Aspectos Formais da Computação

Prof. Sergio D. Zorzo
Departamento de Computação - UFSCar
1º semestre / 2017
Aula 11

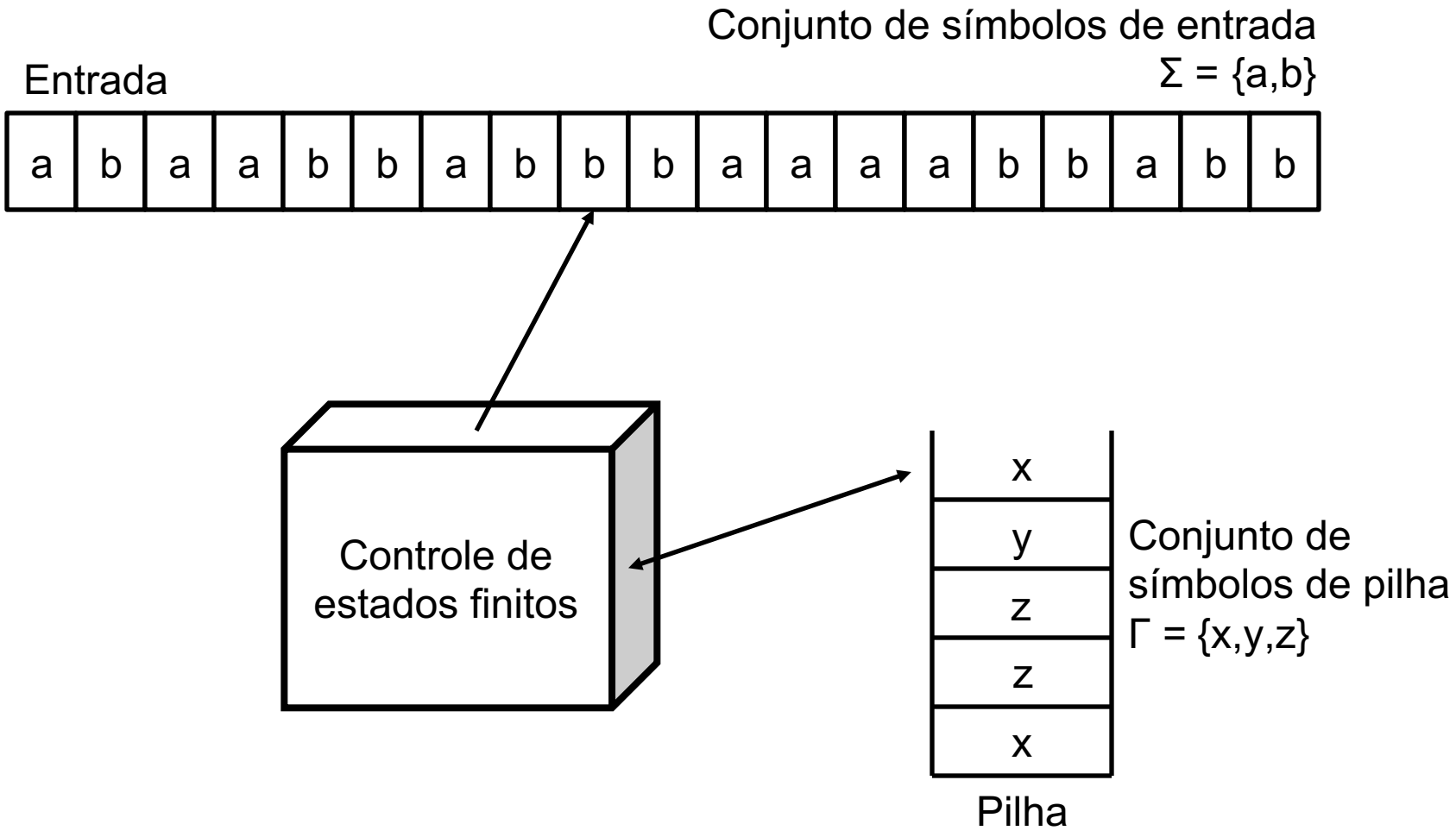
Autômatos de Pilha (Pushdown Automata – PDA)

Autômatos de pilha

- é um ε-NFA com a inclusão de uma pilha

- Pilha = memória adicional
  - Além dos estados finitos
  - Quantidade infinita de informações
- A pilha pode ser lida, aumentada e diminuída apenas no topo
- Autômatos de pilha reconhecem todas as linguagens livres de contexto e apenas as linguagens livre de contexto.....

Autômatos de pilha
(possivel interpretação deste mecanismo ...)



Autômatos de pilha
(possivel interpretação deste mecanismo ...)

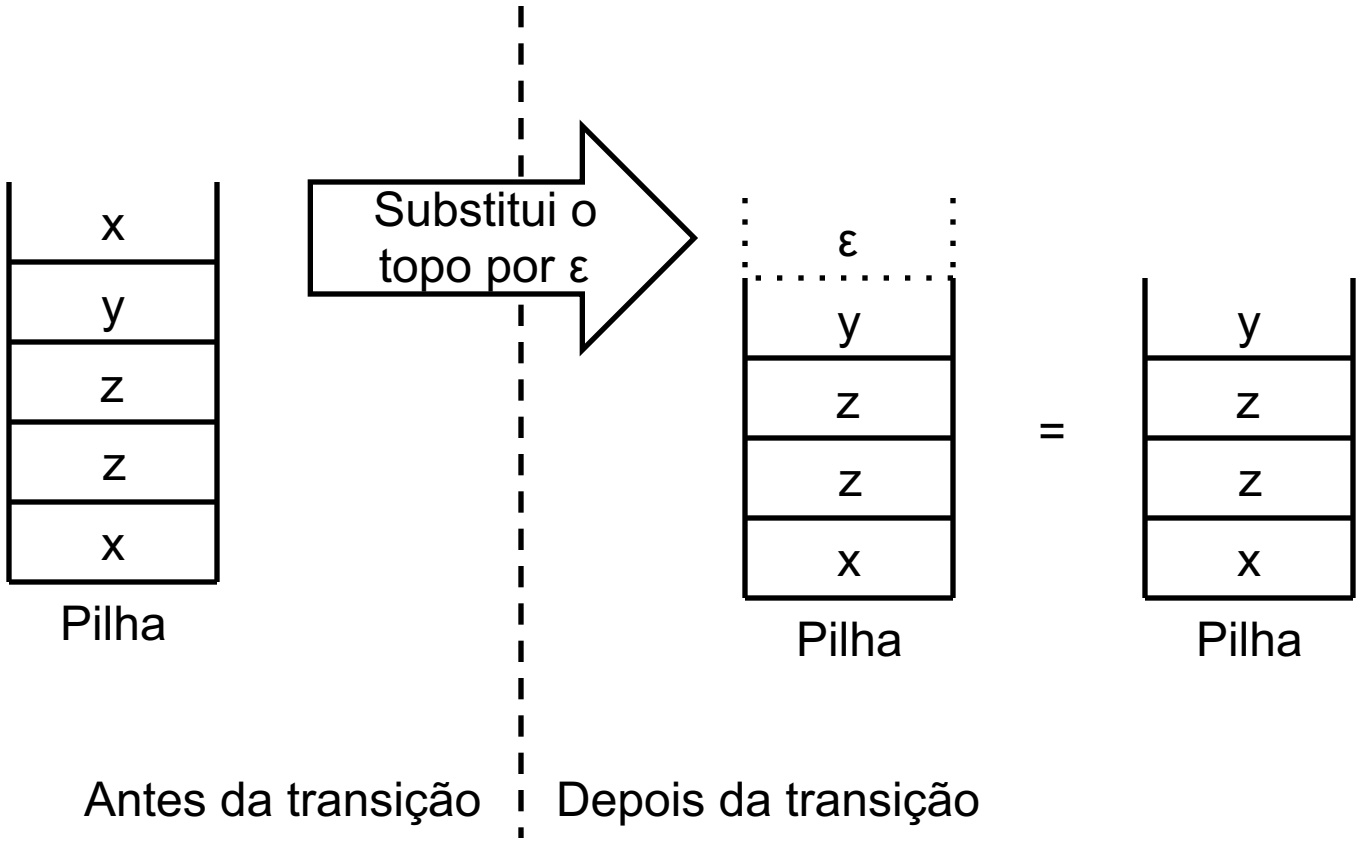
- O controle de estados finitos lê as entradas, um símbolo de cada vez
- O controle tem permissão para observar o símbolo no topo da pilha
  - Pode basear a transição:
    - em seu estado atual
    - no símbolo de entrada
    - no símbolo presente no topo da pilha
  - Opcionalmente, a entrada pode ser ε
    - Ou seja, podem haver transições “espontâneas”

Autômatos de pilha
(possivel interpretação deste mecanismo ...)

- Em uma transição, o autômato de pilha:
  - Consome da entrada o símbolo que utiliza na transição
    - Se for uma transição espontânea, nenhum símbolo de entrada é consumido
  - Vai para um novo estado, que pode ou não ser o mesmo estado anterior
  - Substitui o símbolo no topo da pilha por qualquer cadeia

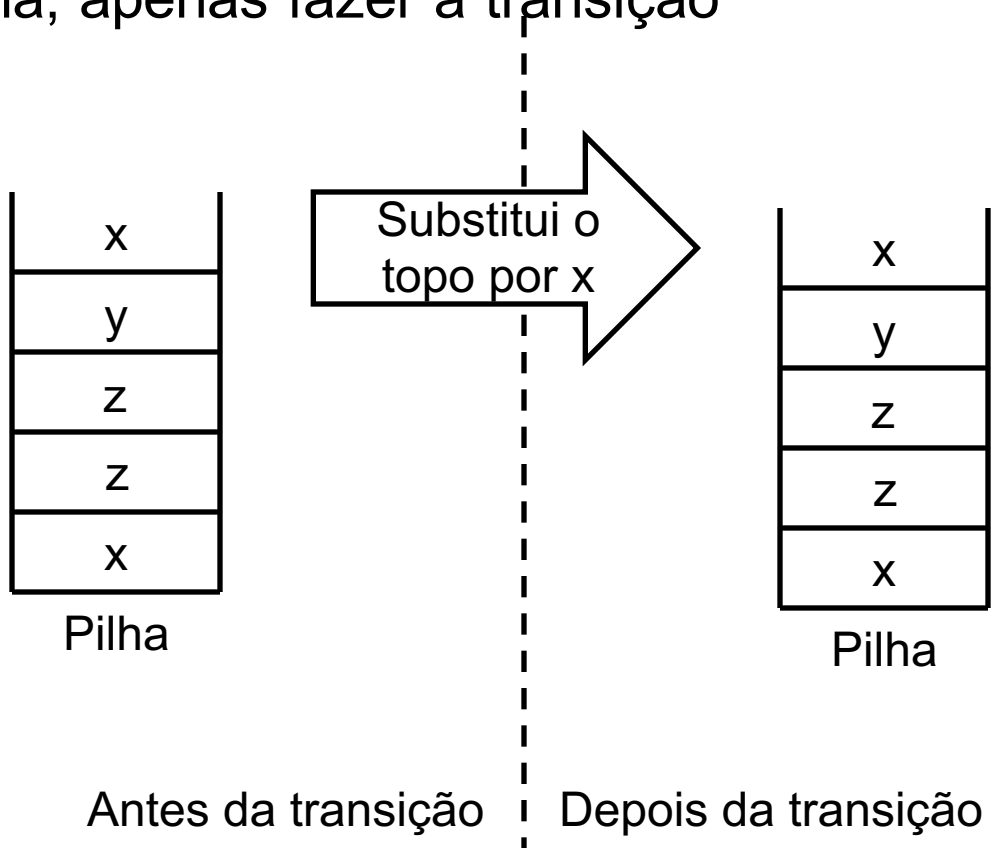
Autômatos de pilha

- Substituição de símbolo no topo da pilha
  - Se for ε, equivale a uma extração (pop) da pilha



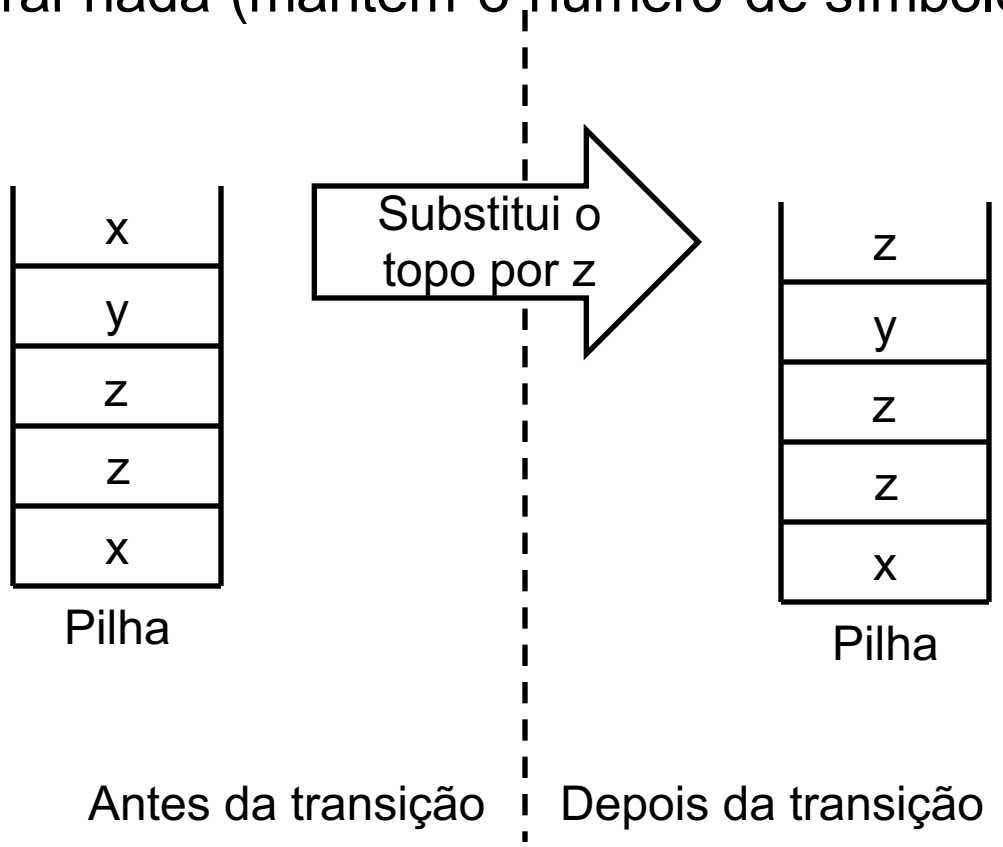
Autômatos de pilha

- Substituição de símbolo no topo da pilha
  - Se for o mesmo que já estava, equivale a não mudar a pilha, apenas fazer a transição



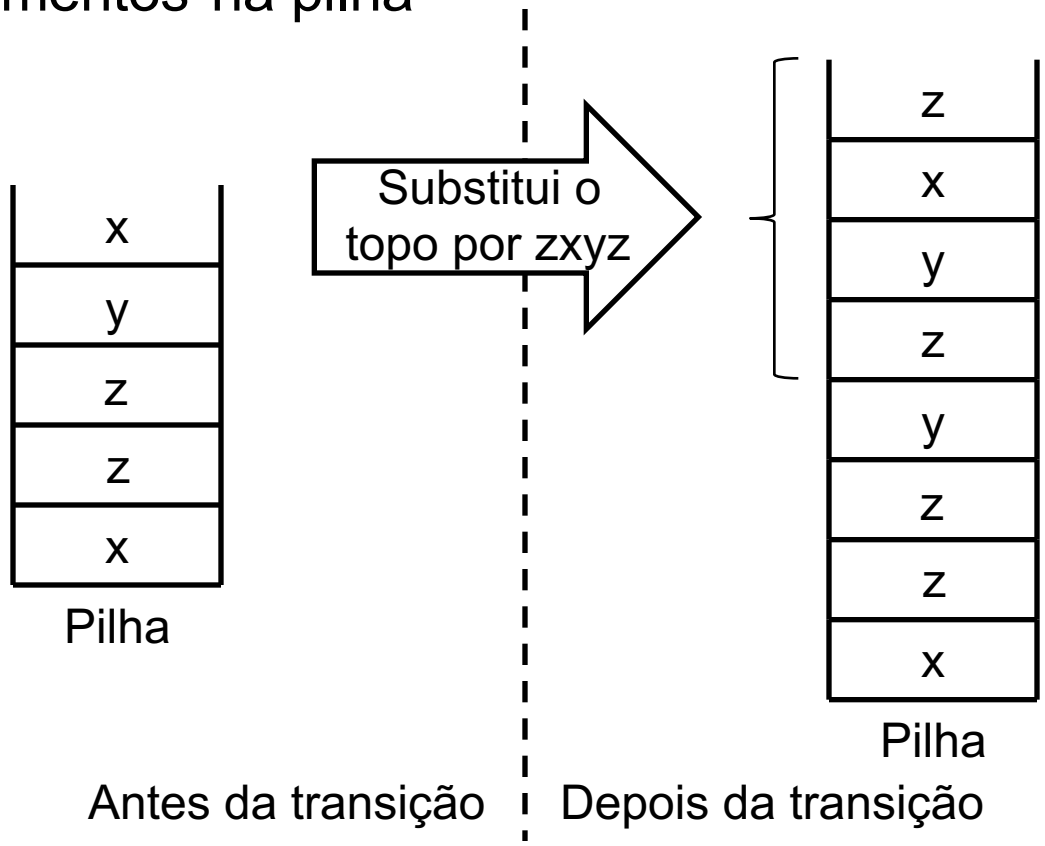
Autômatos de pilha

- Substituição de símbolo no topo da pilha
  - Se for outro símbolo, altera o topo, mas não insere nem extrai nada (mantém o número de símbolos na pilha)



Autômatos de pilha

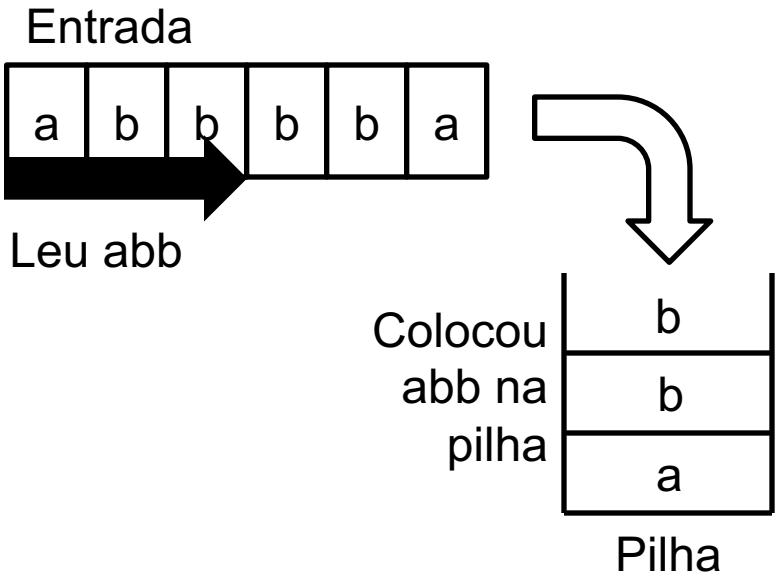
- Substituição de símbolo no topo da pilha
  - Se for uma cadeia com dois ou mais símbolos, insere elementos na pilha



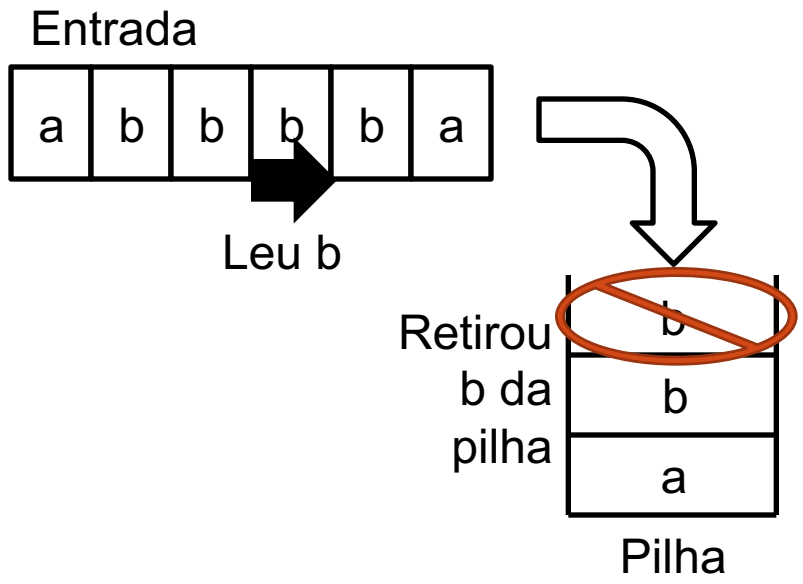
Autômatos de pilha

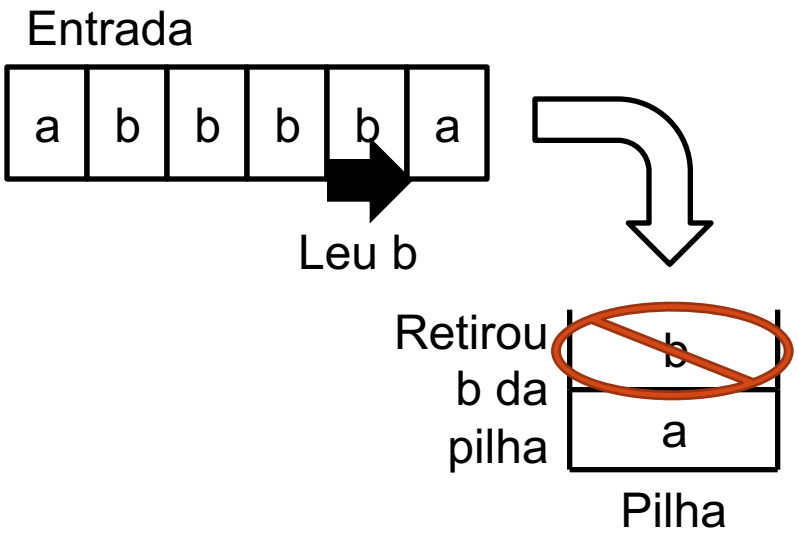
- Ex: L = {ww^R | w está em {0,1}*}
  - Três estados: q0, q1 e q2
  - q0 = ainda não chegou no meio da entrada
  - q1 = passou do meio da entrada
  - q2 = chegou no fim da entrada
- Funcionamento:
  - Começa em q0
  - Estando em q0, vai lendo a entrada e colocando uma cópia do símbolo lido no topo da pilha
  - No meio da pilha (usando o poder de oráculo do não-determinismo), muda para q1
  - Estando em q1, vai lendo a entrada. Compara-se o símbolo lido com o símbolo no topo da pilha. Se for igual, substitui o topo da pilha por ε
  - No final da entrada, se a pilha estiver vazia (topo = ε), aceita a cadeia

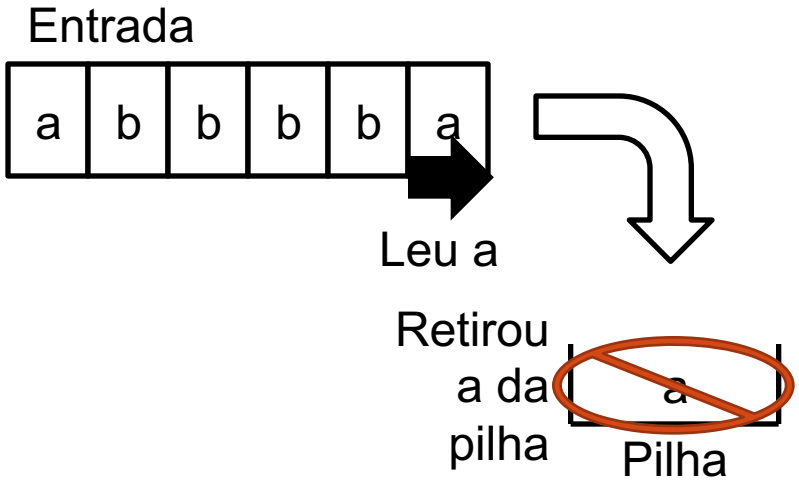
Autômatos de pilha

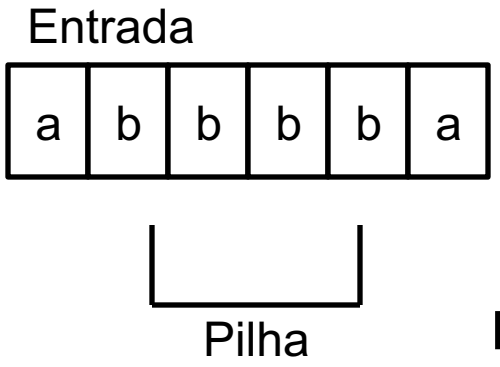


Autômatos de pilha









Fim da entrada
+ pilha vazia
= cadeia aceita

Autômatos de pilha

- Definição formal de um PDA (*PushDown Automata*)

$$\mathbf{P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)}$$

- Onde:
  - Q = Um conjunto finito de estados
  - Σ = Um conjunto finito de símbolos de entrada
  - Γ = Um alfabeto de pilha finito – conjunto de símbolos que temos permissão para inserir na pilha (pode incluir elementos de Σ)
  - δ = Função de transição – governa o comportamento do autômato
  - q_0 = Estado inicial
  - Z_0 = Símbolo de início – Inicialmente, a pilha do PDA consiste em uma instância desse símbolo e em nada mais
  - F = Conjunto de estados de aceitação

Autômatos de pilha

- Função de transição
  - $\delta : Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \cup \{\varepsilon\} \rightarrow 2^{(Q \times \Gamma^* \cup \{\varepsilon\})}$
  - O argumento é uma tripla (q, a, X), onde:
    - q é um estado em Q
    - a é um símbolo de entrada em Σ ou a=ε (cadeia vazia)
    - X é um símbolo da pilha, isto é, um elemento de Γ
  - A saída de δ é um conjunto finito de pares (p,γ), onde:
    - p é o novo estado
    - γ é a cadeia de símbolos da pilha que substitui X no topo da pilha
      - Se γ = ε, a pilha é extraída
      - Se γ = X, a pilha fica inalterada
      - Se γ = YZ, X é substituído por Z e Y é inserido na pilha

Autômatos de pilha

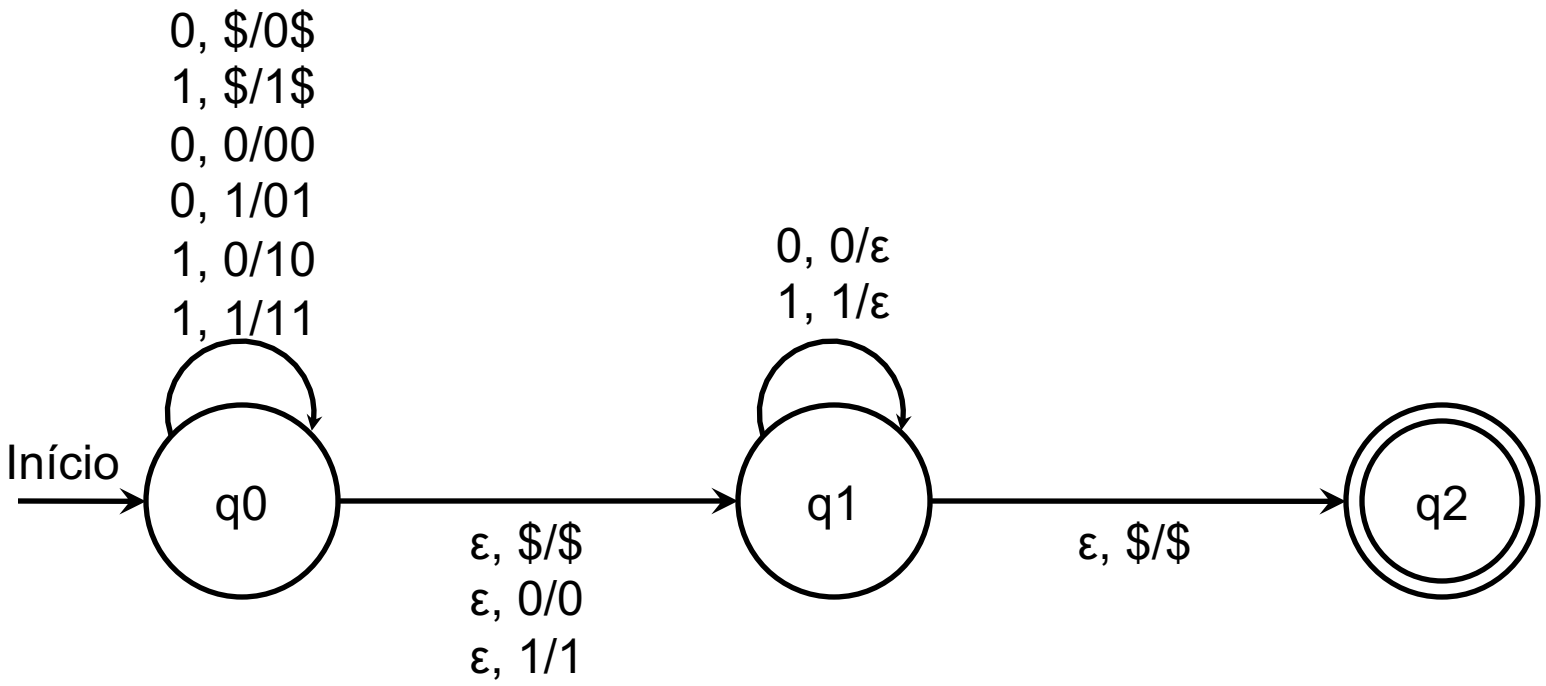
- Exemplo: Vamos projetar um PDA P para aceitar a linguagem L = {ww^R | w está em {0,1}*}:
- P = ({q_0,q_1,q_2}, {0,1}, {0,1,$}, δ, q_0, $, {$q_2$})
  - Obs: $ é um símbolo que fica no fundo da pilha, inicialmente, e será usado para marcar quando a pilha está vazia
- Onde δ é definido pelas seguintes regras:
  1. Empilhando
     - δ(q0,0,$)={(q0,0$)} e δ(q0,1,$)={(q0,1$)}
     - δ(q0,0,0)={(q0,00)}, δ(q0,0,1)={(q0,01)}, δ(q0,1,0)={(q0,10)} e δ(q0,1,1)={(q0,11)}
  2. Adivinhando o meio da cadeia
     - δ(q0,ε,$)={(q1,$)}, δ(q0,ε,0)={(q1,0)} e δ(q0,ε,1)={(q1,1)}
  3. Desempilhando
     - δ(q1,0,0)={(q1,ε)} e δ(q1,1,1)={(q1,ε)}
  4. Checando a pilha vazia
     - δ(q1,ε,$)={(q2,$)}

Autômatos de pilha

- A lista de transições para um PDA nem sempre é fácil de acompanhar
- Uma forma melhor é um diagrama de transição para PDAs:
  - Os nós correspondem aos estados do PDA
  - Uma seta identificada por Início indica o estado inicial, e estados com círculos duplos são estados de aceitação
  - Os arcos correspondem a transições do PDA
    - Mas com algumas extensões, conforme a seguir

Autômatos de pilha

- Um arco é identificado por a, X/α
  - a = símbolo de entrada
  - X = símbolo no topo da pilha
  - α = cadeia a substituir o topo da pilha



Autômatos de pilha

- Outra forma é criar uma tabela tridimensional

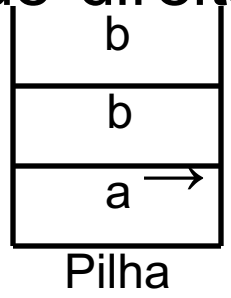
Entrada:	0			1			ε		
Pilha:	0	1	$	0	1	$	0	1	$
q0	{(q0,00)}	{(q0,01)}	{(q0,0$)}	{(q0,10)}	{(q0,11)}	{(q0,1$)}	{(q1,0)}	{(q1,1)}	{(q1,$)}
q1	{(q1,ε)}	Ø	Ø	Ø	{(q1,ε)}	Ø	Ø	Ø	{(q2,$)}
q2	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Autômatos de pilha

- Para facilitar o acompanhamento da execução de um autômato, existe também o conceito de configuração (ou descrição) instantânea
  - Um texto que resume o estado da execução em um determinado momento, com as informações essenciais
    - Estado atual do PDA
    - Entrada a ser lida
    - Conteúdo da pilha

Autômatos de pilha

- A configuração instantânea (CI) de um PDA é representada por uma tripla (q, w, γ), onde:
  - q é o estado
  - w é a parte restante da entrada
  - γ é o conteúdo da pilha
- Convencionalmente, mostramos o topo da pilha na extremidade esquerda e a parte inferior na extremidade direita
  - Ex:



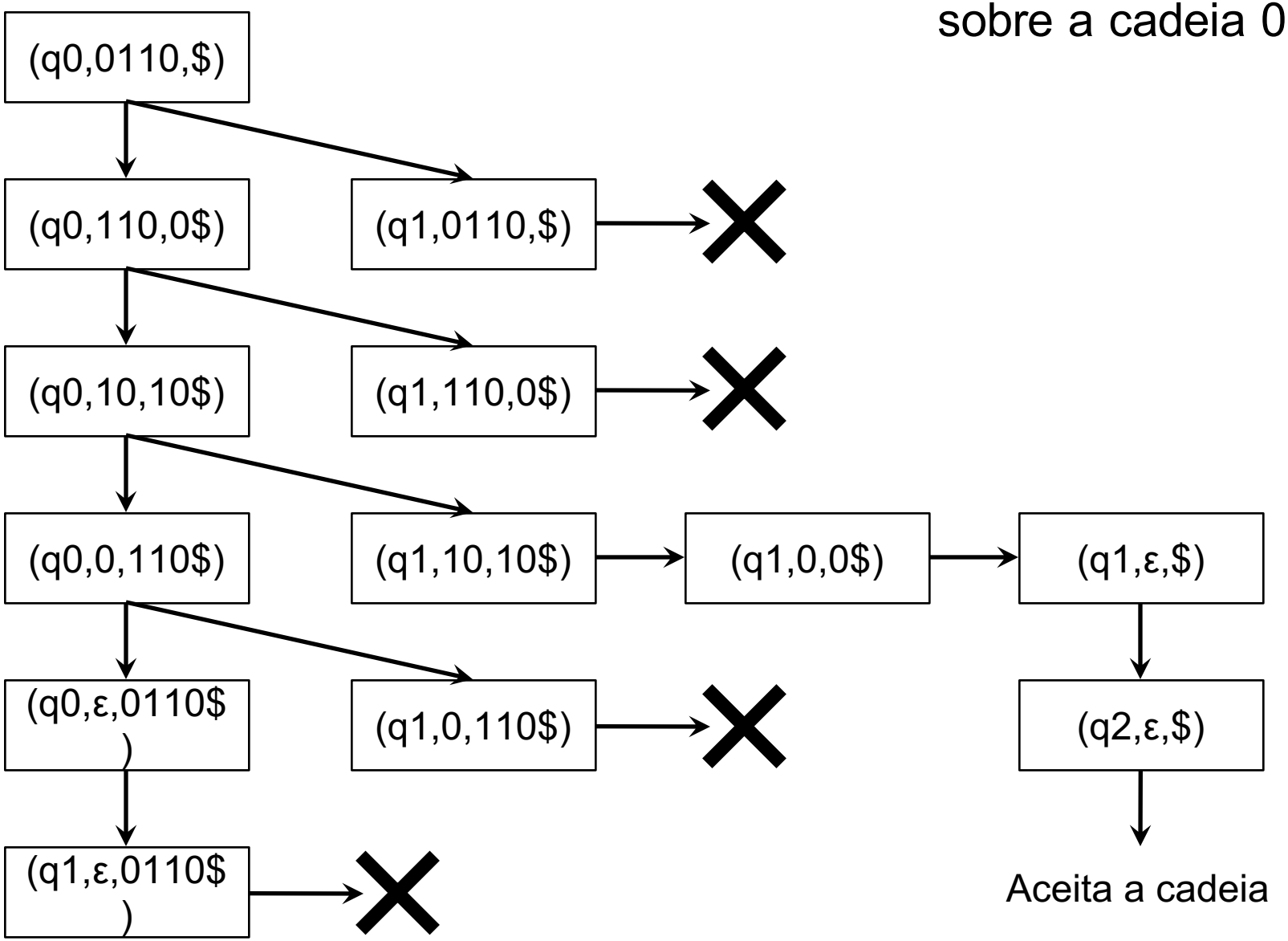
γ = bba

Autômatos de pilha

- Formalmente: um movimento genérico em um PDA é descrito pelas seguintes configurações instantâneas
  - $(q,aw,X\beta) \vdash (p,w,\alpha\beta)$
- Supondo que $\delta(q,a,X)$ contém (p,α)
  - A notação $\vdash^*$ indica uma sequência de movimentos

Autômatos de pilha

Exemplo: o autômato apresentado anteriormente sobre a cadeia 0110



Autômatos de pilha

- Exercício: mostre as sequências de configurações instantâneas para a cadeia 1111

Autômatos de pilha

- As linguagens de um PDA
- Existem duas abordagens para decidir se um PDA aceita ou não uma entrada
  - Aceitação pelo estado final
  - Aceitação por pilha vazia
- Ambas são equivalentes
  - Uma linguagem L tem um PDA que a aceita pelo estado final se e somente se L tem um PDA que a aceita por pilha vazia
  - Ou seja, é possível converter um PDA que aceita L por estado final em outro PDA que aceita L por pilha vazia
    - E vice-versa

Autômatos de pilha

- Aceitação por estado final
  - Seja $P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ um PDA
    - Então L(P), a linguagem aceita por P pelo estado final, é:
    - $L(P) = \{w \mid (q_0,w,Z_0) \vdash^* (q,\varepsilon,\alpha)\}$
    - Para algum estado q em F e qualquer cadeia de pilha α
- Ou seja, é o conjunto de todas as cadeias que o PDA pode processar, a partir da configuração instantânea inicial, consumindo todos os símbolos da cadeia, que chegam a um estado de aceitação
  - Não interessa o que "sobrar" na pilha

Autômatos de pilha

- Aceitação por pilha vazia
  - Seja $P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ um PDA
    - Então N(P), a linguagem aceita por P por pilha vazia, é:
    - $N(P) = \{w \mid (q_0,w,Z_0) \vdash^* (q,\varepsilon,\varepsilon)\}$
    - Para qualquer estado q (não necessariamente em F)
- Ou seja, é o conjunto de cadeias que o PDA pode processar, a partir da configuração instantânea inicial, consumindo todos os símbolos da cadeia e deixando a pilha vazia no final
  - Não interessa em qual estado o autômato parou
  - Portanto, quando a aceitação é por pilha vazia, a descrição pode omitir o último elemento da tupla:
    - $P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0)$

Autômatos de pilha

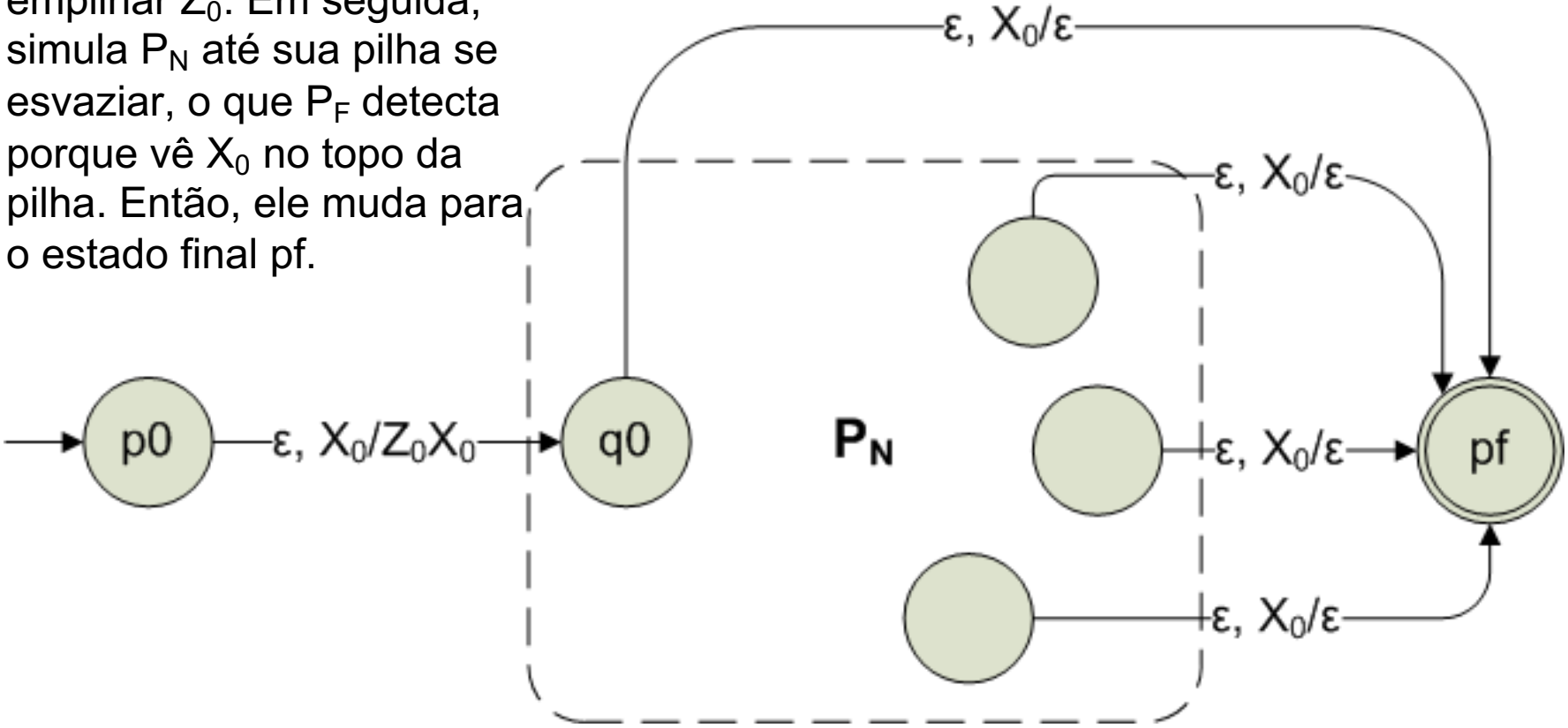
- Aceitação por pilha vazia = Aceitação por estado final
- Prova: por construção
  - Conversão de pilha vazia → estado final
  - Conversão de estado final → pilha vazia

De pilha vazia ao estado final

- Dado um PDA $P_N = (Q,\Sigma,\Gamma,\delta_N,q0,Z_0,F)$ que aceita por pilha vazia:
- Criaremos um PDA P_F que aceita por estado final:
  - $P_F = (Q \cup \{p0, pf\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p0, X_0, \{pf\})$
  - Novo símbolo X_0
  - Novo estado inicial p0
  - Novo estado final pf
  - Novas transições (δ_F) conforme a seguir

De pilha vazia ao estado final

P_F começa com X_0. O primeiro movimento é empilhar Z_0. Em seguida, simula P_N até sua pilha se esvaziar, o que P_F detecta porque vê X_0 no topo da pilha. Então, ele muda para o estado final pf.

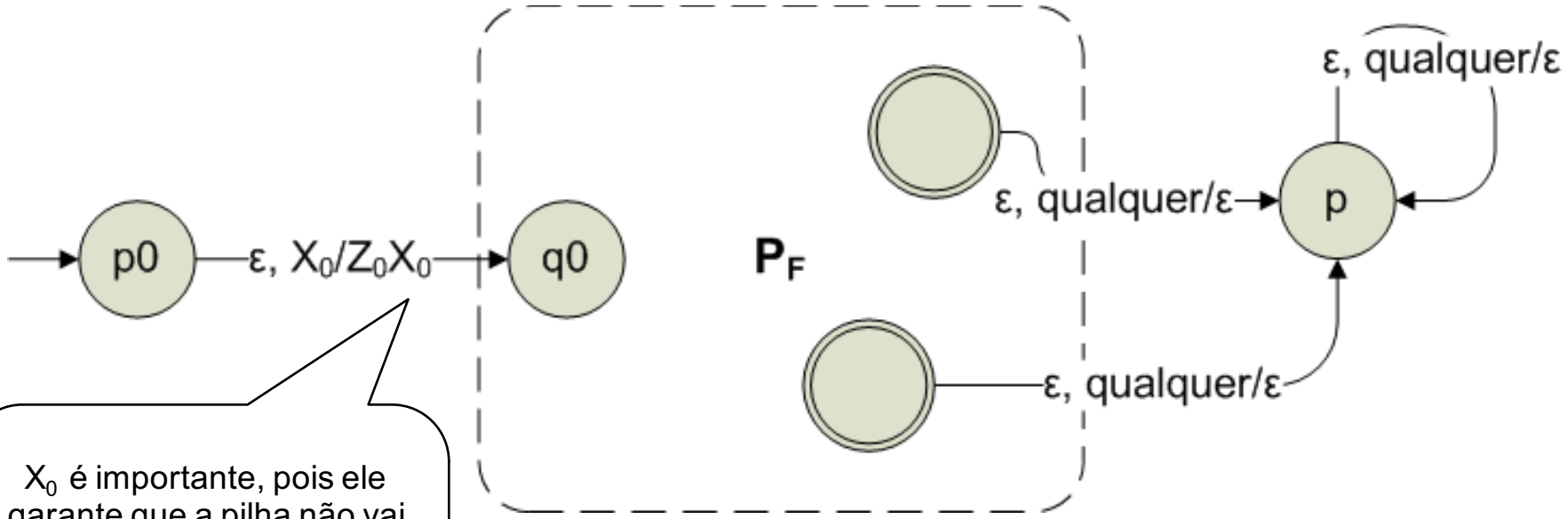


De estado final para pilha vazia

- Dado um PDA $P_F = (Q,\Sigma,\Gamma,\delta_F,q0,Z_0,F)$ que aceita por estado final:
- Criaremos um PDA P_N que aceita por pilha vazia:
  - $P_F = (Q \cup \{p0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p0, X_0)$
  - Novo símbolo X_0
  - Novo estado inicial p0
  - Novo estado p (que representa a pilha vazia)
  - Novas transições (δ_N) conforme a seguir

De estado final para pilha vazia

P_N começa com X_0. Inicialmente, empilha Z_0 e simula P_F até chegar a um estado de aceitação. Então, ele muda para um estado p que nada faz, a não ser esvaziar a pilha.



X_0 é importante, pois ele garante que a pilha não vai esvaziar acidentalmente, durante a simulação de PF (o que pode acontecer, pois PF aceita por estado final)

Autômatos de pilha

- Se uma linguagem é L(P) ou N(P) para algum PDA, então L é livre de contexto
  - Prova: por construção
    - Conversão de CFG → PDA
      - (Essencialmente, é o conteúdo da disciplina de compiladores, não vamos cobrir aqui)
    - Conversão de PDA → CFG
      - (Não tem muita utilidade prática)

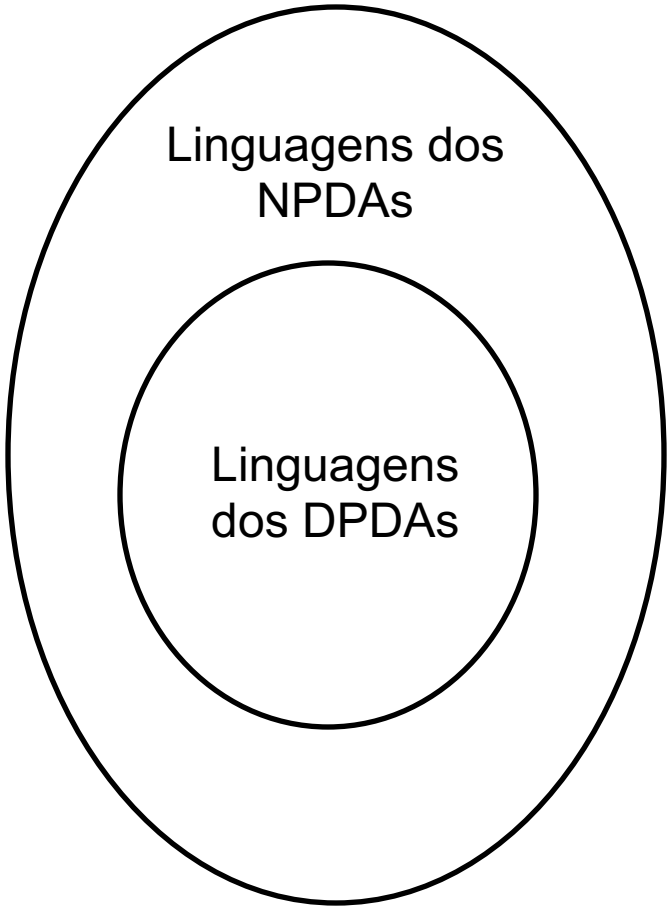
Determinismo em PDAs

Determinismo em PDAs

- O não-determinismo é um bom “truque de programação”, pois ajuda a projetar linguagens/autômatos
  - Para autômatos finitos, o não-determinismo não dá poder aos autômatos, ou seja:
    - NFAs reconhecem as mesmas linguagens que DFAs
    - É possível converter NFAs em DFAs e vice-versa
  - A escolha de quando converter (em tempo de execução ou pré-execução) fica a cargo do projetista, e depende do cenário de aplicação

Determinismo em PDAs

- Para autômatos de pilha, o mesmo não acontece
- O não-determinismo AUMENTA a capacidade reconhecedora de um PDA
  - Ou seja, PDAs determinísticos (DPDAs) reconhecem menos linguagens do que PDAs não-determinísticos (NPDAs)
    - Equivalente a dizer que existem linguagens reconhecidas por NPDAs que não são reconhecidas por DPDAs



PDA determinístico - DPDA

- Definição informal é a mesma do que para autômatos finitos determinísticos
  - Ou seja, em um DPDA, nunca há alternativa de escolha para movimento, e sempre o autômato sabe o que fazer
  - Isso significa que, para toda combinação de entrada, estado e topo da pilha, há uma e somente uma possibilidade de transição
- Importante: a transição vazia (ε) aqui não é indicativo de não-determinismo!!
  - Pois pode haver uma decisão com base no topo da pilha.
  - O problema é haver conflito entre consumir a entrada ou não!

PDA determinístico - DPDA

- O caso da entrada vazia
  - Para um determinado topo da pilha X, e uma entrada a
  - O DPDA:
    - Ou define uma transição com base em a (e a transição com base em ε fica vazia)
    - Ou define uma transição com base em ε (e a transição com base em a fica vazia)
- Na tabela, as células correspondentes às colunas ε nunca sobrepõem o que foi definido nas células das colunas das entradas

Fim