Linguagens Formais
Capítulo 5: Linguagens e gramáticas livres de contexto

José Lucas Rangel, maio 1999

5.1 - Introdução

Vimos no capítulo 3 a definição de gramática livre de contexto (glc) e de linguagem livre de contexto (llc). As regras de uma glc são da forma $A \rightarrow \alpha$, onde α é uma cadeia qualquer de terminais e nãoterminais, possivelmente vazia. Como vimos, o que caracteriza a gramática livre de contexto é a propriedade de que o símbolo não terminal A pode ser substituído pela cadeia α do lado direito da regra, onde quer que A ocorra, independentemente do contexto, isto é, do resto da cadeia que está sendo derivada. Por essa razão, é possível representar derivações em glc's através de árvores de derivação: para usar a regra $A \rightarrow \alpha$, acrescentamos à árvore, como filhos de A, nós correspondentes aos símbolos de α.

5.2 - Árvores de derivação

Uma árvore de derivação é uma árvore composta da seguinte maneira:

- a raiz tem como rótulo o símbolo inicial S da gramática.
- a cada nó rotulado por um nãoterminal A corresponde uma regra de A. Se a regra for $A \rightarrow X_1X_2 \ldots X_m$, os filhos do nó são rotulados, da esquerda para a direita, por X_1, X_2, ..., X_m. (cada um dos X_i pode ser um terminal ou um nãoterminal.)
- um nó rotulado por um terminal é sempre uma folha da árvore, e não tem filhos.

Os nós interiores da árvore são sempre (rotulados por) nãoterminais. Se a um nó rotulado pelo nãoterminal A for aplicada a regra $A \rightarrow \varepsilon$, considera-se o nó como interior, embora ele tenha zero filhos. Na representação gráfica da árvore, é costume indicar, neste caso, os zero filhos através de um nó ε, que não contribui para o resultado da árvore.

Uma sub-árvore de uma árvore de derivação é um nó da árvore com todos seus descendentes, as arestas que os conectam e seus rótulos. Uma sub-árvore sempre corresponde a uma derivação parcial, a partir do símbolo que rotula a raiz da sub-árvore. O resultado de uma árvore de derivação é a cadeia formada pelos terminais (que aparecem como folhas da árvore), lidos da esquerda para a direita. Note que a ordenação dos filhos de cada nó é fundamental para que se possa definir a ordenação das folhas da árvore.

Exemplo 5.1: Seja a gramática G, dada por suas regras:

$S \rightarrow AB$
$A \rightarrow aaA \mid \varepsilon$
$B \rightarrow Bbb \mid \varepsilon$

A árvore de derivação para a sequência `aaaabb` está representada na Fig. 1. O resultado da árvore é `aaaabb`, sendo as regras escolhidas de acordo com a derivação

$S \Rightarrow B \Rightarrow aaAB \Rightarrow aaaaAB \Rightarrow aaaaB \Rightarrow aaaaBbb \Rightarrow aaaabb.$

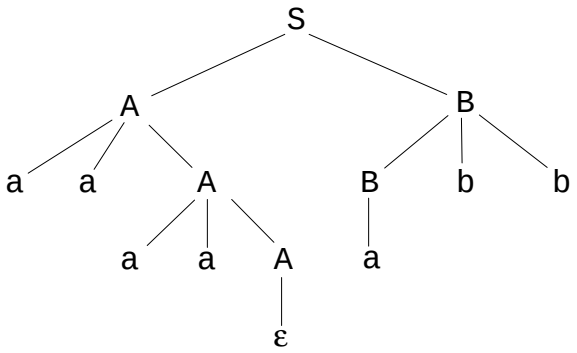

Fig. 1 - Árvore de derivação de aaaabb

Teorema 5.1: Seja G = (N, Σ, P, S) uma glc. Então, para qualquer cadeia x ∈ Σ*,

x ∈ L(G) se e somente se existe uma árvore de derivação A na gramática G, cujo resultado é x.

Demonstração: Basta observar a correspondência entre a substituição de nãoterminais pelos lados direitos de suas regras na derivação e a criação dos filhos correspondentes na árvore. Usando essa correspondência é possível construir uma árvore de derivação cujo resultado é x a partir de uma derivação S ⇒* x; é possível também construir uma derivação S ⇒* x a partir de uma árvore de derivação cujo resultado é x. Os detalhes da demonstração são deixados como exercício.

Observação. A partir de uma derivação. só é possível construir uma árvore; entretanto, na direção oposta, é possível construir várias derivações, dependendo da ordem em que os nós são considerados. O Exemplo 5.2 mostra algumas das várias derivações que correspondem à mesma árvore.

Derivações esquerdas e direitas.

Diz-se que uma derivação é uma derivação esquerda (*leftmost derivation*) se a cada passo da derivação o nãoterminal A escolhido para aplicação de uma regra A → α for sempre aquele que fica mais à esquerda. Simetricamente, fala-se em derivação direita (*rightmost derivation*) se o nãoterminal escolhido é sempre o que estiver mais à direita.

Exemplo 5.2: Seja a gramática G_0 abaixo, dada por suas regras. (Esta gramática será usada em vários exemplos, no que se segue.

```
E → E + T | T
T → T * F | F
F → ( E ) | a
```

Considere a cadeia x = a*(a+a)+a. Temos abaixo três derivações de:

```
E ⇒ E+T ⇒ T+T ⇒ T*F+T ⇒ F*F+T ⇒ a*F+T ⇒ a*(E)+T
      ⇒ a*(E+T)+T ⇒ a*(T+T)+T ⇒ a*(F+T)+T ⇒ a*(a+T)+T
      ⇒ a*(a+F)+T ⇒ a*(a+a)+T ⇒ a*(a+a)+F ⇒ a*(a+a)+a
E ⇒ E+T ⇒ E+F ⇒ E+a ⇒ T+a ⇒ T*F+a ⇒ T*(E)+a
      ⇒ T*(E+T)+a ⇒ T*(E+F)+a ⇒ T*(E+a)+a ⇒ T*(T+a)+a
      ⇒ T*(F+a)+a ⇒ T*(a+a)+a ⇒ F*(a+a)+a ⇒ a*(a+a)+a
E ⇒ E+T ⇒ T+T ⇒ T+F ⇒ T*F+F ⇒ T*F+a ⇒ F*F+a
      ⇒ F*(E)+a ⇒ a*(E)+a ⇒ a*(E+T)+a ⇒ a*(T+T)+a
      ⇒ a*(T+F)+a ⇒ a*(F+F)+a ⇒ a*(a+F)+a ⇒ a*(a+a)+a
```

Note que a primeira derivação é uma derivação esquerda, e a segunda é uma derivação direita. Todas as três derivações correspondem à mesma árvore de derivação, apresentada na Figura 2. Como se pode observar, em todas elas aparecem as mesmas regras, aplicadas nos mesmos lugares, variando apenas a ordem em que as regras são aplicadas.

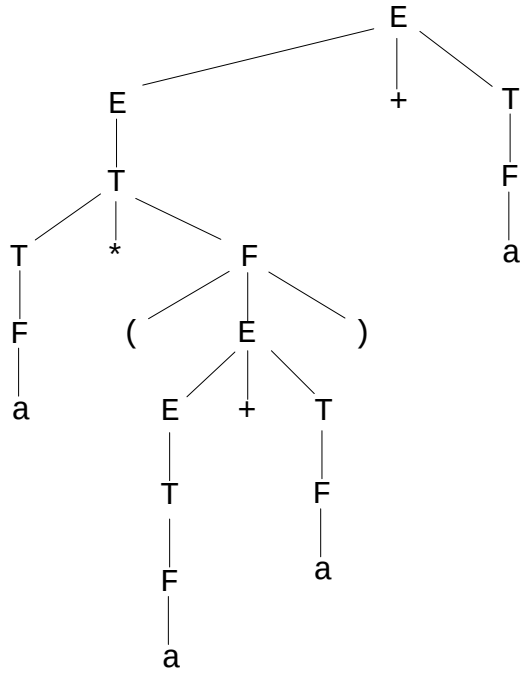

Fig. 2- Árvore de derivação de a*(a+a)+a

Como todas as derivações correspondentes à mesma árvore de derivação descrevem a mesma forma de construção da cadeia derivada - *as mesmas regras aplicadas nos mesmos lugares* - consideramos que a forma de construção da cadeia pode ser representada pela árvore ou por uma derivação esquerda, ou por uma derivação direita. Entretanto, se existem duas ou mais árvores de derivação (duas ou mais derivações esquerdas, duas ou mais derivações direitas), para a mesma cadeia, consideramos que a gramática não define de forma única a maneira pela qual a cadeia é derivada, e dizemos que a gramática é *ambígua*.

Exemplo 5.3: Seja a gramática G_1, dada por suas regras:

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

Pode-se verificar que G_1 é equivalente a G_0, vista no exemplo 5.2 acima. Entretanto, diferentemente de G_0, G_1 é uma gramática ambígua. Considere, por exemplo a cadeia a+a*a. As duas derivações (esquerdas) abaixo correspondem a duas árvores de derivação distintas.

$E \Rightarrow E+E \Rightarrow a+E \Rightarrow a+E*E \Rightarrow a+a*E \Rightarrow a+a*a$

$E \Rightarrow E*E \Rightarrow E+E*E \Rightarrow a+E*E \Rightarrow a+a*E \Rightarrow a+a*a$

A construção das duas árvores fica como exercício.

Uma linguagem livre de contexto cujas gramáticas são todas ambíguas é chamada uma gramática inerentemente ambígua.

Exemplo 5.4: Sejam $L_1 = \{a^i b^j c^k \mid i=j\}$ e $L_2 = \{a^i b^j c^k \mid j=k\}$. A linguagem L definida por $L = L_1 \cup L_2$ é inerentemente ambígua. As linguagens L_1 e L_2 não são inerentemente ambíguas, como se pode ver pelas suas respectivas gramáticas G_1 e G_2.

G_1: $S_1 \rightarrow T\ C$ $\quad$ G_2: $S_2 \rightarrow A\ V$

$T \rightarrow a\ T\ b \mid \varepsilon$ $\quad$ $A \rightarrow a\ A \mid \varepsilon$

$C \rightarrow c\ C \mid \varepsilon$ $\quad$ $V \rightarrow b\ V\ c \mid \varepsilon$

É fácil construir um exemplo G de gramática ambígua para L, a partir de G_1 e G_2:

G: $S \rightarrow S_1 \mid S_2$

$S_1 \rightarrow T\ C$ $\quad$ $S_2 \rightarrow A\ V$

$T \rightarrow a\ T\ b \mid \varepsilon$ $\quad$ $A \rightarrow a\ A \mid \varepsilon$

$C \rightarrow c\ C \mid \varepsilon$ $\quad$ $V \rightarrow b\ V\ c \mid \varepsilon$

Para verificar que G é ambígua, basta observar que todas as cadeias pertencentes à interseção $M = L_1 \cap L_2 = \{a^i\ b^j\ c^k \mid i=j=k\ \}$ podem ser derivadas de duas formas distintas, dependendo da regra inicial escolhida para a derivação. Por exemplo, aabbcc pode ser obtida por uma das duas derivações esquerdas abaixo:

$S \Rightarrow S_1 \Rightarrow TC \Rightarrow aTbC \Rightarrow aaTbbC \Rightarrow aabbC \Rightarrow aabbcC$
$\Rightarrow aabbccC \Rightarrow aabbcc$

$S \Rightarrow S_2 \Rightarrow AV \Rightarrow aAV \Rightarrow aaAV \Rightarrow aaV \Rightarrow aabVc \Rightarrow aabbVcc$
$\Rightarrow aabbcc$

Naturalmente, isto prova apenas que G é ambígua, e não que todas as gramáticas de L são ambíguas. Esta demonstração não será incluída aqui.

Observamos também que M não é uma llc. Isto será visto no Exemplo 5.7.

5.3 - Simplificação de gramáticas livres de contexto

Não existe a possibilidade de simplificação de gramáticas livres de contexto, no mesmo sentido da minimização de automatos finitos vista anteriormente. É, entretanto possível fazer uma simplificação que elimina todos os símbolos e regras inúteis da gramática. Podemos dizer que um símbolo terminal é *inútil* quando não aparece em alguma cadeia da linguagem; podemos dizer que um símbolo não terminal é *inútil* quando não aparece em alguma derivação de alguma cadeia da linguagem. Uma regra é *inútil* contém algum símbolo inútil.

Em alguns casos, a gramática é ambígua, e algumas regras podem ser removidas sem que a linguagem se altere, mas pode não ficar claro qual regra deve ser considerada inútil. Por exemplo, se tivermos

1, 2. $S \rightarrow A\ X \mid Y\ C$

3. $X \rightarrow B\ C$

4. $Y \rightarrow A\ B$

há duas maneiras de gerar ABC a partir de S. Quais as regras que devem ser retiradas? 1 e 3 ou 2 e 4? Tanto faz.

Todas as simplificações que podem ser feitas, entretanto, não alteram a essência de uma gramática: apenas a tornam mais limpa. Algumas transformações de gramáticas visam obter uma gramática equivalente à inicial que tem alguma forma particular, por exemplo, que simplifique a demonstração de algum teorema. Para ver alguns algoritmos para simplificação ou transformação de gramáticas sugerimos a mesma referência citada anteriormente.

O exemplo a seguir procura esclarecer alguns dos conceitos mencionados.

Exemplo 5.5: Uma gramática com regras e símbolos inúteis. Seja a gramática

1, 2, 3: S → A B | A C | B D
4, 5: A → a A | a
6, 7: B → b B | b
8, 9: C → c D | d C
10, 11: Y → y | z Z
12, 13: Z → z | y Y

Símbolos não terminais acessíveis:

Em derivações a partir de S podem aparecer S A B C D (regras 1, 2, 3).

Símbolos não terminais produtivos:

Derivações que levam a cadeias de terminais: A (regra 5), B (regra 7), Y (regra 10), Z (regra 11), S (regra 1, já que A e B são produtivos.)

Logo, todos os nãoterminais, exceto S A B são inúteis. Retirando todas as regras que fazem referência a nãoterminais inúteis, temos:

1: S → A B
4, 5: A → a A | a
6, 7: B → b B | b

Os símbolos restantes podem ser considerados inúteis: C D Y Z c d y z.

5.4 - O lema do bombeamento para linguagens livres de contexto.

Vamos examinar agora um resultado que nos permitirá provar que algumas linguagens não são livres de contexto. O resultado é conhecido como Lema do Bombeamento, ou *Pumping Lemma*, e é semelhante ao resultado correspondente visto para linguagens regulares.

Teorema 5.2: Lema do Bombeamento. Seja L uma llc. Então, existe um número n, que só depende de L, tal que qualquer cadeia z de L com comprimento maior ou igual a n pode ser decomposta de maneira que z = uvwxy e

$|vx| \geq 1$

$|vwx| \leq n$

para todo $i \geq 0$, uv^iwx^iy pertence a L.

Demonstração (simplificada). Se L é uma llc, existe uma glc G tal que L(G)=L. Se z tem um comprimento suficientemente longo, não será possível gerar z sem que na derivação de z ocorra um não terminal A repetido, de forma que a partir de A é derivada uma cadeia que contém outra ocorrência de A. Para que isto ocorra, as duas ocorrências de A devem estar num mesmo caminho da raiz da árvore de derivação até as folhas. (Cadeias mais curtas podem ser geradas sem que essa repetição aconteça.) Através (possivelmente) de uma rearrumação dos passos da derivação, temos

$S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uvwxy$

ou seja,

$S \Rightarrow^* uAy$,

$A \Rightarrow^* vAx$,

$A \Rightarrow^* w$

Portanto, podemos derivar

i=0:	uwy	$S \Rightarrow^* uAy \Rightarrow^* uwy$
i=1:	uvwxy	$S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uvAxy \Rightarrow^* uvwxy$
i=2:	uvvwxxy	$S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uvvAxxy \Rightarrow^* uvvwxxy$
i=3:	uvvvwxxxy	$S \Rightarrow^* uAy \Rightarrow^* uvAxy \Rightarrow^* uvvAxxy$ $\Rightarrow^* uvvvAxxxy \Rightarrow^* uvvvwxxxy$

…

Uma demonstração completa do Lema do Bombeamento pode ser encontrada na referência citada.

Observação. A recíproca do Lema do Bombeamento não é verdadeira, isto é, existem linguagens que não são livres de contexto, mas que tem a propriedade da decomposição.

Exemplo 5.6: Considere a gramática G_0, a cadeia z = `a*(a+a)+a`, e a árvore de derivação correspondente a z (Exemplo 5.2), reproduzida na Figura 3. Podemos ver que existem vários casos de repetição de nãoterminais da forma indicada no teorema acima. Por exemplo, vamos considerar as duas ocorrências de T indicadas pelas setas:

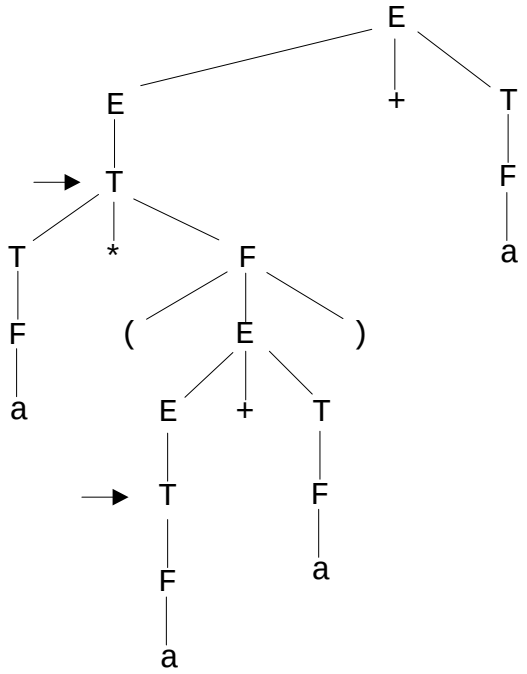

Fig. 3- Árvore de derivação de `a*(a+a)+a`

Temos: `E` $\Rightarrow^*$ `T+a`, `T` $\Rightarrow^*$ `a*(T+a)`, `T` $\Rightarrow^*$ `a`. Ou seja, u = ε, v = `a*(`, w = `a`, x = `+a)`, y=`+a`. Ou seja, as seguintes cadeias devem ser da linguagem:

i=0:	uwy	a +a
i=1:	uvwxy	a*(a +a) +a
i=2:	uv^2wx^2y	a*(a*(a +a) +a) +a
i=3:	uv^3wx^3y	a*(a*(a*(a +a) +a) +a)+a

…

Exercício 5.2: (Ver Exemplo 5.3)

1. Construa árvores de derivação para as cadeias uv^iwx^iy, para i=0, 1, 2, 3, observando que essas árvores são construídas de pedaços da árvore de derivação de x. (Nem todos esses pedaços são sub-árvores.)
2. Verifique todas as combinações de nãoterminais repetidos que satisfazem as condições do teorema, e quais as decomposições possíveis para a cadeia z.
3. Estime o valor de n para a linguagem considerada.

Exercício 5.3: Considere a llc

$$L=\{ x\, x^R \mid x \in \{a, b\}^* \} \cup \{ aaaab \}$$

e a cadeia z = aabaabaa. Estime n para essa linguagem. Determine todas as decomposições possíveis de z, de acordo com o teorema.

Exemplo 5.7: Vamos agora mostrar que $L = \{ a^m b^m c^m \mid n \geq 0 \}$ não é livre de contexto, usando o teorema acima. A demonstração é por contradição: suporemos que L é livre de contexto, e deduziremos um absurdo.

Se L é llc, L satisfaz o teorema acima para algum n. Suponha que a cadeia $z=a^k b^k c^k$ é suficientemente longa: $| z | \geq n$. Então z pode ser decomposta, z = uvwxy, de forma que para qualquer i, $z_i = uv^iwx^iy$ pertence a L. A contradição está em que qualquer decomposição, existem cadeias z_i que não pertencem a L: ou tem o número errado de a's, b's, e c's, ou aparecem símbolos fora da ordem a - b - c: as combinações ba, cb e ca não podem ocorrer em L.

Para eliminar todas as decomposições:

- se v e x não tem o mesmo número de a's, b's e c's, algum z_i terá números diferentes dos três símbolos;
- se v e x tem o mesmo número de a's, b's e c's, devemos ter $v = a^j$ e $x = b^j c^j$, ou $v = a^j b^j$ e $x=c^j$. No primeiro caso, z_2 contém $x^2 = b^j c^j b^j c^j$, que contém a combinação cb, que não ocorre em L; no segundo caso, de forma semelhante, ocorre a combinação ba.

Logo, L não é uma llc.

Exercício 5.4: Mostre que a linguagem $\{ x\, x \mid x \in \{a, b\}^* \}$ não é uma llc.

Nota: a primeira versão deste capítulo contou com a colaboração de Luiz Carlos Castro Guedes

(maio 1999)