Gramáticas - II

O texto apresentado neste documento é uma adaptação de:

Newton Vieira, 2006. Introdução aos fundamentos da computação: Linguagens e máquinas. CENGAGE Learning.

Na aula anterior, vimos a definição de Gramática e de um tipo especial, chamado Gramática Livre de Contexto, em que todas as regras de produção têm exatamente um único símbolo não terminal no lado esquerdo.

Um dos exemplos apresentados é repetido abaixo, como motivação para o próximo tópico a ser estudado.

Seja a gramática G = ({P}, {0, 1}, R, P), onde R é formado pelas 3 regras a seguir:

$$P \rightarrow 0P1P \mid 1P0P \mid \lambda$$

Vimos que L(G) = {w ∈ {0,1}* | w tem um número igual de 0's e 1's}. Uma derivação para a palavra 01010110 é apresentada abaixo:

$$\begin{aligned} P &\Rightarrow 0P1P \\ &\Rightarrow 01P0P1P \\ &\Rightarrow 010P1P \\ &\Rightarrow 0101P \\ &\Rightarrow 01010P1P \\ &\Rightarrow 010101P \\ &\Rightarrow 0101011P0P \\ &\Rightarrow 01010110P \\ &\Rightarrow 01010110 \end{aligned}$$

Em geral, uma palavra da linguagem pode ser gerada por várias derivações diferentes. No exemplo, inicia-se uma derivação de 01010110 por

$$P \Rightarrow 0P1P \quad (\text{regra } P \rightarrow 0P1P).$$

Em seguida, a variável expandida é sempre a mais à esquerda. Pode-se ver que a mesma palavra pode ser derivada expandindo-se sempre a variável mais à direita ao invés da variável mais à esquerda. E mais: a mesma palavra pode ser derivada expandindo-se variáveis em ordem aleatória. Isto mostra que existem várias derivações distintas para a palavra 01010110. O que tais derivações têm em comum? Vamos abordar esse assunto na sequência dos estudos.

1. Derivações e Ambiguidade

Um conceito bastante útil, base para muitas implementações de compiladores de linguagens de programação, é o de árvore de derivação (AD). Uma AD captura a essência de uma derivação, a história da obtenção de uma forma sentencial que não depende da ordem de aplicação das regras da GLC. A cada derivação irá corresponder uma única AD, mas a uma AD irá corresponder, quase sempre, uma quantidade muito grande de possíveis derivações.

Assim, pode-se dizer que as AD's particionam o conjunto de todas as derivações de uma GLC em "derivações equivalentes": duas derivações seriam equivalentes se, e somente se, correspondessem à mesma AD.

Definição 1.1

Seja uma GLC G = (V, Σ,R, P). Uma árvore de derivação (AD) de uma forma sentencial de G é uma árvore ordenada construída recursivamente como segue:

(a) uma árvore sem arestas cujo único vértice tem rótulo P é uma AD de P;

(b) se X ∈ V é rótulo de uma folha f de uma AD A, então:

- se $X \rightarrow \lambda \in R$, então a árvore obtida acrescentando-se a A mais um vértice v com rótulo λ e uma aresta {f, v} é uma AD;
- se $X \rightarrow x_1x_2 \ldots x_n \in R$, onde $x_1, x_2 \ldots, x_n \in V \cup \Sigma$, então a árvore obtida acrescentando-se a A mais n vértices $v_1, v_2, \ldots, v_n$ com rótulos $x_1, x_2 \ldots, x_n$, nesta ordem, e n arestas $\{f, v_1\}, \{f, v_2\}, \ldots, \{f, v_n\}$, é uma AD.

Se a sequência dos rótulos da fronteira da AD é a forma sentencial w, diz-se que a AD é uma árvore de derivação de w.

Exemplo 1.1

Seja a gramática do exemplo 2.4 da aula anterior, cujas regras são reproduzidas abaixo:

$$E \rightarrow E{+}T \mid T$$

$$T \rightarrow T{*}F \mid F$$

$$F \rightarrow (E) \mid \mathtt{t}$$

A Figura 1.1 mostra uma AD de t*(t+t). Para a construção de tal árvore, tomou-se como ponto de partida a derivação:

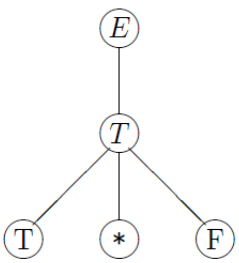
$$E \Rightarrow T \qquad (\text{regra } E \rightarrow T)$$

produzindo-se:

Em seguida, a derivação evoluiu para:

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

e a árvore correspondente para:



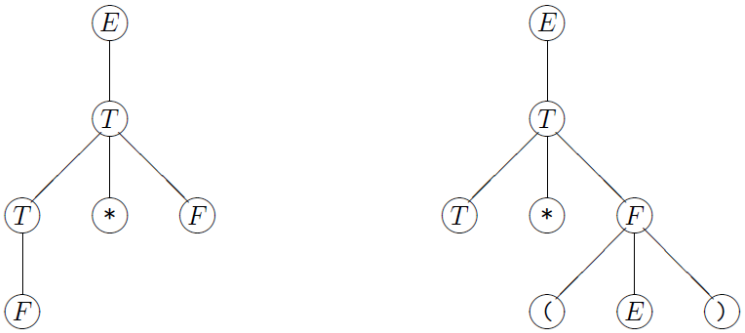
Nesse ponto, tinha-se duas opções para continuar a derivação:

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

ou então:

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

À esquerda, mostra-se a AD correspondente à primeira derivação, e à direita, a AD correspondente à segunda derivação:



Prosseguindo-se por qualquer uma destas alternativas, chega-se, após uma derivação de 11 passos, à AD mostrada na Figura 1.1. ❑

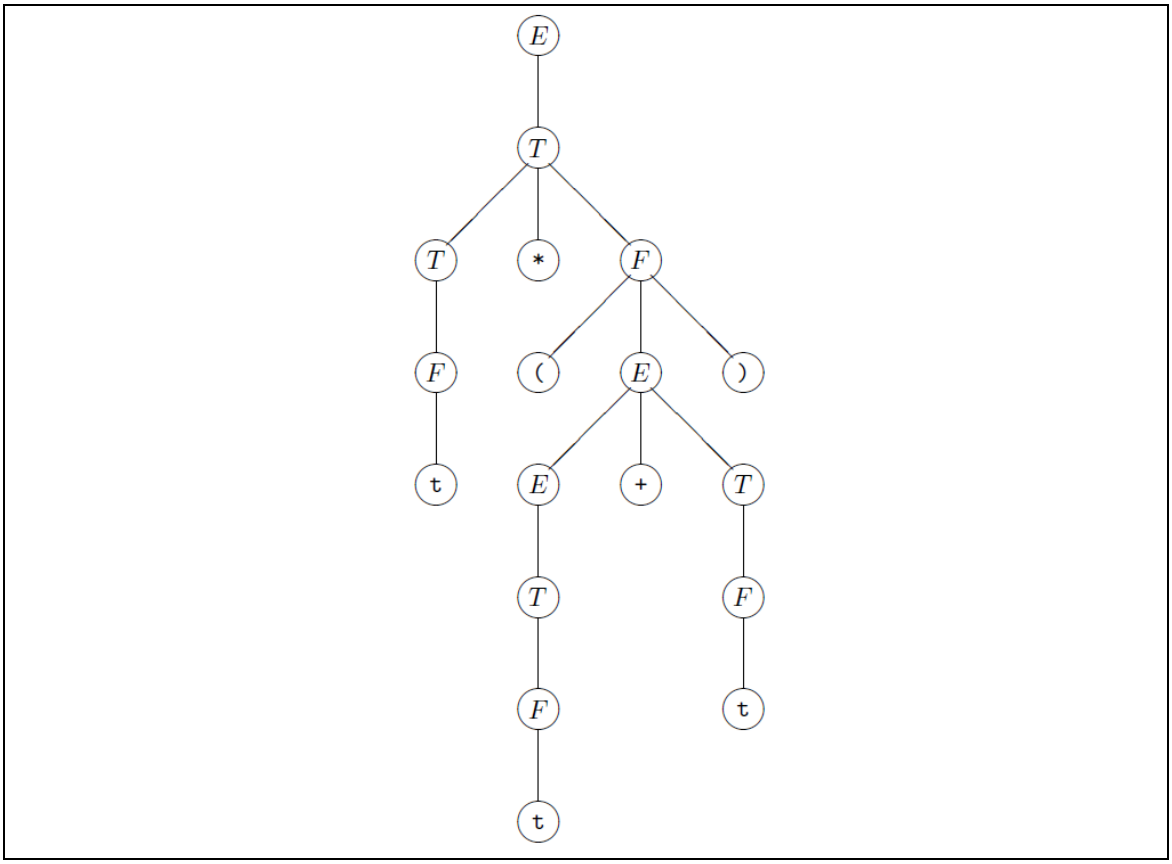


Figura 1.1: Uma árvore de derivação.

O número de passos de qualquer derivação que leva a uma AD X é o número de vértices internos de X, já que a cada vértice interno corresponde a aplicação de uma regra (e vice-versa).

Muitas vezes, a estrutura da árvore de derivação é utilizada para associar significado para as sentenças de uma linguagem, de forma similar ao que se faz em análise sintática de sentenças na língua portuguesa (onde se identifica sujeito, verbo, predicado, etc.). Em português, se a mesma sentença pode ser desmembrada de mais de uma forma durante a análise, então ela tem vários significados, e diz-se que ela é ambígua. De forma análoga, se existir mais de uma AD de uma mesma palavra, ela poderá mais de um significado. Isto inspira a definição a seguir.

Dada uma árvore de derivação, é simples identificar qual é a palavra derivada. Basta concatenar os símbolos das folhas da árvore, realizando um caminhamento em profundidade da esquerda para a direita. A Figura 1.2 mostra a mesma AD da Figura 1.1, ressaltando os símbolos das folhas que, conectados, formam a palavra t*(t+t).

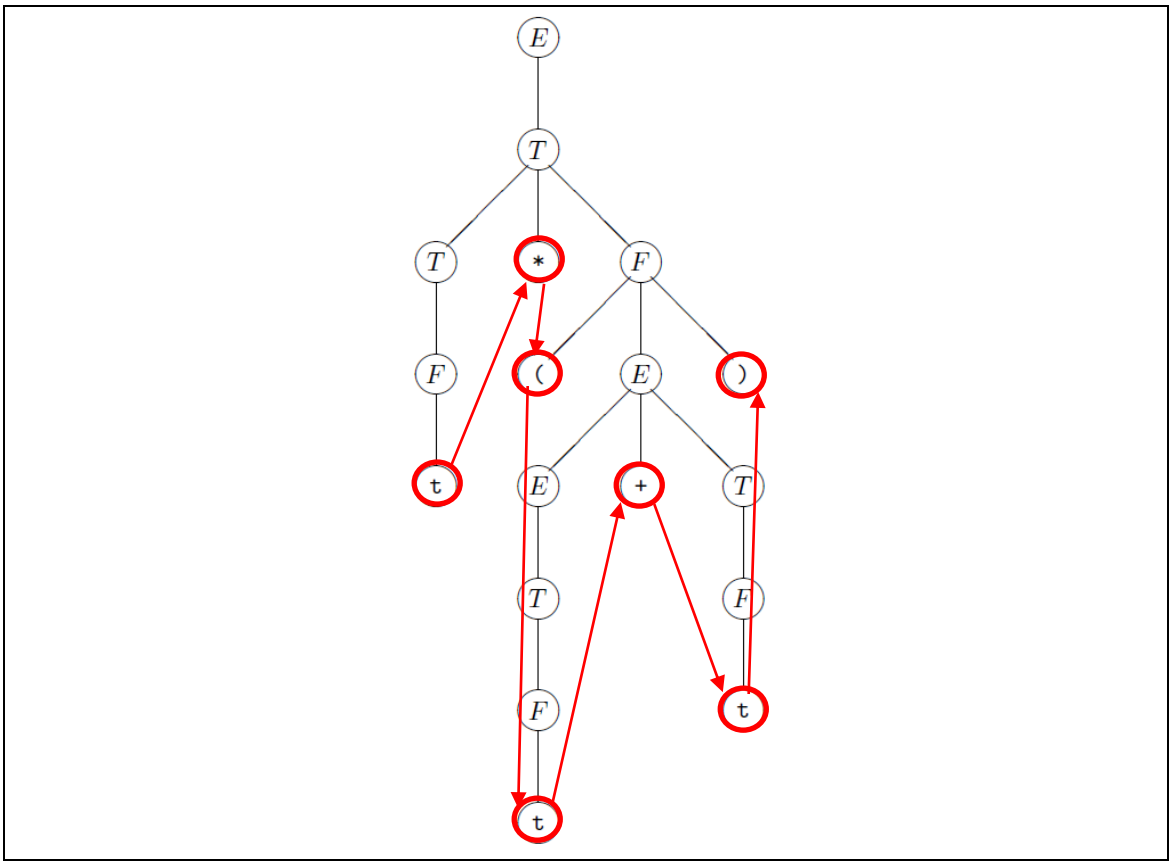


Figura 1.2: Palavra associada à árvore de derivação.

Definição 1.2

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

Observe que é a gramática que é dita ambígua. Não é a linguagem que ela gera, nem as sentenças para as quais haja mais de uma AD. Afinal, pode haver outras GLC's equivalentes a uma GLC ambígua que não sejam ambíguas.

A gramática do Exemplo 1.1 não é ambígua, e a do exemplo apresentado no início deste texto (mesmo número de 0's e 1's) é uma gramática ambígua. O exemplo a seguir apresenta uma gramática ambígua que gera a linguagem de expressões aritméticas, especificada no Exemplo 1.1 usando uma gramática não ambígua.

Exemplo 1.2

Seja a GLC ({E}, {t,+, _, (,)}, R, E), onde R é formado pelas regras:

$$E \to E{+}E \mid E{*}E \mid (E) \mid \mathtt{t}$$

Tal gramática é ambígua, já que existem duas AD's para a palavra t+t+t, as quais são exibidas na Figura 1.2. À árvore da esquerda corresponde, dentre outras, a derivação:

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

À árvore da direita corresponde a seguinte derivação, dentre outras:

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

Como já foi dito anteriormente, normalmente o significado é associado a uma palavra de acordo com a AD obtida. Por exemplo, a AD do lado esquerdo da Figura 1.3 leva à interpretação de t+t+t como sendo a soma de um elemento com a soma de dois elementos, isto é, t+(t+t), enquanto que a AD do lado direito da mesma figura leva à interpretação de t+t+t como sendo a soma da soma de dois elementos com um elemento, isto é, (t+t)+t. ❑

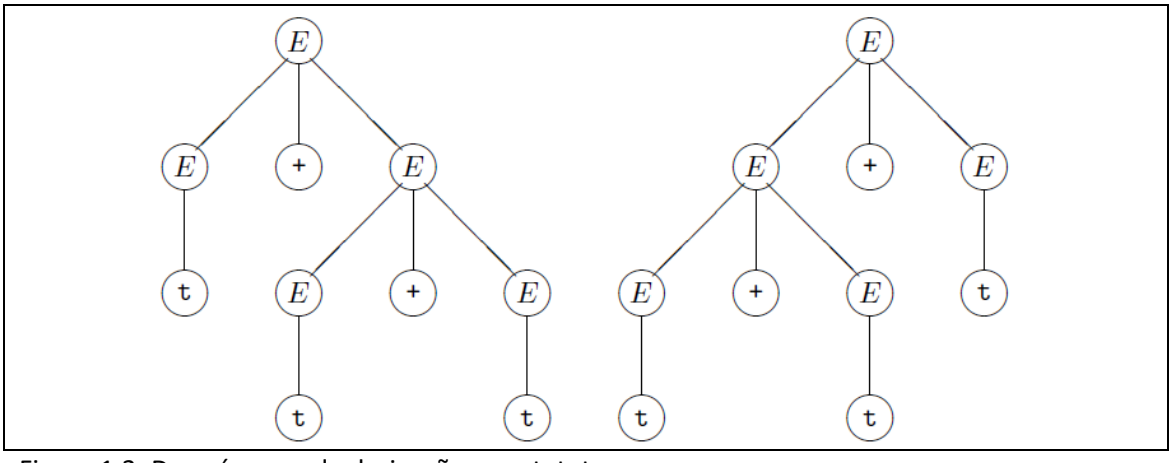


Figura 1.3: Duas árvores de derivação para t+t+t.

Indicamos que a gramática do Exemplo 1.2 é ambígua, mas que a gramática do Exemplo 1.1 não é ambígua, sendo que ambas geram a mesma linguagem. Novamente ressaltamos que associamos a propriedade de ambiguidade às **gramáticas**, e não às **linguagens**. Mas existem algumas linguagens para as quais é impossível definir alguma gramática que não seja ambígua. Para essas linguagens, dizemos que são **linguagens inerentemente ambíguas**. A linguagem gerada pelos exemplos 1.1 e 1.2 é a mesma. Essa linguagem obviamente não é inerentemente ambígua, pois foi possível apresentar uma gramática não ambígua que a gera.

Dentre as derivações correspondentes a uma AD, existem duas de particular interesse: as *derivações mais à esquerda* e as *derivações mais à direita*.

Definição 1.3

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

Existe uma única DME e uma única DMD correspondente 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.

Como existe essa correspondência um-a-um entre árvores de derivação e derivações mais à esquerda / mais à direita, podemos apresentar um conceito alternativo para ambiguidade de gramáticas:

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

e também que:

> *uma GLC é ambígua se, e somente se existe mais de uma DMD para alguma sentença que ela gera.*

Exemplo 1.3

No Exemplo 1.2, foram mostradas as duas DME's que correspondem às AD's da Figura 1.2.

A DMD para a primeira AD é:

$$
\begin{aligned}
E &\Rightarrow_D E{+}E && (\text{regra } E \rightarrow E{+}E)\\
&\Rightarrow_D E{+}E{+}E && (\text{regra } E \rightarrow E{+}E)\\
&\Rightarrow_D E{+}E{+}\texttt{t} && (\text{regra } E \rightarrow \texttt{t})\\
&\Rightarrow_D E{+}\texttt{t}{+}\texttt{t} && (\text{regra } E \rightarrow \texttt{t})\\
&\Rightarrow_D \texttt{t}{+}\texttt{t}{+}\texttt{t} && (\text{regra } E \rightarrow \texttt{t})
\end{aligned}
$$

e para a segunda AD:

$$
\begin{aligned}
E &\Rightarrow_D E{+}E && (\text{regra } E \to E{+}E)\\
&\Rightarrow_D E{+}\mathtt{t} && (\text{regra } E \to \mathtt{t})\\
&\Rightarrow_D E{+}E{+}\mathtt{t} && (\text{regra } E \to E{+}E)\\
&\Rightarrow_D E{+}\mathtt{t}{+}\mathtt{t} && (\text{regra } E \to \mathtt{t})\\
&\Rightarrow_D \mathtt{t}{+}\mathtt{t}{+}\mathtt{t} && (\text{regra } E \to \mathtt{t}).
\end{aligned}
$$

❑

Existem linguagens livres do contexto (LLC's) para as quais existem apenas gramáticas ambíguas. Tais linguagens são denominadas *linguagens inerentemente ambíguas*. Um exemplo de linguagem inerentemente ambígua é;

$$\{\mathtt{a}^m\mathtt{b}^n\mathtt{c}^k \mid m = n \text{ ou } n = k\}$$

Pode-se mostrar que qualquer GLC que gere tal linguagem terá mais de uma AD para palavras da forma $a^n b^n c^n$.

A detecção e remoção de ambiguidade em GLC's é muito importante, por exemplo, como um passo prévio ao uso de uma gramática para geração de um compilador para uma linguagem de programação. Mais tarde veremos algumas técnicas de modificação de GLC's, que permitem chegar a gramáticas com propriedades desejáveis, sem alterar a linguagem gerada. No entanto, infelizmente, o problema de determinar se uma GLC arbitrária é ambígua é indecidível.

2. Gramáticas Regulares

Vimos que as Gramáticas Livres de Contexto são um tipo especial de gramática cujas regras de produção têm exatamente um único símbolo terminal do lado esquerdo. Agora veremos outro tipo especial de gramática livre de contexto, chamado *Gramática Regular*.

Definição 2.1

Uma *Gramática Regular* (GR) é uma gramática (V, Σ ,R, P), em que cada regra tem uma das formas:

- X → a
- X → aY
- X → λ

onde X, Y ∈ V e a ∈ Σ.

Importante: toda gramática regular é também livre de contexto, por definição. Gramáticas regulares são um tipo especial de gramáticas livres de contexto.

Exemplo 2.1

Seja G uma gramática especificada pelas regras abaixo.

S → aA
A → aA | bA | cA | λ

Aplicando a simplificação que apresentamos na aula anterior para quando forem apresentadas apenas as regras de produção, podemos neste caso assumir que V = {S,A}, Σ = {a,b,c} e o símbolo de partida é S.

Pode-se ver que as 5 regras de produção de G atendem às restrições impostas para gramáticas regulares. Assim, G é uma gramática regular.

A linguagem gerada por G é o conjunto de palavras do alfabeto {a,b,c} que começam com 'a'. Exemplos de palavras derivadas:

$S \Rightarrow aA \Rightarrow a$

$S \Rightarrow aA \Rightarrow aaA \Rightarrow aacA \Rightarrow aac$

$S \Rightarrow aA \Rightarrow abA \Rightarrow abbA \Rightarrow abbbA \Rightarrow abbb$ ❑

Uma característica interessante das GR's, que pode ser observada nas derivações do Exemplo 2.1, é o formato das formas sentenciais geradas. Toda forma sentencial tem o formato wA, onde w só tem terminais e A é uma variável.

Exemplo 2.2

Seja G a gramática representada simplificadamente pelas regras a seguir:

$S \rightarrow aA \mid bA$
$A \rightarrow aS \mid bS \mid \lambda$

Pode-se ver que as 5 regras de produção de G atendem às restrições impostas para gramáticas regulares. Assim, G é uma gramática regular.

Exemplos de palavras derivadas:

$S \Rightarrow aA \Rightarrow a$

$S \Rightarrow aA \Rightarrow aaS \Rightarrow aabA \Rightarrow aab$

$S \Rightarrow aA \Rightarrow abS \Rightarrow abbA \Rightarrow abbbS \Rightarrow abbbbA \Rightarrow abbbb$

A cada símbolo terminal gerado, o não terminal no final da forma sentencial se alterna entre S e A. As formas sentencias que terminam com S sempre têm um número par de terminais e as formas sentenciais que terminam com A sempre têm um número de terminais ímpar. Usar a regra $A \rightarrow \lambda$ é a única forma de se encerrar uma sequência de derivações nessa gramática. Assim, a linguagem gerada são as palavras sobre o alfabeto {a,b} com comprimento ímpar. ❑

Exemplo 2.3

Seja G a gramática regular representada simplificadamente pelas regras a seguir:

$A \rightarrow aB \mid bA \mid cA \mid \lambda$
$B \rightarrow aB \mid bC \mid cA \mid \lambda$
$C \rightarrow aB \mid bA \mid \lambda$

A linguagem aceita são as palavras sobre {a,b,c} que não contêm abc.

As formas sentenciais que terminam com B indicam que o último terminal produzido foi ‘a’. Neste estado, se for produzido em seguida um ‘b’, as formas sentencias irão terminar com C. Observe que as regras de C não permitem que seja produzido um símbolo ‘c’, assim a sequência abc nunca será produzida. ❑

Vimos que as gramáticas livres de contexto definem restrições sobre o formato das regras de uma gramática, exigindo que o lado esquerdo das regras seja formado exatamente por um símbolo terminal. E que as gramáticas regulares definem restrições adicionais sobre o formato das regras. Mais tarde, veremos que essas restrições reduzem o poder de expressividade dos modelos. As linguagens livres de contexto, geradas pelas gramáticas livres de contexto, são um subconjunto próprio das linguagens que podem ser geradas por gramáticas sem restrição nas regras. As linguagens geradas pelas gramáticas regulares são exatamente as linguagens regulares (definidas por operações de conjuntos regulares), e são um subconjunto próprio das linguagens livres de contexto.

Exemplo 2.4

Seja G a gramática representada simplificadamente pelas regras a seguir:

S → aSb | λ

A linguagem aceita é { $a^n b^n$, para n ≥ 0 }.

A gramática G é livre de contexto, assim a linguagem gerada é livre de contexto.

A gramática G não é regular, pois a regra S → aSb não satisfaz as restrições de gramáticas regulares. Na realidade, é impossível construir uma gramática regular que gere a linguagem {$a^n b^n$, para n ≥ 0}. Essa é uma das linguagens livres de contexto que não é regular. Mais tarde, vamos demonstrar isso formalmente e apresentar um método que facilita a demonstração de que uma linguagem não é regular. ❑

Definição 2.2

Uma linguagem é dita ser uma *linguagem regular* se existe uma gramática regular que a gera.

Podemos dizer que a linguagem das palavras sobre o alfabeto {a,b} com comprimento ímpar é uma linguagem regular, já que foi possível apresentar uma gramática regular que gera essa linguagem, no Exemplo 2.2. Os exemplos 2.1 e 2.3 também apresentam linguagens regulares. A linguagem {$a^n b^n$, para n ≥ 0} é um exemplo de linguagem livre de contexto que não é regular.