

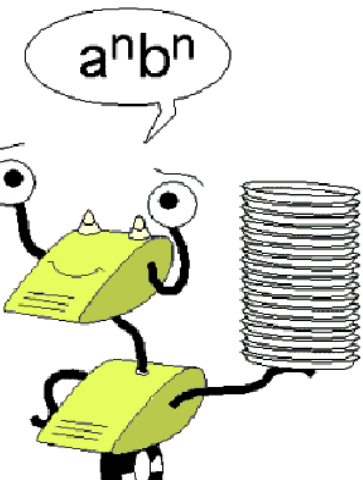
Autômatos com Pilha



Autômatos com Pilha: Definição
Linguagem Aceita por um AP

Notação gráfica para AP

APD X APND



Autômatos com Pilha (AP)

Definições alternativas para Linguagens Livres de Contexto

Extensão de AFND- ϵ com uma pilha, que pode ser lida, aumentada e diminuída apenas no topo.

2 variações de AP:

- Aceita cadeias por estados de aceitação
- Aceita cadeias por pilha vazia

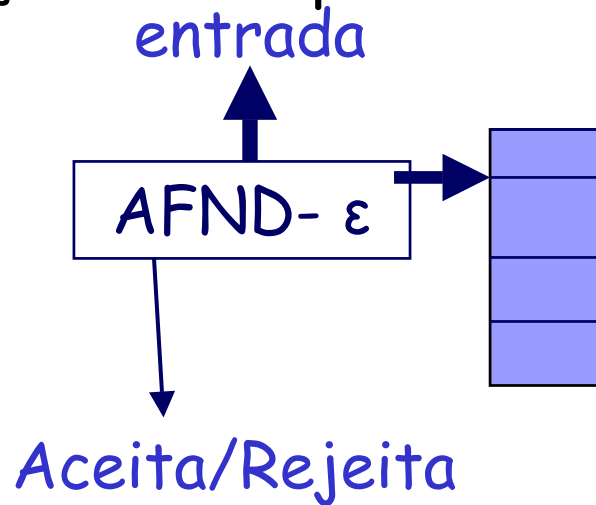
As 2 variações aceitam exatamente as LLC

(GLC \leftrightarrow AP)

AP Determinísticos: aceitam todas as LR e parte das LLC

AP - Definição Informal

- A pilha permite memorizar uma quantidade infinita de informações. Ao contrário de um computador, que também pode armazenar quantidade arbitrária de informações, o AP tem a restrição do comportamento da pilha (FIFO).



Portanto, existem linguagens que poderiam ser reconhecidas por algum programa de computador, mas que não são reconhecidas por qualquer AP.

O AP decide a mudança de estado baseado no símbolo atual da entrada, no estado atual, e no símbolo do topo da pilha.

Ele pode, alternativamente, fazer uma transição espontânea, usando ϵ como entrada. Em uma transição, o AP:

- Consome da entrada o símbolo que ele utiliza na transição. Se ϵ for usado, nenhum símbolo da entrada é consumido.
- Vai para um novo estado, que pode ou não ser o mesmo estado anterior.
- Substitui o símbolo do topo da pilha por qualquer cadeia. A cadeia pode ser ϵ (eliminação na pilha); pode ser o mesmo símbolo do topo (nenhuma alteração), ou um ou mais símbolos distintos (eliminação + inserção).

Exemplo

- $L_{wwr} = \{ww^r \mid w \text{ está em } (0+1)^*\}$ - palíndromos de comprimento par sobre $\{0,1\}$.
- GLC: $P \rightarrow OPO \mid 1P1 \mid \varepsilon$
- Projetando um AP para L_{wwr}
 1. Começamos em um estado q_0 que representa um palpite de que ainda não vimos o meio. Enquanto estamos em q_0 , lemos símbolos e os armazenamos na pilha.
 2. Em qualquer momento, podemos supor que chegamos ao meio, então w estará na pilha, com a extremidade direita no topo. Nessa hora, vamos para q_1 . Como AP é não determinístico, optamos também por ficar em q_0 .
 3. Em q_1 , comparamos os símbolos da entrada com os da pilha. Se coincidirem, consumimos o da entrada e extraímos o da pilha, e prosseguimos. Se não coincidirem, nosso palpite foi errado. Essa ramificação falha, mas outras podem eventualmente ter sucesso.
 4. Se esvaziamos a pilha, então vimos alguma entrada w seguida por seu reverso. Aceitamos a entrada que foi lida até esse ponto

Um AF M é uma 5-tupla $(K, \Sigma, \Gamma, q_0, Z_0, F)$ onde:

1. K é um conjunto finito de estados
2. Σ (sigma) é um alfabeto finito chamado alfabeto de entrada
3. Γ (gama) é um *alfabeto da pilha* finito
4. $q_0 \in K$ é o estado inicial. A máquina começa nele.
5. $Z_0 \in \Gamma$ é o símbolo inicial da pilha. Aparece inicialmente na pilha.
6. F é o conjunto de estados finais $F \subseteq K$
7. δ (delta) é um mapeamento de
$$K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \text{subconjuntos de } K \times \Gamma^*$$

(topo)

O resultado da aplicação de δ é um conjunto finito de pares (p, γ) onde p é o novo estado e γ é a cadeia de símbolos da pilha que substitui o topo da pilha. Se γ é ε , o topo é eliminado; se $\gamma = YZ$, o topo é substituído por Z e Y é o novo topo.

$\Lambda = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, \Delta_0\}, \delta, q_0, \Delta_0, \{q_2\})$

• $\delta(q_0, 0, \Delta_0) = \{(q_0, 0\Delta_0)\}$ e $\delta(q_0, 1, \Delta_0) = \{(q_0, 1\Delta_0)\}$. Uma dessas regras se aplica inicialmente. Empilhamos o primeiro símbolo

• $\delta(q_0, 0, 0) = \{(q_0, 00)\}$ e $\delta(q_0, 0, 1) = \{(q_0, 01)\}$ e $\delta(q_0, 1, 0) = \{(q_0, 10)\}$ e $\delta(q_0, 1, 1) = \{(q_0, 11)\}$. Essas regras nos permitem ficar no estado q_0 e apenas empilhar a entrada.

• $\delta(q_0, \varepsilon, \Delta_0) = \{(q_1, \Delta_0)\}$, $\delta(q_0, \varepsilon, 0) = \{(q_1, 0)\}$ e $\delta(q_0, \varepsilon, 1) = \{(q_1, 1)\}$. Essas regras permitem fazer uma transição espontânea de q_0 para q_1 , deixando a pilha inalterada.

• $\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$ e $\delta(q_1, 1) = \{(q_1, \varepsilon)\}$. Em q_1 , comparamos a entrada com o topo. Se coincidirem, desempilhamos.

• $\delta(q_1, \varepsilon, \Delta_0) = \{(q_2, \Delta_0)\}$. Se encontrarmos a base da pilha e estivermos em q_1 , então encontramos $w = \varepsilon$ e vamos para o estado de aceitação q_2 .

Notação gráfica para AP

O diagrama de transição para APs que aceitam a linguagem por estado final segue:

Os nós correspondem aos estados do AP

Existem o estado inicial e os finais

Um arco rotulado com $a, X/a$ do estado q para p significa que $\delta(q, a, X)$ contém o par (p, a) entre os pares.

Convencionalmente, Z_0 é o símbolo da pilha (no JFLAP é Z).

Leia-se: símbolo, topo_antigo/novo_topo

AP de Lww^r

0, Z₀/0 Z₀

1, Z₀/1 Z₀

0, 0/0 0

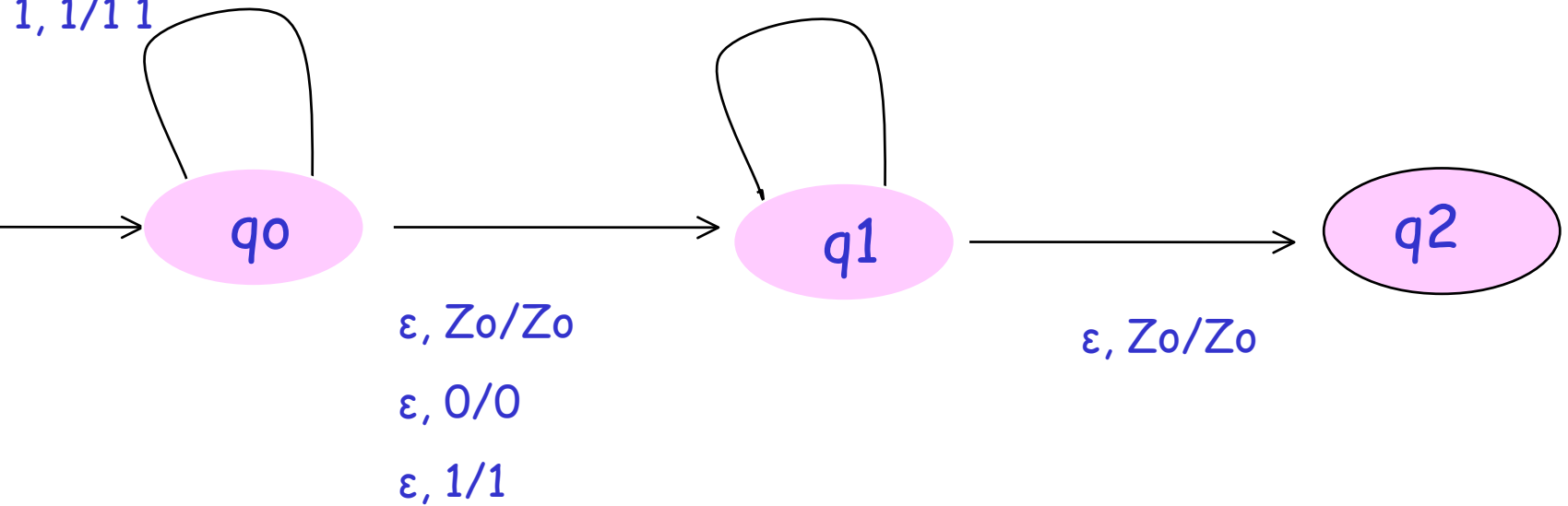
0, 0/0 1

1, 0/1 0

1, 1/1 1

0, 0/ε

1, 1/ε



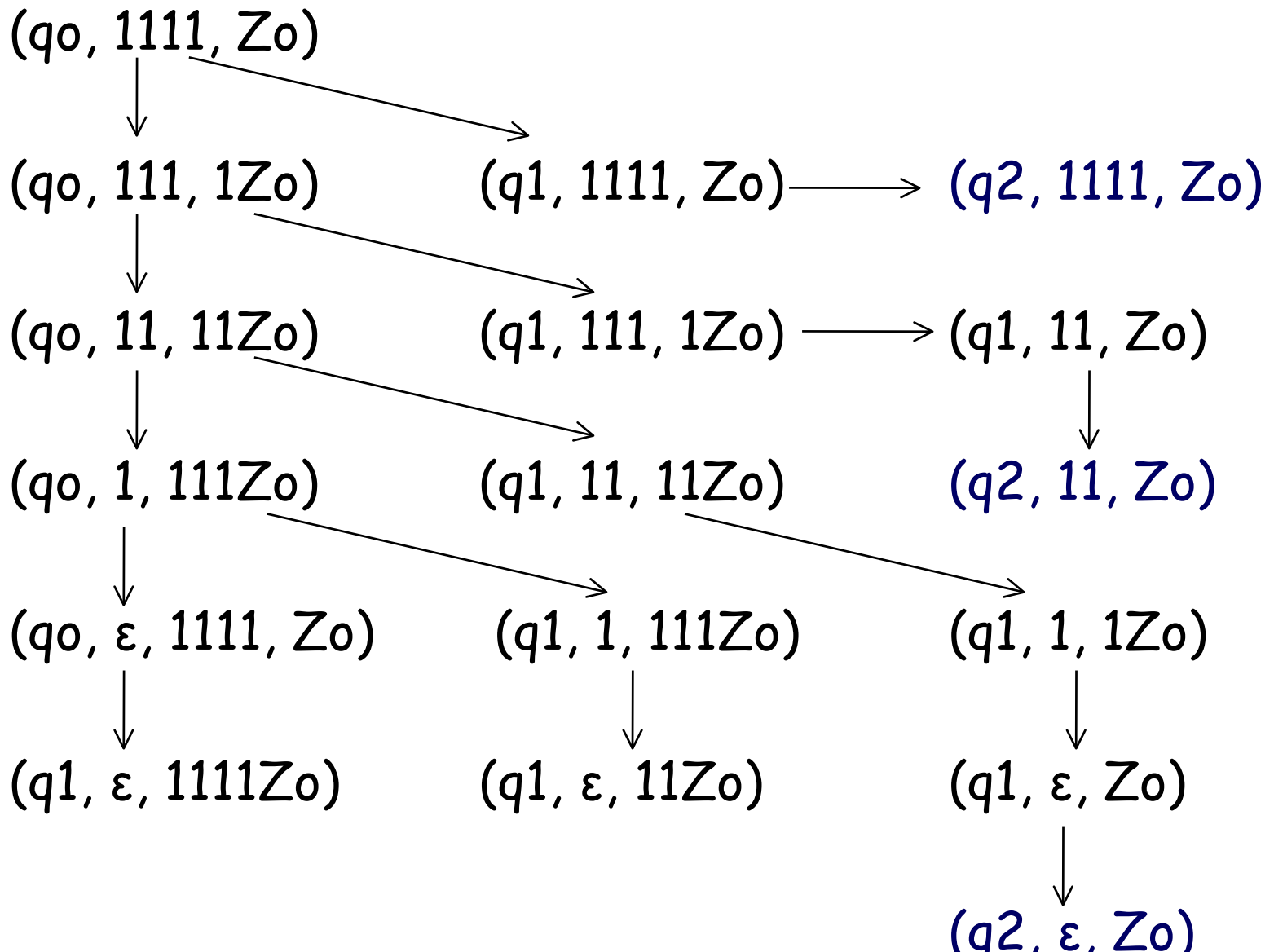
Descrição Instantânea (DI) de um AP

- Para um AF, a DI é o estado atual. Num AP, usamos a "dedução" para indicar mudanças no estado, na entrada e na pilha.
- Seja $P = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ um AP. Suponha que $\delta(q, a, X)$ contém (p, α) . Então, para todas as cadeias w em Σ^* e β em Γ^* :

$$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$$

- Esse movimento reflete a idéia de que, consumindo a e substituindo o topo X por α , podemos ir do estado q para o estado p . Repare que o que fica na entrada (w) e o que está abaixo do topo (β) não influenciam a ação do AP.
- \vdash^* representa zero ou mais movimentos do AP.

Exemplo: derivação de 1111 de L_{ww}^r



Exercício

• Seja $P = (\{p, q\}, \{0, 1\}, \{Z_0, X\}, \delta, q, Z_0, \{p\})$ e :

1. $\delta(q, 0, Z_0) = \{(q, XZ_0)\}$

2. $\delta(q, 0, X) = \{(q, XX)\}$

3. $\delta(q, 1, X) = \{(q, X)\}$

4. $\delta(q, \varepsilon, X) = \{(p, \varepsilon)\}$

5. $\delta(p, \varepsilon, X) = \{(p, \varepsilon)\}$

6. $\delta(p, 1, X) = \{(p, XX)\}$

7. $\delta(q, 1, Z_0) = \{(p, \varepsilon)\}$

A partir da DI inicial (q, w, Z_0) , mostre todas as DI's acessíveis quando a entrada é:

(a) 01

(b) 0011

(c) 010

Linguagem aceita por um AP

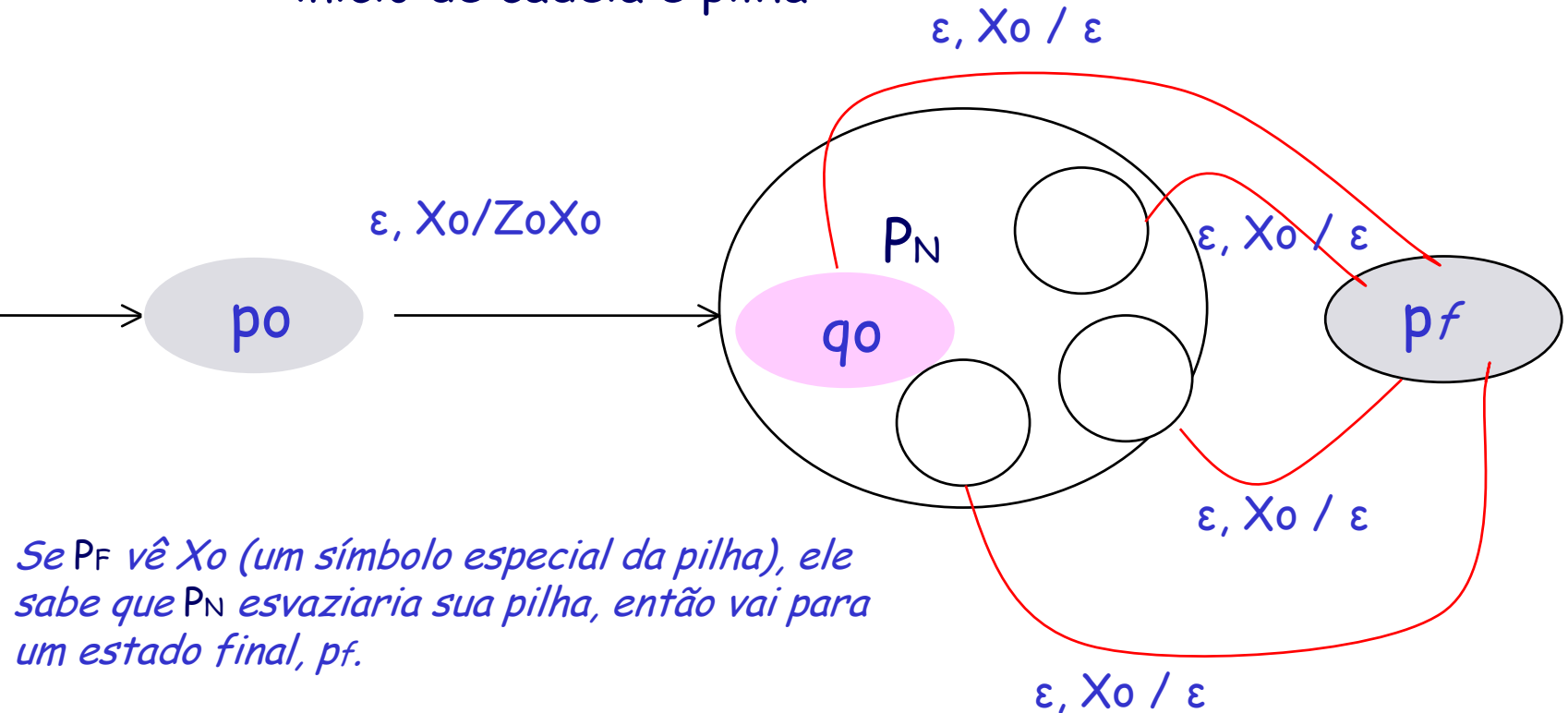
1. **Aceitação por Estado Final:** (Similar a AF)
Conjunto de todas as cadeias para as quais alguma sequência de movimentos faz o AP entrar num estado final → **Linguagem aceita por estado final**, $L(P) = \{w \mid (q_0, w, Z_0) \xrightarrow{*} (q, \varepsilon, \alpha); q \in F\}$
2. **Aceitação por Pilha Vazia:** Conjunto de todas as cadeias para as quais alguma sequência de movimentos, a partir do inicial, faz com que a pilha fique vazia → **Linguagem aceita por Pilha vazia** $N(P) = \{w \mid (q_0, w, Z_0) \xrightarrow{*} (q, \varepsilon, \varepsilon)\}$
OBS: quando aceitamos por pilha vazia o conjunto de estados finais é irrelevante.

Os dois métodos acima são equivalentes.

Equivalência entre os dois tipos de aceitação

Teorema 1: Se $L = N(P_N)$ para algum AP $P_N = (K, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, existe um AP P_F tal que $L = L(P_F)$.

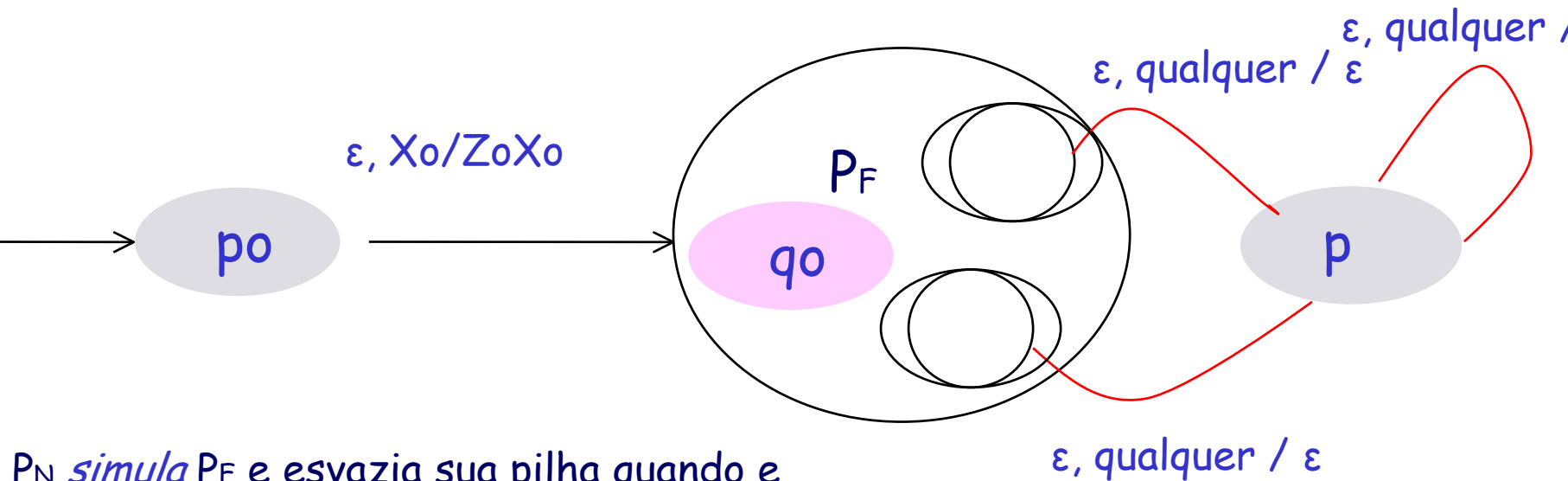
Idéia: Seja X_0 um símbolo especial de início de cadeia e pilha



Se P_F vê X_0 (um símbolo especial da pilha), ele sabe que P_N esvaziaria sua pilha, então vai para um estado final, p_f .

Teorema 2: Se $L = L(P_F)$ para algum AP $P_F = (K, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$, existe um AP P_N tal que $L = N(P_N)$.

Idéia: Seja X_0 um símbolo especial de início de cadeia e pilha



P_N *simula* P_F e esvazia sua pilha quando e somente quando P_N *entra em um estado de aceitação*.

Exercício (para casa)

- Faça um AP que reconhece $L = \{ww^R \mid w \in \{0,1\}^*\}$ por pilha vazia
- Vejam que o AP deve aceitar a cadeia vazia.
- Procure fazer diretamente e também a partir do AP por estado final já construído.

Faça

- Encontre um AP M que reconheça o conjunto $L = \{ w \mid w \in \{0,1\}^* \text{ e } w \text{ tem igual número de } 0 \text{ e } 1 \}$, por pilha vazia.
- Mostre as configurações do AP com a entrada 0101

Dica:

- associar 0 com Z e 1 com U ; $Z, U \in \Gamma$
- "matar" 0 com 1 e 1 com 0
- fazer regra para aceitar cadeia vazia

$$M = (\{q1\}, \{0,1\}, \{B,Z,U\}, \delta, q1, B, \emptyset)$$

1. $\delta(q1, 0, B) = \{(q1, ZB)\}$

empilha primeiro 0

2. $\delta(q1, 1, B) = \{(q1, UB)\}$

empilha primeiro 1

3. $\delta(q1, 0, Z) = \{(q1, ZZ)\}$

empilha 0 consecutivos

4. $\delta(q1, 1, U) = \{(q1, UU)\}$

empilha 1 consecutivos

5. $\delta(q1, 1, Z) = \{(q1, \epsilon)\}$

desempilha se 1 com 0

6. $\delta(q1, 0, U) = \{(q1, \epsilon)\}$

desempilha se 0 com 1

7. $\delta(q1, \epsilon, B) = \{(q1, \epsilon)\}$

desempilha no fim da cadeia

Configurações para 0101

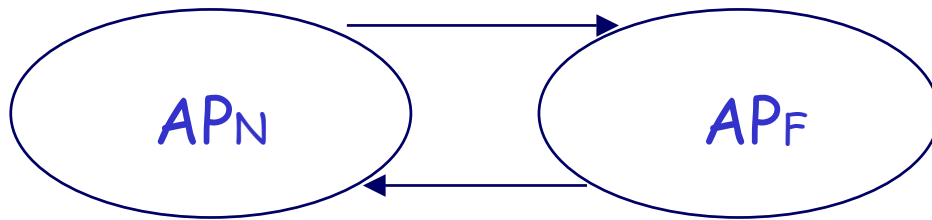
Entrada	Configuração
	$(q1, B)$
0	$(q1, ZB)$
01	$(q1, B)$
010	$(q1, ZB)$
0101	$(q1, B)$
ε	$(q1, \varepsilon)$

Exercício 2 (para casa)

- Encontre um AP M que reconheça o conjunto $L = \{ 0^n 1^n \mid n > 0 \}$ por pilha vazia.

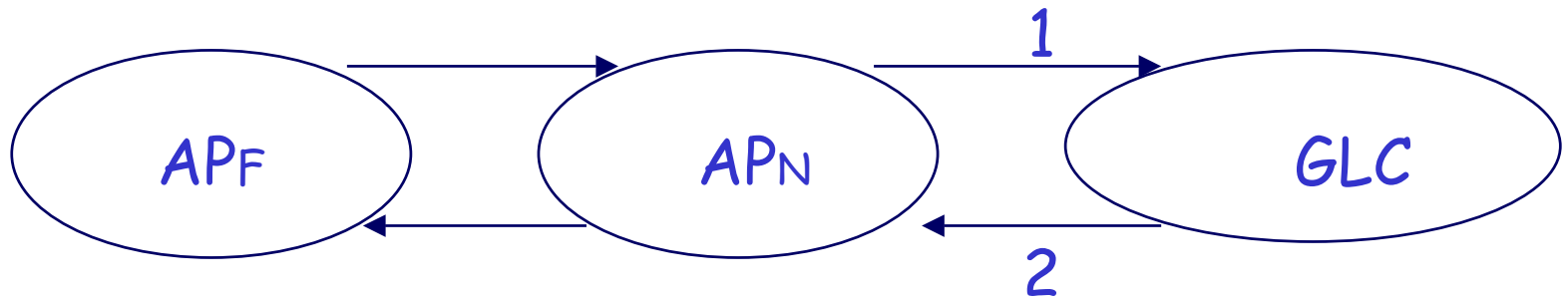
Equivalência de AP e Gramática Livre de Contexto

já mostramos a equivalência entre:



Equivalência de AP e Gramática Livre de Contexto

agora vamos mostrar que:



1. De GLC para AP

- Idéia: fazer o AP simular a sequência de formas sentenciais à esquerda usada para gerar uma dada cadeia w . O AP tem apenas um estado, e empilhamos as formas sentenciais de acordo com as regras da GLC. Num determinado instante, se x é o prefixo de w já reconhecido, então o AP estará numa $DI = (q, y, A\alpha)$ representando a forma sentencial à esquerda $xA\alpha$. Se $w=xy$, então $A\alpha$ deve gerar y nos próximos passos. Se $A \rightarrow \beta$, então o AP passará a $(q, y, \beta\alpha)$. Os terminais que porventura estiverem no topo da pilha (prefixo de $\beta\alpha$) são comparados com a entrada e desempilhados até que o próximo não terminal (mais a esquerda) esteja na pilha, ou essa se esvazie. A aceitação é por pilha vazia.

1. De GLC para AP

Seja $G = (V_n, V_t, Q, S)$ uma GLC. Construimos um AP P_N que aceite $L(G)$ por pilha vazia, da seguinte forma:

$$P_N = (\{q\}, V_t, V_nUV_t, \delta, q, S)$$

$F = \emptyset$

vocab. entrada vocab. pilha estado inicial símbolo inicial pilha

onde δ é definida por:

1. Para cada não terminal A ,

$\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \rightarrow \beta \text{ é uma produção de } G\}$ *(aplica regra e não consome entrada)*

2. Para cada terminal a , $\delta(q, a, a) = \{(q, \varepsilon)\}$ *(consome entrada e desempilha)*

Exemplo

Seja G :

$$1. E \rightarrow I \mid E + E \mid E * E \mid (E)$$

$$2. I \rightarrow a \mid b \mid Ia \mid Ib \mid IO \mid I1$$

Então

$$\Sigma = \{a, b, 0, 1, (,), +, *\}$$

$$\Gamma = \Sigma \cup \{I, E\} \text{ e}$$

$$\delta(q, \varepsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, IO), (q, I1)\}$$

$$\delta(q, \varepsilon, E) = \{(q, I), (q, E+E), (q, E*E), (q, (E))\}$$

$$\delta(q, a, a) = \{(q, \varepsilon)\} \dots\dots\dots \delta(q, *, *) = \{(q, \varepsilon)\}$$

Faça as DIs para $w = a0+b*b1$

Teorema (1)

Se o AP é construído a partir da GLC
pela construção anterior, então
 $N(P) = L(G)$.

demonstração: (A,U&M, 2003 pt), p.256

2. De AP para GLC

Teorema (2): Seja $P = (K, \Sigma, \Gamma, \delta, q_0, Z_0)$ um AP. Então, existe uma gramática livre de contexto G tal que $L(G) = N(P)$.

A prova desse teorema é por construção de G a partir de P . Ao contrário do Teorema (1), que tem também um lado funcional (de construir analisadores sintáticos, p.ex.), o valor do Teorema (2) é apenas teórico, e não vamos demonstrá-lo aqui.

demonstração: (A,U&M, 2003 pt), p.256

Exercício (para casa)

- Converta a gramática

$$S \rightarrow 0S1 \mid A$$

$$A \rightarrow 1A0 \mid S \mid \varepsilon$$

em um AP que aceite a mesma linguagem por pilha vazia.

AP Determinísticos (APD)

Por definição, os AP podem ser não-determinísticos, e a classe de AP determinísticos são especiais: embora limitada, a classe de linguagens reconhecidas por eles é interessante no contexto de linguagens de programação.

APD

Dizemos que um AP $M = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ é determinístico se:

• $\delta(q, a, Z)$ tem no máximo um elemento para $\forall q \in K$,
 $a \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$

(em cada transição, há no máximo uma alternativa de mudança)

• Se $\delta(q, a, Z)$ é não vazio para algum $a \in \Sigma$, então $\delta(q, \varepsilon, Z)$ deve ser vazio.

(impede a possibilidade de escolha entre um mov- ε e um outro que consome um símbolo de entrada)

Não existe um APD que reconheça

$L_{wwr} = \{ww^r \mid w \text{ está em } (0+1)^*\}$ - palíndromos de comprimento par sobre $\{0,1\}$.

Mas, se inserirmos um "marcador de centro" c , tornamos a linguagem reconhecível por um APD:

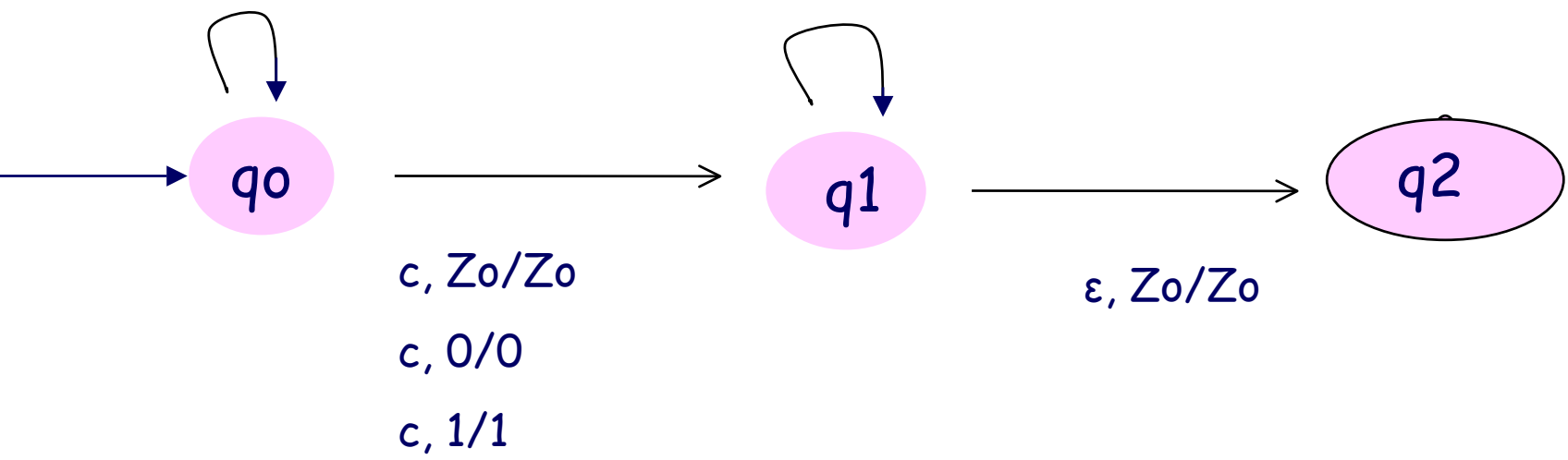
$L_{wwr} = \{wcw^r \mid w \text{ está em } (0+1)^*\}$

Estratégia: empilhar 0s e 1s até encontrar c . Então muda de estado em que compara cada novo símbolo com o topo da pilha, desempilhando se forem iguais, até encontrar Z_0 no topo; caso contrário, pára num estado não final.

APD para $L_{w^r} = \{wcw^r \mid w \text{ está em } (0+1)^*\}$

0, Z0/0 Z0
1, Z0/1 Z0
0, 0/0 0
0, 0/0 1
1, 0/1 0
1, 1/1 1

0, 0/ ϵ
1, 1/ ϵ



APD e Linguagens Regulares

Os APD aceitam uma classe de linguagens que estão entre as LR e as LLC.

Teorema 1: Se L é uma LR, então $L=L(P)$ para algum APD P_F (por estado final).

Prova: Um APD pode simular um AFD, simplesmente ignorando a pilha.

Seja um AFD $A = (Q, \Sigma, \delta_A, q_0, F)$

Construímos um APD $P = (Q, \Sigma, \{Z_0\}, \delta_P, q_0, Z_0, F)$, definindo $\delta_P(q, a, Z_0) = \{(p, Z_0)\}$ para todos os estados p e q em Q , tais que $\delta_A(q, a) = p$.

Mostre, por indução sobre $|w|$, que:

$(q_0, w, Z_0) \vdash_P^* (p, \varepsilon, Z_0)$ se e somente se $\delta_A'(q_0, w) = p$

APD por estado final e as LLC

- As linguagens aceitas por APDs por estado final incluem as LRs, mas estão incluídas nas LLC.
- Ou seja, há LLCs para as quais não existe um APD que as reconhece (há apenas APND).
- Ex. L_{wwr} -

Observação

- D. Knuth definiu, em 1965, as gramáticas LR(k), a subclasse das GLC que geram exatamente as linguagens reconhecidas por APDs.
- As gramáticas LR(k) formam a base para o YACC (*yet another compiler compiler*) do Unix, que gera um programa em C, que é um analisador sintático para a linguagem definida por uma LR(k), dada como entrada.

APD e Gramáticas Ambíguas

- **Teorema:** Se $L = N(P)$ para algum APD P (por pilha vazia), então L tem uma GLC não-ambígua.
- **Teorema:** Se $L = L(P)$ para algum APD P (por estado final) então L tem uma GLC não-ambígua.

(isso nos indica que as linguagens inerentemente ambíguas só podem ser reconhecidas por APND)

Resumo

- *Autômato com pilha:* um AP é um AFND acoplado a uma pilha que pode ser usada para armazenar uma cadeia de comprimento arbitrário.
- *Aceitação por AP:* existem dois modos pelos quais o AP pode indicar aceitação. Um deles é entrar em um estado final; o outro é esvaziar sua pilha. Esses modos são equivalentes, no sentido de que qualquer linguagem aceita por um modo é aceita (por algum outro AP) pelo outro modo.

Resumo

- *Descrição instantânea:* usamos uma DI que consiste no estado, na entrada restante e no conteúdo da pilha para descrever a "condição atual" de um AP. Uma função de transição $|$ - entre DIIs representa movimentos isolados de um AP.
- *APs e Gramáticas:* as linguagens aceitas por APs pelo estado final ou por pilha vazia são exatamente as LLCs.
- *AP determinístico:* é aquele que nunca tem mais de uma opção de movimento para um dado estado, um símbolo de entrada (inclusive ϵ) e um símbolo de pilha. Além disso, ele nunca tem uma escolha entre um ϵ -movimento e consumir um símbolo da entrada.

Resumo

- *Aceitação por APD:* os dois modos - estado final e pilha vazia - **não** são os mesmos para APDs. As linguagens aceitas por pilha vazia são um subconjunto das aceitas por estado final (aquelas que têm a *propriedade de prefixo*: nenhuma cadeia da linguagem é um prefixo de outra cadeia da linguagem).
- *Linguagens aceitas por APD:* todas as linguagens regulares são aceitas (por estado final) por APDs e existem linguagens não regulares aceitas por APDs. As linguagens de APDs são LLC e, de fato, são linguagens que têm GLC não-ambíguas.