Gramáticas Livres do Contexto

Prof. Hamilton José Brumatto

CIC-UESC

29 de dezembro de 2024

Gramática Livre de Contexto

- Vimos até o momento que a definição de uma linguagem depende de quem a reconheça.
- As linguagens regulares são obtidas de um autômato finito que a reconheça, ou de uma Expressão Regular que a gere.
- Mas vimos que as linguagens regulares são limitadas.
- Vamos conhecer aqui uma nova forma de definir uma linguagem, a gramática.
- Definindo a Gramática Livre de Contexto, definiremos por consequência a linguagem livre do contexto associada à gramática.

Gramáticas Livres do Contexto

- A gramática livre do contexto utiliza uma estrutura recursiva.
- O "bombeamento" ocorre, principalmente, quando uma estrutura é composta por ela mesma.
- Sendo recursiva (falamos em indução finita), é necessário haver um terminal.
- Assim, temos a definição de variável e terminal:
  - Uma variável pode ser constituída por outra variável, ou até mesmo terminal.
  - A linha de constituição de qualquer variável tem de chegar a um terminal (base da indução).
- A definição das variáveis representa: Regras de Substituição.

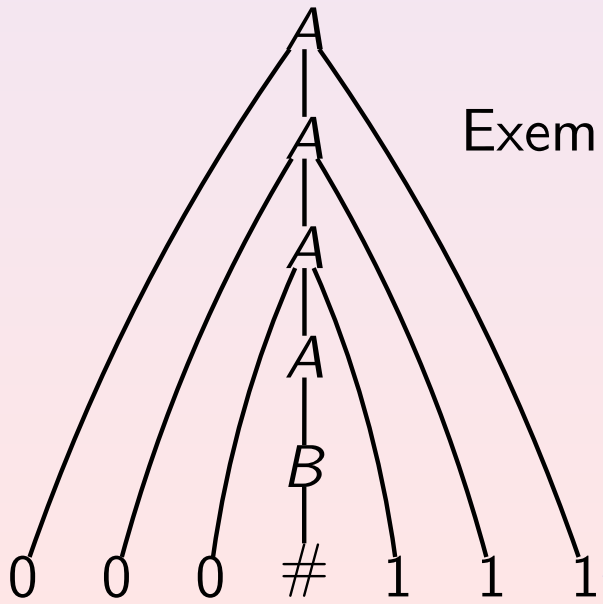
Sintaxe

Exemplo de gramática

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

Uma cadeia gerada (ou derivada) pela gramática são investigadas por uma *Análise Sintática* para identificar se está correta. Exemplo de derivação:

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$$



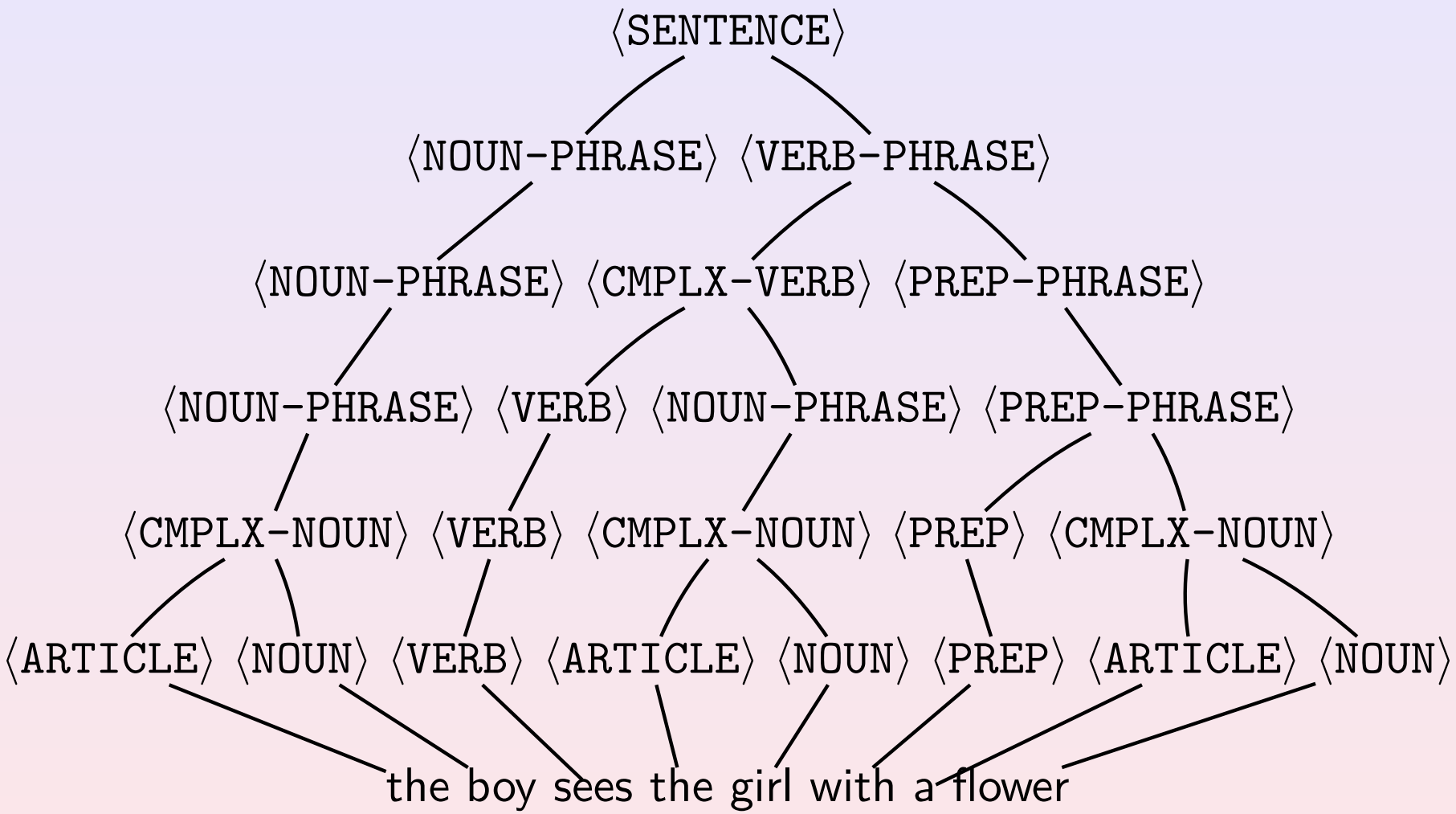
Exemplo de Árvore sinática

Linguagem de Programação

A gramática livre de contexto é utilizada para dar regras na formação da linguagem de programação. Também se aplica à linguagem como conhecemos. A partir da gramática se definem termos como: *nome*, *verbo* e *preposição*.

$$
\begin{aligned}
\langle \texttt{SENTENCE} \rangle &\rightarrow \langle \texttt{NOUN-PHRASE} \rangle \langle \texttt{VERB-PHRASE} \rangle \\
\langle \texttt{NOUN-PHRASE} \rangle &\rightarrow \langle \texttt{CMPLX-NOUN} \rangle \mid \langle \texttt{CMPLX-NOUN} \rangle \langle \texttt{PREP-PHRASE} \rangle \\
\langle \texttt{VERB-PHRASE} \rangle &\rightarrow \langle \texttt{CMPLX-VERB} \rangle \mid \langle \texttt{CMPLX-VERB} \rangle \langle \texttt{PREP-PHRASE} \rangle \\
\langle \texttt{PREP-PHRASE} \rangle &\rightarrow \langle \texttt{PREP} \rangle \langle \texttt{CMPLX-NOUN} \rangle \\
\langle \texttt{CMPLX-NOUN} \rangle &\rightarrow \langle \texttt{ARTICLE} \rangle \langle \texttt{NOUN} \rangle \\
\langle \texttt{CMPLX-VERB} \rangle &\rightarrow \langle \texttt{VERB} \rangle \langle \texttt{NOUN-PHRASE} \rangle \\
\langle \texttt{ARTICLE} \rangle &\rightarrow \texttt{a} \mid \texttt{the} \\
\langle \texttt{NOUN} \rangle &\rightarrow \texttt{boy} \mid \texttt{girl} \mid \texttt{flower} \\
\langle \texttt{VERB} \rangle &\rightarrow \texttt{touches} \mid \texttt{likes} \mid \texttt{sees} \\
\langle \texttt{PREP} \rangle &\rightarrow \texttt{with}
\end{aligned}
$$

Exemplo de linguagem



Linguagem de Programação: Gramática

Trechos da gramática do Turbo Pascal 3.0:

actual-parameter :: = *expression* | *variable*
adding-operator :: = + | - | **or** | **xor**
array-constant :: = (*structured-constant* { , *structured-constant* })
array-type :: = **array** [*index-type* { , *index-type* }] **of** *component-type*
array-variable :: = *variable*
assignment-statement :: = *variable* : = *expression* |
function-identifier :: = *expression*
base-type :: = *simple-type*
block :: = *declaration-part statement-part*
case-element :: = *case-list* : *statement*
case-label :: = *constant*
case-label-list :: = *case-label* { , *case-label* }
case-list :: = *case-list-element* { , *case-list-element*}
case-list-element :: = *constant* | *constant* .. *constant*
case-statement :: = **case** *expression* **of** *case-element* { ; *case-element* } **end** |

Definição Formal

Uma ***gramática livre-do-contexto*** é uma 4-upla (V, Σ, R, S), onde:

1. V é um conjunto finito denominado ***variáveis***;
2. Σ é um conjunto finito, disjunto de V, denominado ***terminais***;
3. R é um conjunto finito de ***regras***, com cada regra sendo uma variável e uma cadeia de variáveis e terminais; e
4. $S \in V$ é a variável inicial.

Seja u, v e w cadeias de variáveis e terminais, e $A \to w$ uma regra da gramática, dizemos que uAv ***origina*** uwv, escrito $uAv \Rightarrow uwv$.
Dizemos que u ***deriva*** v, escrito $u \overset{*}{\Rightarrow} v$, se $u = v$ ou se existe uma sequência $u_1, u_2, \ldots, u_k$ para $k \geqslant 0$, onde

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$$

A linguagem é:

$$\{w \in \Sigma^* | S \overset{*}{\Rightarrow} w\}$$

Exemplo

Considere a gramática: $(\{S\}, \{a, b\}, S \rightarrow aSb|SS|\varepsilon, S)$.
Cadeias que são geradas: *abab*, *aaabbb*, *aababb*.
Uma análise mais cuidadosa, podemos ver que um *a* (abrindo) tem sempre um *b* que o fecha.
Esta gramática gera uma linguagem onde em cada cadeia, um *a* está associado a um *b* específico, após o *a*, numa estrutura de abrir e fechar aninhados.

Exemplo

Considere a gramática: $(V, \Sigma, R, \langle \texttt{EXPR} \rangle)$, onde:
$V = \{\langle \texttt{EXPR} \rangle, \langle \texttt{TERM} \rangle, \langle \texttt{FACTOR} \rangle\}$, e
$\Sigma = \{a, +, \times, (,)\}$, com as seguintes regras:

$$R : \begin{cases} \langle \texttt{EXPR} \rangle \rightarrow \langle \texttt{EXPR} \rangle + \langle \texttt{TERM} \rangle | \langle \texttt{TERM} \rangle \\ \langle \texttt{TERM} \rangle \rightarrow \langle \texttt{TERM} \rangle \times \langle \texttt{FACTOR} \rangle | \langle \texttt{FACTOR} \rangle \\ \langle \texttt{FACTOR} \rangle \rightarrow \langle (\texttt{EXPR}) \rangle | a \end{cases}$$

A partir desta gramática podemos escrever: $a + a \times a$ e $(a + a) \times a$. Ou seja, expressões que respeitam a ordem de precedência.
Neste ponto existe a questão de ambiguidade: "Uma cadeia é derivada **ambiguamente** na gramática livre-do-contexto G se ela tem duas ou mais derivações mais à esquerda diferentes."
A gramática acima não é ambigua, mas a gramática a seguir é:

$$\langle \texttt{EXPR} \rangle \rightarrow \langle \texttt{EXPR} \rangle + \langle \texttt{EXPR} \rangle | \langle \texttt{EXPR} \rangle \times \langle \texttt{EXPR} \rangle | (\langle \texttt{EXPR} \rangle) | a$$

$a + a \times a$ tem origem ambígua nesta gramática.

Construção de Gramáticas Livres do Contexto

A construção das GLC são complexas e requer criatividade:

- Muitas GLC são uniões de GLCs mais simples, então é interessante construir as simples e depois uni-las na mais complexa.
- Se a linguagem for regular, primeiro se constrói o AFD para a linguagem e depois a GLC a partir do AFD.
- Quando há cadeias ligadas: o tanto da cadeia u é o mesmo da cadeia v, é possível ligá-las na forma $R \rightarrow uRv$.
- Nas gramáticas mais complexas é necessário criar estruturas recursivas e que são baseadas em outras estruturas

União de GLCs

Vamos considerar a linguagem: $L = \{0^n1^n \vee 1^n0^n | n \geqslant 0\}$
Vemos aí duas formas distintas unidas na mesma linguagem, poderíamos escrever:

$$L = \{0^n1^n | n \geqslant 0\} \cup \{1^n0^n | n \geqslant 0\}$$

- Para a primeira: $G_1 : S \rightarrow 0S1|\varepsilon$
- Para a segunda: $G_2 : S \rightarrow 1S0|\varepsilon$
- Então nossa linguagem fica:

$$\begin{aligned} S &\rightarrow S_1|S_2 \\ S_1 &\rightarrow 0S_11|\varepsilon \\ S_2 &\rightarrow 1S_20|\varepsilon \end{aligned}$$

Exemplo: Definindo um AFN

A construção da GLC segue:

- Para cada estado q_i do AFD, crie a variável R_i.
- Adicione a regra $R_i \to aR_j$ quando $\delta(q_i, a) \to q_j$
- Adicione a regra $R_i \to \varepsilon$ se q_i for um estado de aceitação.
- Para q_0 o estado inicial do AFD, R_0 é a variável inicial da gramática.

Linguagem regular: $\Sigma = \{0, 1\}$

$L(R) = \{w | w$ é vazia ou contém um número par de 1s e termina com $0\}$

AFD para a linguagem: $F = (\{S, P, I\}, \{0, 1\}, \delta, S, S)$ onde

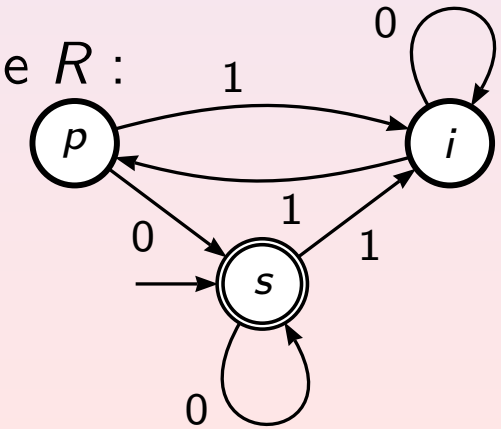
δ	0	1
s	s	i
p	s	i
i	i	p

$G : (\{P, I, S\}, \emptyset, R, S)$, onde R :

$S \to 0S|1I|\varepsilon$

$I \to 0I|1P$

$P \to 0S|1I$



Forma Normal de Chomsky

O modelo recursivo das GLCs são às vezes complexos. Muitas vezes é interessante criar um modelo mais simples da gramática.
Uma das formas mais simples é a **Forma Normal de Chomsky**. Esta forma é interessante para gerar algoritmos.

Na ***forma normal de Chomsky*** todas as regras segue a forma:

$$A \rightarrow BC$$
$$A \rightarrow a$$

Sendo a qualquer terminal, e A, B, e C são variáveis. B e C não pode ser variável inicial
Também é permitida a regra $S \rightarrow \varepsilon$, sendo S a variável inicial.

Qualquer GLC pode ser transformado em uma Forma normal de Chomsky

Forma Normal de Chomsky

Teorema

Qualquer GLC pode ser transformado em uma Forma normal de Chomsky

Prova

Para transformar uma Gramática G na forma normal de Chomsky, seguimos um conjunto de passos:

- *Criamos nova variável inicial para evitar a variável inicial no lado direito.*
- *Todas as regras do tipo $A \to \varepsilon$ são eliminadas.*
- *Todas as regras do tipo $A \to B$ são eliminadas.*
- *Finalmente convertemos as regras remanescentes para atender à forma apropriada.*

Nestas eliminações corrigimos a gramática

Forma Normal de Chomsky - Prova

Vamos demonstrar a prova, com exemplo na seguinte gramática:

$$S \rightarrow ASA|aB$$
$$A \rightarrow B|S$$
$$B \rightarrow b|\varepsilon$$

Prova

Primeiro adicionamos uma variável inicial S_0 e a regra $S_0 \rightarrow S$. Com esta regra a variável inicial não irá aparecer do lado direito.

$$\mathbf{S_0 \rightarrow S}$$
$$S \rightarrow ASA|aB$$
$$A \rightarrow B|S$$
$$B \rightarrow b|\varepsilon$$

Forma Normal de Chomsky - Prova

Prova

Em seguida removemos as regras $A \to \varepsilon$ em que A não é a variável inicial

Primeiro $B \to \varepsilon$

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA|aB|\mathbf{a} \\
A &\to B|S|\varepsilon \\
B &\to b
\end{aligned}
$$

Em seguida removemos $A \to \varepsilon$

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA|\mathbf{SA}|\mathbf{AS}|\mathbf{S}|aB|a \\
A &\to B|S \\
B &\to b
\end{aligned}
$$

Forma Normal de Chomsky - Prova

Prova

Removemos a regra unitária $A \to B$, Se houver a regra $B \to u$ substituimos $A \to B$ por $A \to u$, sendo u uma cadeia qualquer da gramática.

Substituindo $S_0 \to S$

$$
\begin{aligned}
S_0 &\to \mathbf{ASA}|\mathbf{SA}|\mathbf{AS}|\mathbf{aB}|\mathbf{a} \\
S &\to ASA|SA|AS|S|aB|a \\
A &\to B|S \\
B &\to b
\end{aligned}
$$

Substituindo $S \to S$

$$
\begin{aligned}
S_0 &\to ASA|SA|AS|aB|a \\
S &\to ASA|SA|AS|aB|a \\
A &\to B|S \\
B &\to b
\end{aligned}
$$

Forma Normal de Chomsky - Prova

Substituindo $A \to B$ e $A \to S$

$$
\begin{aligned}
S_0 &\to ASA|SA|AS|aB|a \\
S &\to ASA|SA|AS|aB|a \\
A &\to \mathbf{b}|\mathbf{ASA}|\mathbf{SA}|\mathbf{AS}|\mathbf{aB}|\mathbf{a} \\
B &\to b
\end{aligned}
$$

Prova

Por fim adequa-se todas as cadeias no forma normal:

- *Para cada sequência de terminais:* $W = w_1 w_2 w_3 \ldots u_k$ *fazemos:* $U_1 \to w_1 U_2$, $U_2 = w_2 U_3, \ldots$ *também fazemos* $W_1 \to w_1, W_2 \to w_2$ *e substituímos na sequência anterior.*
- *Nas regras* $A \to A_1 A_2 \ldots A_3$, *substituimos* $B_1 \to A_1 A_2$, $B_2 \to B_1 A_3$ *e assim sucessivamente até sobrar só cadeias com duas variáveis.*

□

Forma Normal de Chomsky - Prova

$$
\begin{aligned}
S_0 &\rightarrow AA_1|SA|AS|UB|a \\
S &\rightarrow AA_1|SA|AS|UB|a \\
A &\rightarrow b|AA_1|SA|AS|UB|a \\
A_1 &\rightarrow SA \\
U &\rightarrow a \\
B &\rightarrow b
\end{aligned}
$$

Definição

Uma linguagem é denominada ***Linguagem Livre do Contexto*** se existe uma *Gramática Livre do Contexto* que a gere.

Atividades

- Ler a seção 2.1 do Sipser
- Fazer os exercícios 2.1, 2.2, 2.3, 2.4