

Introdução à Teoria da Computação

Gramáticas - Parte 2

Professor Murilo V. G. da Silva

Departamento de Informática
Universidade Federal do Paraná

2025/2

Produção de strings: formalizando

Produção de string (um passo)

Seja $G = (V, T, P, S)$ uma gramática.

- Seja uma string $\alpha A \beta$ onde $A \in V$ e $\alpha, \beta \in (V \cup T)^*$.
- Seja $A \rightarrow \gamma$ uma regra de P .

Dizemos que $\alpha A \beta$ produz em um passo $\alpha \gamma \beta$ em G .

Notação: $\alpha A \beta \Rightarrow_G \alpha \gamma \beta$ (por conveniência, às vezes escrevemos $\alpha A \beta \Rightarrow \alpha \gamma \beta$)

- Exemplo: Na gramática das expressões aritméticas, temos $a * (I + E) \Rightarrow a * (a + E)$

Produção de string (múltiplos passos)

Seja $G = (V, T, P, S)$ uma gramática. Sejam $\alpha, \beta \in (V \cup T)^*$,

Exercício: Leia no livro a definição para $\alpha \Rightarrow_G^* \beta$

Novamente, por conveniência, às vezes escrevemos $\alpha \Rightarrow^* \beta$.

- Exemplo: Na gramática das expressões aritméticas, temos $E \Rightarrow^* a * (a + b00)$

Linguagem de uma gramática

Dada uma gramática $G = (V, T, P, S)$, a linguagem de G é

$$L(G) = \{w \in T^* : S \Rightarrow_G^* w\}.$$

- Dizemos que a linguagem L é *gerada* por G se $L = L(G)$.

Teorema: Uma linguagem L é livre de contexto $\Leftrightarrow \exists$ uma gramática G tal que $L = L(G)$.

Ideia da prova: Argumentar que $w \in L(G) \Leftrightarrow$ existe uma AP P tal que $w \in N(P)$.

Derivações e árvores sintáticas

Relembrando a gramática anterior:

$G_{\text{EX}} = (\{I, E\}, \{a, b, 0, 1, +, *, (,)\}, A_1, E)$ tal que as regras de A_1 são:

$$E \rightarrow I \mid E + E \mid E * E \mid (E)$$

$$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

Vamos derivar $E + E \Rightarrow$

Qual variável escolher para aplicar regra? $E + E \Rightarrow I + E$

$$E + E \Rightarrow E + I$$

Produzindo $a + b$ de duas maneiras diferentes:

$$(i) E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + I \Rightarrow a + b$$

$$(ii) E \Rightarrow E + E \Rightarrow E + I \Rightarrow I + I \Rightarrow a + I \Rightarrow a + b$$

A string pode ser produzida por várias **derivações** diferentes

Derivações e árvores sintáticas

Uma *derivação* de uma gramática $G = (V, T, P, S)$ é uma sequência de passos

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_k$$

sendo $S = \alpha_1$ e cada $\alpha_i \Rightarrow \alpha_{i+1}$ é uma produção válida na gramática G .

Duas *derivações* diferentes de $a + b * c$:

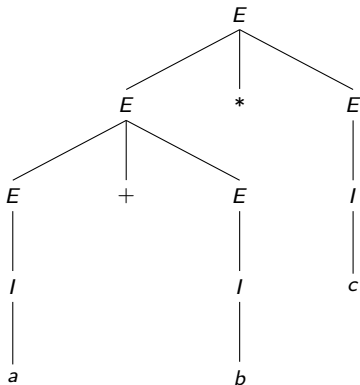
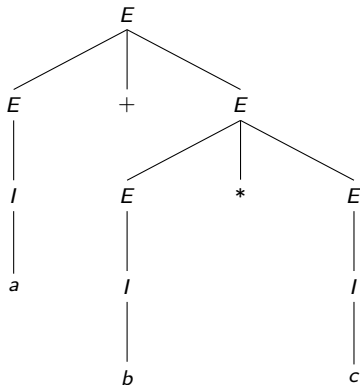
$$\begin{aligned} E &\Rightarrow E + E \Rightarrow E + E * E \Rightarrow E + E * I \Rightarrow E + E * c \Rightarrow I + E * c \Rightarrow a + E * c \Rightarrow a + I * c \Rightarrow a + b * c \\ E &\Rightarrow E * E \Rightarrow E + E * E \Rightarrow E + E * I \Rightarrow E + E * c \Rightarrow I + E * c \Rightarrow a + E * c \Rightarrow a + I * c \Rightarrow a + b * c \end{aligned}$$

Derivações e árvores sintáticas

Duas **derivações** diferentes de $a + b * c$:

$E \Rightarrow E + E \Rightarrow E + E * E \Rightarrow E + E * I \Rightarrow E + E * c \Rightarrow I + E * c \Rightarrow a + E * c \Rightarrow a + I * c \Rightarrow a + b * c$
 $E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow E + E * I \Rightarrow E + E * c \Rightarrow I + E * c \Rightarrow a + E * c \Rightarrow a + I * c \Rightarrow a + b * c$

Árvores sintáticas das derivações:



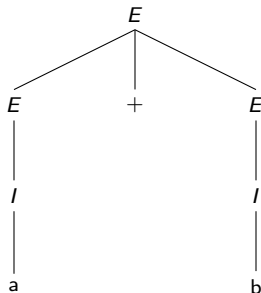
Derivações e árvores sintáticas

Duas derivações para a string $a * b$:

(i) $E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + I \Rightarrow a + b$

(ii) $E \Rightarrow E + E \Rightarrow E + I \Rightarrow I + I \Rightarrow a + I \Rightarrow a + b$

Ambas derivações (i) e (ii) tem a mesma árvore sintática:



Derivações e árvores sintáticas

Seja $G = (V, T, P, S)$ uma gramática e seja $w = w_1 w_2 \dots w_k$ uma string de $L(G)$.

Seja $\mathcal{D}$ uma derivação w em G . A *árvore sintática da derivação $\mathcal{D}$* é uma árvore com raiz S com as seguintes relações entre os nós:

- Se o passo $\alpha A \beta \Rightarrow \alpha \gamma \beta$ aparece na derivação $\mathcal{D}$ e $\gamma = \gamma_1 \gamma_2 \dots \gamma_l$, então o nó A é pai dos nós $\gamma_1, \gamma_2, \dots, \gamma_l$, da esquerda para a direita.

O conjunto das árvores sintáticas de todas as possíveis derivações de w é chamado de conjunto das *árvores sintáticas da string w* .

- Exemplo:** As duas árvores do *slide 5* estão contidas no conjunto das *árvores sintáticas de $a + b * c$* .
- Exercício:** Mostrar que todas as *derivações* tem como árvore exatamente uma das duas árvores do *slide 5* anterior

Mostrar que a árvore do *slide 6* é a única *árvore sintática de $a + b$* .

Derivações e árvores sintáticas

Seja $G = (V, T, P, S)$ uma gramática e seja $w = w_1 w_2 \dots w_k$ uma string de $L(G)$.

Dada uma $\mathcal{D}$ uma derivação w em G , a *árvore sintática da derivação $\mathcal{D}$* é uma árvore com raiz S com as seguintes relações entre os nós:

- Se o passo $\alpha A \beta \Rightarrow \alpha \gamma \beta$ aparece na derivação $\mathcal{D}$ e $\gamma = \gamma_1 \gamma_2 \dots \gamma_l$, então o nó A é pai dos nós $\gamma_1, \gamma_2, \dots, \gamma_l$, da esquerda para a direita.

O conjunto das árvores sintáticas de todas as possíveis derivações de w é chamado de conjunto das *árvores sintáticas da string w* .

Def.: A gramática G é *ambígua* se existe $w \in L(G)$ que possui mais de uma árvore sintática. Caso contrário G é dita *não ambígua*.

Exemplos:

- A gramática G_{PAL} não é ambígua.
- A gramática G_{EX} é ambígua.

Gramáticas ambíguas e derivações mais a esquerda

Relembrando a gramática anterior:

$G_{\text{EX}} = (\{I, E\}, \{a, b, 0, 1, +, *, (,)\}, A_{\text{EX}}, E)$ tal que as regras de A_1 são:

$$E \rightarrow I \mid E + E \mid E * E \mid (E)$$

$$I \rightarrow a \mid b \mid a \mid b \mid 0 \mid 1$$

Voltando a questão anterior: qual variável aplicar em $E + E$ na produção?

Derivação mais a esquerda

Na derivação $\alpha \Rightarrow \beta$ sempre escolha aplicar a regra à variável mais a esquerda de α

- Isso é dito uma **produção mais a esquerda em um passo**.
- Notação: $\Rightarrow_{1m}$

e.g., $E + E \Rightarrow_{1m} I + E$

Analogamente também definir **produção mais a esquerda** ($\Rightarrow_{1m}^*$).

Gramáticas ambíguas

Exemplo de produção mais à esquerda:

Relembre a derivação $\mathcal{D}$ abaixo:

$$\begin{aligned} E &\Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow a * (E) \Rightarrow a * (E + E) \Rightarrow a * (I + E) \\ &\Rightarrow a * (a + E) \Rightarrow a * (a + I) \Rightarrow a * (a + I0) \Rightarrow a * (aI00) \Rightarrow a * (a + b00) \end{aligned}$$

(note: é uma derivação mais a esquerda)

Portanto, também podemos escrever $E \Rightarrow_{1m}^* a * (a + b00)$.

- Similarmente definimos *derivações mais a direita* ($\Rightarrow_{rm}$ e $\Rightarrow_{rm}^*$).

Teorema: Seja G uma gramática. Toda string de $L(G)$ tem exatamente uma derivação mais à esquerda $\Leftrightarrow G$ não é ambígua.

Teorema: Seja G uma gramática. Toda string de $L(G)$ tem exatamente uma derivação mais à direita $\Leftrightarrow G$ não é ambígua.

Ambiguidade e APDs

- Obs: Se L admite uma gramática ambígua G tal que $L(G) = L$ não significa que toda gramática que gere a linguagem seja ambígua
- **Exemplo:** Podemos construir uma gramática não ambígua G'_{EX} tal que $L(G'_{EX}) = L(G_{EX})$

Def.: Seja L uma linguagem. Se existe uma gramática não ambígua G tal que $L(G) = L$, então L é dita uma **linguagem (livre de contexto) não ambígua**.

Entretanto, considere a seguinte linguagem:

- $L_{AMB} = \{a^n b^n c^m d^m : n > 0, m > 0\} \cup \{a^n b^m c^m d^n; n > 0, m > 0\}$
obs: L_{AMB} é livre de contexto (**exercício** no livro)

Teorema: Toda gramática G tal que $L_{AMB} = L(G)$ é ambígua.

A linguagem livre de contexto acima é dita **inerentemente ambígua**.

Ambiguidade e APDs

Teorema: Seja L uma linguagem.

$\exists$ APD P tal que $L = L(P) \Rightarrow \exists$ uma gramática não ambígua G tal que $L(G) = L$.

- Entretanto, a outra direção não vale:

Teorema: Não existe nenhum APD que aceite a linguagem L_{PAL} .

Isto é, existe pelo menos uma linguagem não ambígua que não admite APD

Aplicação de gramáticas em compiladores:

- Gramáticas de Linguagens de Programação normalmente estão contidas em um subconjunto estritamente contido no conjunto das linguagens de APDs
exemplo: conjunto das gramáticas $LR(k)$
 - O analisador sintático do compilador: funciona como um APD e constrói uma árvore de uma derivação mais a direita.
 - Realidade prática: as coisas são mais complicadas
 - “grosso” da gramática é do tipo $LR(k)$: compilador usa um algoritmo $LR(k)$
 - pedaços pequenos da gramática tratados separadamente.
- (às vezes nem mesmo livre de contexto)