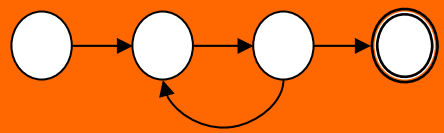
Automata e Linguagens Formais

CTC 34



7

Prof. Carlos H. C. Ribeiro

carlos@ita.br

Automata de Pilha

APs e Linguagens Livres de Contexto

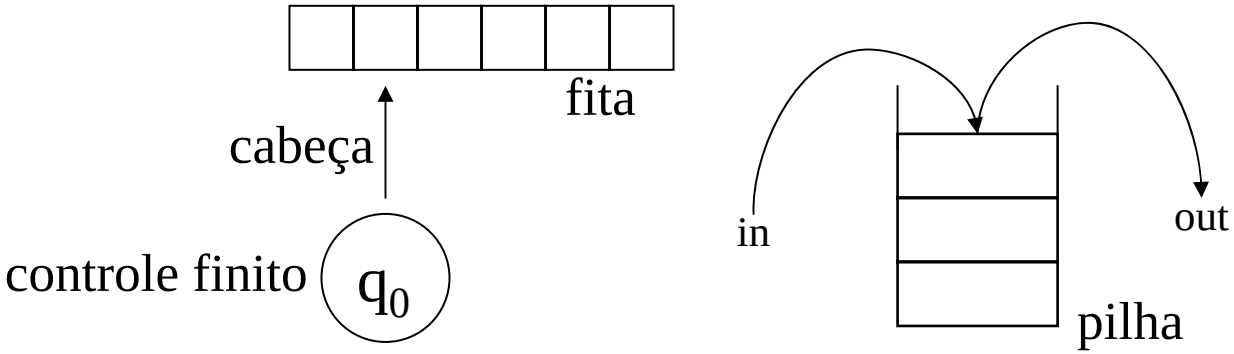
O *Pumping Lemma* para LLCs

Propriedades de LLCs

Automata de Pilha e Máquinas de Estados

Um autômato de pilha é uma **máquina de estados com uma pilha LIFO.**

- um registrador de estado interno (controle finito) + uma fita segmentada + cabeça de leitura + pilha LIFO

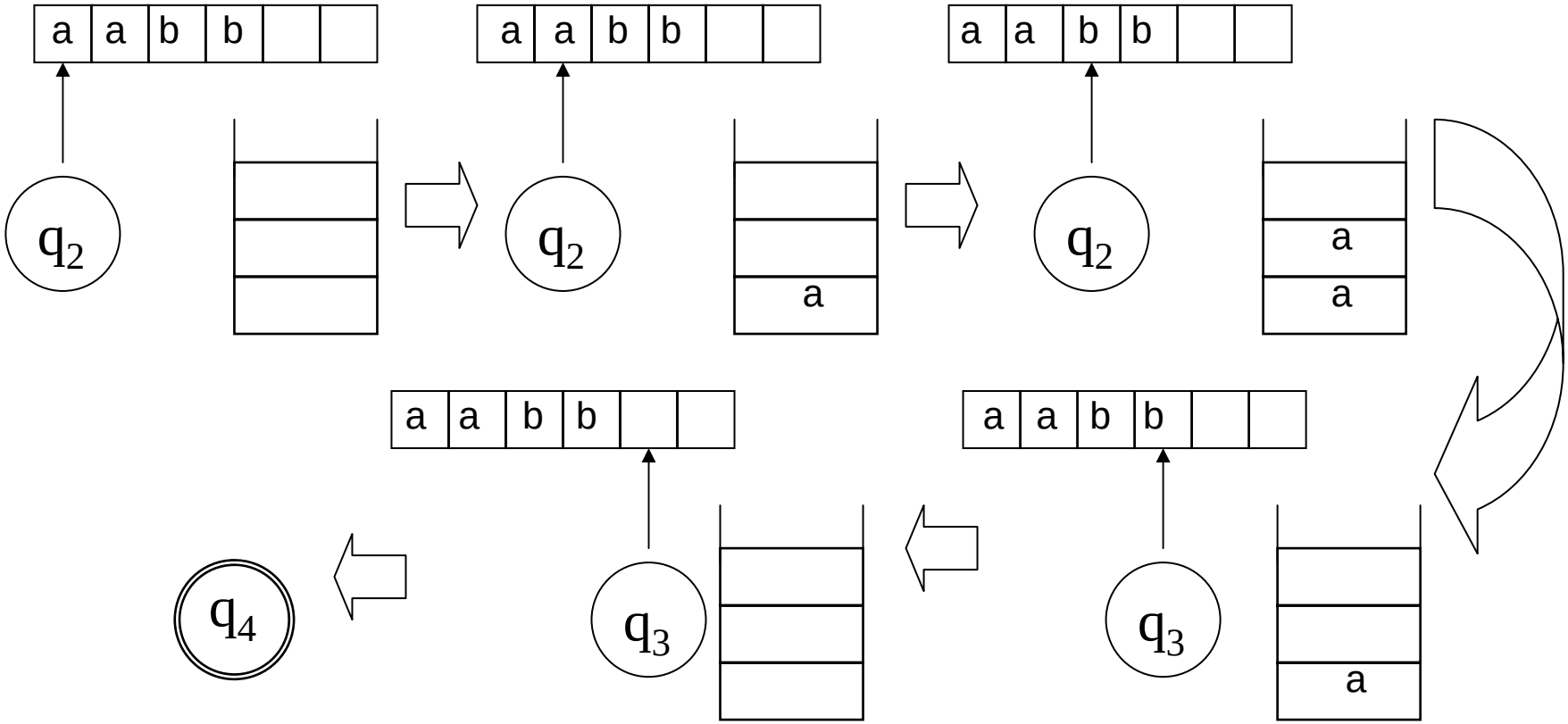


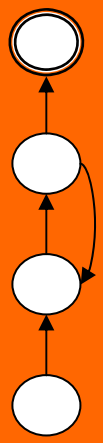
- fita: armazena uma cadeia de Σ (1 símbolo/segmento).
- pilha: armazena uma cadeia de Γ (1 símbolo/segmento)
- cabeça: lê segmento da fita
- registrador: altera estado e conteúdo do topo da pilha de acordo com δ, move fita um segmento para a esquerda.
- uma computação **termina** quando a cadeia da fita “acaba”.

Um Exemplo

AP para reconhecer $\{a^n b^n \mid n \geq 0\}$

- Ao ler a na fita, copio-o para a pilha. Ao ler b na fita, retiro o a do topo da pilha.
- Se a cadeia termina exatamente quando a pilha esvazia, aceito a entrada.





Autômato de Pilha: Definição Formal

A pilha adiciona memória ao autômato finito determinístico usual.
Também chamado **autômato *pushdown***.

Formalmente: $M=(Q,\Sigma,\Gamma,\delta,q_0,F)$, onde

Q = conjunto de estados
Σ=alfabeto da fita
Γ=alfabeto da pilha (inclui um símbolo especial \$, indicador de pilha vazia)
$q_0 \in Q$ = estado inicial
δ=função de transição $Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\})$ em subconjuntos de $Q \times (\Gamma \cup \{\varepsilon\})$
F=conjunto de estados finais

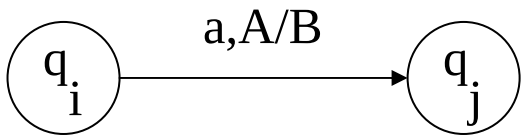
- Uma **computação** em um AP é uma sequência de transições a partir da situação inicial.
- Cadeia é **aceita** pelo AP se a computação termina em um estado $q \in F$.
- Cadeia é **rejeitada** pelo AP se a computação termina em um estado $q \notin F$.
- Situação inicial: estado inicial q_0 e pilha vazia.
- Primeiro passo: transição ε para colocar \$ na pilha, indicando pilha vazia.

A Função de Transição δ

$Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow$ subconjuntos de $Q \times (\Gamma \cup \{\varepsilon\})$

- estado
- símbolo lido na fita
- símbolo lido no topo da pilha e retirado
- possíveis configurações (estado, símbolo colocado no topo da pilha) resultantes

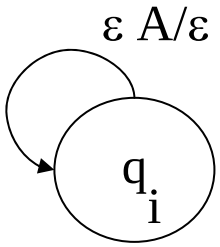
$\delta(q_i, a, A) = \{[q_j, B]\}$:



Estado q_i, símbolo a lido na fita, símbolo A no topo da pilha lido e retirado. Estado resultante q_j, símbolo B colocado no topo da pilha.

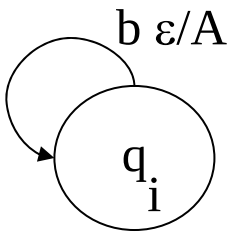
Obs: Poderia ter, por exemplo: $\delta(q_i, a, A) = \{[q_j, B], [q_k, C]\}$ (**Um AP admite transições não-determinísticas!)**

Representação por Diagrama de Estados: Exemplos



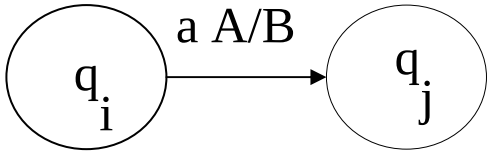
Nenhum símbolo lido na fita, *A* é lido e retirado da pilha, nada é colocado no topo da pilha.

$$\delta(q_i, \varepsilon, A) = \{[q_i, \varepsilon]\}$$



Símbolo *b* lido na fita, nada lido da pilha, *A* é colocado no topo da pilha.

$$\delta(q_i, b, \varepsilon) = \{[q_i, A]\}$$



Símbolo *a* lido na fita, símbolo *A* lido e retirado do topo da pilha, símbolo *B* colocado na pilha.

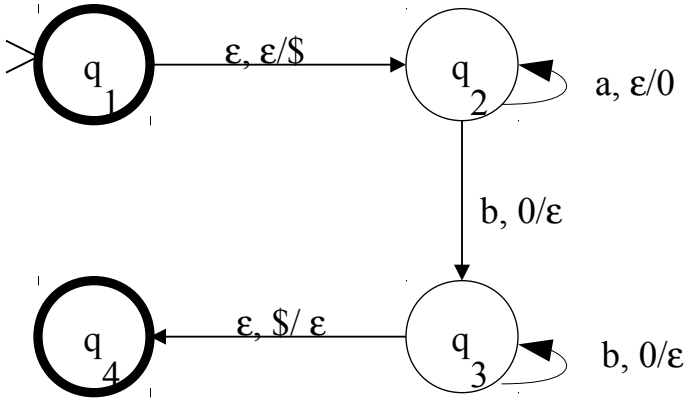
$$\delta(q_i, a, A) = \{[q_j, B]\}$$

Exemplo 1: AP para $L=\{a^n b^n \mid n \geq 0\}$

$Q=\{q_1,q_2,q_3,q_4\}$, $\Sigma=\{a,b\}$, $\Gamma=\{0,\$\}$, $F=\{q_1,q_4\}$

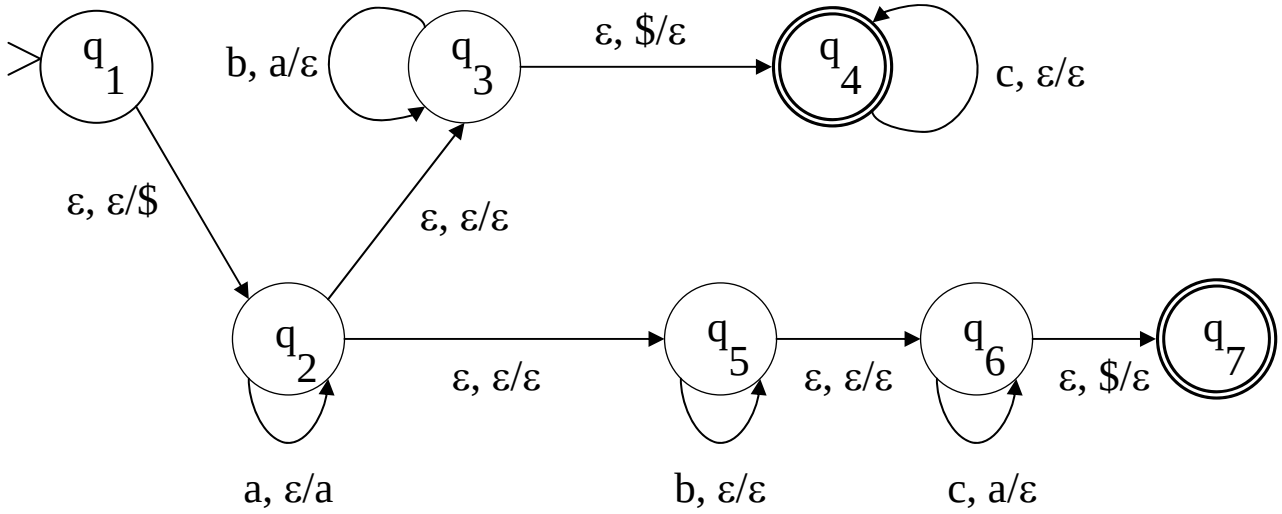
δ:

Fita	a			b			ε		
Pilha	0	$	ε	0	$	ε	0	$	ε
q_1									$\{(q_2,\$)\}$
q_2			$\{(q_2,0)\}$	$\{(q_3, \varepsilon)\}$					
q_3				$\{(q_3, \varepsilon)\}$				$\{(q_4, \varepsilon)\}$	
q_4									



Exemplo 2: AP para $L=\{a^i b^j c^k \mid i,j,k \geq 0 \text{ e } i=j \text{ ou } i=k\}$

$Q=\{q_1,q_2,q_3,q_4,q_5,q_6,q_7\}$, $\Sigma=\{a,b\}$, $\Gamma=\{a,\$\}$, $F=\{q_4,q_7\}$



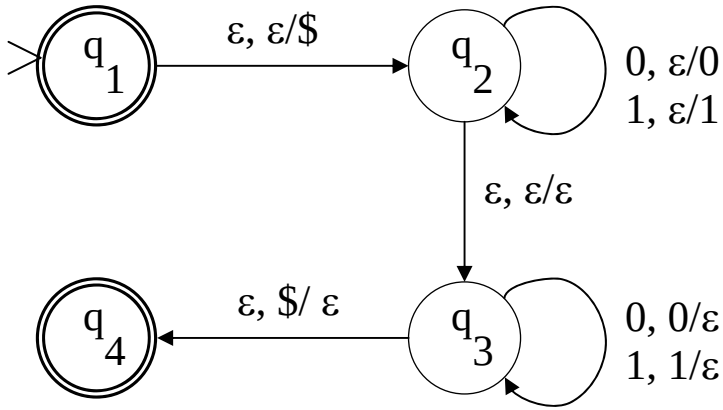
Observe que este é não-determinístico...

Pode ser provado que o não determinismo deste AP é <u>essencial</u>

para o reconhecimento de L

Exemplo 3: AP para $L=\{ww^R \mid w \in \{0,1\}^*\}$

$Q=\{q_1,q_2,q_3,q_4\}$, $\Sigma=\{0,1\}$, $\Gamma=\{0,1,\$\}$, $F=\{q_1,q_4\}$



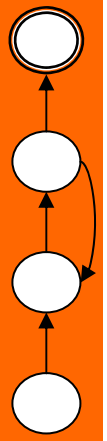
- Copia símbolos lidos para pilha.
- A cada passo, tenta “adivinhar” se o meio da cadeia foi atingido, e retira um símbolo da pilha verificando se é o mesmo lido na fita.
- Para cadeias aceitas, a pilha vai se esvaziar no momento que termina a leitura da cadeia termina.

Configurações em APDs

- **Def.** Uma **configuração instantânea** de um APD é um tripla $[q_i, w, \alpha]$, em que q_i é o estado, *w* é a cadeia não processada e α é o conteúdo da pilha (símbolo mais à esquerda no topo).
- **Def.** A função $\vdash_M$ é definida por $[q_i, w, \alpha] \vdash_M [q_j, v, \beta]$ e indica que a configuração $[q_j, v, \beta]$ pode ser obtida a partir da configuração $[q_i, w, \alpha]$ por uma transição do APD *M*.

  A notação $[q_i, w, \alpha] \vdash_M^* [q_j, v, \beta]$ indica que a configuração $[q_j, v, \beta]$ pode ser obtida a partir da configuração $[q_i, w, \alpha]$ por zero ou mais transições do APD *M*.

Configurações em APDs: Exemplos



M: $Q=\{q_0, q_1, q_2, q_3\}$ $\Sigma=\{a,b\}$

$\Gamma=\{A,\$\}$ $F=\{q_3\}$

$\delta(q_0,\varepsilon,\varepsilon)=\{[q_1,\$]\}$ $\delta(q_1,\varepsilon,\varepsilon)=\{[q_2,A]\}$

$\delta(q_2,a,A)=\{[q_2,A]\}$

$\delta(q_2,b,A)=\{[q_3,A]\}$

$\delta(q_3,b,A)=\{[q_3,A]\}$

$[q_0,aabb,\varepsilon] \vdash [q_1, aabb, \$] \vdash [q_2, aabb, A\$] \vdash [q_2, abb, A\$] \vdash [q_2, bb, A\$] \vdash [q_3, b, A\$] \vdash [q_3, \varepsilon, A\$]$

Computação para *aabb* terminou em estado final → cadeia **aceita** pelo AP.

M: $Q=\{q_0, q_1, q_2, q_3, q_4\}$ $\Sigma=\{a,b\}$

$\Gamma=\{A,\$\}$ $F=\{q_4\}$

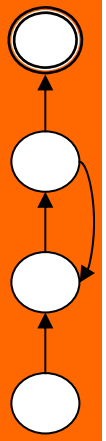
$\delta(q_0,\varepsilon,\varepsilon)=\{[q_1,\$]\}$ $\delta(q_1,\varepsilon,\varepsilon)=\{[q_2,A]\}$

$\delta(q_2,a,A)=\{[q_3,A]\}$ $\delta(q_3,\varepsilon,\varepsilon)=\{[q_3, A]\}$

$\delta(q_2,b,A)=\{[q_3,A]\}$ $\delta(q_3,b,A)=\{[q_4, \varepsilon]\}$

$[q_0,abb,\varepsilon] \vdash [q_1,abb,\$] \vdash [q_2,abb,A\$] \vdash [q_3,bb,A\$] \vdash [q_3,bb,AA\$] \vdash [q_4,b,A\$]$

Computação para *abbb* não terminou → cadeia não **aceita** pelo AP.

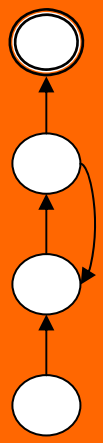


APDs e Linguagens Livres de Contexto

Teorema (*Chomsky, 1962*): L é uma linguagem livre de contexto sss L é aceita por algum APD *M* que aceita por pilha vazia (ou por estado final).

Análogo ao teorema de Kleene para AFs e linguagens regulares!

Prova: Sipser, págs. 107/114



O *Pumping Lemma* para Linguagens Livres de Contexto

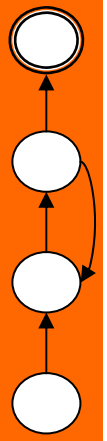
Seja L uma linguagem livre de contexto. Então existe uma constante n (dependente apenas da linguagem L) tal que, se $z \in L$ e $length(z) \geq n$, podemos escrever z=uvwxy de modo que:

- $length(vx) \geq 1$
- $length(vwx) \leq n$
- $u\ v^i\ w\ x^i\ y \in L$, para todo $i \geq 0$

Corresponde ao bombeamento de duas subcadeias separadas por w.

Como no caso de linguagens regulares, posso usar o *Pumping Lemma* para mostrar que uma linguagem não é livre de contexto. A idéia é prestar atenção especial às subcadeias bombeadas v e x, e, assumindo que a linguagem seja LC, tentar chegar a alguma contradição.

Exemplo 1



Prove que L={$a^i b^i c^i$ | i ≥ 1} não é livre de contexto.

Suponha que L seja LC. Então, vale o *pumping lemma* e existe um n correspondente. Seja z = $a^n b^n c^n$ ∈ L. Como length(z)≥n, então posso escrever:

z = uvwxy, com length(vx)≥1, length(vwx)≤n, e $uv^i wx^i y$ ∈ L

i) vx não pode conter a's e c's, pois teria que colocar n b's entre eles (impossível, pois length (vwx)≤n).

ii) v e x não podem ter apenas a's, pois se assim fosse teríamos

$uv^0 wx^0 y$ = uwy com n b's e n c's, mas menos do que n a's, já que alguns destes estariam em vx (pois length(vx)≥1). Assim, não poderia escrever uwy ∈ L como $a^k b^k c^k$.

iii) vx não pode conter apenas a's e b's, pois se assim fosse teríamos

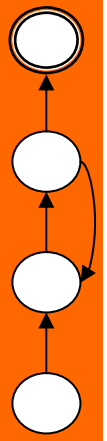
$uv^0 wx^0 y$ = uwy com n c's, mas menos do que n a's ou b's, já que alguns destes estariam em vx (pois length(vx)≥1). Assim, não poderia escrever uwy ∈ L como $a^k b^k c^k$.

Por raciocínio análogo a ii), v e x não podem ter apenas b's ou c's.

Por raciocínio análogo a iii), vx não pode conter apenas a's e c's ou b's e c's.

Assim, z=uvwxy ∈ L (por hipótese), mas v e x não podem ser formados por nenhuma combinação de símbolos do alfabeto da linguagem L: **contradição ⇒ L não é livre de contexto.**

Exemplo 2



Prove que $L=\{a^i b^k c^i d^k \mid i, k \geq 1\}$ não é livre de contexto.

Suponha que L seja L.C. Então, vale o *pumping lemma* e existe um n correspondente. Seja $z = a^n b^n c^n d^n \in L$. Como length(z)≥n, então posso escrever:

$z = uvwxy$, com $length(vx)\geq 1$, $length(vwx)\leq n$, e $uv^i wx^i y \in L$

i) vx não pode conter mais do que dois símbolos diferentes e consecutivos, pois senão teríamos length(vwx)≤n.

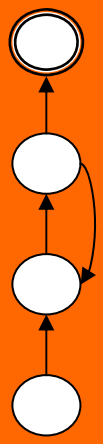
ii) Se vx tivesse apenas a's, então uwy teria menos a's do que c's e não poderíamos escrever $uwy \in L$ como $a^k b^{k'} c^k d^{k'}$.

iii) Se vx tivesse apenas a's e b's, então ainda assim uwy teria menos a's do que c's e não poderíamos escrever $uwy \in L$ como $a^k b^{k'} c^k d^{k'}$.

Por raciocínio análogo a ii), v e x não podem ter só b's, c's ou d's.

Por raciocínio análogo a iii), vx não pode conter apenas b's e c's ou c's e d's.

Assim, $z=uvwxy \in L$ (por hipótese), mas v e x não podem ser formados por nenhuma combinação de símbolos do alfabeto da linguagem L: **contradição ⇒ L não é livre de contexto.**



Exemplo 3

Prove que L={w $\in$ **a*** | length(w) é primo} não é livre de contexto.

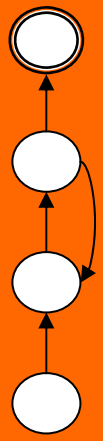
Suponha que L seja L.C. Então, vale o *pumping lemma* e existe um n correspondente: Seja z = $a^k \in$ L, com k um número primo maior do que n. Como length(z)=k$\geq$n, então posso escrever:

z = uvwxy, com length(vx)$\geq$1, length(vwx)$\leq$n, e $uv^i wx^i y \in$ L

Seja m = length(u) + length(w) + length(y). Então, o comprimento de qualquer cadeia $z = uv^i wx^i y$ é:

$length(uwy)+ length(v^i x^i) = length(uwy) + i\ length(vx) = m + i\ (k-m)$

Em particular, $length(uv^{k+1} wx^{k+1} y) = m + (k+1)(k-m) = k(k-m+1)$, que é divisível por k! **Contradição...**



LLCs : Propriedades de Fechamento

Teorema: Linguagens Livres de Contexto são fechadas sob as operações de:

União / Concatenação / Fechamento de Kleene / Homomorfismo / Homomorfismo inverso

Teorema: Seja L_R uma linguagem regular e L_L uma linguagem livre de contexto. Então $L = L_R \cap L_L$ é livre de contexto.

Obs: Linguagens Livres de Contexto **não** são fechadas sob as operações de:

Intersecção / Complementação