aB, A→b, (A→ε apenas para o símbolo inicial, se permitido) A, B ∈ V (variáveis) a, b ∈ T (terminais)	• { ε, b, ab, aab, aaab, ... } = $\{ a^n b \mid n \geq 0\} \cup \{\varepsilon\}$ • Autômato finito
GLC (tipo 2) Livres de Contexto	A → α A ∈ V , $\alpha \in (V \cup T)^*$ A: 1 única variável	• { ab, aabb, aaabbb, aaaabbbb, ... } = $\{ a^n b^n \mid n > 0 \}$ • Autômato com pilha
GSC (tipo 1) Sensíveis ao Contexto	α → β $\alpha \in (V \cup T)^+$, $\beta \in (V \cup T)^*$ $\mid \alpha \mid \leq \mid \beta \mid$ S → ε, se S não aparece do lado direito de nenhuma regra	• { abc, aabbcc, aaabbbccc, ... } = $\{ a^n b^n c^n \mid n > 0 \}$ • Autômato linearmente limitado
GI (tipo 0) Irrestrita ou geral	α → β $\alpha, \beta \in (V \cup T)^*$ α: pelo menos 1 símbolo de V	• { a, aa, aaaa, aaaaaaaa, ... } = $\{ a^{2^n} \mid n \geq 0\}$ • Máquina de Turing

Autômato com pilha (AP)

AF-ε = um tipo de AFND

- AP : tipo de autômato que reconhece as linguagens livres de contexto (LLC)

- Considere o alfabeto $\Sigma = \{ a , b \}$, e L_1 a linguagem sobre Σ definida como $L_1 = \{ a^n b^n \mid n \geq 1 \}$

- A L_1 é livre de contexto (LLC)

- Uma vez que L_1 não é regular (tipo 3), não existe um AFD (sem plilha) que reconheça L_1 (visto anteriormente)
  - Vamos propor um algoritmo para reconhecer cadeias da LLC L_1 que estende o comportamento de um AF-ε ao adicionar uma pilha
    - inspirado no funcionamento de autômatos de pilha (AP)

Um AF-ε (Autômato Finito ε) pode, a qualquer momento, sem ler nenhum símbolo da entrada, passar para o estado q2. Isso introduz não determinismo, pois o autômato pode escolher "fazer" ou "não fazer" a transição ε.

Autômato com pilha (AP) para L_1

- a pilha é inicializada com Z
- lê
- topo da pilha: compara "a" e o desempilha
- empilha "aa"

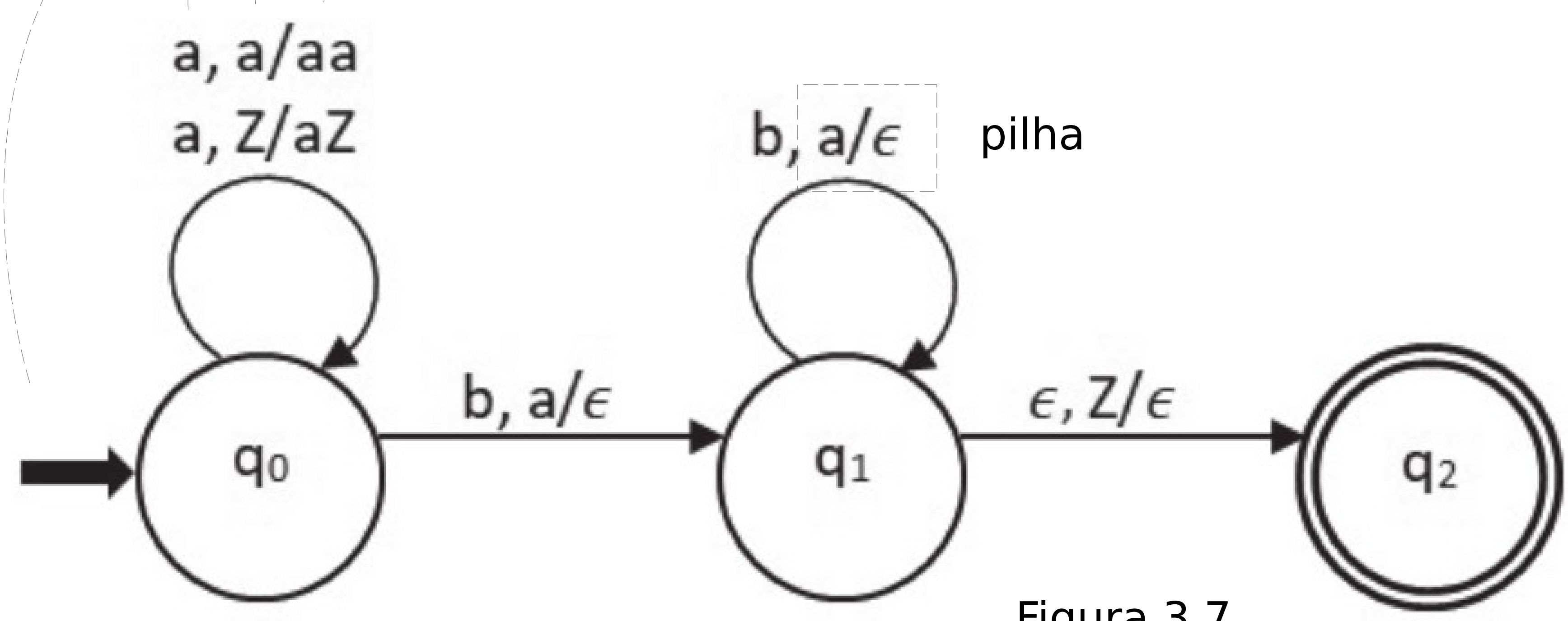


Figura 3.7

- Cada teste de condição depende de 3 variáveis
  - Estado, Entrada, Topo da pilha

Autômato com pilha para L_1

- A pilha é inicializada com o símbolo Z e a transição entre os estados q_0 e q_1 tem o texto b, a/ε

- b, a/ε : estando no estado q_0, ao ler um b e tendo um símbolo a no topo da pilha, desempilha a, e vai para o estado q_1
  - ε significa que nada é empilhado

- a, Z/aZ : estando no estado q_0, ao ler um a, e tendo um Z no topo da pilha, irá desempilhar Z, empilhar aZ, e permanecer no estado q_0 (vide a seta no diagrama)

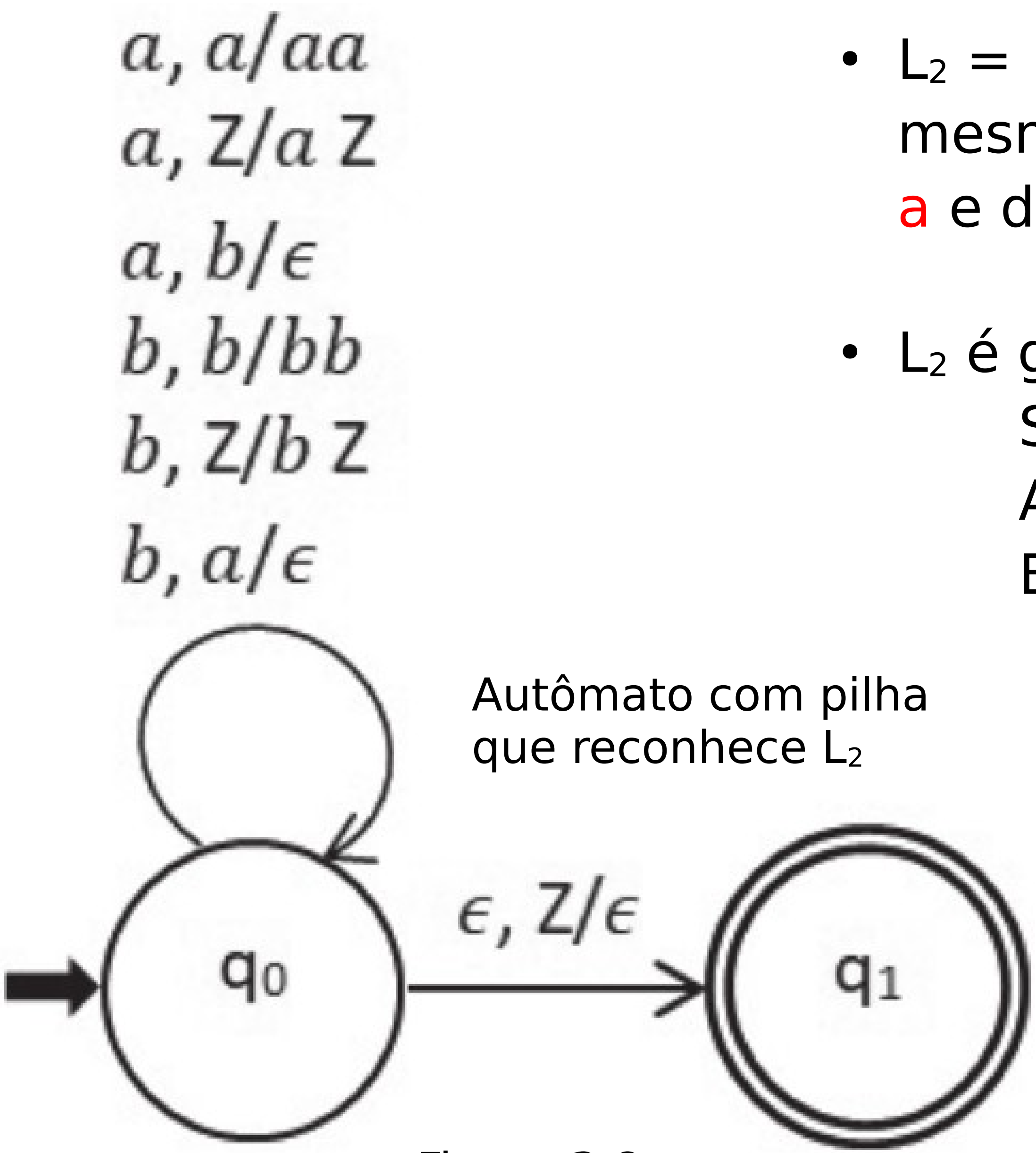
- A aceitação se dá se o autômato parar no estado final (q_2) após ler toda a entrada

Exemplo de reconhecedor para L_1

Uma das possíveis implementações

```
estado = 0
pilha = ['Z']  # Inicializa a pilha com o símbolo inicial
while True:
    entrada = input()  # Lê o próximo caractere da entrada
    topo = pilha[-1]  # Obtém o topo da pilha
    if estado == 0 and entrada == 'a' and (topo == 'a' or topo == 'Z'):
        estado = 0
        pilha.append('a')  # Empilha 'a' para cada 'a' na entrada
    elif estado == 0 and entrada == 'b' and topo == 'a':
        estado = 1
        pilha.pop()  # Remove 'a' da pilha ao encontrar 'b'
    elif estado == 1 and entrada == 'b' and topo == 'a':
        estado = 1
        pilha.pop()  # Continua desempilhando 'a' para cada 'b'
    elif estado == 1 and entrada == 'a' and topo == 'a':
        estado = 1
        pilha.pop()  # Desempilha 'a' se encontrar um 'a' após 'b'
    elif estado == 1 and entrada == '' and topo == 'Z':  # EOF = ‘’
        return 1  # Aceita a cadeia = pilha está na configur. inicial
    else:
        return 0  # Rejeita a cadeia
```

AP para L_2



Autômato com pilha
que reconhece L_2

Figura 3.8

- $L_2 = \{ w \in \{ a, b \}^* \mid w$ tem a mesma quantidade de caracteres a e de caracteres b $\}$
- L_2 é gerada pela GLC

$$S \rightarrow aB \mid b \mid \varepsilon$$
$$A \rightarrow bS \mid aBB$$
$$B \rightarrow aS \mid bAA$$

- "L dois é igual ao conjunto de todas as strings w pertencentes ao alfabeto { a, b } estrela tal que ...”
- { a, b }*: "conjunto de todas as strings possíveis formadas pelos caracteres 'a' e 'b', incluindo a cadeia vazia"

AP para L_2

- A regra de transição a, Z/aZ significa que ao ler um a, se Z está no topo da pilha, irá desempilhar este símbolo e empilhar aZ

- a, a/aa ; a, Z/aZ ; a, b/ε : ao ler um a o autômato pode:
  - se o topo da pilha for a ou Z , desempilha a ou Z , e empilha aa ou aZ
  - se o topo da pilha for b , desempilha o b   (ε = não faz nada)

- Como ele vai empilhando, a quantidade de símbolos na pilha indica quantas vezes aquele símbolo apareceu na entrada

- Ao final da leitura, se a pilha está apenas com o símbolo inicial, usa-se a a transição ε, Z/ε que leva ao estado final q_1

Definição de AP

- Um Autômato com Pilha é uma tupla

$$P = (Q , \Sigma , \Gamma , \delta , q_0 , Z , F) \qquad , \text{onde:}$$

  - Q : conjunto finito de estados
  - Σ : alfabeto de entrada
  - Γ : alfabeto da pilha (os símbolos que podem ser escritos na pilha)
  - δ : função de transição de um AP
  - $q_0 \in Q$: estado inicial
  - $Z \in \Gamma$: símbolo inicial da pilha
  - $F \subseteq Q$: conjunto de estados finais

Σ sigma
Γ gama
ε épsilon
δ delta
$\wp$ conjunto potência
$\in$ pertence a
$\subseteq$ está contido em

(continua)

Função de transição (δ) de um AP

- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \wp(Q \times \Gamma^*)$
  - δ : função de transição; define como o AP muda de estado e manipula a pilha ao ler um símbolo de entrada ou a cadeia vazia
  - $(\Sigma \cup \{\varepsilon\})$: símbolo de entrada atual; pode ser um símbolo do alfabeto de entrada Σ ou a string vazia ε
    - A inclusão de ε permite transições que não consomem nenhum símbolo da entrada
  - $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$: produto cartesiano que representa as possíveis combinações de:
    - Um estado atual (de Q)
    - Um símbolo de entrada OU a string vazia (de $(\Sigma \cup \{\varepsilon\})$)
    - Um símbolo no topo da pilha (de Γ)

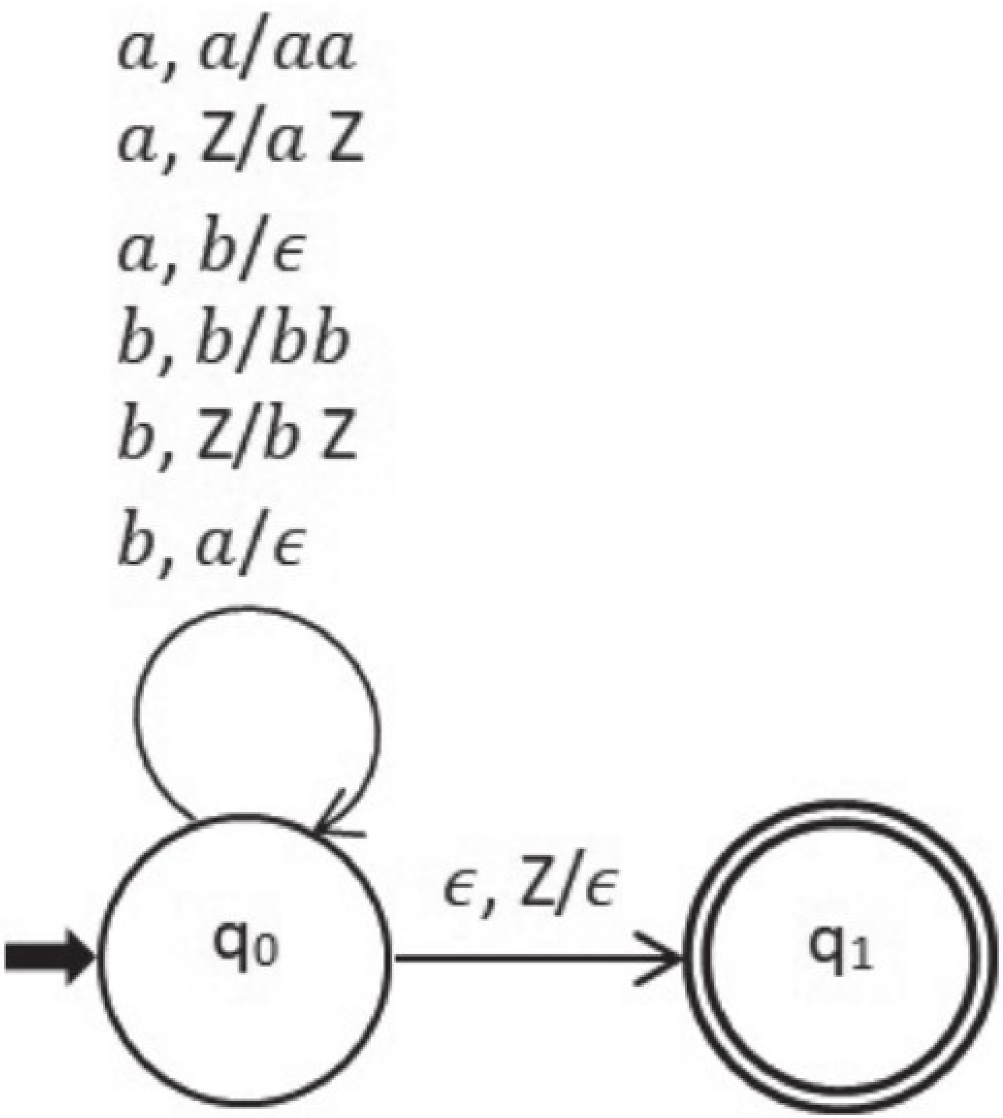
(continua)

Função de transição (δ) de um AP

(continuação)

- $\rightarrow \wp(Q \times \Gamma^*)$: indica que o resultado de δ (função de transição) é um conjunto potência "$\wp$" (conjunto de todos os subconjuntos possíveis de $(Q \times \Gamma^*)$)
  - Cada par dentro desse subconjunto consiste em:
    - Um novo estado (em Q): q estado para o qual o autômato transita
    - Uma sequência de símbolos de pilha (em Γ^*)
      - A sequência de símbolos que será colocada na pilha
        - Pode ser vazia (ε), o que significa que nada é adicionado à pilha, ou
        - pode conter um ou mais símbolos, que são empilhados na ordem em que aparecem na sequência

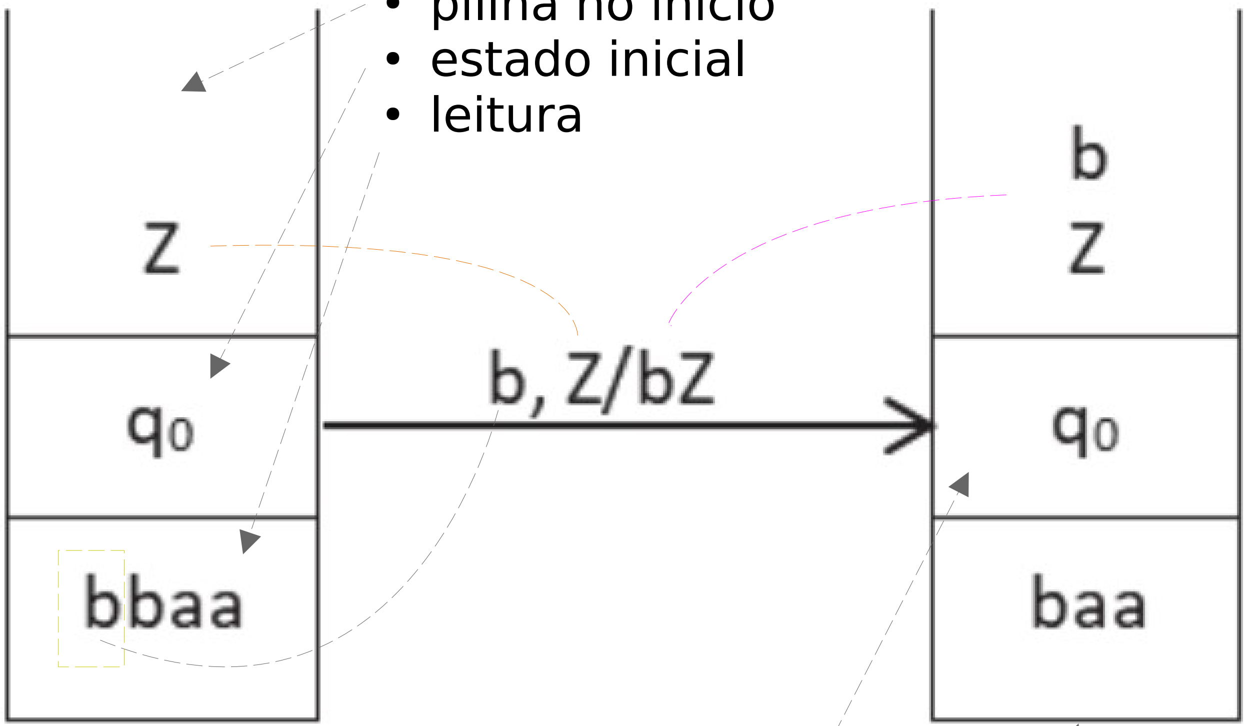
Exemplo - Fig. 3.8 (slide 7)



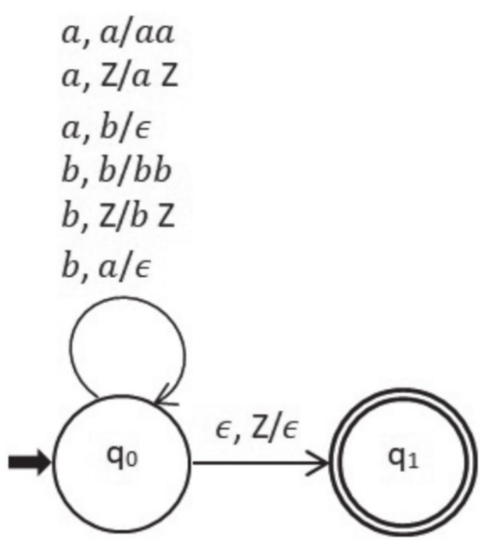
- $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, onde:
  - $Q = \{ q_0, q_1 \}$
  - $\Sigma = \{ a, b \}$
  - $\Gamma = \{ a, b, Z \}$
  - $\delta (q_0, a, a) = \{(q_0, aa)\}$
    $\delta (q_0, a, Z) = \{(q_0, aZ)\}$
    $\delta (q_0, a, b) = \{(q_0, \varepsilon)\}$
    $\delta (q_0, b, b) = \{(q_0, bb)\}$
    $\delta (q_0, b, Z) = \{(q_0, bZ)\}$
    $\delta (q_0, b, a) = \{(q_0, \varepsilon)\}$
    $\delta (q_0, \varepsilon, Z) = \{(q_1, \varepsilon)\}$
  - $F = \{ q_1 \}$

Funcionamento do AF

- pilha no início
- estado inicial
- leitura



Primeiro passo do autômato com pilha para a linguagem L_2 lendo bbaa



- A linha mais abaixo significa o que falta ler de uma cadeia
- Na linha imediatamente superior temos o estado
- Acima do estado temos o conteúdo da pilha
- À esquerda é a configuração inicial do autômato antes de começar a ler a cadeia bbaa ; à direita, a configuração seguinte, após ler o primeiro b

(continua)

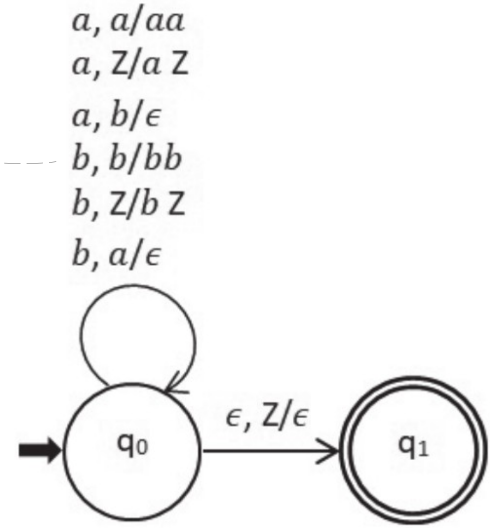
= slide anterior

b,b/bb

Z	b Z	b b Z	b Z	Z	ϵ
q_0	q_0	q_0	q_0	q_0	q_1
bbaa	baa	aa	a	ϵ	ϵ

nada é empilhado

não consome nenhum símbolo da entrada



- Funcionamento passo a passo do autômato até atingir o estado final no reconhecimento da cadeia bbaa

- Quando o autômato vai para o estado final q_1, a entrada já foi totalmente lida, situação representada pelo caractere ε na linha de baixo, e portanto a cadeia é reconhecida

- Os dados representados em cada coluna (caracteres restantes a serem lidos; estado e conteúdo da pilha) são chamados de conjuntos de configuração do autômato com pilha

- Em geral são representados como uma tripla ordenada da forma: (estado, entrada, conteúdo da pilha)

- A configuração inicial pode assim ser representada por: (q_0 , bbaa, Z), enquanto o último estado ser representado como (q_1 , ε , ε)

- $\vdash$: "deriva", símbolo que representa a relação de transição entre configurações, ex.: (q_0 , bbaa, Z) $\vdash$ (q_0 , baa, bZ)

Outra forma de representar

- Transição entre configurações do AP para a linguagem L_2 lendo bbaa :

topo da pilha

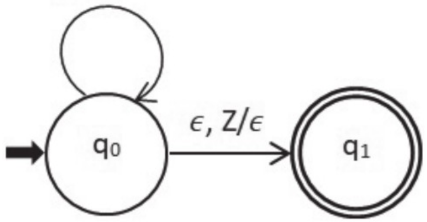
$(q_0 , bbaa, Z) \vdash (q_0 , baa, bZ) \vdash (q_0 , aa, bbZ) \vdash (q_0 , a, bZ) \vdash (q_0 , \varepsilon, Z) \vdash (q_1 , \varepsilon, \varepsilon)$

- Quando a transição entre configurações se dá em zero ou mais passos, representamos esta sequência de transições por $\vdash^*$

- Para as transições: $(q_0 , bbaa, Z) \vdash^* (q_1 , \varepsilon , \varepsilon)$
  - q_1 = estado final = bbaa aceito pelo autônomo de pilha

- Dado um autômato com pilha $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$,
  - a linguagem reconhecida por P por estado final é definida como
    - o conjunto de todas as cadeias $w \in \Sigma^*$ que, quando processadas a partir da configuração inicial (q_0, w, Z),
      - levam P a uma configuração da forma $(q_f, \varepsilon, \gamma)$,
        - onde $q_f \in F$ e $\gamma \in \Gamma^*$

- Definimos a linguagem aceita por P por estado final, L(P), como
  - $L(P)=\{w \in \Sigma^* \mid (q_0, w, Z) \vdash^* (q, \varepsilon, \alpha)$ para algum $\alpha \in \Gamma^*$ e algum $q \in F\}$
    - Uma palavra w é aceita se, partindo do estado inicial q_0 com w na entrada e Z na pilha, o autômato atinge um estado final ($q \in F$) após consumir toda a entrada (ε) , ou seja,
      - a palavra é aceita se existe um caminho que leva a um estado final após processar toda a entrada



- Para o autômato P_2 da Fig. 3.8, $(P_2) = L_2$
  - As cadeias que levam ao estado final q_1 são exatamente aquelas que deixam a pilha vazia
  - Para facilitar a escrita de um autômato com pilha P,
    - deixamos de colocar um estado final e dizemos que a linguagem reconhecida pelo AP são as sentenças que levam o autômato a uma configuração onde a pilha está vazia
      - Neste caso, dizemos que a linguagem é reconhecida por pilha vazia e escrevemos esta linguagem como N(P)

- para P_2 : $N(P_2) = L(P_2) = L_2$

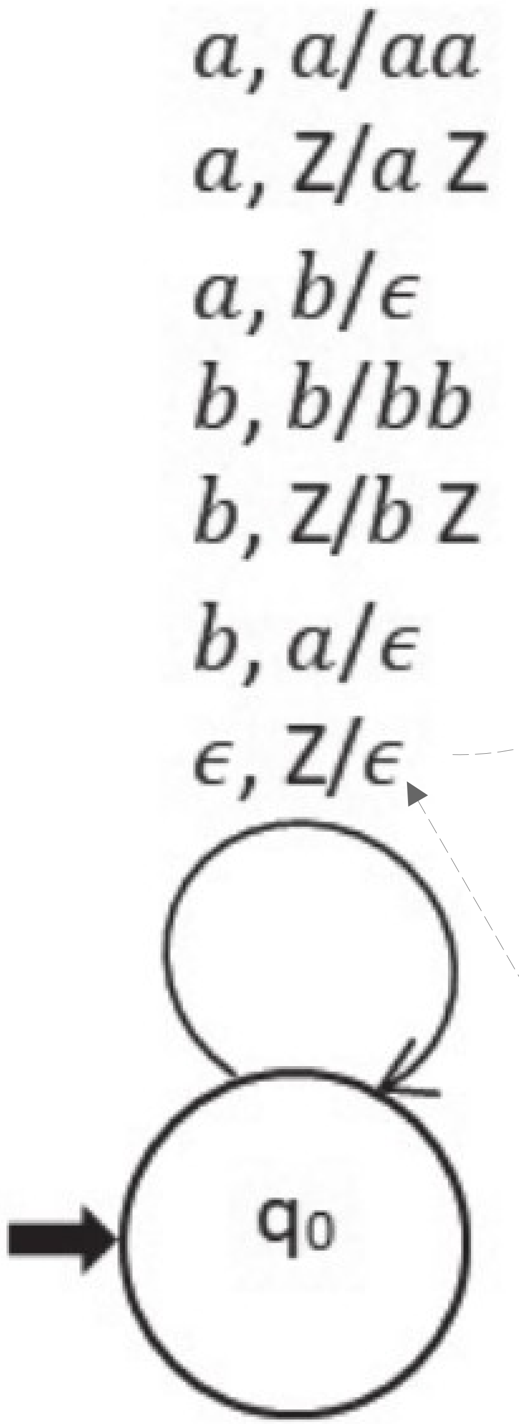
- $N(P_2)$: linguagem reconhecida, pelo autômato com pilha P_2 , por pilha vazia
- $L(P_2)$: linguagem aceita pelo autômato com pilha P_2 , por estado final
- L_2 : linguagem que o autômato reconhece

Pilha vazia X estado final

- Dado um autômato com pilha que reconheça uma linguagem L por pilha vazia, sempre é possível obter um outro autômato com pilha que reconheça L por estado final e vice-versa

- O reconhecimento por pilha vazia pode simplificar o desenho de um autômato

L2 por pilha vazia



- Este autômato reconhece, por pilha vazia, a mesma linguagem (L_2) que o autômato da Fig. 3.8, mas tem um estado a menos
- A regra δ (q_0 , ε , Z) = (q_0 , ε) indica que o autômato pode esvaziar a pilha ao encontrar o símbolo Z no topo da pilha, sem consumir nenhum símbolo de entrada (ε)
  - essa regra pode ser aplicada a qualquer momento, esvaziando a pilha

AP que reconhece GLC

- Existe um método para, dada uma Gramática Livre de Contexto (GLC), produzir um AP que reconhece a linguagem gerada pela gramática (L)

- Ex.: dada a Gramática Livre de Contexto G_{exp}
  - $S \rightarrow S + S$
  - $S \rightarrow S \times S$
  - $S \rightarrow (S)$
  - $S \rightarrow 1$

gera as expressões aritméticas formadas com as operações soma e multiplicação e o número '1'

(continua)

AP equivalente

- $P = (\{ q_0 \}, \{ 1, +, \times, (,) \}, \{ S, 1, +, \times, (,) \}, \delta, q_0, S, \varnothing)$
  - Estados
  - Alfabeto de entrada
  - Alfabeto da pilha
  - Função de Transição δ

    ex.: “S → S + S”, A = “S”, w = “S + S”

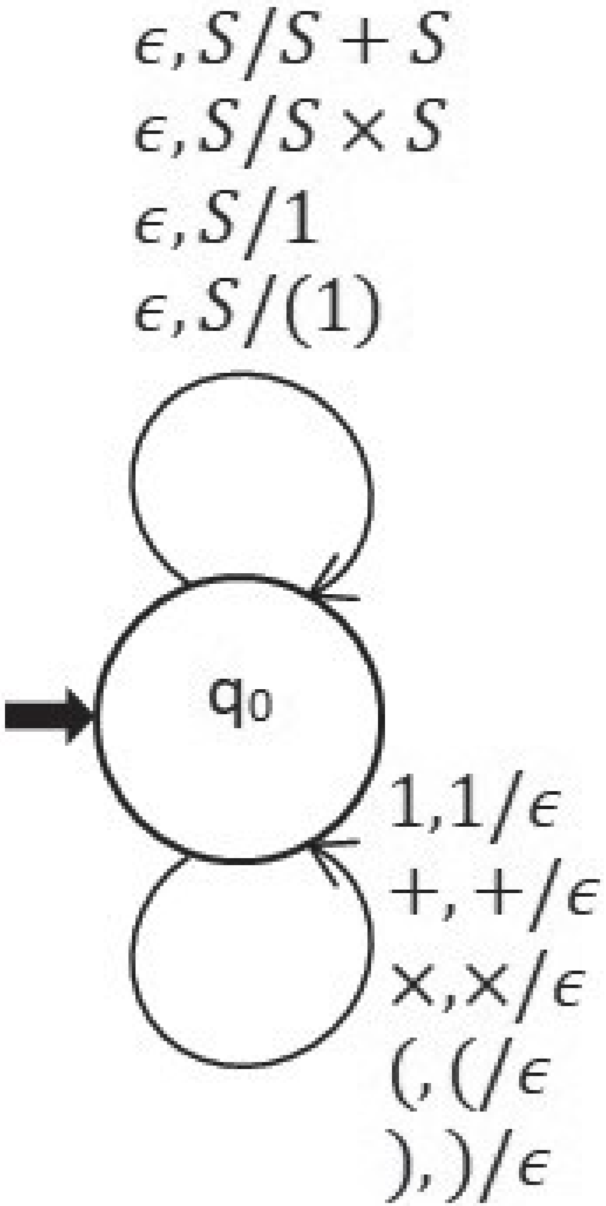
    - Para cada regra A → w da gramática, o autômato desempilha A e empilha w sem consumir nenhum símbolo da entrada (=ε)
    - Para cada símbolo a ∈ { 1 , + , × , (,) }, se a está no topo da pilha, ele é desempilhado ao ler a da entrada. Nada é empilhado.
  - Estado inicial
  - Símbolo inicial da pilha
  - Conjunto de estados finais: ∅ (reconhecimento por pilha vazia)

AP produzido

S = símbolo inicial

- Regras de redução (empilhamento)
  - ε, S/S + S
  - ε, S/S × S
  - ε, S/1
  - ε, S/(1)

- Regras de leitura e desempilhamento
  - 1, 1/ε
  - +, +/ε
  - ×, ×/ε
  - (, (/ε
  -),)/ε



- O autômato reconhece L(Gexp) por pilha vazia
- Para reconhecer a linguagem por estado final, é necessário mais um estado
- Conforme já visto é possível construir um AP equivalente a partir de uma GLC
  - Também é possível fazer o processo inverso: dado um AP construir uma GLC equivalente
- A versão determinística do AP é amplamente usada nos compiladores para programar a análise sintática descendente
- Também existem os autômatos com pilha ascendentes (APA), cuja versão determinística é usada na análise sintática ascendente