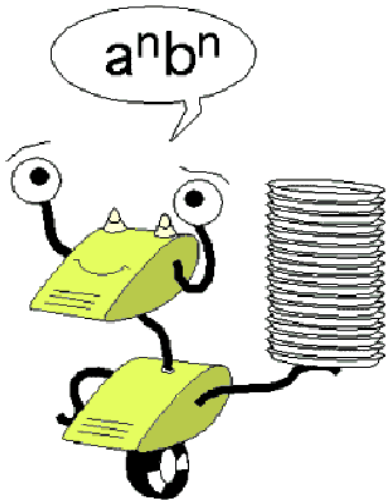
Autômatos com Pilha



Autômatos com Pilha: Definição

Linguagem Aceita por um AP

Notação gráfica para AP

APD X APND

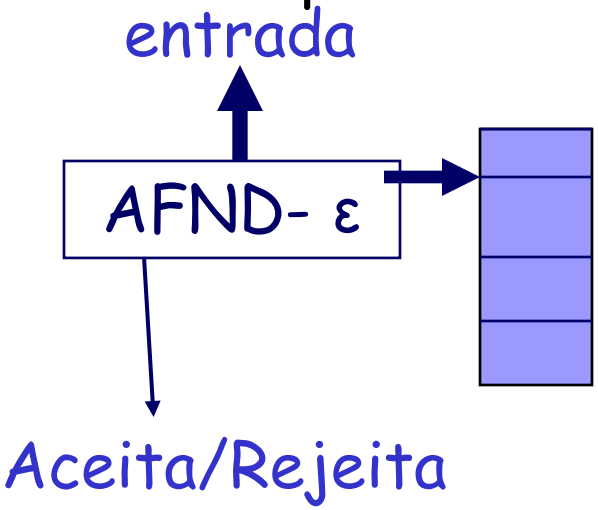
Autômatos com Pilha (AP)

- 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 ⇐⇒ AP)
- APDeterminísticos: 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.

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 -> 0P0 | 1P1| ε
- Projetando um AP para L_{wwr}

1. Começamos em um estado qo que representa um palpite de que ainda não vimos o meio. Enquanto estamos em qo, 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 q1. Como AP é não determinístico, optamos também por ficar em qo.
3. Em q1, 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 AP M é uma sétupla (K, Σ, Γ, δ, qo, Zo,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. qo $\in$ K é o estado inicial. A máquina começa nele.
5. Zo $\in \Gamma$ 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; senão o topo é substituído por γ de tal forma que o 1º. símbolo de γ é o novo topo.

$M_{Lww^r}=(\{qo, q1,q2\}, \{0,1\}, \{0,1, Zo\},\delta, qo,Zo, \{q2\})$

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

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

3. $\delta(qo, \varepsilon, Zo) = \{(q1,Zo)\}$, $\delta(qo, \varepsilon,0) = \{(q1,0)\}$ e $\delta(qo,\varepsilon,1)=\{(q1,1)\}$. Essas regras permitem fazer uma transição espontânea de qo para q1, deixando a pilha inalterada. É a aposta de que se chegou na metade da cadeia.

4. $\delta(q1,0,0) = \{(q1, \varepsilon)\}$ e $\delta(q1,1) = \{(q1, \varepsilon)\}$. Em q1, comparamos a entrada com o topo. Se coincidirem, desempilhamos.

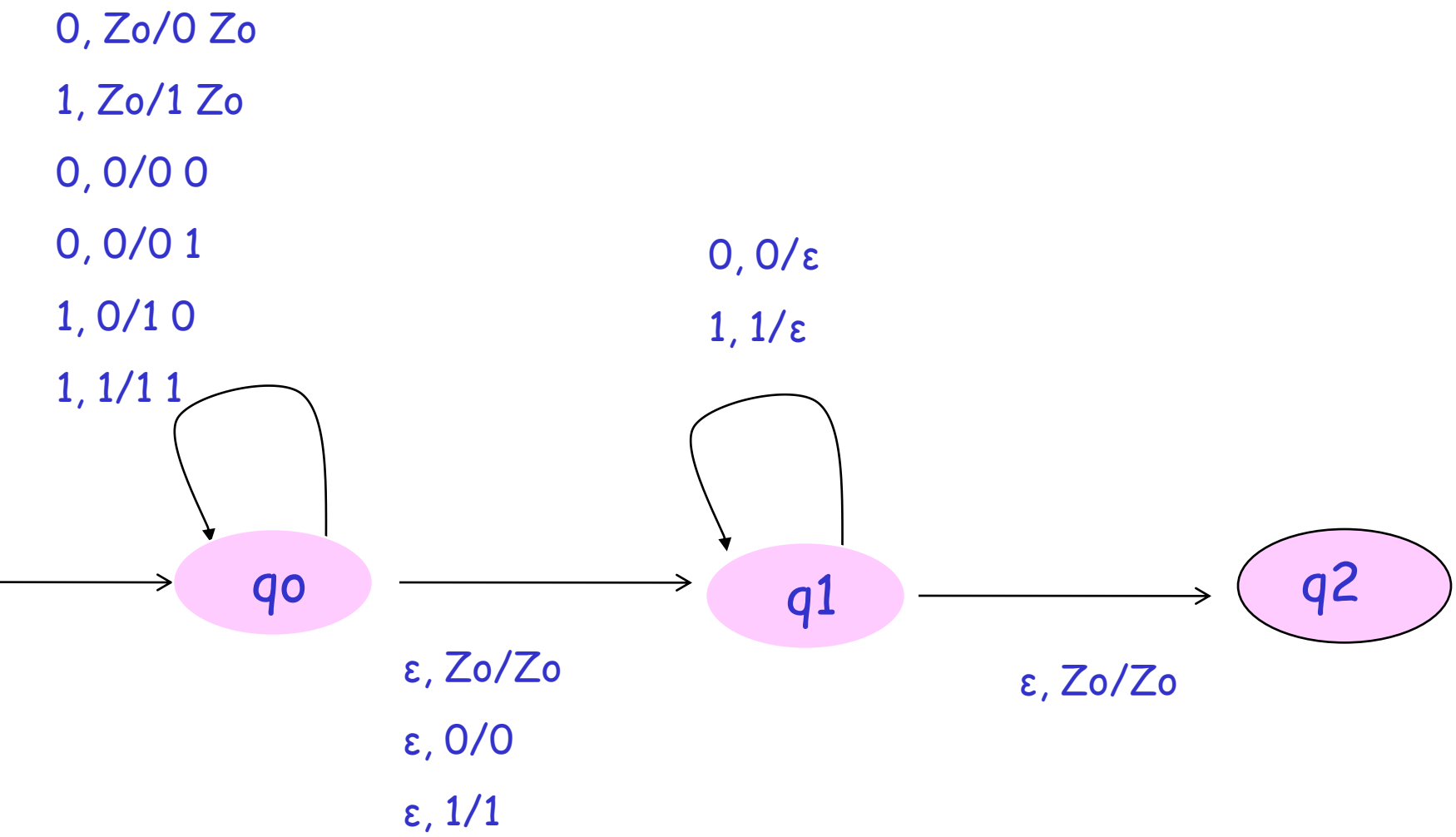
5. $\delta(q1, \varepsilon,Zo) = \{(q2,Zo)\}$. Se encontrarmos a base da pilha e estivermos em q1, então encontramos ww^r e vamos para o estado de aceitação q2.

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/α do estado q para p significa que $\delta(q,a,X)$ contém o par (p, α) entre os pares.
- Convencionalmente, Zo é o símbolo da pilha (no JFLAP é Z).

- Leia-se: símbolo, topo_antigo/novo_topo

AP de Lww^r


0, Zo/0 Zo
1, Zo/1 Zo
0, 0/0 0
0, 0/0 1
1, 0/1 0
1, 1/1 1
q0
ε, Zo/Zo
ε, 0/0
ε, 1/1
0, 0/ε
1, 1/ε
q1
ε, Zo/Zo
q2

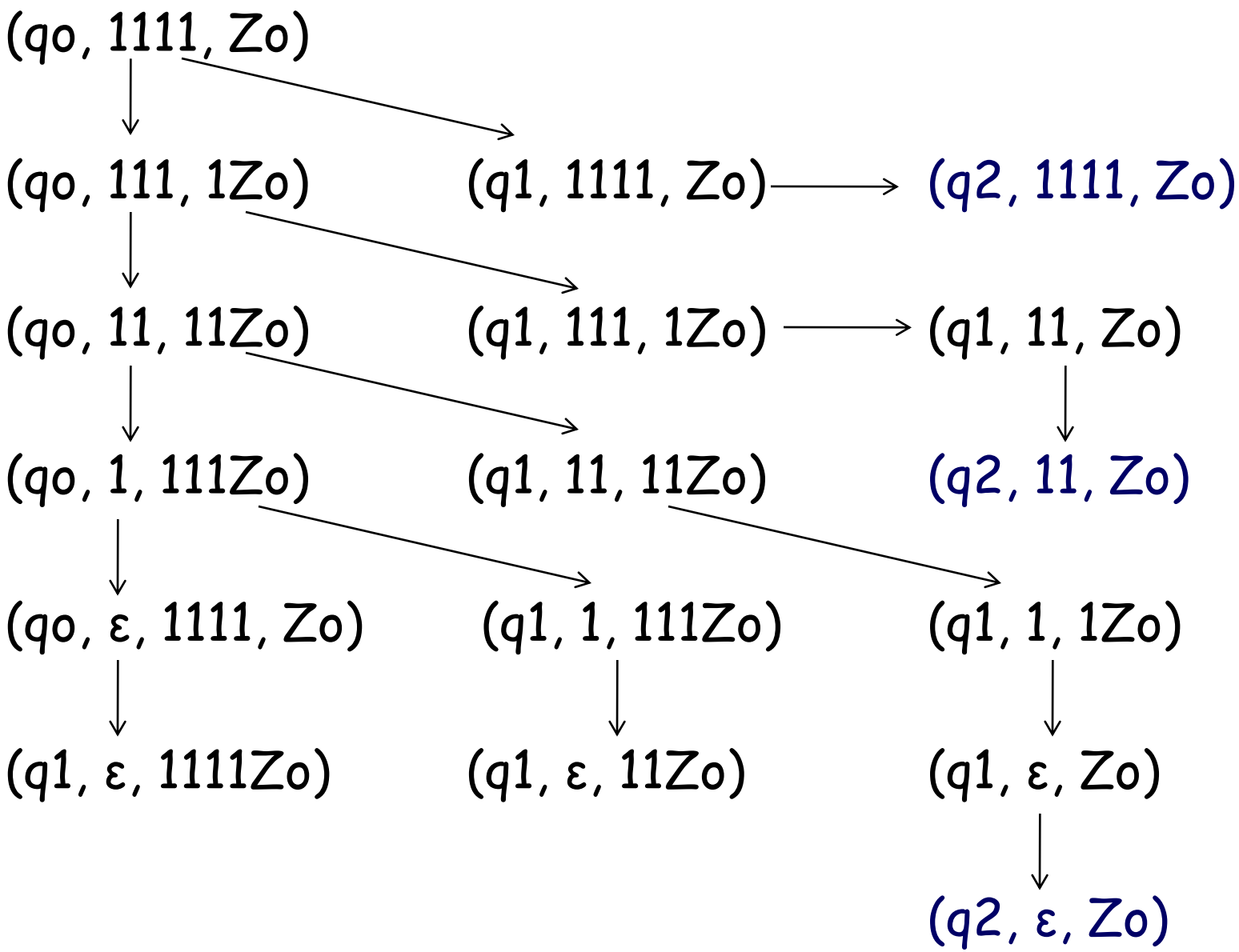
Descrição Instantânea (DI) de um AP

- 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, Σ, Γ, δ, qo, Zo,F) um AP. Suponha que δ(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 Lww^r



Exercício

- Seja P=({p,q}, {0,1}, {Zo,X}, δ, q, Zo, {p}) e :

1. $\delta(q, 0, Zo) = \{(q, XZo)\}$
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. $\delta(q, 1, Zo) = \{(p, \varepsilon)\}$

A partir da DI inicial (q, w, Zo), mostre todas as DIs 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, $L(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \alpha); q \in F\}$

2. Aceitação por Pilha Vazia: Conjunto de todas as cadeias para as quais alguma seqüência de movimentos, a partir do inicial, faz com que a pilha fique vazia → Linguagem aceita por Pilha vazia $N(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$

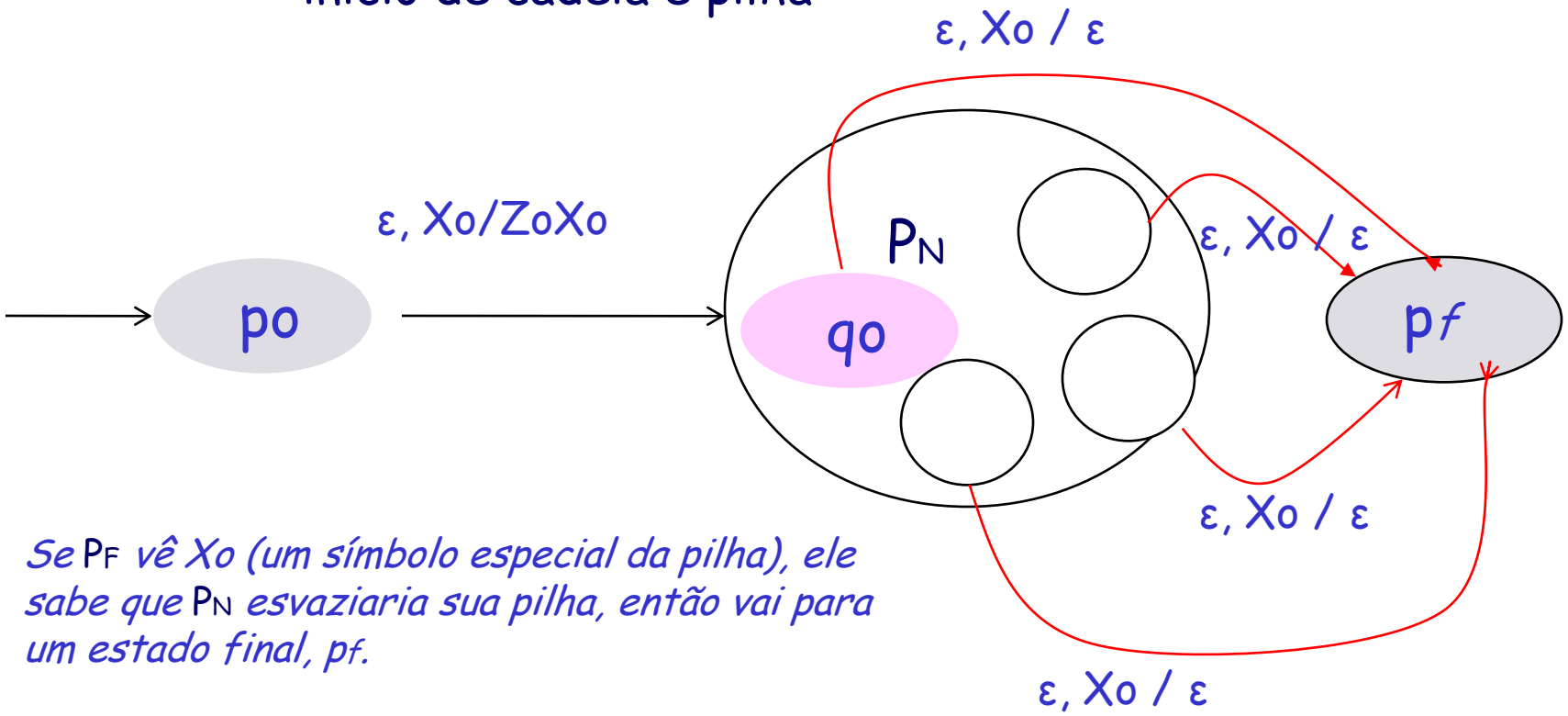
   OBS: quando aceitamos por pilha vazia o conjunto de estados finais é irrelevante.

Os dois métodos acima são equivalentes.

Equivalência entre os dois tipos de aceitação

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

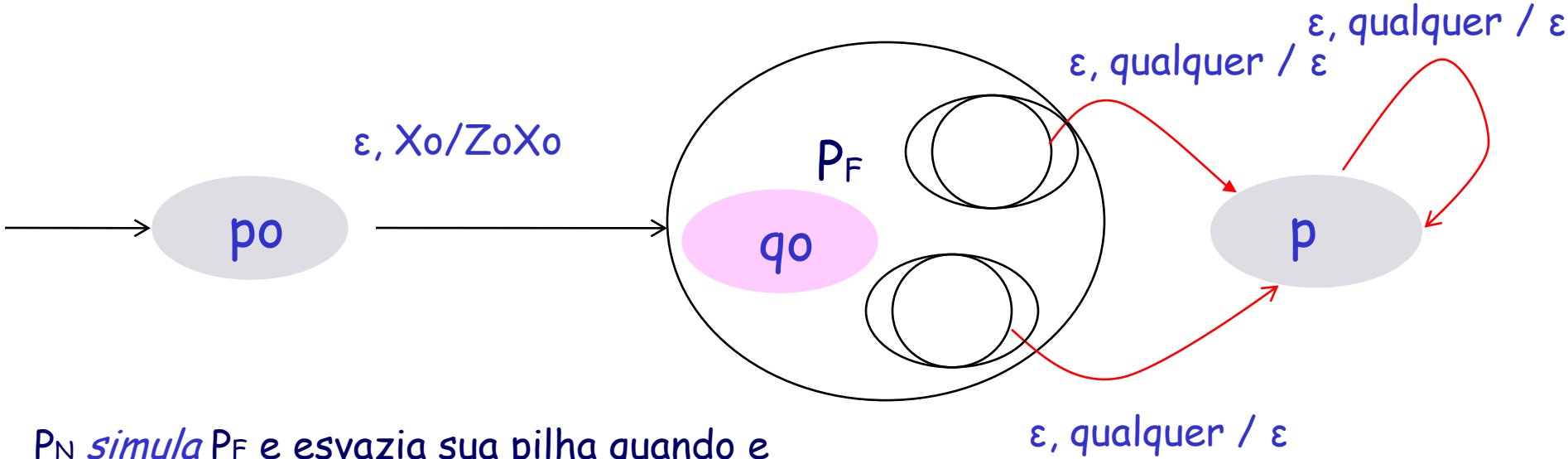
Idéia: Seja Xo um símbolo especial de início de cadeia e pilha



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

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

Idéia: Seja Xo 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 $L = \{ w \mid w \in \{0,1\}^* \text{ e } w \text{ tem igual número de 0 e 1} \}$, 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

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

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. $\delta(q1, 1, Z) = \{(q1, \varepsilon)\}$ *desempilha se 1 com 0*
6. $\delta(q1, 0, U) = \{(q1, \varepsilon)\}$ *desempilha se 0 com 1*
7. $\delta(q1, \varepsilon, B) = \{(q1, \varepsilon)\}$ *desempilha no fim da cadeia*

Configurações para 0101

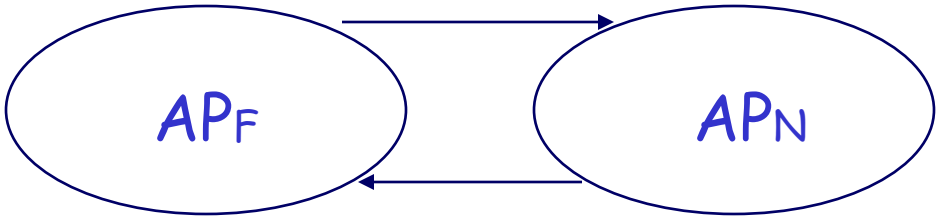
Entrada	Configuração
	(q1,B)
0	(q1,ZB)
01	(q1,B)
010	(q1,ZB)
0101	(q1,B)
ε	(q1, ε)

Exercício 2 (para casa)

- Encontre um AP M que reconheça o conjunto $L = \{ 0^n1^n \mid n > 0\}$ por pilha vazia.

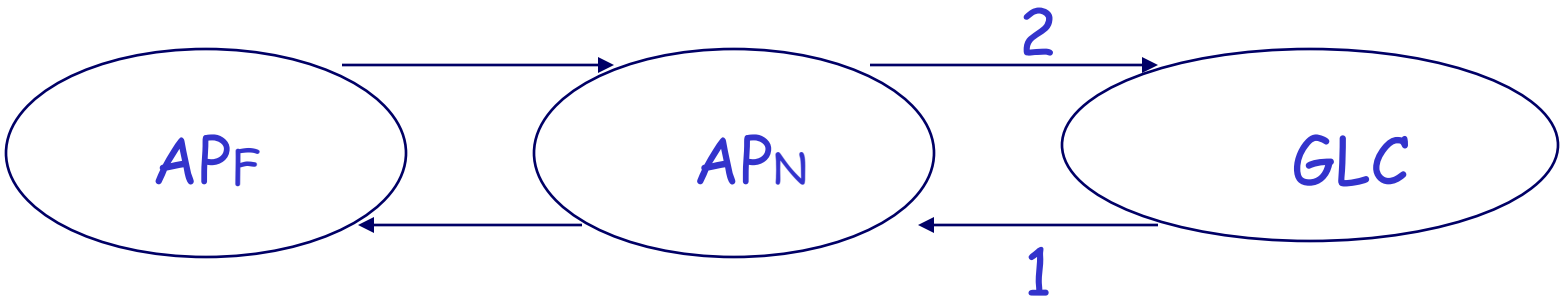
Equivalência de AP e Gramática Livre de Contexto

já mostramos a equivalência entre:



Equivalência de AP e Gramática Livre de Contexto

agora vamos mostrar que:



1. De GLC para AP

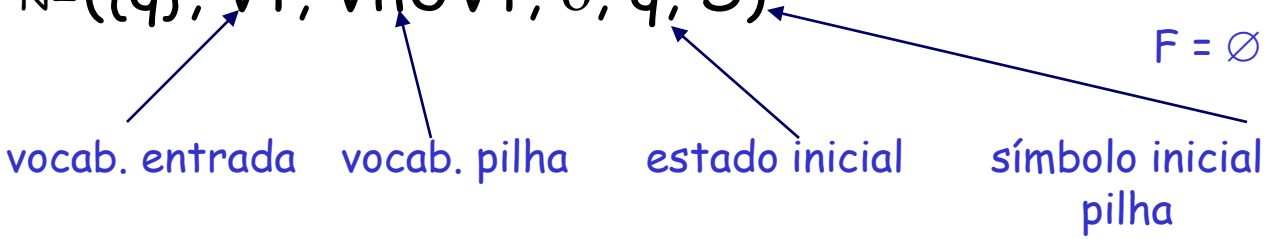
- Idéia: fazer o AP simular a sequência de formas sentenciais à esquerda usada para gerar uma dada cadeia w. O AP tem apenas um estado, e empilhamos as formas sentenciais de acordo com as regras da GLC.

  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->β, 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= (Vn, Vt, Q, S) uma GLC. Construímos um AP P_N que aceite L(G) por pilha vazia, da seguinte forma:

P_N=({q}, Vt, VnUVt, δ, q, S)



onde δ é definida por:

1. Para cada não terminal A,

$\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \rightarrow \beta$ 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 -> I | E + E |E * E|(E)
2. I -> a|b|Ia|Ib|I0|I1

Então

Σ = {a, b, 0, 1, (,), +, *}

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

$\delta(q, \varepsilon, I) = \{(q,a), (q,b), (q,Ia), (q, Ib), (q, I0), (q, I1)\}$

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

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

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, Σ, Γ, δ, qo, Zo) 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, 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:

1. $\delta(q,a,Z)$ 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)

2. 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)

APD

Não existe um APD que reconheça

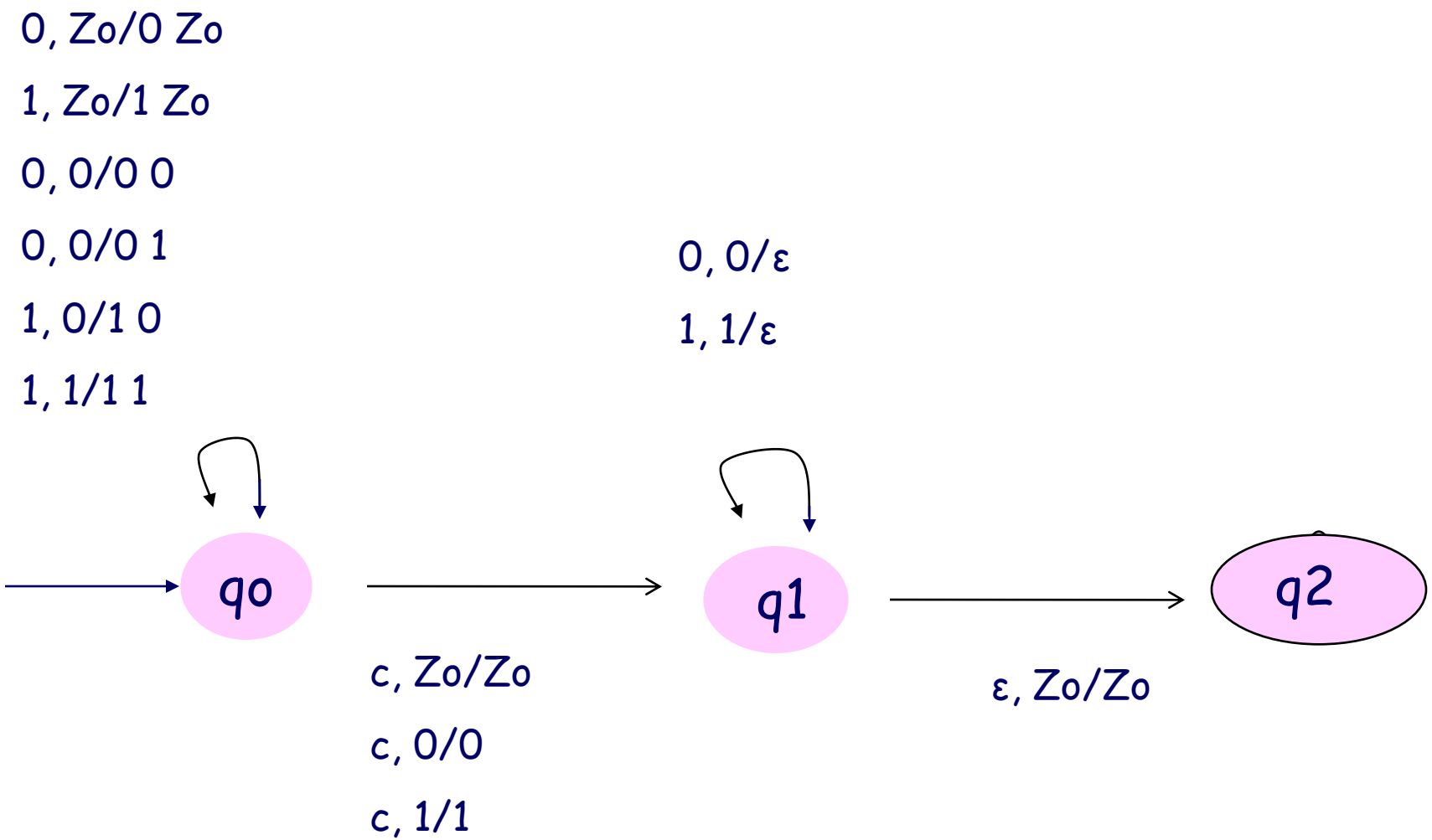
L_{wwr} = {ww^r | w está em (0+1)*} – palíndromos de comprimento par sobre {0,1}.

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

L_{wwr} = {wcw^r | w 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 Zo no topo; caso contrário, pára num estado não final.

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



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 $A = (Q, \Sigma, \delta_A, qo, F)$

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

Mostre, por indução sobre |w|, que:

$(qo,w,Zo) \vdash^*_P (p, \varepsilon, Zo)$ se e somente se $\delta_A'(qo,w)=p$

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 DIs 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.