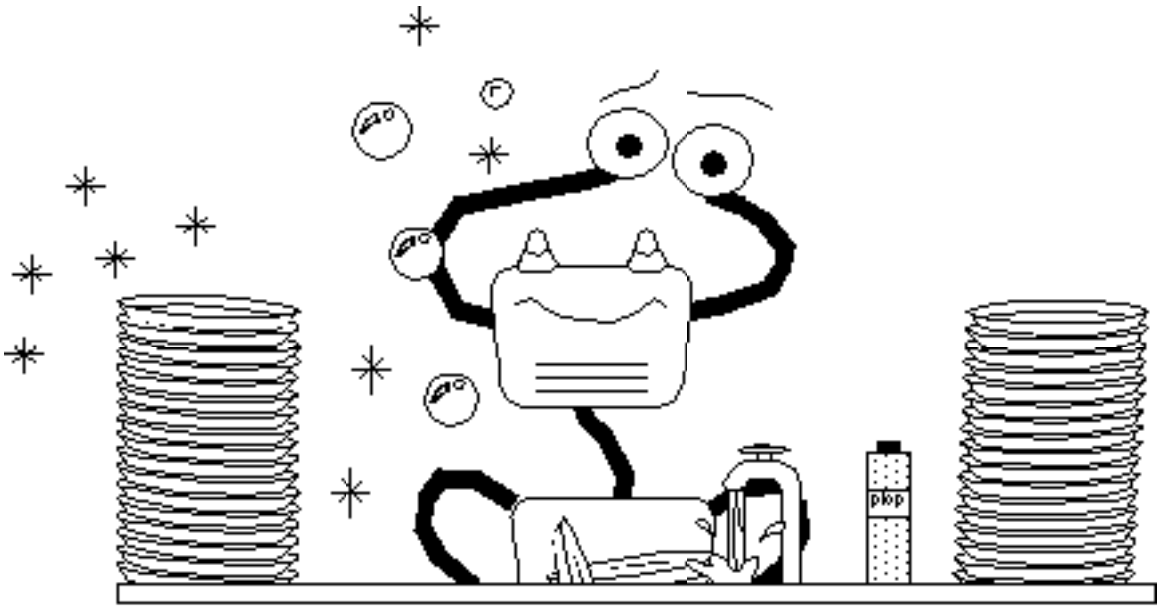


3 Linguagens Livres do Contexto

3 Linguagens Livres do Contexto

♦ Classe das LLC ou Tipo 2

- contém propriamente a Classe das LR

♦ LLC são importantes pois

- universo mais amplo de linguagens que as LR
- trata questões típicas de linguagens de programação
- exemplo
  - parênteses balanceados
  - construções bloco-estruturadas, etc
- algoritmos reconhecerdores e geradores
  - *relativamente* simples
  - *boa* eficiência
- exemplos de aplicações
  - analisadores sintáticos
  - tradutores de linguagens
  - processadores de texto em geral

♦ O estudo é desenvolvido a partir de

- formalismo axiomático ou gerador
  - *Gramática Livre do Contexto*
- formalismo operacional ou reconhecedor
  - *Autômato com Pilha*

♦ Gramática Livre do Contexto

- regras de produção são definidas de forma mais livre que na GR

♦ Autômato com Pilha

- a estrutura básica é análoga à do AF
- memória auxiliar tipo pilha
  - leitura
  - *gravação*
- facilidade de não-determinismo

3.1 Gramática Livre do Contexto

♦ *Gramática Livre do Contexto* (GLC)

- é uma gramática $G = (V, T, P, S)$
- qq regra de produção é da forma

$$A \rightarrow \alpha$$

  - A é uma variável de V
  - α uma palavra de $(V \cup T)^*$

♦ *Ling. Livre do Contexto* (LLC)

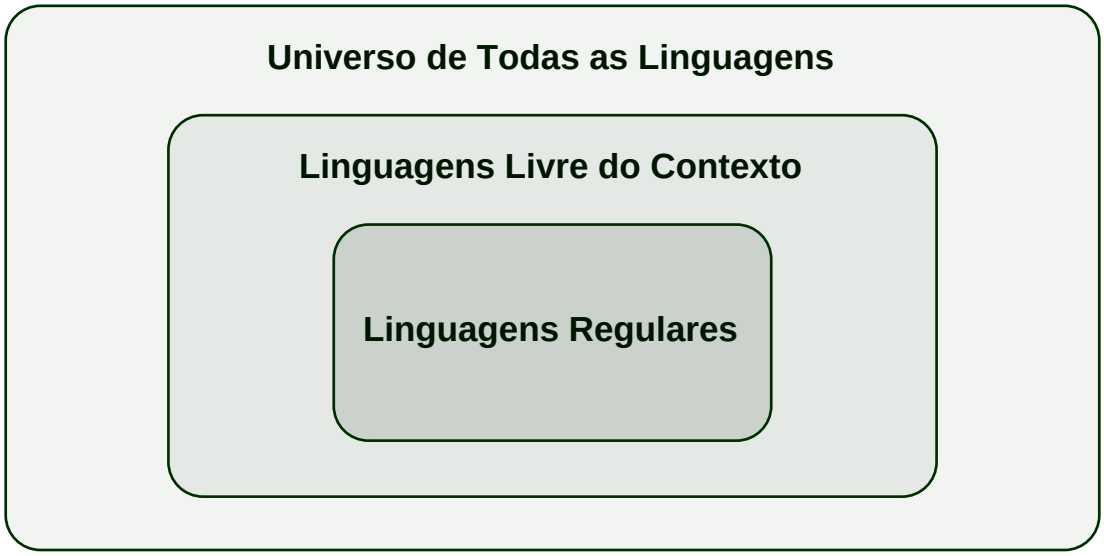
- gerada por uma gramática livre do contexto

♦ Nome "Livre do Contexto"

- mais geral classe de linguagens com produções
  - da forma $A \rightarrow \alpha$
- em uma derivação
  - A deriva α
  - "livre" de qq análise dos símbolos que antecedem ou sucedem A ("contexto") na palavra que está sendo derivada

♦ Regular × Livre do Contexto

- claramente, toda LR é LLC



♦ *Exemplo*: $L = \{a^n b^n \mid n \geq 0\}$

- $G = (\{S\}, \{a, b\}, P, S)$
- $P = \{S \rightarrow aSb \mid S \rightarrow \varepsilon\}$.

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aa\varepsilon bb = aabb$

♦ $L = \{a^n b^n \mid n \geq 0\}$

- Importante: analogia entre $a^n b^n$ e linguagens
  - bloco-estruturadas: $BEGIN^n END^n$
  - parênteses balanceados: $(^n)^n$

♦ *Exemplo*: Expressões Aritméticas

- parênteses balanceados
- operadores $*$, $+$
- operando representado por x
- GLC
  - $G = (\{E\}, \{+, *, [,], x\}, P, E)$, onde
  - $P = \{E \rightarrow E+E \mid E*E \mid [E] \mid x\}$
- $[x+x]*x$ pode ser gerada pela derivação

$$E \Rightarrow E*E \Rightarrow [E]*E \Rightarrow$$

$$[E+E]*E \Rightarrow [x+E]*E \Rightarrow$$

$$[x+x]*E \Rightarrow [x+x]*x$$

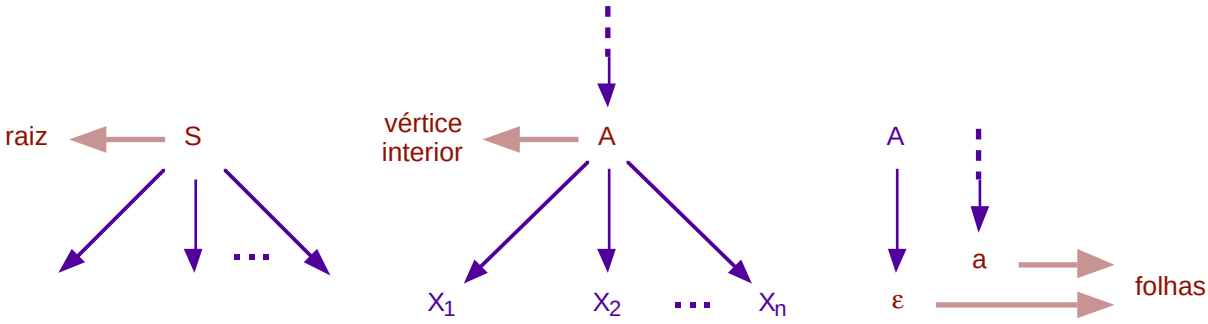
- é possível gerar a mesma expressão
  - com outra seqüência de derivação?
- quantas seqüências distintas são possíveis?

3.2 Árvore de Derivação

- **Freqüentemente é conveniente**
  - representar a derivação de palavras
  - na forma de árvore
- **Exemplos**
  - compiladores
  - processadores de textos
- **representação na forma de árvore**
  - *raiz*
    - símbolo inicial
  - *folhas*
    - símbolos terminais

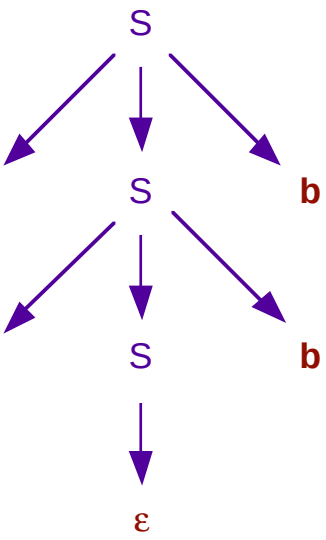
♦ *Árvore de Derivação*

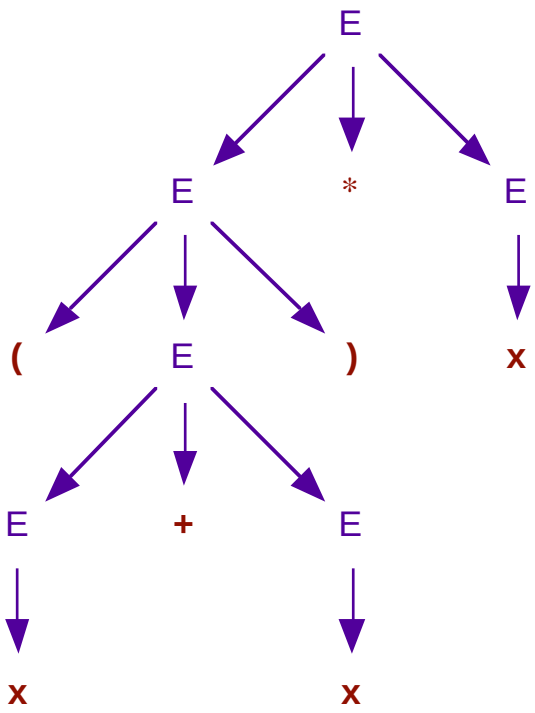
- *raiz*
  - símbolo inicial da gramática
- *vértices interiores*
  - variáveis
- se A é um vértice interior e $X_1, X_2, ..., X_n$ são os filhos de A, então
  - $A \rightarrow X_1X_2...X_n$ é uma produção da gramática
  - os vértices $X_1, X_2, ..., X_n$ estão ordenados da esquerda para a direita
- *folha*
  - terminal ou o símbolo vazio
  - se vazio, então é o único filho de seu pai



♦ *Exemplo*:

- aabb
- (x+x)*x

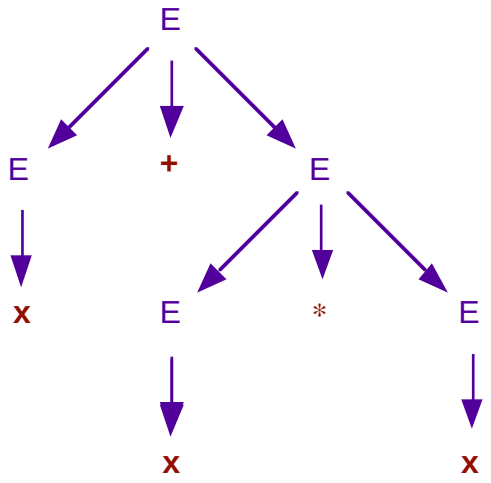




♦ Uma única árvore de derivação

- pode representar derivações *distintas*
- de uma mesma palavra

♦ *Exemplo*: x+x*x pode ser gerada por



- E ⇒ E+E ⇒ X+E ⇒ x+E*E ⇒ x+x*E ⇒ x+x*x
- E ⇒ E+E ⇒ E+E*E ⇒ E+E*x ⇒ E+x*x ⇒ x+x*x
- E ⇒ E+E ⇒ E+E*E ⇒ X+E*E ⇒ x+x*E ⇒ x+x*x
- etc...

♦ *Derivação mais à Esquerda (Direita)*

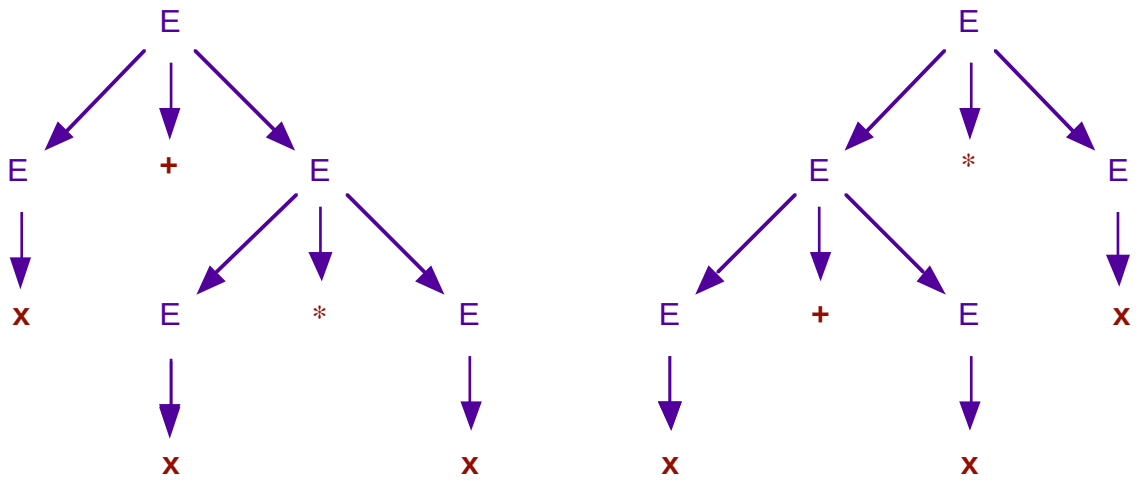
* a seqüência de produção aplicada sempre à variável mais à esquerda (direita)

3.3 Ambigüidade

♦ *Gramática Ambigüa*

- Gramática Livre do Contexto
- se existe uma palavra que possua
  - duas ou mais árvores de derivação

♦ *Exemplo*: x+x*x



♦ x+x*x possui mais de uma derivação à esquerda (direita)

- derivação mais à esquerda

$$E \Rightarrow E+E \Rightarrow x+E \Rightarrow x+E*E \Rightarrow x+x*E \Rightarrow x+x*x$$
$$E \Rightarrow E*E \Rightarrow E+E*E \Rightarrow x+E*E \Rightarrow x+x*E \Rightarrow x+x*x$$

- derivação mais à direita

$$E \Rightarrow E+E \Rightarrow E+E*E \Rightarrow E+E*x \Rightarrow E+x*x \Rightarrow x+x*x$$
$$E \Rightarrow E*E \Rightarrow E*x \Rightarrow E+E*x \Rightarrow E+x*x \Rightarrow x+x*x$$

♦ Forma equivalente de definir ambigüidade

- a existência de uma palavra
- com duas ou mais derivações mais à esquerda (direita)

♦ Ambigüidade ???

- em muitas aplicações, é conveniente que a
  - gramática usada não seja ambígua
- nem sempre é possível eliminar ambigüidades
- é fácil definir linguagens para as quais
  - qq GLC é ambígua

♦ *Linguagem Inerentemente Ambigüa*

- LLC
- qq GLC que a define é ambígua

♦ *Exemplo*:

$$L = \{w \mid w = a^n b^n c^m d^m \text{ ou}$$

$$w = a^n b^m c^m d^n,\ n \geq 1,\ m \geq 1\}$$

3.4 Simplificação de GLC

- **Simplificações**
  - *não* reduzem o poder de expressão das GLC
- **Simplificações são importantes para**
  - construção e otimização de algoritmos
  - demonstração de teoremas
- **Simplificações**
  - exclusão de *símbolos inúteis*
    - variáveis ou terminais não-usados
    - para gerar palavras de terminais
  - exclusão de *produções vazias* da forma $A \rightarrow \varepsilon$
    - se ε pertence à linguagem,
    - é incluída uma produção vazia específica
  - exclusão de produções da forma $A \rightarrow B$
    - substituem uma variável por outra
    - não adicionam qq informação de geração de palavras.

Símbolos Inúteis

- **Símbolos inúteis**
  - variáveis ou terminais
  - não-usados na geração de palavras de terminais
- **Simplificação**
  - *exclui* as produções que fazem referência aos símbolos inúteis
  - análise das produções a partir
    - de terminais gerados e
    - do símbolo inicial
  - não é necessária qq modificação adicional nas produções da gramática
  - *exclui* os símbolos que não são referenciados em qq produção

♦ Algoritmo

- *Qualquer variável gera palavra de terminais*
  - gera um novo conjunto de variáveis
  - inicialmente, considera todas as variáveis que geram terminais diretamente (ex: $A \rightarrow a$)
  - a seguir, são adicionadas as variáveis que geram palavras de terminais indiretamente (ex: $B \rightarrow Ab$)
- *Qualquer símbolo é atingível a partir do símbolo inicial*
  - analisa as produções da gramática a partir do símbolo inicial
  - inicialmente, considera exclusivamente o símbolo inicial
  - após, as produções da gramática são aplicadas e os símbolos referenciados adicionados aos novos conjuntos

♦ *Algoritmo. Eliminação dos Símbolos Inúteis*

- GLC $G = (V, T, P, S)$
- *Etapa 1: garante que qualquer variável gera terminais*
- gramática resultante: $G_1 = (V_1, T, P_1, S)$
- construção de V_1

$V_1 = \varnothing$;
faça $V_1 = V_1 \cup \{A \mid A \rightarrow \alpha \in P \text{ e } \alpha \in (T \cup V_1)^*\}$
até cardinal de V_1 não aumente;

- construção de P_1
  - mesmas produções que P
  - excetuando-se as produções cujas variáveis não pertencem a V_1

- *Etapa 2: garante que qualquer símbolo é atingível a partir do símbolo inicial*

- gramática resultante: $G_2 = (V_2, T_2, P_2, S)$
- construção de V_2 e T_2

$T_2 = \varnothing$;
$V_2 = \{S\}$;
faça $V_2 = V_2 \cup \{A \mid X \rightarrow \alpha A \beta \in P_1, X \in V_2\}$;
$T_2 = T_2 \cup \{a \mid X \rightarrow \alpha a \beta \in P_1, X \in V_2\}$
até cardinais de V_2, T_2 não aumentem;

- construção de P_2
  - mesmas produções que P_1
  - excetuando-se as produções cujos símbolos não pertencem a V_2 ou T_2

- *se as etapas acima forem executados em ordem inversa (etapa 2 antes da 1)*
  - pode não gerar o resultado desejado
  - para demonstrar, é suficiente apresentar um contra-exemplo (exercício)

♦ *Exemplo*: Considere a GLC

- $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$, onde
  $P = \{S \rightarrow aAa \mid bBb, A \rightarrow a \mid S, C \rightarrow c\}$
- *Qualquer variável gera palavra de terminais*

iteração	variáveis
início	$\varnothing$
1	{A, C}
2	{A, C, S}
3	{A, C, S}

- produção $S \rightarrow bBb$
  - excluída
  - B não pertence ao novo conjunto de variáveis

- *Qualquer símbolo é atingido a partir de* S
- G = ({S, A, ~~B~~, C}, {a, b, c}, P, S), onde
  P = {S → aAa | ~~bBb~~, A → a | S, C → c}

iteração	variáveis	terminais
início	{S}	∅
1	{S, A}	{a}
2	{S, A}	{a}

- produção C → c
  - excluída
  - C e c não pertencem aos novos conjuntos de variáveis e terminais
- gramática resultante
  - G = ({S, A}, {a}, P, S), onde
  - S = {S → aAa, A → a | S}

Produções Vazias

♦ Exclusão de produções vazias

- da forma $A \rightarrow \varepsilon$
- pode determinar
  - modificações diversas
  - nas produções da gramática

♦ Algoritmo

- *Variáveis que constituem produções vazias*
  - $A \rightarrow \varepsilon$. variáveis que geram ε diretamente
  - $B \rightarrow A$. variáveis que geram ε indiretamente
- *Exclusão das produções vazias*
  - todas as produções não-vazias
  - a seguir, cada produção que possui uma variável que gera ε, determina uma produção adicional, sem esta variável
- *Inclusão de geração da palavra vazia*
  - se necessário

♦ *Algoritmo. Eliminação das Produções Vazias*

- GLC $G = (V, T, P, S)$
- *Etapa 1: Conjunto de variáveis que constituem produções vazias*.
  - V_ε : conjunto das variáveis que geram ε

$V_\varepsilon = \{A \mid A \rightarrow \varepsilon\}$;
faça $V_\varepsilon = V_\varepsilon \cup \{X \mid X \rightarrow X_1 \ldots X_n \in P$ tq
$X_1, \ldots, X_n \in V_\varepsilon\}$
até que o cardinal de V_ε não aumente;

- *Etapa 2: Conjunto de produções sem produções vazias*

- gramática resultante: $G_1 = (V, T, P_1, S)$
- construção de P_1

$P_1 = \{A \rightarrow \alpha \mid \alpha \neq \varepsilon \text{ e } \alpha \notin V_\varepsilon\}$;
faça para $A \rightarrow \alpha \in P_1$ e
$X \in V_\varepsilon$ tq $\alpha = \alpha_1 X \alpha_2$ e $\alpha_1\alpha_2 \neq \varepsilon$
faca $P_1 = P_1 \cup \{A \rightarrow \alpha_1\alpha_2\}$
até que o cardinal de P_1 não aumente;

- *Etapa 3: inclusão de geração da palavra vazia, se necessário*

- se a palavra vazia pertence à linguagem
- gramática resultante:

  $G_2 = (V, T, P_2, S)$ onde

  $P_2 = P_1 \cup \{S \rightarrow \varepsilon\}$

♦ *Exemplo*: Considere a GLC

- $G = (\{S, X, Y\}, \{a, b\}, P, S)$, onde
  $P = \{S \rightarrow aXa \mid bXb \mid \varepsilon, X \rightarrow a \mid b \mid Y, Y \rightarrow \varepsilon\}$

- *Conjunto de variáveis que geram ε*

iteração	V_ε
início	{S, Y}
1	{S, Y, X}
2	{S, Y, X}

- *Conjunto de produções sem produções vazias*

iteração	produções
inicial	$\{S \rightarrow aXa \mid bXb, X \rightarrow a \mid b\}$
1	$\{S \rightarrow aXa \mid bXb \mid aa \mid bb, X \rightarrow a \mid b\}$
2	$\{S \rightarrow aXa \mid bXb \mid aa \mid bb, X \rightarrow a \mid b\}$

- *Inclusão da geração da palavra vazia*
  - como ε pertence à linguagem
  - $S \rightarrow \varepsilon$ é incluída no conjunto de produções

- *gramática resultante* (note-se Y *é inútil*)
  - $G = (\{S, X, Y\}, \{a, b\}, P, S)$, onde
    $P = \{S \rightarrow aXa \mid bXb \mid aa \mid bb \mid \varepsilon, X \rightarrow a \mid b\}$

Produções da Forma $A \rightarrow B$

- **$A \rightarrow B$**
  - neste caso
    - A pode ser substituída por B
    - não adiciona informação alguma em termos de geração de palavras
  - adicionalmente, se $B \rightarrow \alpha$, então
    - $A \rightarrow B$ pode ser substituída por $A \rightarrow \alpha$
    - generalização desta idéia: algoritmo proposto
- **Algoritmo**
  - *Construção do fecho da cada variável*
    - variáveis que podem substituí-la transitivamente
    - ex.: se $A \rightarrow B$ e $B \rightarrow C$, então B e C pertencem ao fecho de A
  - *Exclusão das produções $A \rightarrow B$*
    - se α é atingível a partir de A através de seu fecho
      então substitui $A \rightarrow B$ por $A \rightarrow \alpha$

♦ *Algoritmo. Eliminação de Produções da Forma* $A \rightarrow B$

- GLC $G = (V, T, P, S)$
- *Construção do fecho da cada variável*

```
para  A ∈ V
faça  FECHO-A = {B | A ⇒+ B} usando exclusivamente
      produções da forma X → Y;
```

- *Exclusão das produções* $A \rightarrow B$
- gramática resultante: $G_1 = (V, T, P_1, S)$
- construção de P_1

```
P1 = {A → α | α ∉ V};
para  A ∈ V e B ∈ FECHO-A
faça  se    B → α ∈ P e α ∉ V
      então P1 = P1 ∪ {A → α};
```

♦ *Exemplo*: Considere a GLC

- G = ({S, X}, {a, b}, P, S), onde
  P = {S → aXa | bXb, X → a | b | S | ε}

- *Construção do fecho de cada variável*
  * FECHO-S = ∅
  * FECHO-X = {S}

- *Construção do conjunto de produções*

iteração	produções
inicial	{S → aXa | bXb, X → a | b | ε}
S	{S → aXa | bXb, X → a | b | ε}
X	{S → aXa | bXb, X → a | b | ε | aXa | bXb }

- *gramática resultante*
  * G = ({S, X}, {a, b}, P, S), onde
  * P = {S → aXa | bXb, X → a | b | ε | aXa | bXb}

Simplificações combinadas

- **Não é qualquer seqüência de simplificação de gramática que atinge os resultados desejados**
  - por quê?
- **Seqüência recomendada**
  - exclusão de produções vazias
  - exclusão de produções da forma $A \rightarrow B$
  - exclusão de símbolos inúteis

3.5 Formas Normais

♦ Formas Normais

- restrições rígidas na definição de produções
- *não* reduzem o poder de geração das GLC
- aplicações
  - desenvolvimento de algoritmos (com destaque para reconhecedores de linguagens)
  - demonstração de teoremas

♦ *Forma Normal de Chomsky*

- todas as produções são da forma
  - $A \rightarrow BC$ ou
  - $A \rightarrow a$

♦ *Forma Normal de Greibach*

- Todas as produções são da forma
  - $A \rightarrow a\alpha$.
  - α é palavra de variáveis

3.5.1 Forma Normal de Chomsky

♦ Algoritmo de transformação

- transforma uma GLC qq
- a linguagem não pode ter a palavra vazia
- etapas
  - *Simplificação da Gramática*
  - *Variáveis no lado direito das produções*
  - *Exatamente duas variáveis no lado direito das produções*

♦ Simplificação da Gramática

- produções vazias
  - $A \rightarrow \varepsilon$
  - podem ser excluídas pois a ling não possui ε
- produções da forma $A \rightarrow B$
  - um símbolo do lado direito: terminal
- símbolos inúteis
  - opcional

♦ Variáveis no lado direito das produções

- lado direito das produções de comprimento maior ou igual a dois
  - exclusivamente variáveis
- exclusão de um terminal a
  - substitui por uma variável intermediária C_a
  - inclui a produção $C_a \rightarrow a$

♦ Exatamente duas variáveis no lado direito das produções

- produções da forma $A \rightarrow B_1B_2...B_n$
  - é suficiente gerar $B_1B_2...B_n$ em diversas etapas
  - usando variáveis intermediárias

♦ *Algoritmo*

- GLC $G = (V, T, P, S)$
- *Etapa 1: Simplificação da Gramática*.
  - algoritmos de simplificação
  - resulta na gramática $G_1 = (V_1, T_1, P_1, S)$
- *Etapa 2: Variáveis no lado direito das produções*
  - gramática resultante: $G_2 = (V_2, T_1, P_2, S)$
  - construção de V_2 e P_2

$V_2 = V_1$;
$P_2 = P_1$;
para toda $A \rightarrow X_1X_2...X_n \in P_2$ tal que $n \geq 2$
faça se para $r \in \{1,...,n\}$, X_r é terminal
então (suponha $X_r = a$)
$V_2 = V_2 \cup \{C_a\}$;
substitui a por C_a em
$A \rightarrow X_1X_2...X_n \in P_2$;
$P_2 = P_2 \cup \{C_a \rightarrow a\}$;

- *Etapa 3: Exatamente duas variáveis no lado direito das produções*.
  - gramática resultante: $G_3 = (V_3, T_1, P_3, S)$
  - construção de V_3 e P_3

$V_3 = V_2$;
$P_3 = P_2$;
para toda $A \rightarrow B_1B_2...B_n \in P_3$ tal que $n \geq 3$
faça exclui $A \rightarrow B_1B_2...B_n$ de P_3;
$V_3 = V_3 \cup \{D_1,...,D_{n-2}\}$;
$P_3 = P_3 \cup \{A \rightarrow B_1D_1, D_1 \rightarrow B_2D_2, ...,$
$D_{n-3} \rightarrow B_{n-2}D_{n-2}, D_{n-2} \rightarrow B_{n-1}B_n\}$;

♦ *Exemplo*: GLC

- $G = (\{E\}, \{+, *, [,], x\}, P, E)$, onde
- $P = \{E \rightarrow E+E \mid E*E \mid [E] \mid x\}$
- *Simplificação*
  - já está simplificada.
- *Variáveis do lado direito*
  - $E \rightarrow x$ está OK
  - demais produções

    $E \rightarrow EC_+E \mid EC_*E \mid C_[EC_]$

    $C_+ \rightarrow +$

    $C_* \rightarrow *$

    $C_[\rightarrow [$

    $C_] \rightarrow]$

- *Exatamente duas variáveis do lado direito*
  - As produções

$E \rightarrow EC_{+}E \mid EC_{*}E \mid C_{[}EC_{]}$

  - necessitam ser substituídas como segue

$E \rightarrow ED_1 \mid ED_2 \mid C_{[}D_3$

$D_1 \rightarrow C_{+}E$

$D_2 \rightarrow C_{*}E$

$D_3 \rightarrow EC_{]}$

- gramática na FNC

$G = (\{E, C_{+}, C_{*}, C_{[}, C_{]}, D_1, D_2, D_3\},$
$\{+, *, [,], x\}, P_2, E)$, onde

$P_2 = \{E \rightarrow ED_1 \mid ED_2 \mid C_{[}D_3 \mid x,$
$D_1 \rightarrow C_{+}E, D_2 \rightarrow C_{*}E, D_3 \rightarrow EC_{]},$
$C_{+} \rightarrow +, C_{*} \rightarrow *, C_{[} \rightarrow [, C_{]} \rightarrow]\}$

3.5.2 Forma Normal de Greibach (FNG)

♦ Algoritmo de transformação

- transforma uma GLC qualquer
- a linguagem não pode ter a palavra vazia
- seis etapas

- *Simplificação da Gramática*
  - Análoga à FNC

- *Renomeação das variáveis em uma ordem crescente qualquer*
  - as variáveis da gramática são renomeadas em uma ordem crescente qualquer
  - exp: A_1, A_2, ..., A_n, onde n é o cardinal de V
  - diferentes critérios de renomeação:
    podem resultar em diferentes FNG
    entretanto, todas são equivalentes (geram a mesma linguagem)

- *Qq produção é da forma* $A_r \rightarrow A_s\alpha$, *onde* $r \leq s$
  - produções são modificadas garantindo que a primeira variável do lado direito é maior ou igual que a do lado esquerdo
  - $A_r \rightarrow A_s\alpha$ tais que $r > s$ são modificadas
  - substitui a variável A_s pelas suas produções

    $$A_s \rightarrow \beta_1 \mid \ldots \mid \beta_m$$
  - resultando em $A_r \rightarrow \beta_1\alpha \mid \ldots \mid \beta_m\alpha$
  - o conjunto de variáveis é finito:
    existe um limite para as produções crescentes
    geração de um terminal: $A_r \rightarrow a\alpha$
    geração de uma recursão: $A_r \rightarrow A_r\alpha$

- *Exclusão das recursões da forma* $A_r \rightarrow A_r\alpha$.
  - podem existir originalmente na gramática, ou serem geradas pela etapa anterior
  - eliminação da recursão à esquerda:
    introduz variáveis auxiliares
    inclui recursão à direita: $B_r \rightarrow \alpha B_r$

- *Um terminal no início do lado direito de cada produção*
  - todas as produções da forma $A_r \rightarrow A_s\alpha$ são tais que $r < s$
  - conseqüentemente, as produções da maior variável A_n só podem iniciar por terminais no lado direito
  - assim, se em $A_{n-1} \rightarrow A_n\alpha$ for substituído A_n pelas suas correspondentes produções (ex: $A_n \rightarrow a\beta$), o lado direito das produções de A_{n-1} também iniciará por terminal (ex: $A_{n-1} \rightarrow a\beta\alpha$)
  - a repetição do algoritmo para A_{n-2}, ...,A_1 resultará em produções exclusivamente da forma $A_r \rightarrow a\alpha$

- *Produções na forma $A \rightarrow a\alpha$ onde α é composta por variáveis*
  - análoga a correspondente etapa do algoritmo relativo à Forma Normal de Chomsky

♦ *Definição. Algoritmo*

- GLC $G = (V, T, P, S)$
- *Etapa 1: Simplificação da Gramática*
  - algoritmos de simplificação
  - símbolos inúteis: opcional
  - resulta na gramática $G_1 = (V_1, T_1, P_1, S)$
- *Etapa 2: Renomeação das variáveis em uma ordem crescente.*
  - gramática resultante: $G_2 = (V_2, T_1, P_2, S)$
  - suponha que o cardinal de V_1 é n
  - construção de V_2 e P_2

$V_2 = \{A_1, A_2, ..., A_n\}$ é V_1 renomeando os elementos

em uma ordem qualquer

P_2 é P_1 renomeando as variáveis nas produções

- *Etapa 3: Qq produção é da forma* $A_r \rightarrow A_s\alpha$, $r \leq s$
- *Etapa 4: Exclusão das recursões* $A_r \rightarrow A_r\alpha$
  - gramática resultante $G_3 = (V_3, T_1, P_3, S)$
  - suponha que o cardinal de V_2 é n

```
P3 = P2
para r variando de 1 até n
faça
                                        Etapa 3
     para s variando de 1 até r-1
     faça para toda Ar → Asα ∈ P3
          faça excluir Ar → Asα de P3;
               para toda As → β ∈ P3
               faça P3 = P3 ∪ {Ar → βα}
                                        Etapa 4
     para toda Ar → Arα ∈ P3
     faça excluir Ar → Arα de P3;
          V3 = V3 ∪ {Br};
          P3 = P3 ∪ {Br → α} ∪ {Br → αBr};
     para toda Ar → ϕ ∈ P4 tq
          ϕ não inicia por Ar e
          alguma Ar → Arα foi excluída
     faça P3 = P3 ∪ {Ar → ϕBr};
```

- *Etapa 5: Um terminal no início do lado direito de cada produção*
  - gramática resultante: $G_4 = (V_3, T_1, P_4, S)$
  - construção de P_4

```
P4 = P3;
para r variando de n-1 até 1 e
      toda Ar → Asα ∈ P4
faça excluir Ar → Asα de P4;
      para toda As → β de P4
      faça P4 = P4 ∪ {Ar → βα};
```

  - também é necessário garantir que as produções relativas às variáveis auxiliares B_r iniciam por um terminal do lado direito

```
para toda Br → Asβr
faça excluir Br → Asβr de P4;
      para toda As → aα
      faça P4 = P4 ∪ {Br → aαBr};
```

- *Etapa 6: Produções na forma $A \rightarrow a\alpha$ onde α é composta por variáveis*
  - análoga à correspondente etapa do algoritmo relativo à Forma Normal de Chomsky

♦ *Exemplo*: FNG da GLC

- $G = (\{S, A\}, \{a, b\}, P, S)$, onde
- $P = \{S \rightarrow AA \mid a, A \rightarrow SS \mid b\}$

- *Simplificação*
  - já está simplificada.

- *Renomeação das variáveis em ordem crescente*
  - S e A são renomeadas para A_1 e A_2
  - produções da gramática

$A_1 \rightarrow A_2A_2 \mid a$
$A_2 \rightarrow A_1A_1 \mid b$

- $A_r \rightarrow A_s\alpha$, com $r \leq s$
- *Recursões* $A_r \rightarrow A_r\alpha$
  - produção de A_2 *não* está na forma crescente

$A_1 \rightarrow A_2A_2 \mid a$
$A_2 \rightarrow A_2A_2A_1 \mid aA_1 \mid b$

  - recursão: $A_2 \rightarrow A_2A_2A_1$

$A_1 \rightarrow A_2A_2 \mid a$
$A_2 \rightarrow aA_1 \mid b \mid aA_1B \mid bB$
$B \rightarrow A_2A_1 \mid A_2A_1B$

- *Terminal no início de cada lado direito das produções*
  - o lado direito das produções da maior variável A_2 iniciam por terminal
  - substituição de A_2 no lado direito de $A_1 \rightarrow A_2A_2$ pelas correspondentes derivações

$$A_1 \rightarrow aA_1A_2 \mid bA_2 \mid aA_1BA_2 \mid bBA_2 \mid a$$
$$A_2 \rightarrow aA_1 \mid b \mid aA_1B \mid bB$$
$$B \rightarrow A_2A_1 \mid A_2A_1B$$

  - produções referentes à variável B

$$B \rightarrow aA_1A_1 \mid bA_1 \mid aA_1BA_1 \mid bBA_1 \mid aA_1A_1B \mid bA_1B \mid aA_1BA_1B \mid bBA_1B$$

- $A \rightarrow a\alpha$, *com* α *composta por variáveis*
  - as produções já se encontram nesta forma

- *Gramática na FNG resultante*

$G = (\{A_1, A_2, B\}, \{a, b\}, P, A_1)$, onde

$P = \{$

$A_1 \rightarrow aA_1A_2 \mid bA_2 \mid aA_1BA_2 \mid bBA_2 \mid a,$

$A_2 \rightarrow aA_1 \mid b \mid aA_1B \mid bB,$

$B \rightarrow aA_1A_1 \mid bA_1 \mid aA_1BA_1 \mid bBA_1 \mid aA_1A_1B \mid$
$bA_1B \mid aA_1BA_1B \mid bBA_1B\}$

3.6 Recursão à Esquerda

- **Recursão à esquerda**

$$A \Rightarrow^{+} A\alpha$$

  - em algumas situações, como algoritmos reconhecedores, é desejável que a gramática não seja recursiva à esquerda

- **A transformação de uma gramática qualquer em uma sem recursão à esquerda**

  - 4 primeiras etapas do algoritmo
    - Forma Normal de Greibach

  - *Etapa 1*. Simplificação da Gramática
  - *Etapa 2*. Renomeação das variáveis em ordem crescente
  - *Etapa 3*. Qq produção é da forma $A_r \rightarrow A_s\alpha$, $r < s$
  - *Etapa 4*. Exclusão das recursões $A_r \rightarrow A_r\alpha$