



Gramáticas Livre de Contexto

Autômatos, Linguagens e Computação

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Objetivos de Aprendizagem

- Definir formalmente uma Gramática Livre de Contexto.
- Demonstrar o processo de **derivação** (\Rightarrow^*) de uma sentença, aplicando a sequência de regras a partir do símbolo inicial.
- Entender o conceito de Hierarquia de Linguagens.
- Explicar a importância central das GLCs no projeto de um **analisador sintático (parser)** de Linguagens de Programação e
- Entender por que a utilização de **Formas Normais** (como a FNC) é necessária para eliminar ambiguidades e garantir que o parsing seja eficiente.

Gramática

- Formalismo para **definir** linguagens formais.
- Uma gramática mostra como gerar todas as palavras de uma linguagem.
- Utilizam recursão.

Hierarquia de Chomsky

Tipo	Linguagem	Gramática	Máquina de Aceitação
3	Regular	Regular	Autômato Finito
2	Livre de Contexto	Livre de Contexto	Autômato com Pilha
1	Sensível ao Contexto	Sensível ao Contexto	Autômato Linearmente Limitado
0	Rec. Enumerável	Irrestrita	Máquina de Turing

O Poder das GLCs

- **Limitação das LR (Linguagens Regulares)**
 - AFs possuem "memória limitada" (finita).
 - Não conseguem contar ou expressar estruturas de balanceamento, como $\{a^n b^n \mid n \geq 0\}$.
 - O Lema do Bombeamento para LRs comprova que $L = \{a^i b^i \mid i \geq 0\}$ não é regular.
- **LLCs Aumentam a Expressão**
 - GLCs permitem descrever estruturas recursivas平衡adas.
 - Exemplo clássico: Balanço de parênteses $(^n)^n$ ou blocos *begin*ⁿ*end*ⁿ.
 - **Aplicação central:** GLCs são fundamentais no projeto de um **analisador sintático (parser)**, especificando a gramática formal de Linguagens de Programação (LPs).

Definição Formal

- **GLC (Gramática Livre de Contexto)** é uma quádrupla $G = (V, \Sigma, P, S)$.

Símbolo	Componente	Descrição
V	Conjunto de Variáveis	Símbolos não-terminais.
Σ	Alfabeto de Terminais	Símbolos da linguagem gerada.
P	Conjunto de Regras	Regras de produção (substituição). Formato: $\alpha \rightarrow \beta$
S	Símbolo Inicial	Variável de partida para a derivação ($S \in V$).

- **Restrição Principal:** V e Σ devem ser disjuntos ($\Sigma \cap V = \emptyset$).
- **Variável inicial:** Ao lado esquerdo da primeira regra, a menos que seja especificado o contrário.

Exemplo 1

Considere a gramática G_1 com as regras de produção abaixo:

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- $V = \{A, B\}$

- $S = A$

- $P = \{A \rightarrow 0A1 \mid B, B \rightarrow \#\}$

- $\Sigma = \{0, 1, \#\}$

Podemos gerar a cadeia $000\#\underline{1}11$ pela sequência de produções (uma **derivação**):

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#\underline{1}11$$

Gramática da Linguagem C

<https://www.lysator.liu.se/c/ANSI-C-grammar-y.html>

Convenções e Componentes (V e Σ)

- **Variáveis (V)**

- Representam categorias sintáticas (e.g., Sentença, Expressão, Verbo).
- Geralmente representadas por letras maiúsculas (A, B, C...).
- Símbolos **não-terminais**
- **Cada variável representa uma sub-linguagem.**

- **Terminais (Σ)**

- São os símbolos que formam as palavras finais (o alfabeto da linguagem).
- Análogos ao alfabeto de entrada dos Autômatos Finitos.
- Geralmente representados por letras minúsculas, números ou símbolos especiais (e.g., $a, b, 0, 1, +$).

Componentes e a Natureza "Livre de Contexto"

- **Regras de Produção (P)**
 - São regras de substituição no formato: $\mu \rightarrow \nu$.
 - Em GLCs, a estrutura é restrita:
 - **Lado Esquerdo (μ)**: Deve ser uma única variável ($A \in V$).
 - **Lado Direito (ν)**: Pode ser qualquer cadeia de variáveis e/ou terminais ($\nu \in (V \cup \Sigma)^*$).
- **Caractere "Livre de Contexto"**
 - A regra $A \rightarrow w$ significa que o não-terminal A pode ser substituído por w .
 - Essa substituição é **sempre válida**, independentemente do contexto (u e v) em que A aparece (i.e., $uAv \Rightarrow uwv$).
- **Símbolo Inicial (S)**
 - É a variável que inicia o processo de derivação de qualquer sentença da linguagem.

Aplicação de Regras e Notação

- **Derivação (\Rightarrow):** A aplicação de uma regra de substituição.
- **Exemplo de Regra:** $A \rightarrow 0A1$.
- **Mecânica de Substituição:**
 - Se na cadeia atual aparecer A , ele pode ser substituído por $0A1$.
 - Seja $\alpha = \alpha_{esq}A\alpha_{dir}$ e $A \rightarrow \beta$ uma regra, então $\alpha \Rightarrow \alpha_{esq}\beta\alpha_{dir}$.
- **Tipos de Geração (Derivação):**
 - $\alpha \Rightarrow \beta$: β é gerada **diretamente** a partir de α (1 passo).
 - $\alpha \Rightarrow^* \beta$: β é **derivável** a partir de α (aplicando 0 ou mais regras).
 - $\alpha \Rightarrow^+ \beta$: β é derivável a partir de α (aplicando 1 ou mais regras).

A distinção entre \rightarrow (regra/produção) e \Rightarrow (aplicação da regra/derivação) é fundamental.

Exemplo 2

Seja $G = (\{a, b\}, \{S, A\}, P, S)$ com o conjunto de regras
 $P = \{S \rightarrow aAS \mid a, A \rightarrow SbA \mid SS \mid ba\}.$

$$S \rightarrow aAS$$

$$S \rightarrow$$

$$S \rightarrow a$$

$$A \rightarrow SbA$$

$$A \rightarrow SS$$

$$A \rightarrow ba$$

Podemos gerar a cadeia $aabbaa$, pela derivação:

$$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$$

Derivação e Formas Sentenciais

- **Forma Sentencial:** Qualquer cadeia $\alpha \in (V \cup \Sigma)^*$ que é derivável a partir do símbolo inicial ($S \Rightarrow^* \alpha$). Pode conter variáveis e terminais.
- **Sentença:** Uma forma sentencial que contém **somente terminais** ($\alpha \in \Sigma^*$). É uma palavra válida na linguagem $L(G)$.
- **Derivação Mais à Esquerda:** Uma ordem de substituição que, a cada etapa, substitui a variável não-terminal mais à esquerda. Usada por analisadores sintáticos descendentes (Top-Down).

Exemplo de Derivação em Sequência

- **GLC Clássica para Palíndromos:** $G : S \rightarrow aSa|bSb|a|b|\epsilon.$
- **Derivação da Sentença 'baab':**

Etapa	Aplicação	Forma Sentencial	Variável Substituída
1	$S \rightarrow bSb$	bSb	S
2	$S \rightarrow aSa$	$baSab$	S
3	$S \rightarrow \epsilon$	$baab$	S

$$S \Rightarrow bSb \Rightarrow baSab \Rightarrow baab$$

- **Conclusão:** $baab$ é uma sentença de $L(G)$ porque $S \Rightarrow^* baab$.

Árvore de Derivação (Parse Tree)

- **Definição:** Uma estrutura de árvore que representa graficamente as substituições aplicadas na derivação.
- **Componentes:**
 - **Raiz:** O Símbolo Inicial (S).
 - **Nós Internos:** Variáveis ($A \in V$).
 - **Folhas:** Terminais ($a \in \Sigma$) ou ϵ .
 - **Ordem:** Os filhos de um nó interno A correspondem ao lado direito da produção $A \rightarrow X_1X_2 \dots X_n$, lidos da esquerda para a direita.
- **Exemplo:** Árvore para 'baab' (usando a derivação anterior) na lousa.

Linguagem da Gramática

- **Definição Formal:** A linguagem gerada por uma GLC G é o conjunto de todas as sentenças (cadeias de terminais) que podem ser derivadas a partir do símbolo inicial S :

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

- **LLCs vs. LRs:** Todas as Linguagens Regulares (Tipo 3) são também Linguagens Livres de Contexto (Tipo 2), mas o contrário não é verdadeiro.

Exemplos de Linguagens LLC

Linguagem	Descrição	GLC
$L_1 = \{a^i b^i \mid i \geq 1\}$	Cadeias com i 'a's seguidos de i 'b's.	$S \rightarrow aSb \mid ab$
$L_2 = \{w \mid w = w^R\}$	Palíndromos sobre $\{a, b\}$.	$S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$
$L_3 = \{0^n 1^n 2^k \mid n, k \geq 0\}$	Balanço entre 0s e 1s, seguido por qualquer número de 2s.	$S \rightarrow AB$ $A \rightarrow 0A1 \mid \epsilon$ $B \rightarrow 2B \mid \epsilon$
L_4 (Parênteses aninhados)	Cadeias de parênteses bem formadas.	$S \rightarrow (S)S \mid \epsilon$

- **Observação:** A regra $A \rightarrow \epsilon$ permite a geração da palavra vazia.

Propriedades de Fechamento de LLCs

- O conjunto das LLCs é fechado para algumas operações:
 - **União:** Se L_1 e L_2 são LLCs, então $L_1 \cup L_2$ também é LLC.
 - **Esboço de Prova:** Criar uma nova GLC com um novo símbolo inicial S' e regras $S' \rightarrow S_1 \mid S_2$ (onde S_1 e S_2 são símbolos iniciais de L_1 e L_2).
 - **Concatenação:** Se L_1 e L_2 são LLCs, então L_1L_2 também é LLC.
 - **Fecho de Kleene (*):** Se L_1 é LLC, então L_1^* também é LLC.
- **Limitações (Não Fechamento):**
 - **Interseção:** O conjunto das LLCs **NÃO** é fechado para interseção.
 - Exemplo: $L_1 = \{a^i b^i c^j\}$ e $L_2 = \{a^j b^i c^i\}$ são LLCs, mas $L_1 \cap L_2 = \{a^i b^i c^i\}$ **não** é LLC.
 - **Complemento:** O conjunto das LLCs **NÃO** é fechado para complemento.

Formas Normais (FN): Introdução e Pré-processamento

- **Motivação:** GLCs são flexíveis, mas essa liberdade dificulta a construção de analisadores sintáticos e o uso em algoritmos de reconhecimento. FNs impõem restrições nas regras, facilitando o design de parsers e provas.
- **Etapas de Simplificação (Pré-FN):** Para alcançar as FNs, transformações são necessárias.
 - i. **Remoção de Recursividade no Símbolo Inicial:** Introduzir $S' \rightarrow S$ para garantir que S não seja recursivo.
 - ii. **Eliminação de Regras ϵ (Nulidade):** Remover $A \rightarrow \epsilon$ exceto $S \rightarrow \epsilon$ (se $\epsilon \in L(G)$).
 - iii. **Eliminação de Regras de Cadeia (Unitárias):** Remover regras do tipo $A \rightarrow B$.
 - iv. **Remoção de Símbolos Inúteis:** Remover variáveis que não geram terminais (TERM) e aquelas que não são alcançáveis (REACH).

Forma Normal de Chomsky (FNC)

Definição: Uma GLC está na FNC se todas as suas regras de produção têm a seguinte forma:

$$A \rightarrow BC \quad (\text{onde } B, C \in V \setminus \{S\})$$

$$A \rightarrow a \quad (\text{onde } a \in \Sigma)$$

$$S \rightarrow \epsilon \quad (\text{se } \epsilon \in L(G))$$

Mais detalhes na lousa.

Forma Normal de Greibach (FNG)

Definição: Uma GLC está na FNG se todas as suas regras de produção têm a seguinte forma:

$$A \rightarrow aA_1A_2 \dots A_n \quad (\text{onde } a \in \Sigma \text{ e } A_i \in V \setminus \{S\})$$

$$A \rightarrow a \quad \text{ou} \quad S \rightarrow \epsilon$$

O Autômato com Pilha e a Análise

- **AP (Autômato com Pilha):** Máquina aceitadora de LLCs (Tipo 2). Essencialmente um Autômato Finito Não-Determinístico (ϵ) **mais uma pilha** de memória ilimitada.
- **Função de Transição (δ):** Define a mudança de estado e o manuseio da pilha.

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\epsilon\}))$$

- **Mecânica da Transição (Exemplo):** $\delta(q_i, a, A) = [q_j, B]$:

- No estado q_i , lendo a na entrada e tendo A no topo da pilha...
- ...vai para o estado q_j , desempilha A , e empilha B .

Mais informações na próxima aula.

Aplicação em Compiladores

- Análise Léxica (Lexing/Scanning):
 - Função: Agrupar caracteres de entrada em **tokens** (e.g., identificadores, palavras-chave, operadores).
 - Formalismo: Linguagens Regulares (LRs), Expressões Regulares (ERs) e Autômatos Finitos.
- Análise Sintática (Parsing/Sintaxe):
 - Função: Verificar se a sequência de tokens está estruturalmente correta, segundo as regras da Linguagem de Programação (LP).
 - Formalismo: Gramáticas Livres de Contexto (GLCs).

Ambiguidade e a Construção da AST

- **Problema da Ambiguidade:**
 - Uma GLC é ambígua se uma palavra (w) possui duas ou mais árvores de derivação (ou duas ou mais derivações mais à esquerda).
 - Em LPs, isso é indesejável, pois leva a múltiplas interpretações estruturais (e.g., precedência de operadores).
 - Exemplo: $E \rightarrow E + E \mid E \times E \mid id$ para a cadeia $id + id \times id$.
- **Árvore Sintática Abstrata (AST):**
 - O objetivo final do parsing é gerar uma representação hierárquica da estrutura do código.
 - A **AST** (Abstract Syntax Tree) é uma versão simplificada da Árvore de Derivação, eliminando detalhes sintáticos irrelevantes e focando na estrutura semântica para posterior geração de código.

Limitações Intrínsecas

- O que LLCs não conseguem descrever:
 - Enquanto LLCs conseguem fazer **balanceamento duplo** (e.g., $a^n b^n$), não conseguem realizar **balanceamento triplo**.
- Exemplos de Linguagens NÃO Livres de Contexto:
 - **Triplo Balanceamento:** $L = \{a^n b^n c^n \mid n \geq 0\}$.
 - **Intuição:** O Autômato com Pilha (AP) tem apenas uma pilha. Ele consegue contar e balancear a 's com b 's. Mas, se precisar balancear c 's também, a informação original (sobre a 's ou b 's) já foi perdida na pilha.
 - **Cadeia Duplicada:** $L = \{ww \mid w \in \{a, b\}^*\}$.
 - O AP não consegue armazenar uma palavra w inteira na pilha para depois compará-la com o segundo w da entrada.

Lema do Bombeamento para LLCs

- **Propósito:** Técnica usada para provar que uma linguagem **NÃO** é Livre de Contexto.
- **Ideia Central:** Se L é uma LLC infinita, qualquer palavra z suficientemente longa deve ter repetições de variáveis em sua árvore de derivação (ciclos).
- **Estrutura do Lema (Simplificada):**
 - Se L é uma LLC, existe uma constante k tal que para toda palavra $z \in L$ com $|z| \geq k$, z pode ser dividida em cinco partes: $z = uvwxy$.
 - As restrições são:
 - a. $|vwx| \leq k$ (o 'bombeamento' deve ocorrer dentro de uma subárvore limitada).
 - b. $|vx| \geq 1$ (pelo menos uma parte a ser bombeada não é vazia).
 - c. $uv^iwx^i y \in L$ para todo $i \geq 0$ (a repetição das partes v e x deve manter a palavra na linguagem).
- **Diferença chave:** O Lema para LLCs permite o bombeamento de **duas** seções (v e x) simultaneamente, refletindo o poder de empilhar e desempilhar fornecido pela pilha.

Exercícios de Criação de GLCs

1. GLC para Linguagem com Contagem Desbalanceada (LLC):

- Desenvolva uma GLC para $L_a = \{a^n b^m \mid m > n, \text{ e } n, m \geq 0\}$. (O número de b 's é estritamente maior que o número de a 's).
- **Dica:** Gaste os a 's em par com b 's, e garanta que reste pelo menos um b .

2. GLC para Expressão Aritmética:

- Desenvolva uma GLC (usando variáveis a, b, c) que aceite expressões simples com soma e multiplicação (e.g., $a \times a + a$).

Exercícios de Derivação e Ambiguidade

1. Derivação (Sequência):

- Dada a GLC G :

$$S \rightarrow aSa \mid aBa$$

$$B \rightarrow bB \mid \epsilon$$

- Mostre a sequência de derivação mais à esquerda para a sentença aab^2a (i.e., $aabba$).

2. Verificação de Ambiguidade:

- A GLC G a seguir, que gera cadeias com o mesmo número de 'a's e 'b's, é ambígua? Justifique.

$$S \rightarrow SS \mid aSb \mid bSa \mid \epsilon$$

- **Dica:** Tente encontrar duas derivações mais à esquerda (ou duas árvores) para a cadeia ab .

Perguntas para Reflexão (Discussão)

1. **Poder de Expressão:** Por que as GLCs, mesmo sendo mais poderosas que as LRs, ainda não conseguem descrever linguagens como $\{a^n b^n c^n\}$ ou $\{ww\}$? O que falta no modelo do Autômato com Pilha?
2. **Ambiguidade na Prática:** Por que a ambiguidade de uma gramática é um problema **grave** no contexto de um compilador de Linguagem de Programação?

Conclusão

- **Formalismo:** GLCs (Tipo 2) são definidas por $G = (V, \Sigma, P, S)$, onde o lado esquerdo de cada regra é uma única variável.
- **Capacidade:** GLCs descrevem estruturas que exigem **balanceamento** e **memória infinita** (e.g., parênteses aninhados, $a^n b^n$), superando as Linguagens Regulares.
- **Aceitação:** São reconhecidas pelo **Autômato com Pilha (AP)**, que usa uma pilha para memória.
- **Restrições:** FNC e FNG impõem formatos restritos às regras para facilitar algoritmos de análise e garantir a remoção de recursão à esquerda.
- **Limitação:** LLCs não conseguem balancear três ou mais quantidades distintas (e.g., $a^n b^n c^n$).