Fundamentos Teóricos da Computação

CIÊNCIA DA COMPUTAÇÃO

Prof. Dr. João Paulo Aramuni

UNIVERSIDADE FUMEC

Sumário

* **Introdução às Gramáticas**
* **Gramáticas Regulares**
  * Transformação em AFs
* **Gramáticas Livres de Contexto**
  * Técnicas de projeto de GLCs

Introdução às Gramáticas

* **Introdução às Gramáticas**

Introdução às Gramáticas

- Gramáticas são uma outra forma de definir linguagens
  - Existem diversos tipos de gramáticas. Cada tipo é diferido pela forma de suas regras.
    - Neste curso estudaremos dois tipos de gramáticas
      - Gramáticas Regulares
      - Gramáticas Livre de Contexto
  - As gramáticas nada mais são do que definições recursivas
    - O elemento fundamental da gramática é a <u>regra</u>

Introdução às Gramáticas

- Uma regra é um par ordenado (u, v) geralmente escrito na forma:
  - $u \to v$
  - Em que u e v são palavras que podem conter símbolos de dois alfabetos disjuntos
    - Um com símbolos denominados *variáveis*
    - Outro com símbolos denominados *terminais*
  - As *variáveis* são símbolos auxiliares para gerar palavras da linguagem definida pela gramática
    - O alfabeto da linguagem gerada só contém *terminais*

Regra

* Seja a regra:

$$aAB \rightarrow baA$$

* Neste exemplo, usaremos letras maiúsculas para as *variáveis* e letras minúsculas para os *terminais*

* Esta regra define que em qualquer palavra que contenha uma subpalavra aAB, esta subpalavra pode ser substituída por baA

Regra

* Seja a regra:

$$aAB \rightarrow baA$$

* Assim, a partir da palavra $aABBaAB$, deriva-se:
  * $baABaAB$, substituindo a primeira ocorrência de aAB
  * $aABBbaA$, substituindo a segunda ocorrência de aAB

Derivação

- A relação de derivação é representada usando $\Rightarrow$
  - Assim, temos uma possível cadeia de derivações como abaixo:

$$
\begin{aligned}
aABBaAB &\Rightarrow baABaAB \quad (aAB \rightarrow baA) \\
&\Rightarrow bbaAaAB \quad (aAB \rightarrow baA) \\
&\Rightarrow bbaAbaA \quad (aAB \rightarrow baA)
\end{aligned}
$$

Derivação

- De uma gramática, consta um conjunto de regras e uma variável denominada *variável de partida*
  - Todas as derivações devem ser iniciadas com uma palavra constituída apenas pela variável de partida
  - As palavras geradas a partir da variável de partida de uma gramática são chamadas *formas sentenciais*
    - Uma *forma sentencial* sem variáveis é chamada de *sentença*
- A linguagem definida pela gramática é o conjunto de *sentenças* geradas pela gramática

Exemplo 1

- Para uma gramática G, a linguagem gerada por ela é denotada por $L(G)$. Seja a gramática G constituída pela variável de partida P e as seguintes regras:
  1. $P \rightarrow aAbc$
  2. $A \rightarrow aAbC$
  3. $A \rightarrow \lambda$
  4. $Cb \rightarrow bC$
  5. $Cc \rightarrow cc$

Exemplo 1

* Toda derivação de G deve começar com a forma sentencial constituída pela variável de partida P. Um exemplo de derivação:

$$
\begin{aligned}
P &\Rightarrow aAbc && \text{(regra 1)} \\
&\Rightarrow abc && \text{(regra 3)}
\end{aligned}
$$

Exemplo 1

* Toda derivação de G deve começar com a forma sentencial constituída pela variável de partida P. Um exemplo de derivação:

$$
\begin{aligned}
P &\Rightarrow aAbc && \text{(regra 1)} \\
&\Rightarrow abc && \text{(regra 3)}
\end{aligned}
$$

Exemplo 1

* Isso mostra que abc é uma sentença da linguagem gerada por G: $abc \in L(G)$. Outro exemplo de derivação:

$$
\begin{aligned}
P &\Rightarrow aAbc && \text{(regra 1)} \\
&\Rightarrow aaAbCbc && \text{(regra 2)} \\
&\Rightarrow aaaAbCbCbc && \text{(regra 2)} \\
&\Rightarrow aaabCbCbc && \text{(regra 3)} \\
&\Rightarrow aaabbCCbc && \text{(regra 4)} \\
&\Rightarrow aaabbCbCc && \text{(regra 4)} \\
&\Rightarrow aaabbCbcc && \text{(regra 5)} \\
&\Rightarrow aaabbbCcc && \text{(regra 4)} \\
&\Rightarrow aaabbbccc && \text{(regra 5)}
\end{aligned}
$$

Exemplo 1

* Logo, $a^3b^3c^3 \in L(G)$

* Na verdade, pode-se mostrar que $L(G) = \{ a^n b^n c^n \mid n >= 1 \}$

* Relembrar é viver:
  * Essa linguagem não é regular, por esse motivo seria impossível construir um autômato finito que a reconheça. Para que o autômato possa "contar" (guardar em memória) quantos a's, b's, ou c's possua à palavra, seria necessário um autômato de pilha determinístico.

Definição Formal

- Uma gramática é uma quádrupla (V, Σ, R, P), em que:
  - V é um conjunto finito de elementos denominados variáveis;
  - Σ é um alfabeto; $V \cap \Sigma = \emptyset$;
  - R, um subconjunto de $(V \cup \Sigma)^{+} \times (V \cup \Sigma)^{*}$ é um conjunto finito de pares ordenados chamados regras; e
  - $P \in V$ é uma variável conhecida como <u>variável de partida</u>.

Observações

* Variáveis serão escritas em letra maiúscula;
* Terminais serão escritos em letra minúscula;
* A linguagem gerada pela gramática é o conjunto de sentenças geradas pela gramática
  * Sentenças não possuem variáveis !

Observações

* É muito comum uma gramática ter duas ou mais regras com o mesmo lado esquerdo. Por exemplo, a gramática do Exemplo 1, tem as regras 2 e 3 com o mesmo lado esquerdo: A (slide 10).

* Nesse caso, pode ser útil a abreviação, colocando-se apenas um lado esquerdo e também os lados direitos das várias regras separados por "|". As regras 2 e 3 do Exemplo 1 seriam substituídas por:

$$A \to aAbC \mid \lambda$$

Exemplo 2

- Seja a gramática $G = (V, \Sigma, R, E)$, em que:
  - $V = \{E, T, N, D\}$
  - $\Sigma = \{+, -, (,), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - R contém as seguintes regras:
    - $E \rightarrow E + T \mid E - T \mid T$
    - $T \rightarrow (E) \mid N$
    - $N \rightarrow DN \mid D$
    - $D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Exemplo 2

* A gramática G gera expressões aritméticas contendo números na base decimal, operadores de soma e subtração e parênteses.
* Por meio das três regras com E do lado esquerdo, pode-se gerar uma sequência de somas e/ou subtrações de Ts (termos); por exemplo:

$$
\begin{aligned}
E &\Rightarrow E + T && \text{(regra } E \rightarrow E + T\text{)} \\
&\Rightarrow E - T + T && \text{(regra } E \rightarrow E - T\text{)} \\
&\Rightarrow E - T - T + T && \text{(regra } E \rightarrow E - T\text{)} \\
&\Rightarrow T - T - T + T && \text{(regra } E \rightarrow T\text{)}
\end{aligned}
$$

Exemplo 2

* Observe como as regras são *recursivas à esquerda*, levando à produção de uma sequência da direita para a esquerda.
* Entretanto, as regras responsáveis pela produção dos números das expressões são *recursivas à direita*, redundando na produção de número da esquerda para a direita. Por exemplo, para gerar um número de quatro dígitos, pode-se derivar:

$$\begin{aligned} N &\Rightarrow DN && \text{(regra } N \rightarrow DN\text{)} \\ &\Rightarrow DDN && \text{(regra } N \rightarrow DN\text{)} \\ &\Rightarrow DDDN && \text{(regra } N \rightarrow DN\text{)} \\ &\Rightarrow DDDD && \text{(regra } N \rightarrow D\text{)} \end{aligned}$$

Exemplo 2

* E, em seguida, derivar os quatro dígitos usando-se as regras com D do lado esquerdo.
* Observe também que a geração de parênteses se dá por recursão (no caso, indireta), a qual não pode ser classificada como recursão à esquerda nem à direita. Por exemplo, na derivação:

$$
\begin{aligned}
E &\Rightarrow T + T && \text{(regra } E \rightarrow T\text{)} \\
&\Rightarrow T + T && \text{(regra } E \rightarrow T\text{)} \\
&\Rightarrow (E) + T && \text{(regra } T \rightarrow (E)\text{)}
\end{aligned}
$$

A variável E aparece (recursivamente) na forma sentencial entre “(” e “)”.

Gramáticas Regulares

* **Gramáticas Regulares**

Gramáticas Regulares

* Uma GR é uma gramática (V, Σ, R, P), em que cada regra tem uma das seguintes formas:
  * $X \rightarrow a$;
  * $X \rightarrow aY$;
  * $X \rightarrow \lambda$;

* Em que X,Y são variáveis e a é um símbolo do alfabeto
* Toda GR gera uma linguagem regular

Gramáticas Regulares

* Uma característica interessante das GRs é o formato das formas sentenciais geradas: wA, sendo que w só tem terminais e A é uma variável.

Exemplo 3

* A GR abaixo gera a linguagem:
* $L(G) = \{\ w \in \{\mathrm{a,b,c}\}^* \mid w \text{ não contém abc } \}$

* $G = (\{A,B,C\}, \{\mathrm{a,b,c}\}, R, A)$, em que R é:

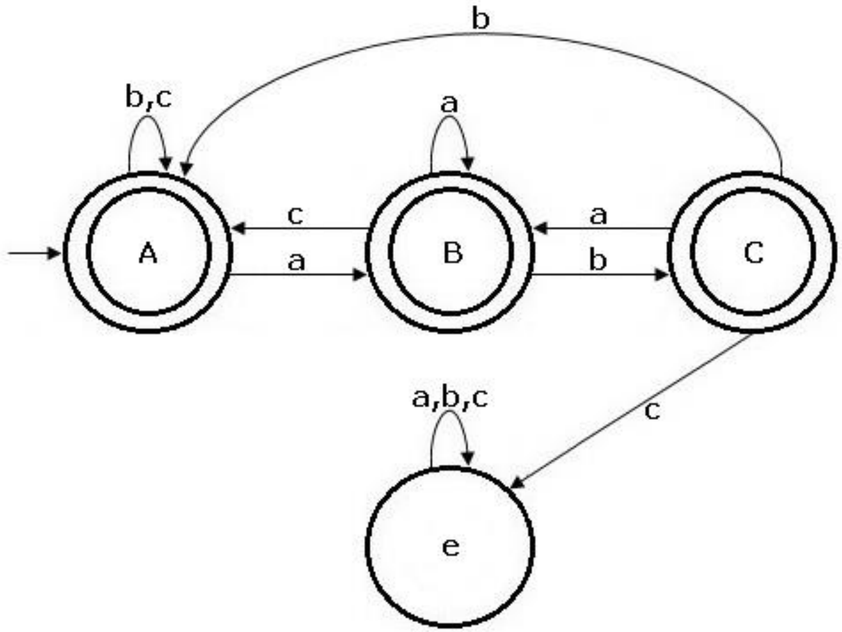
  * $A \rightarrow \mathrm{a}B \mid \mathrm{b}A \mid \mathrm{c}A \mid \lambda$
  * $B \rightarrow \mathrm{a}B \mid \mathrm{b}C \mid \mathrm{c}A \mid \lambda$
  * $C \rightarrow \mathrm{a}B \mid \mathrm{b}A \mid \lambda$

Transformando GRs em AFs

* É possível transformar uma Gramática Regular em um AF, ou vice-versa
  * Cada estado de um AF corresponde a uma variável da gramática
  * Cada regra da gramática é utilizada para criar uma transição entre os estados
    * Uma regra da forma $A \rightarrow aB$, gera uma transição do estado A para o estado B sob o símbolo a
  * A variável de partida corresponde ao estado inicial
  * Qualquer variável que possa ser substituída por λ corresponde a um estado final
  * Qualquer variável que possa ser substituída por apenas um símbolo de Σ leva a um estado final que não possui transições saindo dele

Exemplo 4

- A gramática do exemplo 3 corresponde ao AFD abaixo:
- (Poderia ser um AFN, neste caso foi escolhido AFD)



Gramáticas Livres de Contexto

* **Gramáticas Livres de Contexto**

Gramáticas Livres de Contexto

- GLCs geram Linguagens Livres de Contexto
  - Todas as Linguagens Regulares são também Livres de Contexto
  - Muitas Linguagens de Programação são LLCs

- GLCs são a base para projeto e implementação de Compiladores
  - Especificação de Linguagens
  - Entrada para programas que geram Analisadores Sintáticos Automaticamente (YACC, Bison, etc.)

Gramáticas Livres de Contexto

* Regras para a vida:

"Toda gramática regular é livre de contexto, mas nem toda gramática livre de contexto é regular."

"Toda linguagem regular é livre de contexto, mas nem toda linguagem livre de contexto é regular."

Gramáticas Livres de Contexto

* Regras para a vida:

"Uma linguagem é dita ser uma linguagem livre de contexto se existe uma gramática livre de contexto que a gera."

"Uma linguagem é dita ser uma linguagem sensível ao contexto se existe uma gramática sensível ao contexto que a gera."

Definição

* Uma GLC é uma gramática (V, Σ, R, P), em que cada regra tem a forma:

  * $X \rightarrow w$;

    * Em que $X \in V$
    * $w \in (V \cup \Sigma)^*$

  * Ou seja, cada regra só pode ter uma variável do lado esquerdo e, no lado direito, pode ter variáveis ou terminais

Definição

* Para um GLC, em cada passo de uma derivação deve-se escolher, na forma sentencial, a variável A a ser substituída pelo lado direito de uma regra com A do lado esquerdo. Ao se fazer tal substituição, diz-se que A é <u>expandida</u>.

Exemplo 5

* Seja a GLC $G = (\{P\}, \{0,1\}, R, P)$, cujas únicas duas regras de R são:
  * $P \rightarrow 0P1 \mid \lambda$
* A Linguagem $L(G)$ é:
  * $L(G) = \{0^n 1^n \mid n \in \mathbf{N}\}$

Exemplo 5

- Seja a GLC $G = (\{P\}, \{0,1\}, R, P)$, cujas únicas duas regras de R são:
  - $P \longrightarrow 0P1 \mid \lambda$: Exemplo de GLC que não é uma GR
- A Linguagem $L(G)$ é:
  - $L(G) = \{0^n1^n \mid n \in \mathbf{N}\}$: Exemplo de LLC que não é uma LR

Exemplo 5 – Análise da Regra

* Sejam duas regras
  * $P \rightarrow 0P1 \mid \lambda$

* Toda forma sentencial gerada por estas regras
  * Possui a mesma quantidade de 0s e 1s
  * A primeira metade é composta de 0s
  * A segunda metade é composta de 1s

* A última regra aplicada é sempre uma substituição de P por λ

Exemplo 6

- Seja a GLC $G = (\{P\}, \{0,1\}, R, P)$, cujas regras de R são:
  - $P \rightarrow 0P0 \mid 1P1 \mid 0 \mid 1 \mid \lambda$
- A Linguagem $L(G)$ é:
  - $L(G) = \{w \in \{0,1\}^* \mid w = w^r\}$

Exemplo 6 – Análise da Regra

* Sejam as regras
  * $P \rightarrow 0P0 \mid 1P1 \mid 0 \mid 1 \mid \lambda$
* Toda forma sentencial gerada por estas regras é da forma wPw^{r}
* Á última regra aplicada pode ser uma substituição de P por:
  * 0 – Neste caso o símbolo do meio será 0;
  * 1 – Neste caso o símbolo do meio será 1;
  * λ – Neste caso a palavra terá tamanho par.

Exemplo 7

* Seja a GLC $G = (\{P\}, \{0,1\}, R, P)$, cujas regras de R são:
    * $P \rightarrow 0P1P \mid 1P0P \mid \lambda$
* A Linguagem $L(G)$ é:

$L(G) = \{w \in \{0,1\}^* \mid w \text{ tem um número igual de 0s e 1s}\}$

Um olhar para o futuro

* Seja a GLC $G = (\{P\}, \{0,1\}, R, P)$, cujas regras de R são:

  * $P \rightarrow 0P1P \mid 1P0P \mid \lambda$: Gramática ambígua

* A Linguagem $L(G)$ é:

$$L(G) = \{w \in \{0,1\}^* \mid w \text{ tem um número igual de 0s e 1s}\}$$

Exemplo 7 – Análise da Regra

* Sejam as regras
  * $P \rightarrow 0P1P \mid 1P0P \mid \lambda$

* Toda forma sentencial gerada por estas regras tem a mesma quantidade de 0s e 1s, pois cada regra tem a mesma quantidade de 0s e 1s
  * G gera somente palavras de L

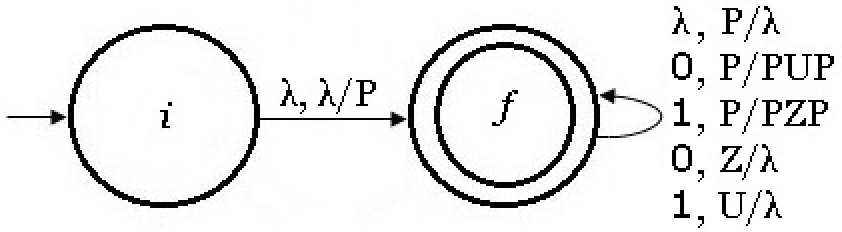
* Mas G produz todas as palavras de L?
  * Toda palavra de L é da forma:
    * $w = \lambda$ (regra 3)
    * $w = 0y$ e $y = x1z$ (regra 1)
    * $w = 1y$ e $y = x0z$ (regra 2)
      * x e z têm a mesma quantidade de 0s e 1s

GLCs e Autômatos de Pilha

- Seja a GLC $G = (\{P\}, \{0,1\}, R, P)$, cujas regras de R são:
  - $P \rightarrow 0P1P \mid 1P0P \mid \lambda$

- A Linguagem $L(G)$ é:
  - $L(G) = \{w \in \{0,1\}^* \mid w$ tem um número igual de 0s e 1s$\}$

- O AP correspondente à GLC é:



GLCs e Autômatos de Pilha

* Regras para a vida:

“É sempre possível obter uma GLC que gera a linguagem reconhecida por um AP.”

Exemplo 8

* Seja a GLC $G = (\{E,T,F\}, \{t,+,*,(,)\}, R, E)$, cujas regras de R são:

    * $E \longrightarrow E+T \mid T$
    * $T \longrightarrow T * F \mid F$
    * $F \longrightarrow (E) \mid \text{t}$

* A Linguagem $L(G)$ é uma linguagem para expressões aritméticas

* Essa gramática G contém a essência da especificação da sintaxe das expressões aritméticas das linguagens de programação usuais

Exemplo 8 – Análise da Regra

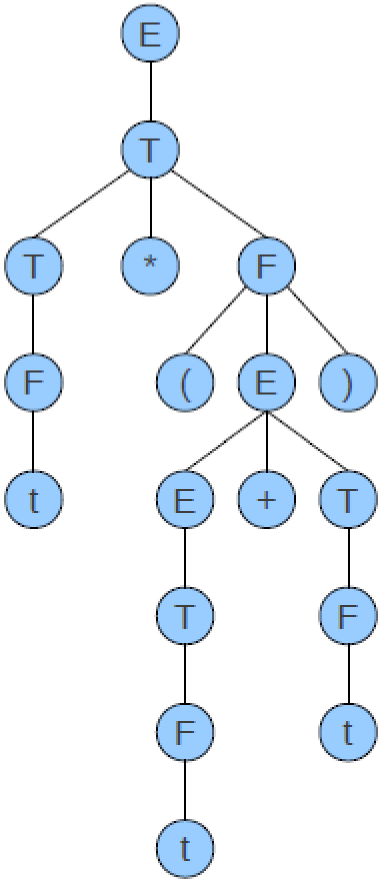
* Sejam as regras
  * $E \longrightarrow E{+}T \mid T$
    * E : Expressão aritmética
      * Um ou mais termos somados
  * $T \longrightarrow T{*}F \mid F$
    * T : Termo
      * Um ou mais fatores multiplicados
  * $F \longrightarrow (E) \mid t$
    * F : Fator
      * Um terminal ou expressão entre parênteses

Árvores de Derivação

- Um tipo de grafo muito comum em computação é a árvore. Uma árvore pode ser definida como sendo um grafo acíclico conexo. Na maioria das aplicações, existe um vértice especial denominado raiz .

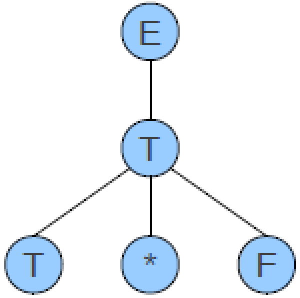
- A árvore de derivação guarda a história da derivação de uma forma sentencial a partir de uma gramática
  - Duas derivações são equivalentes se correspondem à mesma AD
- A cada derivação corresponde uma única AD
  - Uma AD, geralmente, corresponde a várias derivações

Exemplo de Derivação



- t * (t + t)
  - 11 passos de derivação
    - Número de vértices internos

Exemplo de Derivação



- Neste instante, tem-se duas opções para continuar a derivação:

$$
\begin{aligned}
E &\Rightarrow T && \text{(regra } E \to T\text{)} \\
&\Rightarrow T * F && \text{(regra } T \to T * F\text{)} \\
&\Rightarrow F * F && \text{(regra } T \to \mathrm{F}\text{)}
\end{aligned}
$$

$$
\begin{aligned}
E &\Rightarrow T && \text{(regra } E \to T\text{)} \\
&\Rightarrow T * F && \text{(regra } T \to T * F\text{)} \\
&\Rightarrow T * (E) && \text{(regra } F \to (E)\text{)}
\end{aligned}
$$

- À esquerda, a primeira, e à direita, a segunda.

Ambiguidade de GLCs

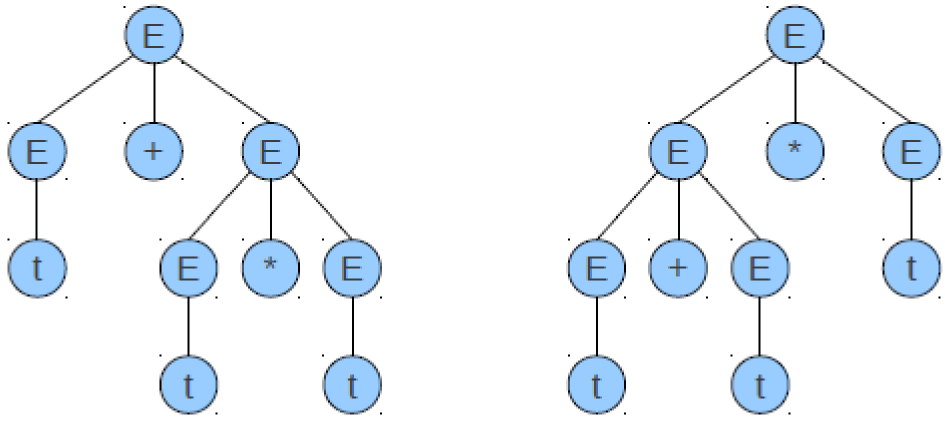
* Uma GLC é denominada ambígua quando existe mais de uma AD para alguma sentença que ela gera

* Observe, no entanto, que a <u>gramática</u> é dita ambígua, não a <u>linguagem</u> que ela gera nem as <u>sentenças</u> para as quais haja mais de uma AD

* Uma linguagem livre de contexto é inerentemente ambígua se somente pode ser gerada por gramáticas ambíguas

Exemplo 9

- Seja a GLC G = ({E}, {t,+,*,(,)}, R, E), cujas regras de R são:
  - $E \rightarrow E+E \mid E*E \mid (E) \mid t$
  - Derivar t + t * t



Exemplo 9

* Primeira árvore, derivação:

$$\begin{aligned}
E &\Rightarrow E + E && \text{(regra } E \to E + E\text{)} \\
&\Rightarrow \mathrm{t} + E && \text{(regra } E \to \mathrm{t}\text{)} \\
&\Rightarrow \mathrm{t} + E * E && \text{(regra } E \to E * E\text{)} \\
&\Rightarrow \mathrm{t} + \mathrm{t} * E && \text{(regra } E \to \mathrm{t}\text{)} \\
&\Rightarrow \mathrm{t} + \mathrm{t} * \mathrm{t} && \text{(regra } E \to \mathrm{t}\text{)}
\end{aligned}$$

* Segunda árvore, derivação:

$$\begin{aligned}
E &\Rightarrow E * E && \text{(regra } E \to E * E\text{)} \\
&\Rightarrow E + E * E && \text{(regra } E \to E + E\text{)} \\
&\Rightarrow \mathrm{t} + E * E && \text{(regra } E \to \mathrm{t}\text{)} \\
&\Rightarrow \mathrm{t} + \mathrm{t} * E && \text{(regra } E \to \mathrm{t}\text{)} \\
&\Rightarrow \mathrm{t} + \mathrm{t} * \mathrm{t} && \text{(regra } E \to \mathrm{t}\text{)}
\end{aligned}$$

Derivações

* DME – Derivação mais à Esquerda
  * Em cada passo, expande-se a variável mais à Esquerda

* DMD – Derivação mais à Direita
  * Em cada passo, expande-se a variável mais à Direita

Derivações

* Uma derivação é dita mais à esquerda (DME) se em cada passo é expandida a variável mais à esquerda.

* É dita mais à direita (DMD) se em cada passo é expandida a variável mais à direita.

Derivações

* Existe uma única DME e uma única DMD correspondentes a uma AD:

* Para obter a DME a partir de uma AD, basta ir gerando os passos de derivação à medida em que se percorre a AD visitando primeiro as subárvores à esquerda, antes de visitar as subárvores à direita;

* Para obter a DMD, visita-se primeiro as subárvores à direita.

Derivações

* Assim sendo, pode-se dizer que:

* uma GLC é ambígua se, e somente se, existe mais de uma DME para alguma sentença que ela gere;

* Uma GLC é ambígua se, e somente se, existe mais de uma DMD para alguma sentença que ela gere;

Derivações

- No caso do Exemplo 9, haviam duas DMEs para a mesma sentença t + t * t

Exercícios

* Construa GLCs para as seguintes linguagens:

* $\{0^n 1^n \mid n >= 0\}$ U $\{0^n 1^{2n} \mid n >= 0\}$;
* $\{0^m 1^n \mid m >= n\}$;
* $\{0^m 1^n \mid m > n\}$;

Exercícios

- Construa GLCs para as seguintes linguagens:

- $\{0^n1^n \mid n >= 0\} \cup \{0^n1^{2n} \mid n >= 0\}$;

Para $\{0^n1^n \mid n \geq 0\} \cup \{0^n1^{2n} \mid n \geq 0\}$:

$$
\begin{aligned}
P &\rightarrow A \mid B \\
A &\rightarrow 0A1 \mid \lambda \\
B &\rightarrow 0B11 \mid \lambda
\end{aligned}
$$

Exercícios

- Construa GLCs para as seguintes linguagens:

- $\{0^m1^n \mid m >= n\}$;

Para $\{0^m1^n \mid m >= n\}$:

$$P \rightarrow 0P1 \mid 0P \mid 0 \mid \lambda$$

Exercícios

* Construa GLCs para as seguintes linguagens:

* $\{0^m 1^n \mid m > n\}$;

Para $\{0^m 1^n \mid m > n\}$:

$$P \rightarrow 0P1 \mid 0P \mid 0$$

Exercícios

* Seja a gramática G, cujas regras são:

$$P \rightarrow \mathrm{a}P\mathrm{b} \mid \mathrm{aa}P\mathrm{b} \mid \lambda$$

* Mostre que G é ambígua.
* Construa uma gramática não ambígua equivalente a G.

* A gramática G gera a linguagem:

$$L = \{\mathrm{a}^n\mathrm{b}^k \mid k <= n <= 2k\}$$

Exercícios

* Seja a gramática G, cujas regras são:

$$P \rightarrow \mathrm{a}P\mathrm{b} \mid \mathrm{aa}P\mathrm{b} \mid \lambda$$

* Mostre que G é ambígua.

* G é ambígua, pois existem duas derivações mais à esquerda da palavra aaabb;
* $P \Rightarrow \mathrm{a}P\mathrm{b} \Rightarrow \mathrm{aaa}P\mathrm{bb} \Rightarrow \mathrm{aaabb}$
* $P \Rightarrow \mathrm{aa}P\mathrm{b} \Rightarrow \mathrm{aaa}P\mathrm{bb} \Rightarrow \mathrm{aaabb}$

Exercícios

* Seja a gramática G, cujas regras são:

$$P \rightarrow \mathrm{a}P\mathrm{b} \mid \mathrm{aa}P\mathrm{b} \mid \lambda$$

* Construa uma gramática não ambígua equivalente a G.

* $P \rightarrow \mathrm{a}P\mathrm{b} \mid X$
* $X \rightarrow \mathrm{aa}X\mathrm{b} \mid \lambda$
* Não é possível trocar X por aPb. Dessa forma, tirou-se a ambiguidade.

Primeira Técnica de Projeto para GLCs

- GLC da união de Linguagens
  - $L = L_1 \cup L_2$
    - Construa a Gramática G_1 para a Linguagem L_1, com a variável de partida sendo P_1;
    - Construa a Gramática G_2 para a Linguagem L_2, com a variável de partida sendo P_2;
    - Combine as duas GLCs em uma GLC com variável de partida P e todas as regras das gramáticas G_1 e G_2, adicionando-se a regra:
      - $P \rightarrow P_1 \mid P_2$
    - Cuidado!! Para esta técnica funcionar, não pode haver variáveis repetidas entre as gramáticas, ou seja:

$$V_1 \cap V_2 = \emptyset$$

Exemplo 10

- GLC para a linguagem:

  $\{0^n1^n \mid n >= 0\} \cup \{0^n1^{2n} \mid n >= 0\}$;

- Regras para a primeira linguagem:
  - $P_1 \rightarrow 0P_11 \mid \lambda$
- Regras para a segunda linguagem:
  - $P_2 \rightarrow 0P_211 \mid \lambda$
- Regra para a GLC da união:
  - $P \rightarrow P_1 \mid P_2$

Segunda Técnica de Projeto para GLCs

- GLC da união de Linguagens
  - $L = L_1L_2$
    - Construa a Gramática G_1 para a Linguagem L_1, com a variável de partida sendo P_1;
    - Construa a Gramática G_2 para a Linguagem L_2, com a variável de partida sendo P_2;
    - Combine as duas GLCs em uma GLC com variável de partida P e todas as regras das gramáticas G_1 e G_2, adicionando-se a regra:
      - $P \rightarrow P_1P_2$
    - Cuidado!! Para esta técnica funcionar, não pode haver variáveis repetidas entre as gramáticas, ou seja:

$$V_1 \cap V_2 = \emptyset$$

Exemplo 11

* GLC para a linguagem:

$$\{0^n1^n \mid n >= 0\}\{0^n1^{2n} \mid n >= 0\};$$

* Regras para a primeira linguagem:
  * $P_1 \rightarrow 0P_11 \mid \lambda$
* Regras para a segunda linguagem:
  * $P_2 \rightarrow 0P_211 \mid \lambda$
* Regra para a GLC da união:
  * $P \rightarrow P_1P_2$

Exercícios

* Construa GLCs para as seguintes linguagens:

* $L_1 = \{0^n 1^k \mid 2n <= k <= 3n\}$;
* $L_2 = \{a^n b^k c^m \mid k = 2n + m\}$;
* $L_3 = (L_1 \cup L_2)^2$.

Obrigado.

joaopauloaramuni@gmail.com
joaopauloaramuni@fumec.br

UNIVERSIDADE FUMEC

50 Anos