UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Disciplina: LINGUAGENS FORMAIS, AUTÔMATOS E COMPUTABILIDADE

Prof. Jefferson Morais
Email: jmorais@ufpa.br

Agenda

- Linguagens Livres de Contexto
- Gramáticas Livres de Contexto
- Árvore de derivação
- Ambiguidade
- Simplificação de GLC
- Formas Normais

Linguagens Livres de Contexto

Linguagens regulares

- Descrevemos linguagens através de autômatos finitos e expressões regulares: formas diferentes, mas equivalentes
- Sabemos que algumas linguagens não são possíveis de serem descritas por eles
  - Ex. $\{0^n1^n \mid n \geq 0\}$ (ver Lema do bombeamento)

Introdução

- Gramática Livre de Contexto: método mais poderoso de descrição de linguagens
- Estruturas recursivas
- LCC são simples e eficientes
- No estudo de linguagens humanas: GLC's podem captar aspectos importantes da relação entre substantivos, verbos e preposições

Introdução

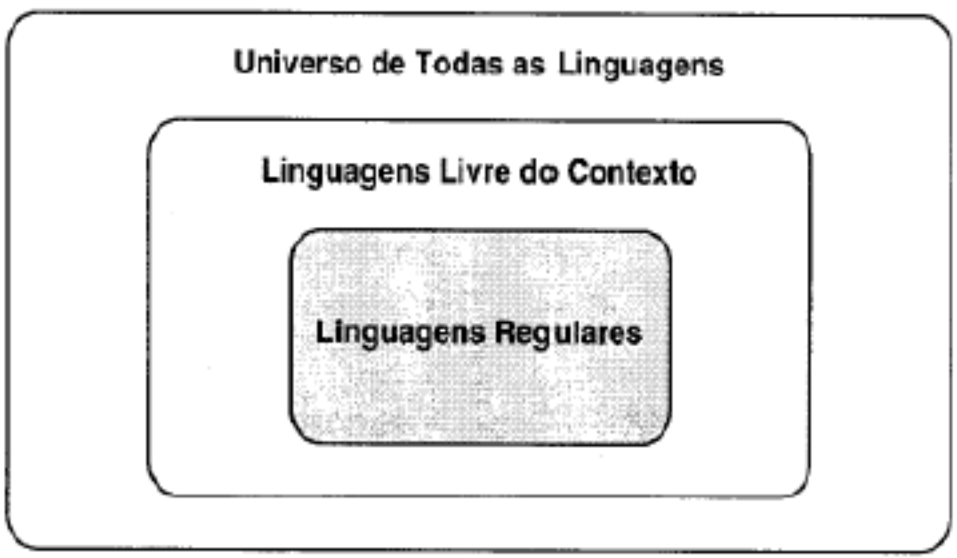
- Aplicação: especificação e compilação de linguagens de programação
  - Elaborar uma gramática para a linguagem: primeiro passo para um compilador e interpretador
  - ***Parser***: extrai o significado do programa para gerar o código compilado ou interpreta a execução
  - GLC's facilitam a produção do *parser*

Introdução

- Uma **Linguagem Livre de Contexto** (LLC) é gerada por uma Gramática Livre de Contexto (GLC)
- LLC's são definidas a partir de um formalismo gerador (gramática) e um reconhecedor (autômato), como:
  - GLC: onde as regras de produção são definidas mais livrementes que em GR
  - Autômato com Pilha (*Pushdown Automata*): estrutura básica e análoga ao AF, incluindo uma memória auxiliar do tipo pilha (pode ser lida ou gravada) e a facilidade do não-determinismo

Gramática Livre de Contexto

- Gramáticas consistem em uma coleção de regras de substituição (produção)
- GLC é uma 4-tupla G=(V,T,P,S) com a restrição de conter, no lado esquerdo da produção, exatamente uma variável (não terminal)



Gramática Livre de Contexto

- “Livre de Contexto” refere-se à possibilidade de representar a mais geral classe de linguagens cuja produção é da forma $A \longrightarrow a$
  - A variável “A” deriva “a” sem depender (livre) de qualquer análise de símbolos que precedem ou sucedem A (contexto) na palavra que está sendo derivada
- Uma LR é uma LLC

Gramática Livre de Contexto

- Exemplo: Duplo Balanceamento
  - L1=$\{a^n b^n \mid n \geq 0\}$
  - G1= ({S}, {a,b}, P={S→aSb, S→ε}, S) pode produzir aaabbb, por exemplo
- Esse exemplo é importante pois permite fazer analogia para problemas do tipo
  - Linguagens bloco-estruturadas $begin^n end^n$
  - Linguagens com parênteses balanceados $(^n)^n$

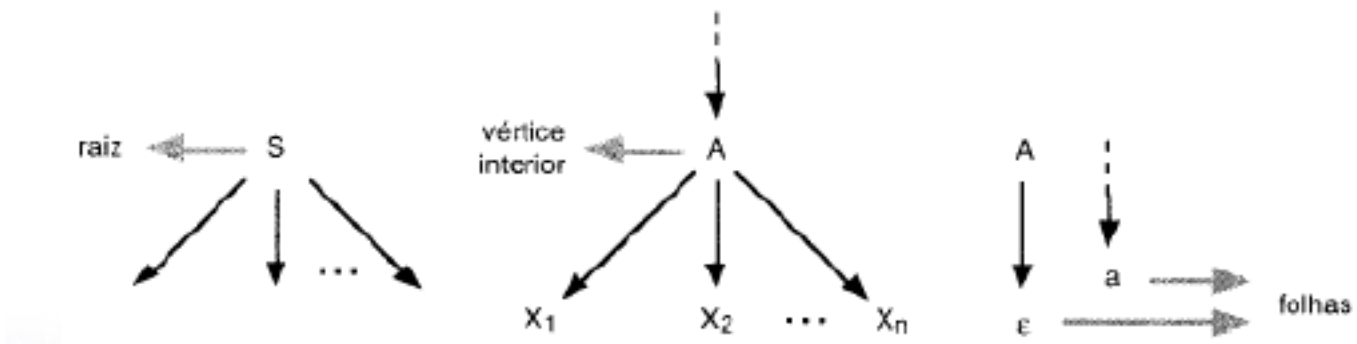
Gramáticas Livre de Contexto

- Ex: Expressões aritméticas
  - G2=({E}, {+,*,[,],x}, P2, E)
  - P2= {E → E+E | E*E | [E] | x }
  - G2 permite gerar [x+x]*x
  - Derivação
    - E=> E*E => [E]*E=>[E+E]*E=> [x+E]*E=>[x+x]*E=>[x+]*x

Árvore de derivação

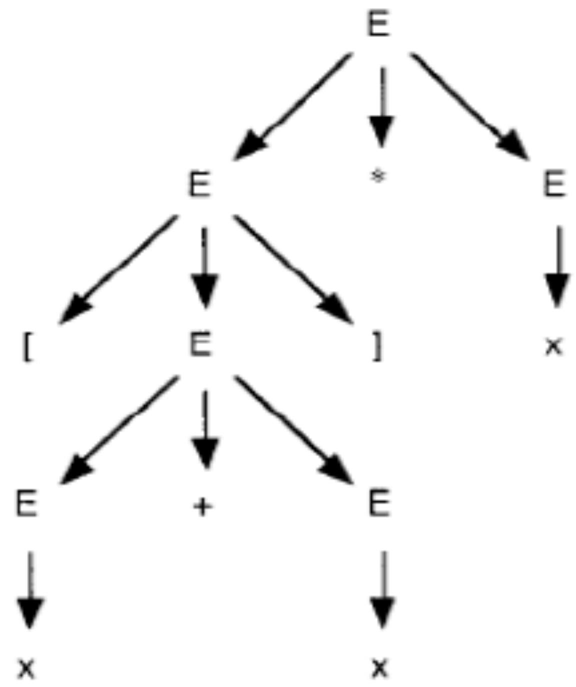
Árvore de derivação

- Derivação de palavras na forma de árvore
  - A raiz é o símbolo inicial da gramática
  - Os vértices (nós) interiores obrigatoriamente são variáveis
  - Um vértice (nó) folha é um símbolo terminal, ou o símbolo vazio. O símbolo vazio é o **único** filho



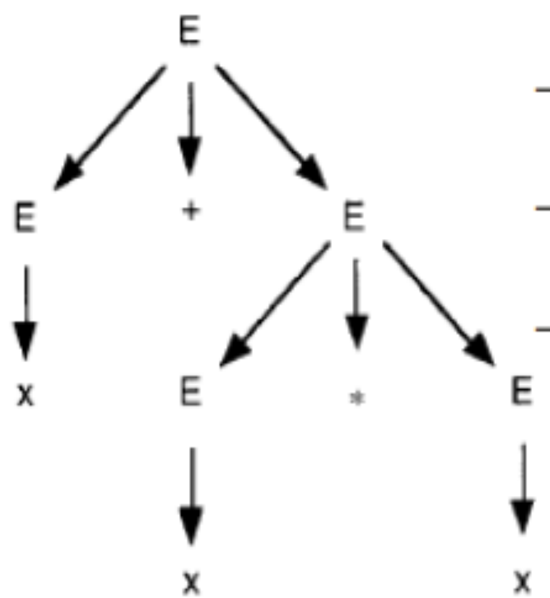
Árvore de derivação

- Exemplo: Voltando à expressão aritmética [x+x]*x



Árvore de derivação

- Exemplo: Árvore de derivação X Derivações palavra = x+x*x



- $E \rightarrow E+E$

  $\rightarrow x+E \rightarrow x+E*E \rightarrow x+x*E \rightarrow x+x*x$

  $\rightarrow E+E*E \rightarrow E+E*x \rightarrow E+x*x \rightarrow x+x*x$

  $\rightarrow E+E*E \rightarrow x+E*E \rightarrow x+x*E \rightarrow x+x*x$

  ...

Árvore de derivação

- Derivação mais à esquerda (direita)
  - É a sequência de produção aplicada sempre à variável mais à esquerda (direita)
  - Mais à esquerda
    - E=>E+E=>x+E=>x+E*E=>x+x*E=>x+x*x
  - Mais à direita
    - E=>E+E=>E+E*E=>E+E*x=>E+x*x=>x+x*x

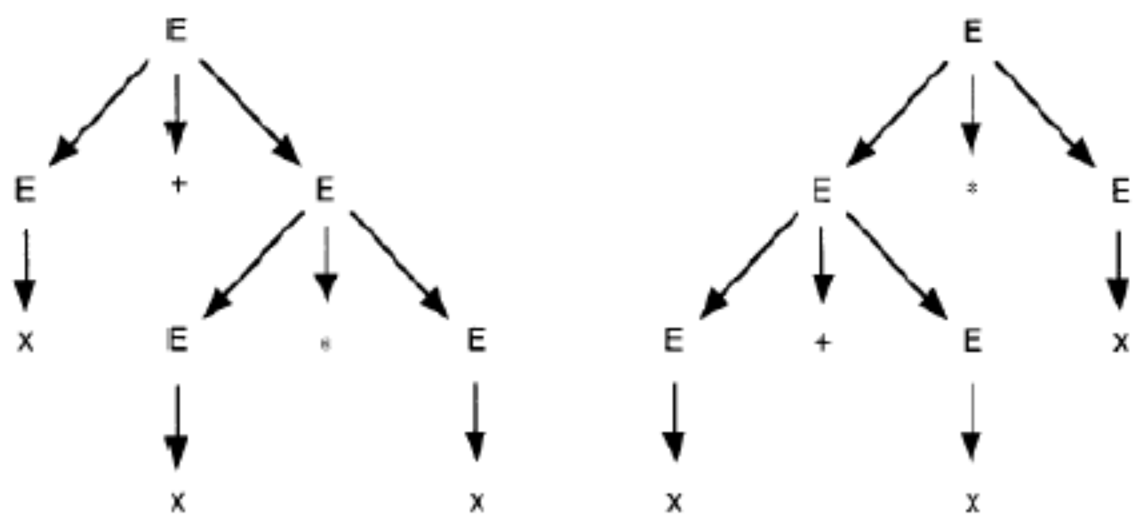
Ambiguidade

Ambiguidade

- Uma mesma palavra associada a duas ou mais árvores de derivação.
- Para algumas aplicações de desenvolvimento e otimização de alguns algoritmos de reconhecimento é conveniente que a gramática não seja ambígua.
- Nem sempre é possível eliminar a ambiguidade.
- É mais fácil definir linguagens para as quais qualquer GLC é ambígua.

Ambiguidade

- Definição: uma GLC é dita uma Gramática ambígua, se existe uma palavra que possua duas ou mais árvores de derivação.
  - Ex: a palavra x+x*x pode ser gerada por árvores distintas, portanto é ambígua.



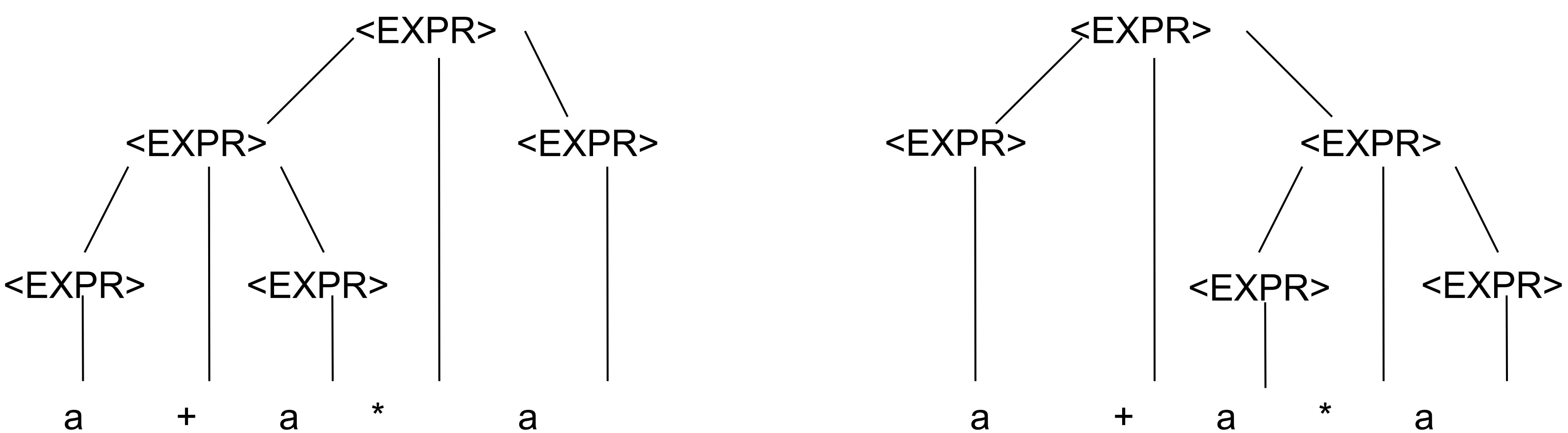
Ambiguidade

- Definição: Uma linguagem é uma Linguagem inerentemente ambígua se qualquer GLC que a define é ambígua
- Ex: $L3=\{w \mid w=a^nb^nc^md^m, n \geq 1, m \geq 1\}$

  $L4=\{w \mid w=a^nb^mc^md^n, n \geq 1, m \geq 1\}$

Exercícios

- Dado G = ({<EXPR>,<TERM>,<FACTOR>}, {a,+,*,(,)}, R, <EXPR>}, onde = {<EXPR> → <EXPR>+<EXPR> | <EXPR>*<EXPR> | (<EXPR>) | a} responda:
  - Para a palavra a+a*a G é ambígua? Prove.
  - Sim.



Simplificação de GLCs

Simplificação de GLC

- Tem por objetivo tornar a gramática mais simples ou de prepará-las para posteriores aplicações.
- É importante notar que, qualquer que seja a transformação efetuada, a linguagem gerada deverá ser sempre a mesma.

Simplificação de GLC

- Não reduzem o poder de expressão das GLC

- São importantes para:
  - Construção e otimização de algoritmos
  - Demonstrações de teoremas

Simplificando GLC

- São 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 qualquer informação de geração de palavra

Eliminação de Símbolos Inúteis

- Símbolos Inúteis : um símbolo (terminal ou não-terminal) é inútil se ele não aparece na derivação de nenhuma sentença.
- Podendo ser:
  - **Estéril**: se não gera nenhuma sequência de terminais pertencente a uma sentença
  - **Inalcançável**:se não aparece em nenhuma forma sentencial da gramática.

Determinação do conjunto de símbolos férteis

- 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:$B \rightarrow Ab$)

Determinação do conjunto de símbolos férteis

- Pode ser efetuada através do seguinte algoritmo:
  - Construir o conjunto $N0 = \varnothing$ e fazer $i = 1$
  - Repetir

    $Ni = Ni\text{-}1 \cup \{ A \mid A \rightarrow \alpha \in P \text{ e } \alpha \in (Ni\text{-}1 \cup T)^* \}$

    $i = i + 1$
  - até que $Ni = Ni\text{-}1$
  - Ni é o conjunto de símbolos férteis.
- Se o símbolo inicial não fizer parte do conjunto de símbolos férteis, a linguagem gerada pela gramática é vazia.

Exemplo

- Retirar os símbolos estéreis da gramática: G = ({S,A,B,C,D}, {a,b,c,d}, P, S)
  - P: S → a A

    A → a | b B

    B → b | d D

    C → cC | c

    D → d D

Exemplo

- Solução
    - N0 = ∅
    - N1 = {A,B,C}
    - N2 = {S,A,B,C}
    - N3 = {S,A,B,C} = N2
- Conjunto de símbolos férteis: {S,A,B,C}
- Gramática simplificada:
  - G’ = ({S,A,B,C}, {a,b,c}, P’, S)
  - P’:
    - S → a A
    - A → a | b B
    - B → b
    - C → cC | c

Determinação do conjunto de símbolos alcançáveis

- 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

Determinação do conjunto de símbolo alcançáveis

- Pode ser efetuada através do seguinte algoritmo:
  - Construir o conjunto $V0 = \{S\}$ (S = símbolo inicial) e fazer $i = 1$
  - Repetir
    - $Vi = \{ X | \text{existe algum } A \rightarrow \alpha X \beta \text{ e } A \in Vi\text{-}1 \text{ e } \alpha, \beta \in (N \cup T)^* \} \cup V\, i\text{-}1$
    - $i = i + 1$
  - até que $Vi = Vi\text{-}1$
  - Vi é o conjunto de símbolos alcançáveis.

Determinação do conjunto de símbolo alcançáveis - Exemplo

- Simplificar a gramática G' do exemplo anterior, retirando os símbolos inalcançáveis.
- Solução:
    - V0 = {S}
    - V1 = {S, a, A}
    - V2 = {S, a, A, b, B}
    - V3 = {S, a, A, b, B} = V2
  - Conjunto de símbolos alcançáveis: {S, a, A, b, B}
  - Gramática simplificada:
  - G' = ({S,A,B}, {a,b}, P'', S)
  - P'': $S \rightarrow a\ A$

    $A \rightarrow a \mid b\ B$

    $B \rightarrow b$

Transformação de uma GLC qualquer para uma GLC **ε**-Livre

- Variáveis que constituem produções vazias
  - A → ε. variáveis que geram ε diretamente
  - B → A. variáveis que geram ε indiretamente

Transformação de uma GLC qualquer para uma GLC **ε**-Livre

- Esta transformação sempre é possível e pode ser efetuada pelo seguinte algoritmo:
  - Reunir em um conjunto os não-terminais que derivam direta ou indiretamente a sentença vazia: $Ne = \{A \mid A \in N$ e $A \overset{+}{\to} \varepsilon\}$
  - Construir o conjunto de regras P' como segue:
    - incluir em P' todas as regras de P, com exceção daquelas da forma $A \to \varepsilon$
    - para cada ocorrência de um símbolo Ne do lado direito de alguma regra de P, incluir em P' mais uma regra, substituindo este símbolo por ε. Isto é, para regra de P do tipo $A \to \alpha B \beta$, $B \in Ne$ e $\alpha, \beta \in V^*$ incluir em P' a regra $A \to \alpha\ \beta$

Transformação de uma GLC qualquer para uma GLC **ε**-Livre

- Se $S \in Ne$, adicionar a P' as regras $S' \rightarrow S$ e $S' \rightarrow \varepsilon$, sendo que N' ficará igual a $N \cup S'$. Caso contrário trocar os nomes de S por S' e N por N'.

- A nova gramática será definida por: $G' = (N', T, P', S')$

Exemplo 1

- Transformar as GLC abaixo, definidas pelo respectivo
- conjunto de regras de produção P, para GLC **ε**-Livres.
  - G = ({S, D, C}, {b,c,d,e}, P, S)
  - P: S → b D C e
    D → d D | **ε**
    C → c C | **ε**
  - Solução:
    - Ne = {D, C}
    - P ‘: S → b D C e | b C e | b D e | b e
      D → d D | d
      C → c C | c

Exemplo 2

- Transformar as GLC abaixo, definidas pelo respectivo conjunto de regras de produção P, para GLC **ε**-Livres.
  - G = ({S}, {a}, P, S)
    - P: S → a S | **ε**
    - Solução:

      Ne = {S}

      P ‘:S‘ → S | **ε**

      S → a S | a

Remoção de Produções Simples

- Produções simples são produções da forma $A \rightarrow B$ onde A e $B \in N$. Onde:
  - A pode ser substituída por B
  - não adiciona informação alguma em termos de geração de palavras

Remoção de Produções Simples

- Podem ser removidas de uma GLC através do seguinte algoritmo:
  - Transformar a GLC em uma GLC **ε**-livre, se necessário
  - Para todo não-terminal de N, construir um conjunto com os não-terminais que ele pode derivar, em um ou mais passos. Isto é, para todo A ∈ N, construir NA = { B | A *→ B }
  - Construir P' como segue:
    - se B **→α** ∈ P e não é uma produção simples, adicione a P' as produções: A **→** **α** para todo A | B ∈ NA
  - A GLC equivalente, sem produções simples, será definida por: G' = (N, T, P', S)

Exemplo 1

- Transformar as GLC abaixo em gramáticas equivalentes que não apresentem produções simples.
  - G = ({S, A}, {a,b}, P, S)
    - P: $S \rightarrow b\,S \mid A$

      $A \rightarrow a\,A \mid a$
    - Solução:
      - Ns = {A}
      - NA = { }
      - P ‘: $S \rightarrow b\,S \mid a\,A \mid a$

        $A \rightarrow a\,A \mid a$

Exemplo 2

- Transformar as GLC abaixo em gramáticas equivalentes que não apresentem produções simples.
  - G = ({S, A, B}, {a,b,c}, P, S)
    - P: S → a S b | A

      A → a A | B

      B → b B c | b c
    - Solução:

      Ns = {A, B}

      NA = {B}

      NB = {}

      P ‘: S → a S b | a A | b B c | b c

      A → a A | b B c | b c

      B → b B c | b c

Simplificações combinadas

- Considerando as simplificações de gramáticas LC, nem todas as combinações de simplificação atingem o resultado desejado. Recomenda-se a seguinte sequência de simplificação:
  1. Exclusão de produções vazias
  2. Exclusão de produções simples
  3. Exclusão de símbolos inúteis

Fatoração de GLC

Fatoração de GLC

- Uma GLC está fatorada se ela é determinística, isto é, não possui produções cujo lado direito inicie com o mesmo conjunto de símbolos ou com símbolos que gerem sequências que iniciem com o mesmo conjunto de símbolos.
- Por exemplo, a gramática fatorada não deverá apresentar as seguintes regras:
  - A → a B | a C
  - pois as duas iniciam com o mesmo terminal a.
- Outro exemplo de gramática não-fatorada é o seguinte:
  - S → A | B
  - A → a c
  - B → a b

Fatoração de GLC

- Para fatorar uma GLC devemos alterar as produções envolvidas no não-determinismo da seguinte maneira:
  - as produções que apresentam não-determinismo direto, da forma
  - $A \rightarrow \alpha\ \beta \mid \alpha\ \delta$
  - serão substituídas por
    - $A \rightarrow \alpha\ A'$
    - $A' \rightarrow \beta \mid \delta$
  - sendo A' um novo não-terminal
- O não-determinismo indireto é retirado fazendo, nas regras de produção, as derivações necessárias para torná-lo um determinismo direto, resolvido posteriormente como no item anterior.

Fatoração de GLC

- Exemplo:
- Fatorar as GLC abaixo
  - G = ({S, A, B}, {a,b}, P, S)
    - P: S → a A | a B
    - A → a A | a
    - B → b
  - Solução:
    - P ‘: S → a S’
    - S’ → A | B
    - A → a A’
    - A’ → A | ε
    - B → b

Fatoração de GLC

- G = ({S, A}, {a,b}, P, S)
- P: S → A b | a b | b a A

  A → a a b | b
- Solução:
  - P ‘: S → a a b b | b b | a b | b a A

    A → a a b | b
  - P’’: S → a S’ | b S’’

    S’ → a b b | b

    S’’ → b | a A

    A → a a b | b

Fatoração de GLC

- Fatoração é importante pois na implementação de um compilador, o mesmo deve seguir uma gramática que não apresente não-determinismo pois, caso contrário, no processo de reconhecimento haveria “retornos” (backtracking) que acabam reduzindo a eficiência do algoritmo.

Eliminar Recursão à Esquerda

- Uma gramática G = (N, T, P, S) tem recursão à esquerda se existe $A \in N$ tal que
  - $A \xrightarrow{+} A\,\alpha$, $\alpha \in (N \cup T)^*$
- Uma gramática G = (N, T, P, S) tem recursão à direita se existe $A \in N$ tal que
  - $A \xrightarrow{+} \alpha\,A$, $\alpha \in (N \cup T)^*$
- A recursão é dita direta se a derivação acima for em um passo, isto é:
  - G tem recursão direta à esquerda se existe produção $A \rightarrow A\,\alpha \in P$
  - G tem recursão direta à direita se existe produção $A \rightarrow \alpha A \in P$

Eliminar a Recursão à Esquerda

- A importância de elminar a recursividade à esquerda é que alguns tipos de compiladores podem executar o processo de reconhecimento como chamadas de rotinas (procedimentos ou funções). Assim, uma gramática recursiva à esquerda, tal como por exemplo $A \rightarrow Aa|a$, acaba gerando um laço infinito $A \Rightarrow Aa \Rightarrow Aaa \Rightarrow Aaaa \Rightarrow$ ... e o processo de reconhecimento não finaliza nunca.

Eliminar a Recursão à Esquerda

Algoritmo:

1. recursões diretas: substituir cada regra $A \rightarrow A\alpha_1|A\alpha_2|\cdots|A\alpha_n|\beta_1|\beta_2|\cdots|\beta_m$, onde nenhum β_i começa por A, por:

$$\begin{array}{lcl} A & \rightarrow & \beta_1 A'|\beta_2 A'|\cdots|\beta_m A' \\ A' & \rightarrow & \alpha_1 A'|\alpha_2 A'|\cdots|\alpha_n A'|\varepsilon \end{array}$$

Eliminar Recursão à esquerda

- Para Recursões indiretas:
  - Transformar produções com recursão indireta em recursão direta (fazer as substituições necessárias)

Eliminar Recursão à esquerda

- Exemplo: G = (N, T, P, S)
  - P: S → A a
  - A → S b | c A | a
  - Solução:
  - P': S → A a

    A → A a b | c A | a

  - P'': S → A a

    A → c A A' | a A'

    A' → a b A' | ε

Formas normais

Formas normais

- Estabelecem restrições rígidas na definição das produções, sem reduzir o poder de geração das GLC.
- Usadas no desenvolvimento de algoritmos(reconhecedores de linguagem) e na prova de teoremas.
  - FN de Chomsky, onde as produções são da forma: $A \rightarrow BC$ ou $A \rightarrow a$
  - FN de Greibach, onde as produções são da forma: $A \rightarrow a\alpha$ (onde α é uma palavra de variáveis)

Formais normais de Chomsky

- Qualquer LLC é gerada por uma GLC na forma normal de Chomsky.
- A conversão segue 3 etapas
  1. Simplificação da gramática;
  2. Transformação do lado direito das produções de comprimento maior ou igual a dois;
  3. Transformação do lado direito das produções de comprimento maior ou igual a três, em produções com exatamente duas variáveis.

Formais normais de Chomsky

- Etapa 1
  - Excluir produções vazias
  - Exclui produções do tipo A → B (se o lado direito da produção tiver só um símbolo, ele deve ser terminal); e
  - Exclui opcionalmente símbolos inúteis
  - Use os algoritmos de simplificação mostrados anteriormente!

Formais normais de Chomsky

- Etapa 2
  - Garante que o lado direito das produções de comprimento maior ou igual a 2 seja composto exclusivamente por variáveis.
  - Exclui um terminal substituindo-o por uma variável intermediária:
    - $A \rightarrow aB$, torna-se:
    - $A \rightarrow TB$, $T \rightarrow a$

Formais normais de Chomsky

- Etapa 3
  - Garante que o lado direito das produções de comprimento maior do que 1 seja composto exclusivamente por duas variáveis.
    - $A \to BCD$
    - $A \to BF$
    - $F \to CD$

Forma normal de Chomsky

- Exemplo: Transformação GLC para FNC. G = ({E},{+,*,[,],x},P={E → E+E | E*E | [E] | x},E)
- E1: tem algo para excluir (vazias, inúteis ou A → B)?

Forma normal de Chomsky

- Exemplo: Transformação GLC para FNC. G = ({E},{+,*,[,],x},P={E → E+E | E*E | [E] | x},E)
- E1: tem algo para excluir (vazias, inúteis ou A → B)? Não
- E2:
  - E → E+E torna-se E→EME, onde M → +
  - E → E*E torna-se E→EVE, onde V → *
  - E → [E] torna-se E→C^1EC^2, onde C^1→[e C^2→].
  - E → x (não mexe).

Forma normal de Chomsky

- Exemplo: Transformação GLC para FNC. G = ({E},{+,*,[,],x},P={E → E+E | E*E | [E] | x},E)
- E3: E → EME | EVE | C^1EC^2 torna-se:
  - E → ED^1 | ED^2 | C^1D^3
  - D^1 → ME
  - D^2 → VE
  - D^3 → EC^2

Forma normal de Chomsky

- Exemplo: Transformação GLC para FNC. G = ({E},{+,*,[,],x},P={E → E+E | E*E | [E] | x},E)
- A gramática resultante é:
  - G = ({E,M,V,C^1,C^2,D^1,D^2,D^3},{+,*,[,],x},P,E}
    - P={ E → ED^1 | ED^2 | C^1D^3 | x,
      D^1 → ME, D^2 → VE, D^3 → EC^2,
      M → +,V → *,C^1→[, C^2→]}

Forma normal de Chomsky

- Colocar a GLC na FN de Chomsky
- $G = (\{S, A, B\}, \{a, b\}, P, S)$
- $P: S \rightarrow A \mid A\ B\ A$

  $A \rightarrow a\ A \mid a$

  $B \rightarrow b\ B \mid b$

Solução

a) P': $S \rightarrow a\ A \mid a \mid A\ B\ A$

$A \rightarrow a\ A \mid a$

$B \rightarrow b\ B \mid b$

b) P'': $S \rightarrow A_a\ A \mid a \mid A\ B\ A$

$A \rightarrow A_a\ A \mid a$

$B \rightarrow A_b\ B \mid b$

$A_a \rightarrow a$

$A_b \rightarrow b$

c) P''': $S \rightarrow A_a\ A \mid a \mid A\ B'$

$B' \rightarrow B\ A$

$A \rightarrow A_a\ A \mid a$

$B \rightarrow A_b\ B \mid b$

$A_a \rightarrow a$

$A_b \rightarrow b$

Forma normal Greibach

- Uma GLC está na Forma Normal de Greibach se ela é **ε**-livre e apresenta todas as produções na forma:
  - A → a **α**
  - onde A ∈ N, a ∈ T e **α** ∈ N*.

Forma normal Greibach

- Para achar a gramática equivalente a G = (N, T, P, S), na GNF, deve-se seguir os seguintes passos:
  1. achar G' = (N', T, P', S) tal que L(G') = L(G) e que G' esteja na CNF (opcional)
  2. ordenar os não-terminais de G' em uma ordem quaisquer – por exemplo: N' = { A1, A2, ..., Am }
  3. modificar as regras de P' de modo a que, se Ai → Aj γ é uma regra de P', então j > i
  4. a gramática obtida do passo anterior, G'', apresentará todas as regras de Am com o lado direito iniciando por um terminal; através de substituições sucessivas dos primeiros termos das regras Ai anteriores, coloca-se estas também nessa forma
  5. se no item 3 tiverem sido incluídos novos não-terminais Bi (para retirar recursões à esquerda), fazer também para as regras correspondentes a estes, as devidas substituições dos primeiros termos (que serão sempre terminais ou Ai)
  6. a gramática final, G''', está na GNF

Exemplo: Obtenha a FNG.

$$\begin{aligned} P: S &\rightarrow AS|a \\ A &\rightarrow SA|b \end{aligned}$$

Solução:

1. G já está na FNC;
2. Renomear os não-terminais: $S = A_1$ e $A = A_2$;

$$\begin{aligned} P: A_1 &\rightarrow A_2A_1|a \\ A_2 &\rightarrow A_1A_2|b \end{aligned}$$

Formal normal de Greibach

3. se $A_i \to A_j\gamma \in P'$, então $j > i$. A única regra a modificar é:

$$A_2 \to A_1 A_2 \,.$$

Substituindo A_1 nesta regra:

$$A_2 \to A_2 A_1 A_2 | a A_2 | b \,.$$

Retirando a recursão à esquerda de

$$\overbrace{A_2}^{A} \rightarrow \overbrace{A_2}^{A} \overbrace{A_1 A_2}^{\alpha_1} \,|\, \overbrace{aA_2}^{\beta_1} \,|\, \overbrace{b}^{\beta_2},$$

obteremos:

$$\begin{aligned}
P'' : A_1 &\rightarrow A_2 A_1 | a \\
\overbrace{A_2}^{A} &\rightarrow \overbrace{aA_2}^{\beta_1} \overbrace{B_2}^{A'} \,|\, \overbrace{b}^{\beta_2} \overbrace{B_2}^{A'} \\
\overbrace{B_2}^{A'} &\rightarrow \overbrace{A_1 A_2}^{\alpha_1} \overbrace{B_2}^{A'} | \varepsilon
\end{aligned}$$

Formal normal de Greibach

Retirando o ε por substituições:

$$
\begin{aligned}
P'' : A_1 &\rightarrow A_2A_1|a \\
A_2 &\rightarrow aA_2B_2|bB_2|aA_2|b \\
B_2 &\rightarrow A_1A_2B_2|A_1A_2
\end{aligned}
$$

4. Fazendo as substituições finais para tornar todos os lados direitos na forma $A \longrightarrow a\alpha$, $A \in N$, $a \in T$ e $\alpha \in N^*$:

$$
\begin{aligned}
P''' : A_1 &\longrightarrow aA_2B_2A_1|bB_2A_1|aA_2A_1|bA_1|a \\
A_2 &\longrightarrow aA_2B_2|bB_2|aA_2|b \\
B_2 &\longrightarrow aA_2B_2A_1A_2B_2|bB_2A_1A_2B_2| \\
&\quad\;\; aA_2A_1A_2B_2|bA_1A_2B_2|aA_2B_2| \\
&\quad\;\; aA_2B_2A_1A_2|bB_2A_1A_2|aA_2A_1A_2| \\
&\quad\;\; bA_1A_2|aA_2
\end{aligned}
$$