Observações:

- Enquanto que todas as Linguagens Regulares são reconhecíveis por Autómatos **Deterministas** (Finitos), nem todas as Linguagens Independentes do Contexto são reconhecíveis por Autómatos **Deterministas** (de Pilha);

- Prova-se que, toda a LIC reconhecível por um ADP (por Estado Final ou por Pilha Vazia) tem uma Gramática geradora não ambígua;

- A existência de Autómatos Deterministas é vital para o Processamento automático de Linguagens. Sistemas como o lex são geradores automáticos de Autómatos Finitos Deterministas, do mesmo modo que sistemas como o yacc são geradores automáticos de Autómatos Deterministas de Pilha;

- É também importante, tanto para o Processamento de Linguagens como para a demonstração de propriedades, que as produções das LIC's tenham "formas" determinadas ...

Operações de "simplificação" de Gramáticas

1. Eliminação de produções-ε :

{ uma **produção-ε** tem a forma $A \rightarrow \varepsilon$ }

Dada uma gramática G = (V, T, P, S), tal que $\varepsilon \notin L(G)$,
construir G', sem produções-ε, de modo a que L(G) = L(G').

Definição: Uma variável $A \in V$ diz-se **anulável** quando $A \Rightarrow^* \varepsilon$.

Algoritmo: { eliminação de variáveis anuláveis }

Para cada produção $A \rightarrow X_1 X_2 \ldots X_n$

construir uma produção $A \rightarrow \alpha_1 \alpha_2 \ldots \alpha_n$

onde $\alpha_i = \begin{cases} X_i & \text{se } X_i \text{ não é anulável} \\ X_i \text{ ou } \varepsilon & \text{se } X_i \text{ é anulável} \end{cases}$

e nem todos os X_i são ε;

Eliminar todas as produções-ε.

Exemplo:

$S \rightarrow AB$
$A \rightarrow aAA \mid \varepsilon$
$B \rightarrow bBB \mid \varepsilon$

As variáveis anuláveis são S, A e B.

As novas produções serão:

$S \rightarrow AB \mid A \mid B$
$A \rightarrow aAA \mid aA \mid a$
$B \rightarrow bBB \mid bB \mid b$

2. Eliminação de produções unitárias :

{ uma **produção unitária** tem a forma $A \rightarrow B$, com $A, B \in V$ }

Dada uma gramática G = (V, T, P, S) construir G',
sem produções unitárias, de modo a que L(G) = L(G').

Definição: $\langle A, B \rangle$ diz-se um **par unitário** quando $A \Rightarrow^* B$, usando só produções unitárias.

Algoritmo: Identificar todos os pares unitários;

Para cada par unitário $\langle A, B \rangle$, e para todas as produções não unitárias de B: $B \rightarrow \beta_1 \mid \beta_2 \mid ... \mid \beta_n$
juntar as produções $A \rightarrow \beta_1 \mid \beta_2 \mid ... \mid \beta_n$;

Eliminar todas as produções unitárias.

Exemplo:

$E \rightarrow T \mid E + T$
$T \rightarrow F \mid T * F$
$F \rightarrow I \mid (E)$
$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Os pares unitários são:
$\langle E, T \rangle, \langle E, F \rangle, \langle E, I \rangle, \langle T, F \rangle, \langle T, I \rangle, \langle F, I \rangle$

As novas produções serão:

$E \rightarrow E + T \mid T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$T \rightarrow T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$F \rightarrow (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

3. Eliminação de símbolos inúteis :

{ um **símbolo** é **inútil** se não for útil ...}

Dada uma gramática G = (V, T, P, S) construir G',
sem símbolos inúteis, de modo a que L(G) = L(G').

Definição: Um símbolo $X \in V \cup T$ diz-se **útil** numa Gramática G = (V, T, P, S) se existir alguma derivação
$S \Rightarrow^* \alpha X \beta \Rightarrow^* w$ com $w \in T^*$ e $\alpha, \beta \in (V \cup T)^*$.

Portanto, se X for um símbolo **útil** então:

1. $S \Rightarrow^* \alpha X \beta$ para alguns $\alpha, \beta \in (V \cup T)^*$ (X é **atingível**)
2. $X \Rightarrow^* w$ para algum $w \in T^*$. (X é **gerador**)

Algoritmo para a determinação do conjunto dos **símbolos geradores**:

Caso Base: Todos os símbolos terminais são geradores;

Indução: Se existir uma produção $A \rightarrow \alpha$, $\alpha \in (V \cup T)^*$ onde todas as variáveis em α são geradoras, então A é um símbolo gerador.

Exemplo:
$S \rightarrow AB \mid C$
$A \rightarrow 0B \mid C$
$B \rightarrow 1 \mid A0$
$C \rightarrow AC \mid C1$

Os terminais	0 e 1 são geradores
por B → 1	B é gerador
por A → 0B	A é gerador
por S → AB	S é gerador

Portanto o símbolo C pode ser **eliminado**, ficando:

S → AB
A → 0B
B → 1 | A0

Algoritmo para a determinação do conjunto dos **símbolos atingíveis**:

Caso Base: S é atingível;

Indução: Se existir uma produção $A \rightarrow \alpha$, $\alpha \in (V \cup T)^*$ onde A é atingível,
então todas as variáveis em α são atingíveis.

Exemplo:
S → A
A → aA | ε
B → bA

A variável B não é atingível, podendo portanto ser eliminada, bem como a produção B → bA.

Para eliminar todos os símbolos inúteis:

1º : eliminar os símbolos **não geradores**;
2º : eliminar os **símbolos não atingíveis**.

(por esta ordem!)

Exemplo: Eliminar todos os símbolos e produções **inúteis** da gramática G = ({S, A, B, C}, {a, b}, P, S) com P:

$S \rightarrow aS \mid A \mid C$
$A \rightarrow a$
$B \rightarrow aa$
$C \rightarrow aCb$

Os símbolos geradores são {S, A, B}, ficando:

$S \rightarrow aS \mid A$
$A \rightarrow a$
$B \rightarrow aa$

e como os símbolos atingíveis são {S, A},

a nova gramática será G' = ({S, A}, {a, b}, P', S) com P':

$S \rightarrow aS \mid A$
$A \rightarrow a$

"Simplificação" de Gramáticas:

Dada uma gramática G = (V, T, P, S), tal que $L(G) \neq \emptyset$, construir G', tal que $L(G') = L(G) \setminus \{\varepsilon\}$ e G' não tem produções-ε, nem produções unitárias nem símbolos inúteis.

1º : eliminar as **produções-ε**;
2º : eliminar as **produções unitárias**;
3º : eliminar os **símbolos** e as **produções inúteis**.

- # Formas Normais

A Forma Normal de Chomsky:

Prova-se que:

> Para toda a Linguagem Independente do Contexto L, existe uma Gramática G que gera L \ {ε}, sem símbolos inúteis, onde cada produção tem uma das formas: $A \rightarrow BC$
> $A \rightarrow a$.

Redução à Forma Normal de Chomsky:

1º : eliminar as **produções-ε**;
2º : eliminar as **produções unitárias**;
3º : eliminar os **símbolos** e as **produções inúteis**.
4º : eliminar **símbolos terminais** do lado direito de produções;
5º : reduzir os lados direitos das produções a **duas variáveis**;

(4º) Eliminar os símbolos terminais do lado direito das produções:

Para cada símbolo terminal a, b, c, ...
juntar novas variáveis X_a, X_b, X_c, ...
bem como novas produções $X_a \rightarrow a$, $X_b \rightarrow b$, $X_c \rightarrow c$, ...
Substituir a por X_a , b por X_b , c por X_c , ...
(quando não aparecerem isolados) nos lados direitos da produções.

(5º) Reduzir os lados direitos das produções a duas variáveis:

Substituir cada produção da forma $A \rightarrow B_1 B_2 B_3$...
por $A \rightarrow B_1 B_{23}$
$B_{23} \rightarrow B_2 B_{34}$
$B_{34} \rightarrow B_3 B_{45}$ ...

Exemplo: Reduzir à Forma Normal de Chomsky:

$S \longrightarrow ASB \mid \varepsilon$
$A \longrightarrow aAS \mid a$
$B \longrightarrow SbS \mid A \mid bb$

1º : eliminar as **produções-ε**;

Como o único símbolo anulável é S, que nunca aparece isolado num lado direito de produção, não há produções a eliminar. Basta detectar as ocorrências (anuláveis) de S:

$S \longrightarrow ASB \mid AB$
$A \longrightarrow aAS \mid aA \mid a$
$B \longrightarrow SbS \mid bS \mid Sb \mid b \mid A \mid bb$

2º : eliminar as **produções unitárias**;

Como a única produção unitária é $B \longrightarrow A$, basta substituir:

$S \longrightarrow ASB \mid AB$
$A \longrightarrow aAS \mid aA \mid a$
$B \longrightarrow SbS \mid bS \mid Sb \mid b \mid aAS \mid aA \mid a \mid bb$

3º : eliminar os **símbolos** e as **produções inúteis**.

Todos os símbolos são geradores e atingíveis.

4º : eliminar os **símbolos terminais** do lado direito das produções, quando não ocorrerem isolados;

Introduzir as variáveis C e D e as produções $C \longrightarrow a$ e $D \longrightarrow b$:

S → ASB | AB
A → CAS | CA | a
B → SDS | DS | SD | b | CAS | CA | a | DD
C → a
D → b

5º : reduzir os lados direitos das produções a **duas variáveis**;

Por fim, para reduzir as sequências ASB, CAS e SDS, introduzir as variáveis E, F e G.
A gramática pretendida, na Forma Normal de Chomsky, será então:

S → AE | AB
A → CF | CA | a
B → SG | DS | SD | b | CF | CA | a | DD
C → a
D → b
E → SB
F → AS
G → DS

A Forma Normal de Greibach:

Prova-se que:

> Para toda a Linguagem Independente do Contexto L, existe uma Gramática G que gera L \ {ε}, onde cada produção tem a forma:
>
> $A \rightarrow a\,\alpha$ com $a \in T$ e $\alpha \in V^*$.

{ Um pequeno regresso às Linguagens Regulares}

A Forma Normal das Gramáticas Regulares:

Prova-se que:

> Toda a Linguagem Regular pode ser gerada por uma Gramática onde cada produção tem uma das formas:
>
> $X \rightarrow a\ Y$ com $a \in T$ e $X, Y \in V$
> $X \rightarrow \varepsilon$ com $X \in V$

Este resultado permite demonstrar de modo bem simples que:

Teorema: Para toda a Gramática Regular G existe um Autómato Finito A (AFND-ε) tal que L(A) = L(G).

Demonstração / Construção:

Seja G = (V, T, P, S) uma Gramática Regular escrita na sua Forma Normal.

Construímos o Autómato Finito A = (Q, T, δ, Q_0, F), fazendo:

- Q = V
- Q_0 = S
- F = { $X \in V \mid [X \rightarrow \varepsilon] \in P$ }
- $\delta(X, a) = Y$ sse $[X \rightarrow a\ Y] \in P$.

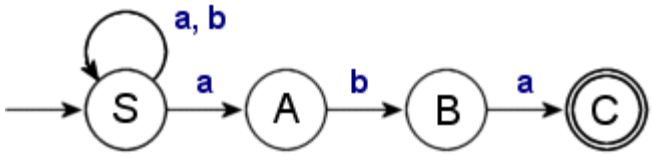
Exemplo: Seja G = ({S, A, B, C}, {a, b}, P, S),

com P dado por:

$$S \rightarrow aS \mid bS \mid aA$$
$$A \rightarrow bB$$
$$B \rightarrow aC$$
$$C \rightarrow \varepsilon$$

O Autómato pretendido é

A = ({S, A, B, C}, {a, b}, δ, S, {C}) com



cuja Expressão Regular é (a + b)* a b a.

- É fácil verificar que toda a palavra de L(G) é reconhecida por A, bem como o inverso.
- Também não é difícil verificar que:

Teorema: Para todo o Autómato Finito Não Determinista A, existe uma Gramática Regular G tal que L(G) = L(A).

O Lema da Bombagem
para Linguagens Independentes do Contexto

Teorema: Seja L uma Linguagem Independente do Contexto.
Então existe um inteiro positivo n tal que, para toda a palavra $z \in L$ com $|z| \geq n$, existem $u, v, w, x, y \in \Sigma^*$ tais que:

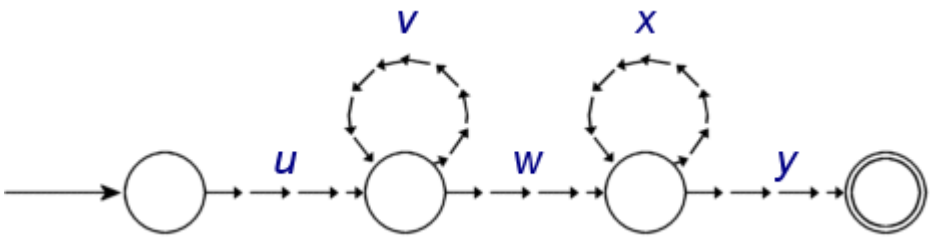
- $z = uvwxy$
- $|vwx| \leq n$
- $|vx| > 0$
- $\forall\, k \geq 0,\ uv^k wx^k y \in L$

de modo mais formal:

$\forall$ L Linguagem Independente do Contexto, $\exists\, n \in \mathbb{N}$:
$\forall\, z \in L$ com $|z| \geq n$, $\exists\, u, v, w, x, y \in \Sigma^*$: $z = uvwxy$
com $|vwx| \leq n$ e $|vx| > 0$: $\forall\, k \in \mathbb{N}_0,\ uv^k wx^k y \in L$.

ou seja:

Toda a palavra (suficientemente longa) de L pode ser dividida em cinco partes de modo a que, com um número igual de repetições da segunda e quarta partes, continue a ser uma palavra de L.



Um exemplo: G = ({S, A}, {a, b, c}, P, S), na Forma Normal de Chomsky com,

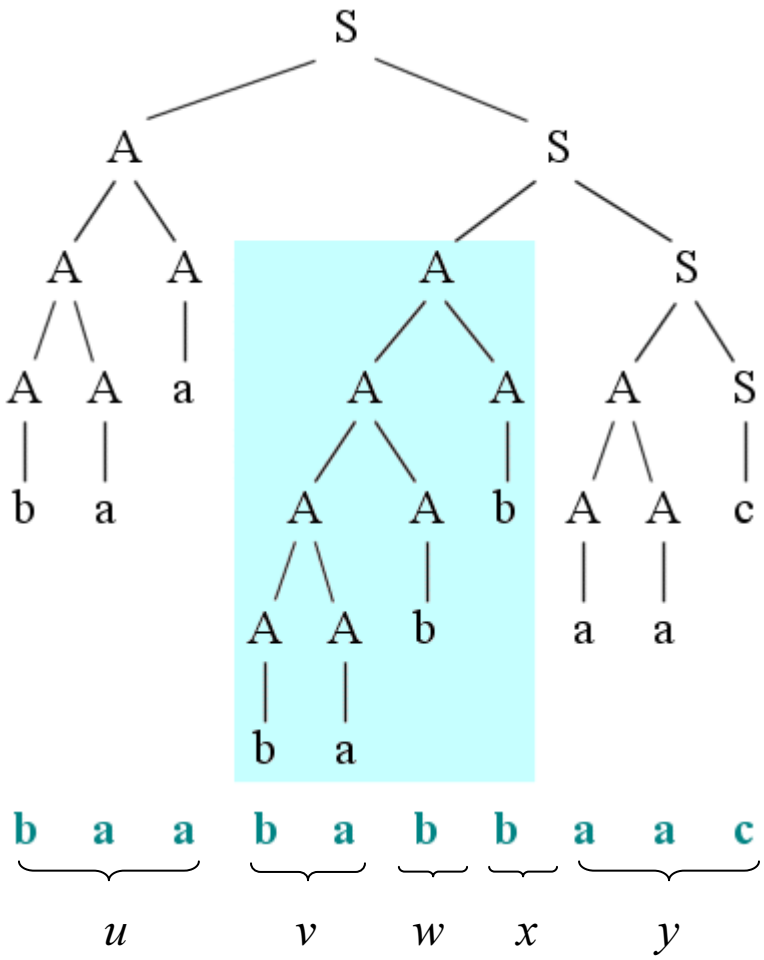
$S \rightarrow AS \mid c$

$A \rightarrow AA \mid a \mid b$

Para uma palavra "suficientemente longa" tal como w = baababbaac temos a árvore de derivação:

$S \Rightarrow AS$

$\Rightarrow^*$ baa A aac



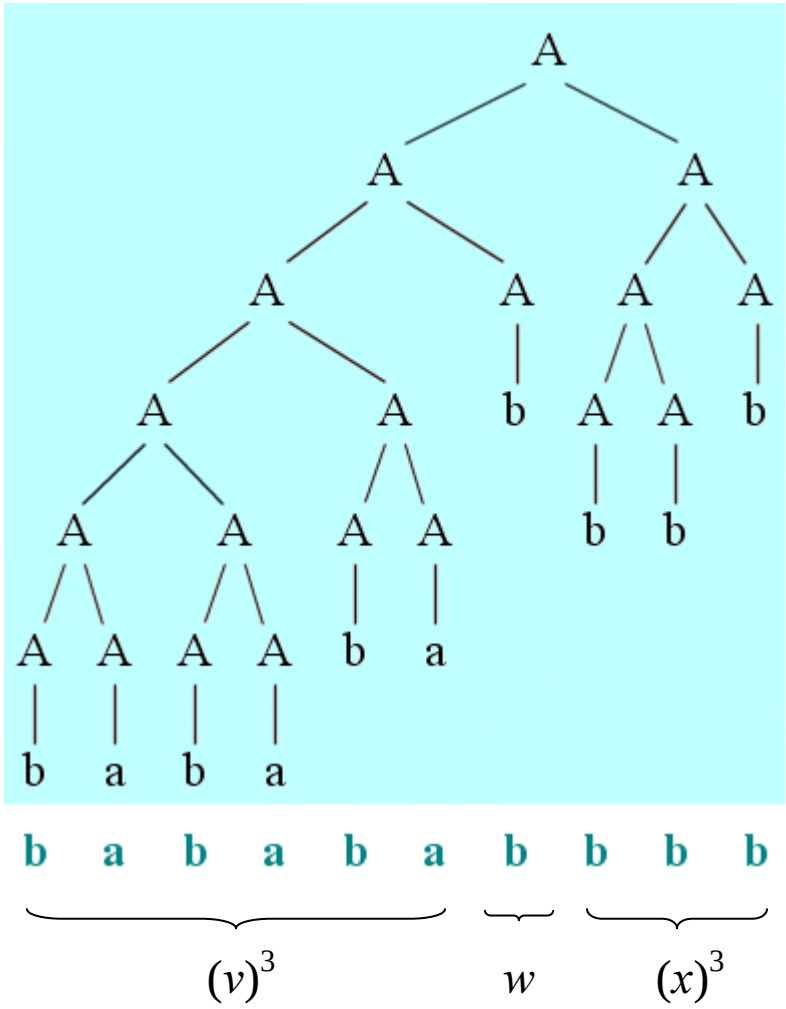
Verificamos que a "parte central" (a mais **longa**) é gerada por:

$$A \Rightarrow^* ba\ A\ b \qquad = v\ w\ x$$

e que pode ser "bombeada" por forma a gerar:

$$A \Rightarrow^* (ba)^k\ b\ (b)^k \qquad = (v)^k\ w\ (x)^k$$

como por exemplo, para k = 3:



Ou seja, $S \Rightarrow^* baa\,(ba)^k\,b\,(b)^k\,aac = u\,(v)^k\,w\,(x)^k\,y \in L, \ \forall\, k \geq 0$

Por outro lado, como a Gramática está na Forma Normal de Chomsky, a Árvore de Derivação é sempre uma **Árvore Binária**, onde:

Teorema: Numa Árvore Binária com n níveis,

$$n \leq |\{\text{folhas}\}| \leq 2^{n-1} .$$

{ ... Prove! }

Teorema: Numa Árvore Binária com 2^m folhas, o número de nós visitados pelo Caminho mais longo da raiz até às folhas, é:

$m + 1 \leq |C_{Max}| \leq 2^m$.

Para m = 3:

(árvore binária total)

(árvore binária degenerada)

No caso das Árvores de Derivação de uma Gramática, na Formal Normal de Chomsky, os resultados anteriores referem apenas às Produções da forma $A \rightarrow BC$ (só geradoras de variáveis).

Demonstração (do Lema da Bombagem para LIC's) **:**

Seja L é uma Linguagem Independente do Contexto e G uma Gramática, na Formal Normal de Chomsky, tal que $L(G) = L \setminus \{\varepsilon\}$
e seja m o número de variáveis de G.

Uma palavra $z \in L$ de comprimento $|z| \geq \mathbf{n = 2^m}$ tem uma Árvore de Derivação com, pelo menos, m +1 níveis (cujos nós são variáveis).

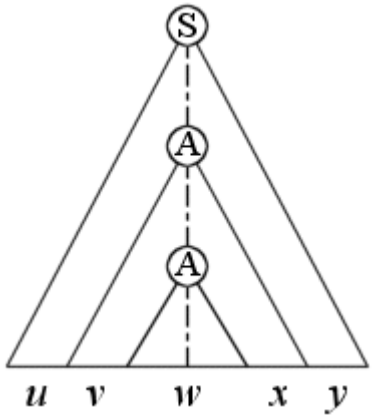
Como só existem m variáveis, alguma variável deverá aparecer repetida.

Assim (pelo menos no caminho mais longo) existem (pelo menos) **duas** ocorrências de uma variável A. Ou seja,

$S \Rightarrow^* u\,A\,y$

$A \Rightarrow^* v\,A\,x = vwx$

$A \Rightarrow^* w$



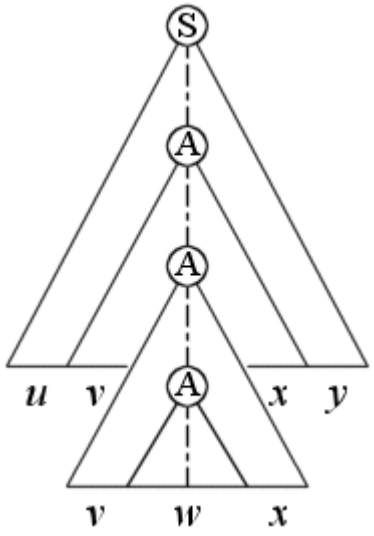
O facto de $S \Rightarrow^* uAy$ (o primeiro A) garante-nos que, $|vwx| \leq n = 2^m$

e por $A \Rightarrow^* vAx$ (como a Gramática não tem produções-ε), $|vx| > 0$.

Então, por $A \Rightarrow^* v\,A\,x$ podemos ir derivando:

$A \Rightarrow^* v\,A\,x$

$\Rightarrow^* v\,v\,A\,x\,x$



para qualquer k,

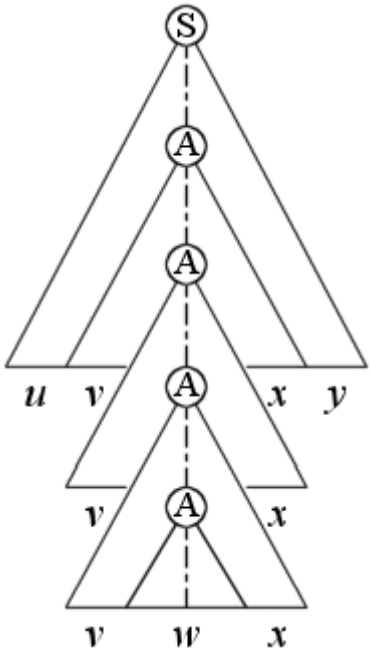
$$A \Rightarrow^* v\,A\,x$$

$$\Rightarrow^* v\,v\,A\,x\,x$$

$$\Rightarrow^* v\,v\,v\,A\,x\,x\,x$$

$$\ldots$$

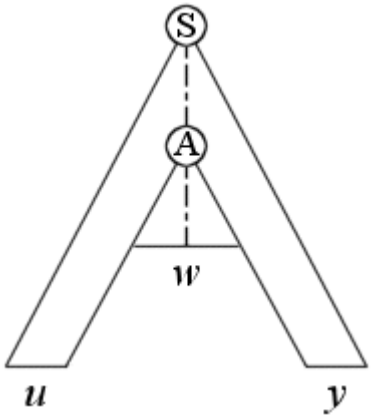
$$\Rightarrow^* v^k\,A\,x^k$$



e mesmo para k = 0,

pois $S \Rightarrow^* u\,A\,y$

$A \Rightarrow^* w$.



□

Nota:

- Comparar esta demonstração com a das Linguagens Regulares;

Exemplo: Provar que **não** é uma Linguagem Independente do Contexto,

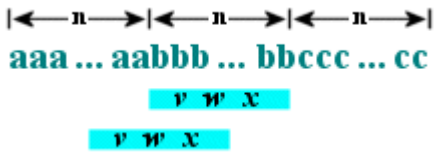
$L = \{ a^m b^m c^m \mid m \geq 0 \}$.

Suponhamos que L é Independente do Contexto.
Então, pelo Lema da Bombagem, existe um $n \in \mathbb{N}$ para o qual todas as palavras $z \in L$ com $|w| \geq n$ satisfazem as condições.

Tomemos $z = a^n b^n c^n$.
Como z é suficientemente longa, existem $u, v, w, x, y \in \Sigma^*$, tais que: $z = uvwxy$ com

$$|vwx| \leq n$$
$$|vx| > 0$$
$$\forall\, k \geq 0,\ uv^k wx^k y \in L$$

Ora, para que se verifique $|vwx| \leq n$,



a palavra vwx não pode conter as 3 letras.

Suponhamos, por exemplo, que vwx não contém c's:

Para que se verifique $|vx| > 0$, pelo menos um deles é não nulo, nenhum contendo c's.
Então, como $uvwxy$ tem o mesmo número de a's, b's e c's, a palavra $uv^0wx^0y = uwy$ consistiria numa redução do número de a's e b's, resultando num excesso de c's.

Assim, como para $k = 0$, $uv^k wx^k y \notin L$
L não é uma Linguagem Independente do Contexo.

Exercícios: Mostrar que **não** são Linguagens Independentes do Contexto:

$L = \{ a^m b^n c^m d^n \mid m, n \geq 0 \}$

$L = \{ a^n \mid \underline{n} \text{ é um Quadrado Perfeito} \}$

$L = \{ a^n \mid \underline{n} \text{ é um Número Primo} \}$

$L = \{ a^{n!} \mid n \geq 0 \}$

$L = \{ a^m b^n \mid m = n^2 \}$

$L = \{ ww \mid w \in \{a, b\}^* \}$

Operações sobre Linguagens Independentes do Contexto

Propriedades do Fecho:

Sejam L_1 a LIC gerada pela Gramática $G_1 = (V_1, T_1, P_1, S_1)$
e L_2 a LIC gerada pela Gramática $G_2 = (V_2, T_2, P_2, S_2)$.

- **União:**

  $L_1 \cup L_2$ é uma Linguagem Independente do Contexto.

  Basta juntar a produção $S \rightarrow S_1 \mid S_2$ e assegurar que $V_1 \cap V_2 = \emptyset$ (se necessário, mudar o nome de algumas variáveis).

  A Gramática pretendida é G = (V, T, P, S) com:

  $$V = V_1 \cup V_2 \cup S$$
  $$T = T_1 \cup T_2$$
  $$P = P_1 \cup P_2 \cup \{ S \rightarrow S_1 \mid S_2 \}$$

- **Concatenação:**

  L_1L_2 é uma Linguagem Independente do Contexto.

  Basta juntar a produção $S \rightarrow S_1S_2$ (com $V_1 \cap V_2 = \emptyset$).

- **Fecho de Kleene:**

  L_1^* é uma Linguagem Independente do Contexto.

  Basta juntar a produção $S \rightarrow S_1S \mid \varepsilon$.

- **Homomorfismo:**

  A Classe das Linguagens Independentes do Contexto também é fechada para Homomorfismos.

  Seja G = (V, T, P, S) a Gramática geradora de L e
  $h : T \longrightarrow \Sigma^*$ um Homomorfismo.

  h(G) é uma Gramática com símbolos terminais em Σ e cuja produções se obtêm a partir de P, substituindo cada terminal $a \in T$ por h(a).

  Não é difícil mostrar que h(G) gera h(L).

  Exemplo: $S \rightarrow 0S0 \mid 1S1 \mid \varepsilon$ com $h(0) = aba$, $h(1) = bb$

  h(G) tem as produções:

  $S \rightarrow abaSaba \mid bbSbb \mid \varepsilon$

Todas estas propriedades são casos particulares de um resultado mais geral ...

O Teorema da Substituição:

A partir de uma LIC L, substituindo cada símbolo terminal a por uma LIC L_a, obtemos uma linguagem da forma,

$\{ w_1 w_2 \ldots w_n \mid \exists\, a_1 a_2 \ldots a_n \in L \text{ e } w_i \in L_{a_i} \}$

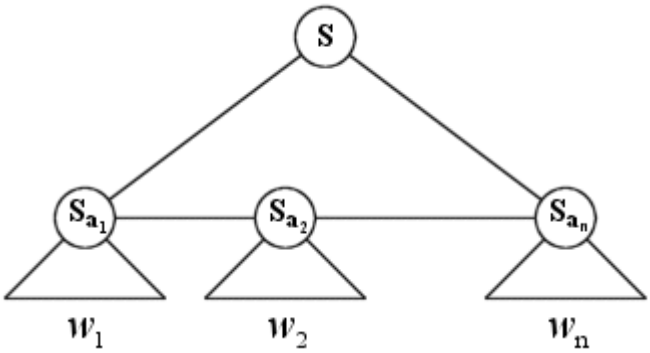
que é também Independente do Contexto.

(Esquema da) Demonstração:

Seja $G = (V, T, P, S)$ a gramática geradora de L
e $G_a = (V_a, T_a, P_a, S_a)$ cada gramática geradora de cada L_a.

Se os conjuntos de variáveis (V e V_a's) forem disjuntos e, se em cada Produção de G, substituirmos cada ocorrência de cada símbolo $a \in T$ pelo respectivo S_a, obtemos uma Linguagem Independente do Contexto.

As palavras da Linguagem resultante são da forma $w_1 w_2 \ldots w_n$,
em substituição de cada palavra $a_1 a_2 \ldots a_n \in L$.



Exemplo:

$L = \{0^n 1^n \mid n \geq 1\}$, gerada pela gramática de produções: $S \rightarrow 0S1 \mid 01$.

Substituímos **0** pela Linguagem $\{a^n b^m \mid m \leq n\}$, gerada por:

$S \rightarrow aSb \mid A$
$A \rightarrow aA \mid ab$

e substituímos **1** pela Linguagem {ab, abc}, gerada por:

$S \rightarrow abA$
$A \rightarrow c \mid \varepsilon$

Nas gramáticas de substituição, a variável S passa a chamar-se S_0 e S_1, respectivamente. Na última gramática, a variável A passa a ser B.
Ou seja,

$$S \rightarrow 0S1 \mid 01$$

$$S_0 \rightarrow aS_0b \mid A$$
$$A \rightarrow aA \mid ab$$

$$S_1 \rightarrow abB$$
$$B \rightarrow c \mid \varepsilon$$

Substituindo, na primeira Gramática, 0 por S_0 and 1 por S_1, obtemos:

$$S \rightarrow S_0\, S\, S_1 \mid S_0\, S_1$$
$$S_0 \rightarrow a\, S_0\, b \mid A$$
$$A \rightarrow a\, A \mid a\, b$$
$$S_1 \rightarrow a\, b\, B$$
$$B \rightarrow c \mid \varepsilon$$

Exercício: Por aplicação directa do Teorema da Substituição, mostre que a Classe das Linguagens Independentes do Contexto é fechada para a União, a Concatenação, o Fecho de Kleene e os Homomorfismos.

- **Intersecção:**

  $\mathbf{L_1 \cap L_2}$ **não é** necessariamente uma Linguagem Independente do Contexto.

Por exemplo, sejam,

$L_1 = \{a^m b^m c^n \mid m, n \geq 0\}$ definida por, $S \rightarrow XC$, $X \rightarrow aXb \mid \varepsilon$, $C \rightarrow cC \mid \varepsilon$

$L_2 = \{a^m b^n c^n \mid m, n \geq 0\}$ definida por, $S \rightarrow AY$, $A \rightarrow aA \mid \varepsilon$, $Y \rightarrow bYc \mid \varepsilon$

A Linguagem $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ não é Independente do Contexto.

- **Complementar:**

  $\overline{L_1}$ **não é** necessariamente uma Linguagem Independente do Contexto.

  Como $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ e porque a classe das LIC's é fechada para a União, se fosse fechada para o Complementar também o seria para a Intersecção.

  E como o Complementar é um caso particular da Diferença ...

- **Diferença:**

  $L_1 \setminus L_2$ **não é** necessariamente uma Linguagem Independente do Contexto.

- **Inversão (no sentido de Reversão):**

  L_1^R é uma Linguagem Independente do Contexto.

Basta inverter a ordem dos símbolos do lado direito de cada Produção.

Exemplo: $L = \{a^m b^m c^n \mid m, n \geq 0\}$ definida por

$$S \rightarrow XC$$
$$X \rightarrow aXb \mid \varepsilon$$
$$C \rightarrow cC \mid \varepsilon$$

$L^R = \{c^n b^m a^m \mid m, n \geq 0\}$ é definida por

$$S \rightarrow CX$$
$$X \rightarrow bXa \mid \varepsilon$$
$$C \rightarrow Cc \mid \varepsilon$$

- ## Homomorfismo Inverso:

$h^{-1}(L_1)$ é uma Linguagem Independente do Contexto.

Recordemos que, dado um Homomorfismo de palavras, $h : \Sigma^* \longrightarrow \Gamma^*$ e uma Linguagem $L \subseteq \Gamma^*$, se define:

$$h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$$

o conjunto das palavras cuja imagem homomórfica pertence a L.

Prova-se que, se L for uma Linguagem Independente do Contexto, então $h^{-1}(L)$ também é Independente do Contexto.