Autômato de Pilha

- Autômato de Pilha (PDA) é um NFA com um componente extra: a pilha.
- A pilha é uma memória ilimitada. O acesso se dá na forma LIFO.
- A pilha possibilita ao PDA reconhecer um conjunto mais abrangente de linguagens que um NFA.

Exemplo

- PDA para a linguagem $\{ 0^n1^n \mid n \geq 0 \}$

1. ler os símbolos na entrada,
2. para cada 0 lido, empilhar um 0,
3. para cada 1 lido, desempilhar um 0,
4. se a pilha estiver vazia quando a entrada for λ, o string é aceito,
5. se a pilha ficar vazia e a entrada não for vazia, ou se aparecer algum 0 na entrada, depois de lido algum 1, o string não é aceito.

Autômato de Pilha

- Autômato de Pilha é uma sêxtupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$:
  - Q, Σ, q_0, F notação análoga ao AFN,
  - Γ é um alfabeto para a pilha,
  - $\delta: Q \times (\Sigma \cup \{ \lambda \}) \times (\Gamma \cup \{ \lambda \}) \rightarrow P (Q \times (\Gamma \cup \{ \lambda \}))$.

- Transições:

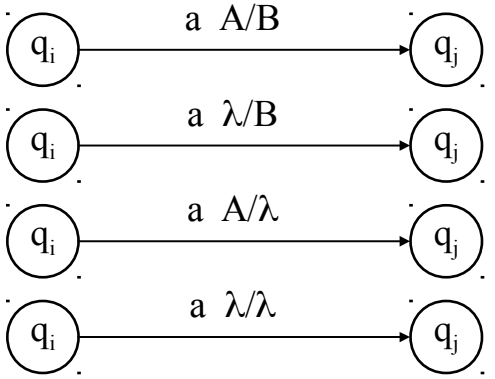
$\forall \quad \delta (q_i, a, A) = [q_j, B]$

$\forall \quad \delta (q_i, a, \lambda) = [q_j, B]$

$\forall \quad \delta (q_i, a, A) = [q_j, \lambda]$

$\forall \quad \delta (q_i, a, \lambda) = [q_j, \lambda]$

$\forall \quad \delta (q_i, \lambda, \lambda) = [q_j, \lambda]$



Autômato de Pilha

- Um string w é aceito por um autômato de pilha M se existir uma computação $[q_0, w, \lambda] \overset{*}{\Rightarrow} [q_i, \lambda, \lambda]$, $q_i \in F$.

L(M), a linguagem do autômato M, é o conjunto de todos os strings aceitos por M.

- Construa M para L(M) = $\{ 0^n1^n \mid n \geq 0 \}$.

- Construa M para L(M) = $\{ w \mid w = w^R, w \in \{ a, b \}^* \}$.

Variações sobre PDA

1. Autômatos Atômicos:
   - executam apenas uma operação em cada transição (operações atômicas):
     - ∀ $\delta(q_i, a, \lambda) = [q_j, \lambda]$ - processa um símbolo da entrada,
     - ∀ $\delta(q_i, \lambda, \lambda) = [q_j, A]$ - empilha um elemento,
     - ∀ $\delta(q_i, \lambda, A) = [q_j, \lambda]$ - desempilha um elemento.

2. Autômatos Estendidos:
   - permitem o empilhamento de N símbolos em uma transição:
     - ∀ $\delta(q_i, a, A) = [\, q_j, BCD \,]$

Variações sobre PDA

3. Autômatos com Aceitação por Estado Final:
   - processam a entrada e terminam em um estado de aceitação.

4. Autômatos com Aceitação por Pilha Vazia:
   - processam a entrada e terminam com pilha vazia.

- Nota:
  - Sempre é possível construir um PDA a partir das variações.

Equivalência entre PDA e CFG

- Teorema: uma linguagem é livre de contexto se, e somente se, é reconhecida por um autômato de pilha.

- Prova:

  1. Dado L uma LLC construir o PDA que reconheça L
     - simular as regras na pilha.

  2. Dado um PDA definir uma GLC
     - para cada par de estados P e Q definir uma regra A_{pq}.

Equivalência entre PDA e CFG

- Considere $S \rightarrow a\ S\ b\ S \mid \lambda$ na forma normal de Greibach:

$S \rightarrow \lambda$

$A \rightarrow a$

$A \rightarrow a\ A_1\ A_2 \ldots A_n$

$S' \rightarrow a\ S\ B\ S \mid a\ S\ B \mid a\ B\ S \mid a\ B \mid \lambda$

$S \rightarrow a\ S\ B\ S \mid a\ S\ B \mid a\ B\ S \mid a\ B$

$B \rightarrow b$

Forma Normal de Greibach

1. **Alterar G, gerando G_1 na FN Chomsky**

2. **Eliminar recursividade à esquerda (direta e indireta) de G_1 gerando G_2**

3. **Efetuar as substituições para colocar G_2 na forma $A \rightarrow a\ A_1\ A_2 \ldots A_n$**

Forma Normal de Greibach

G: $S \rightarrow SaB \mid a\,B$

$B \rightarrow bB \mid \lambda$

G_1:

$S' \rightarrow ST \mid AB \mid SA \mid a$

$S \rightarrow ST \mid AB \mid SA \mid a$

$B \rightarrow CB \mid b$

$A \rightarrow a$

$C \rightarrow b$

$T \rightarrow AB$

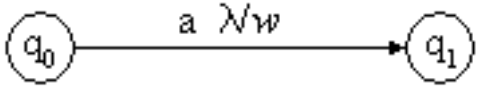
G_2: ?

Equivalência entre PDA e CFG

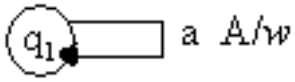
1. Seja a gramática G = (V, Σ, P, S), na forma normal de Greibach. O autômato estendido M = (Q, Σ_M, Γ, δ, q_0, F) é:

   Q = $\{q_0, q_1\}$ $\quad$ $\Sigma_M = \Sigma$ $\quad$ $\Gamma = V - \{S\}$ $\quad$ $F = \{q_1\}$

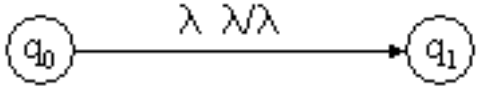
   - para cada produção S → a*w* crie a transição

   - para cada produção A → a*w* crie a transição

   - se existe a produção S' → λ crie a transição

2. Dado um PDA definir uma GLC: livro.

Pumping Lemma para LLC

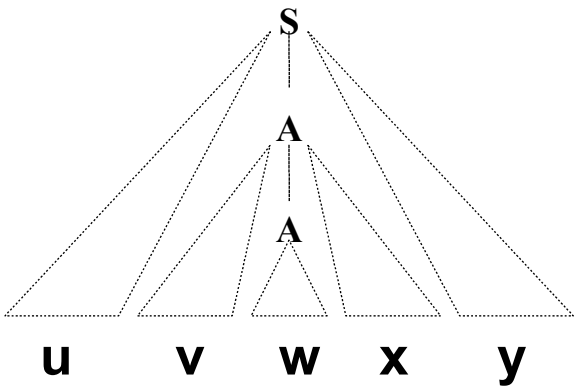
- Considere:
  - G = (V, Σ, P, S) uma gramática na forma normal de Chomsky,
  - *w* um string gerado por G e,
  - T a árvore de parse de *w*, com altura n.

- Como G está na FN Chomsky, o número de folhas de T é no máximo 2^{n-1} (árvore binária - $|w| \leq 2^{n-1}$).

Pumping Lemma para LLC

- Seja n o número de símbolos não-terminais de G.
  - se existe um string z gerado por G com $|z| \geq 2^n$, então a árvore de parse de z tem altura $\geq n+1$:
    - no caminho de maior tamanho que parte da raiz e chega a alguma folha, existe pelo menos um símbolo que se repete.

Pumping Lemma para LLC



$S \overset{*}{\Rightarrow} uAy$, $A \overset{*}{\Rightarrow} vAx$, $A \overset{*}{\Rightarrow} w$, $S \overset{*}{\Rightarrow} uAy \overset{*}{\Rightarrow} uvAxy \overset{*}{\Rightarrow} uvwxy$.

- O processo $A \Rightarrow vAx$ pode ser omitido, ou repetido i vezes, logo:
  - uv^iwx^iy também pode ser gerado pela gramática, $i \geq 0$.

Pumping Lemma para LLC

- Seja L uma LLC gerada por uma gramática G na FN de Chomsky com K símbolos não-terminais. Qualquer string $z \in L$, tal que $|z| > 2^k$ pode ser escrito na forma z = uvwxy, onde:

1. $| vwx | \leq 2^k$,
2. $| v | + | x | > 0$ e,
3. $\forall i \geq 0$, $uv^iwx^iy \in L$.

Pumping Lemma para LLC

- Provar que $L = \{ a^i b^i c^i \mid i \geq 0 \}$ não é livre do contexto:

  1. faça $w = a^k b^k c^k$, com $k = 2^n$, (n o número de variáveis da gramática G que gere L),

  2. como $|w| > 2^n$ aplique o lema do bombeamento.

Propriedades das Linguagens Livres do Contexto

- Teorema: se L_1 e L_2 são linguagens livres do contexto, então:
  - $L_1 \cup L_2$, $(L_1)^*$, L_1L_2 são LLC, mas
- Teorema: o conjunto das LLC's não é fechado sob interseção ou complemento:
  1. $L_1 \cap L_2$ pode não ser LLC - $a^ib^ic^k \cap a^kb^kc^i = a^kb^kc^k$ $i, k > 0$.

Propriedades das Linguagens Livres do Contexto

2. $(L_1)'$ pode não ser LLC. Prova:

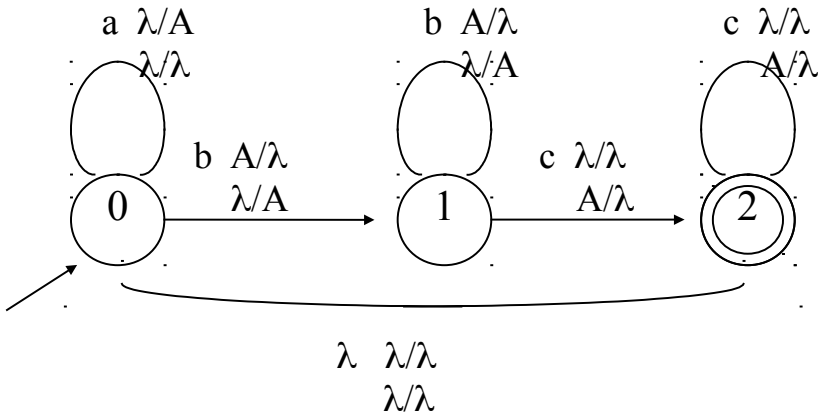
   - considere L_1 e L_2 LLC. Faça $L = ((L_1)' \cup (L_2)')'$. Note que o resultado contradiz (1).
   - $L = \{ ww \mid w \in \{ a,b \}^* \}$ não é LLC. Mas L' é LLC.

- Teorema: seja L_1 uma LR e L_2 uma LLC, então:
  - $L_1 \cap L_2$ é uma LLC. Prova por construção do PDA.

Autômatos com 2 Pilhas

- Em cada transição o autômato verifica o símbolo no topo de cada pilha e executa uma operação em cada pilha.

- Exemplo: autômato para L = { $a^i b^i c^i$ | i ≥ 0 }.



Autômatos com 2 Pilhas

- Nem toda linguagem reconhecida por um autômato com duas pilhas é livre do contexto.

- E para $L = \{ a^i b^i c^i d^i \mid i \geq 0 \}$. Precisamos de 3 pilhas?