CAPÍTULO 8

PROPRIEDADES DAS LINGUAGENS LIVRES DE CONTEXTO

8.1. Introdução

Tal como no Capítulo 4 estudámos propriedades fundamentais das linguagens regulares, iremos estudar neste capítulo algumas propriedades das linguagens livres de contexto.

Uma questão importante é a de se saber se uma dada linguagem é ou não livre de contexto. Encontraremos aqui lemas da bombagem, com algumas semelhanças com o que vimos nas linguagens regulares, mas agora um pouco mais complicados. Eles não servem para provar que uma linguagem é livre de contexto, mas servem para provar o contrário: que uma dada linguagem não é livre de contexto, por não possuir certas propriedades estruturais da classe.

Quando aplicamos certas operações a CFL, importa saber se o resultado é ainda uma CFL, e isto leva-nos às propriedades de fecho.

Há questões aparentemente simples sobre linguagens livres de contexto para as quais ainda não foi possível encontrar resposta. Veremos algumas, no quadro dos algoritmos de decisão.

8.2. Dois lemas de bombagem

Os lemas de bombagem servem para mostrar que uma linguagem não pertence a uma dada família.

8.2.1 Lema de bombagem para linguagens livres de contexto

Os lemas de bombagem aplicam-se a linguagens infinitas. Qualquer gramática tem um número finito de produções e por isso, para uma linguagem infinita, há produções que têm que ser usadas repetidas vezes na mesma derivação, criando-se assim ciclos que permitem a geração de um número infinito de cadeias de comprimento arbitrário. Numa CFL têm que existir ciclos com certas características, definidas pelo seguinte teorema.

Teorema 8.2.1.1

Seja L uma linguagem infinita livre de contexto.

Então existe algum inteiro positivo m tal que toda a $w \in L$ com $|w| \geq m$ se pode decompor em

$$w = uvxyz$$

com

$$|vxy| \leq m$$

e

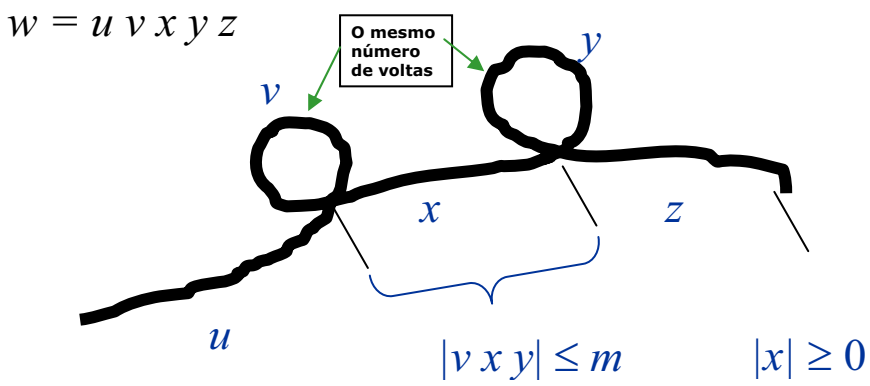
$$|vy| \geq 1$$

tal que bombeando os segmentos v e y se obtêm cadeias de L, ou seja,

$$uv^ixy^iz \in L$$

para todo o $i = 0, 1, 2, ...$

Graficamente pode ilustrar-se pela figura seguinte. Note-se que $|vxy| \leq m \Rightarrow |v| \leq m$ e $|y| \leq m$.



$|u| \geq 0,\ |z| \geq 0$, arbitrários

bombagem : $u\, v^{\,i}\, x\, y^{\,i}\, z$

Ou v ou y é não vazia (uma delas pode sê-lo)

Figura 8.2.1.1.Bombagem nas CFLs.

A característica fundamental é a igualdade das voltas nos dois ciclos. Essa igualdade requer que o autómato que aceita a linguagem tenha a possibilidade de se lembrar do número de voltas no primeiro ciclo. Podendo ser um número arbitrariamente grande, não pode ser um autómato finito, como já sabemos. O PDA, com pilha infinita, tem essa possibilidade. A pilha serve precisamente para de algum modo contar as voltas do primeiro ciclo.

Repare-se que se v ou y for vazia (uma delas pode sê-lo, mantendo $|vy| \geq 1$), temos o caso de uma linguagem regular. Este facto é natural: as linguagens regulares também são livres de contexto e por isso este lema também se lhes aplica. Neste caso não é necessário contar o número de voltas do ciclo, porque não há comparação a fazer, e por isso um autómato finito (ou um PDA coma pilha desactivada) é suficiente para a linguagem.

Aqui consideraremos o caso em que nem v nem y são vazias.

Vejamos uma demonstração do lema.

Considere-se uma Linguagem $L - \{\lambda\}$, sem o carácter vazio, com uma gramática G sem produções unitárias e sem produções -λ.

O comprimento do corpo de uma produção é sempre limitado; seja k esse limite. Para uma cadeia $w \in L$, o comprimento da sua derivação (número de produções necessárias para a obter) será por isso de pelo menos $n = | w | / k$.

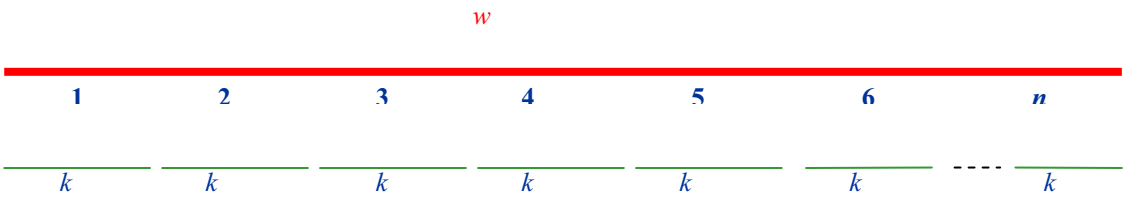


Figura 8.2.1.2 Uma cadeia é produzida por sucessivas produções de comprimento k.

Sendo L infinita, existem derivações arbitrariamente longas a que correspondem árvores de derivação de altura arbitrária.

Se considerarmos uma derivação suficientemente longa e um caminho longo desde a raiz até a uma folha, haverá no seu percurso um número ilimitado de variáveis; ora como o número de variáveis diferentes é finito, haverá necessariamente repetições de variáveis, tal como ilustrado na Fig. 8.2.1.3

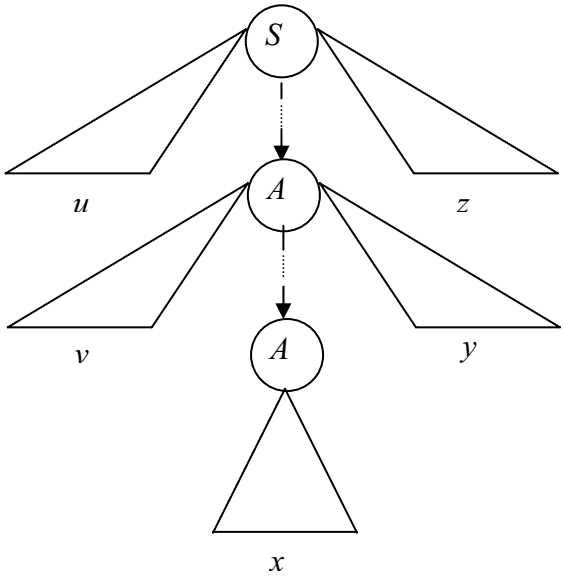


Figura 8.2.1.3. Uma derivação arbitrariamente longa repete necessariamente variáveis. Um triângulo representa um conjunto de derivações a partir da variável central até à cadeia-folha do triângulo. No fundo cada triângulo é uma sub-árvore.

Da figura poderemos escrever a sequência de derivações

$$S \overset{*}{\Rightarrow} uAz \overset{*}{\Rightarrow} uvAyz \overset{*}{\Rightarrow} uvxyz, \qquad uvxyz \in L$$

sendo u, v, x, y, z cadeias de símbolos terminais.

Quer dizer, tendo em conta que de cima se tira $uAz \overset{*}{\Rightarrow} uvAyz$ o que implica que

$$A \overset{*}{\Rightarrow} vAy.$$

Pode-se então desenhar uma nova árvore de uma derivação com mais um conjunto de produções que completem um ciclo de A a A, obtendo-se a Fig.8.2.1.4.

Agora temos

$$S \overset{*}{\Rightarrow} uAz \overset{*}{\Rightarrow} uvAyz \overset{*}{\Rightarrow} uvvAyyz \overset{*}{\Rightarrow} uvvxyyz$$

e portanto a cadeia $uvvxyyz = uv^2xy^2z \in L$.

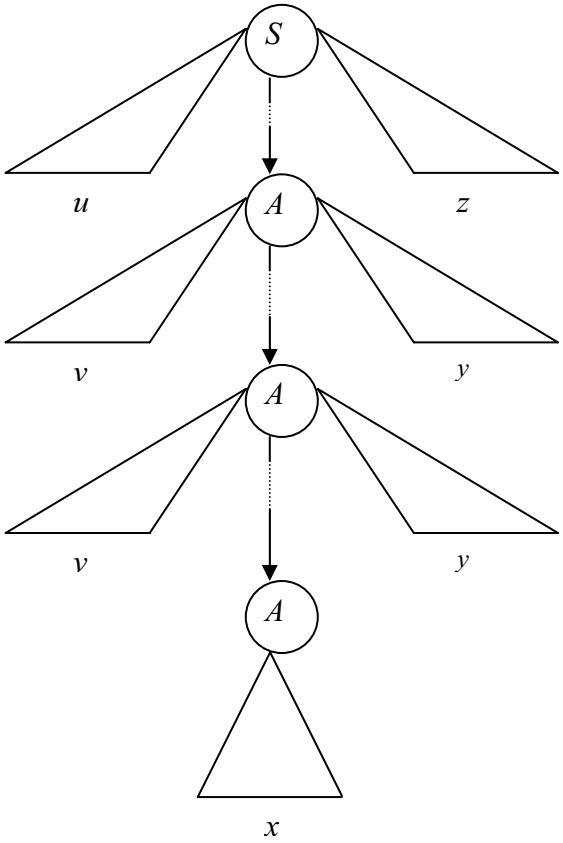


Figura 8.2.1.4. Mais um ciclo de A a A em relação à figura anterior.

Pode-se repetir o procedimento um número arbitrário de vezes, *i,* de tal modo que

$$uv^i xy^i z \in L$$

Fica assim demonstrado que bombeando v e y o mesmo número de vezes se obtêm cadeias na linguagem. É como se fossem duas bombas síncronas.

Este Lema aplica-se de modo análogo ao do regular: para provar que uma linguagem **não é** livre de contexto. Como apresenta duas "bombas" funcionando sincronamente, é de aplicação mais delicada. Basicamente prova-se que para uma dada linguagem não é possível encontrar as duas bombas síncronas satisfazendo o lema, e por isso ela não é livre de contexto.

Note-se que agora a parte a bombear, *vxy,* pode estar em qualquer parte de w, seja no princípio ($u=\lambda$), seja no meio (u e z diferentes de λ), ou no fim ($z=\lambda$), sempre com $|vxy| \leq m$, o que implica que na prova se tenham que experimentar todas essas possibilidades.

Mas a filosofia de aplicação é exactamente a mesma. Poderemos repetir aqui a Tabela 4.4.9, com a única diferença na definição da decomposição de w em partes.

Tabela 8.2.1.1 O lema da bombagem para CFLs.

O Lema pela afirmativa	**O Lema pela negativa**
- existe **um** m tal que para	- para **qualquer** valor de m sou capaz de encontrar
- para **qualquer** cadeia $\|w\| \geq m$ pertencente a L existe	- (pelo menos) **uma** cadeia $\|w\| \geq m$ pertencente a L em que
- **uma** decomposição $uvxyz$ que produz, pela bombagem de v e y, $\|vxy\| \leq m$	- **qualquer** decomposição $uvxyz$ produz, pela bombagem de v e y, $\|vxy\| \leq m$
- **todas** cadeias pertencentes a L $\forall i \geq 0, uv^i xy^i z \in L$	- (pelo menos) **uma** cadeia que não pertence à linguagem $\exists\, i \geq 0, uv^i xy^i z \notin L$

Exemplo 8.2.1.1.

Seja $L=\{a^n b^{2n} c^n,\ n \geq 0\}$. Provar que não é livre de contexto.

A característica essencial da linguagem é a da relação entre o número de *a*'s, *b*'s e *c*'s. Se uma bombagem destruir essa relação, a cadeia bombeada não pertence à linguagem.

Para fazer com que isso aconteça, basta impossibilitar a bombagem simultânea de *a´*s, *b*'s e *c*'s, dado que a bombagem de apenas *a'*s e *b*'s, ou de apenas *b*'s e *c*'s, destrói a relação.

Considere-se por exemplo, para um dado m, a cadeia com m *a*'s, $2m$ *b*'s e m *c*'s.Ela é maior do que m e portanto deve verificar o lema.

Verifica-se que qualquer que seja o posicionamento da bombagem obtêm-se sempre cadeias fora da linguagem. Como a subcadeia a bombear, vxy, tem que ser menor ou igual a m, se a distância do último *a* ao primeiro *c* for maior do que m, como é o caso, nunca se podem bombear simultaneamente *a´*s, *b´*s e *c´*s.

Note-se que esta construção é válida para qualquer valor de m. Pode-se por isso concluir que não existe um tal m que verifique o Lema, ficando assim provado que a linguagem não é livre de contexto.

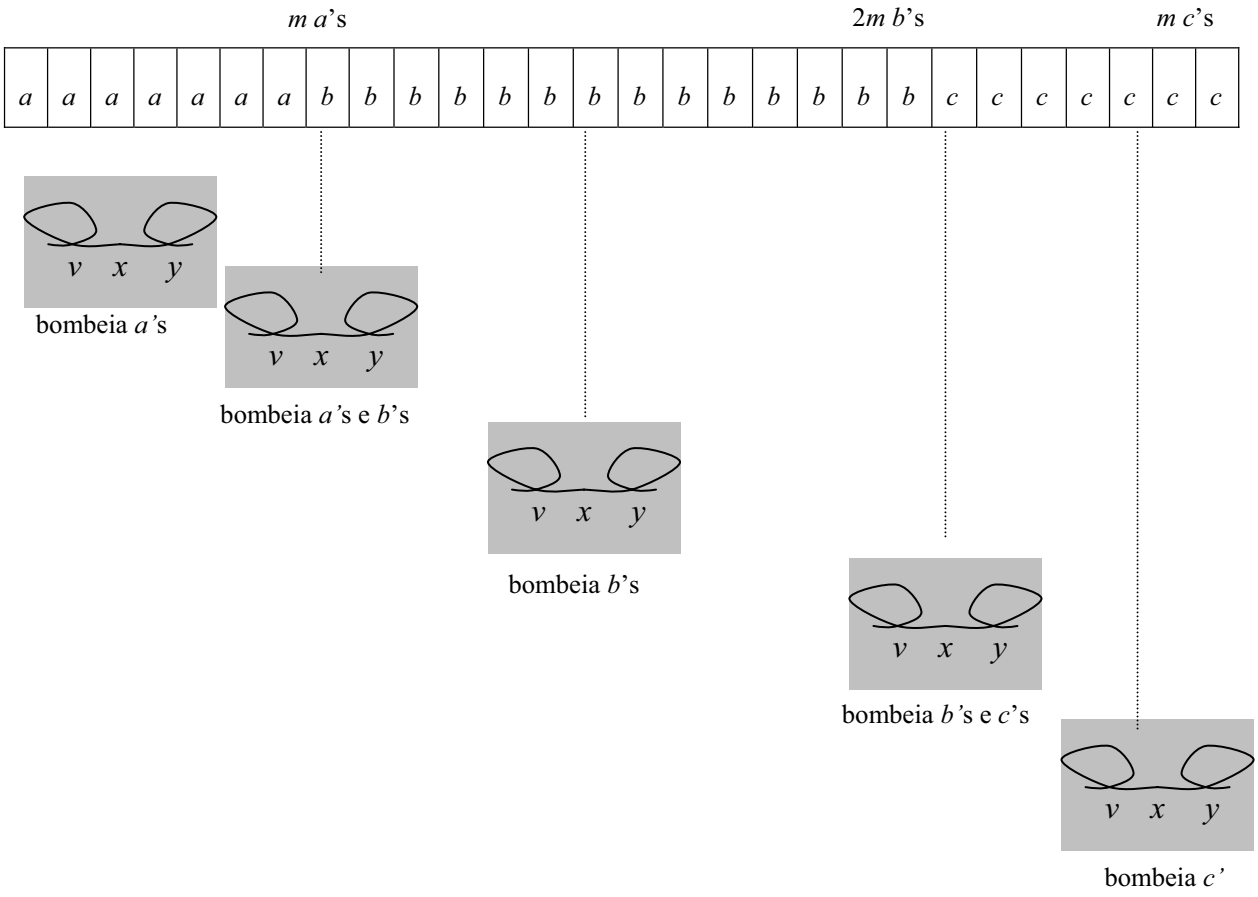


Figura 8.2.1.5 É necessário tentar todas as localizações possíveis, ao longo da cadeia de teste, das partes bombeadas.

Também se pode fazer a prova como um jogo entre dois parceiros, um procurando o *m*, o outro procurando uma cadeia que "encoste o adversário à parede". Repete-se aqui, na Tabela 8.2.1.2, a Tabela 4.4.10 agora com uma nova decomposição *uvxyz*.

Tabela 8.2.1.2 O jogo do lema da bombagem

Jogador 1	Jogador 2
Define a linguagem L	
	Escolhe um *m*
Indica uma cadeia de L maior do que *m*	
	Define uma decomposição $\|uvxyz\| \leq m$
Indica um *i* que bombeie para fora	

Exemplo 8.2.1.2

$L = \{a^n b^p c^q,\ n > p \geq q > 0\ \}$

Aqui temos uma desigualdade. Interessa, para um dado valor de *m,* escolher uma cadeia na situação limite e depois, com a bombagem, destruir a desigualdade. Para isso é conveniente que não se possam bombear simultaneamente *a*'s, *b*'s e *c*'s (se se pudesse, poder-se-ia eventualmente manter a desigualdade).

Tabela Exemplo 8.2.1.2

Jogador 1	Jogador 2
Define a linguagem:	
	Escolhe um *m*: 6
Indica uma cadeia: *aaaaaaaabbbbbbbccccccc* estratégia: *m+1 a*'s, *m b*'s, *m c*'s	
	Decomposição: *uvxyz*= |*aaaaaa*|*a*|*bbbb*|*b*|*bccccc*
Indica um *i* que bombeie para fora: *i* = 0 *aaaaaabbbbbccccccc* não pertence a *L* (*p*<*q*)	Perde, em qualquer decomposição com $\|vxy\| \leq 6$
	Escolhe outro *m*: 8
Indica uma cadeia: *aaaaaaaaabbbbbbbbbccccccccc* (estratégia, *m+1 a*'s, *m b*'s, *m c*'s)	
	Decomposição *uvxyz* = |*aaaaaaaa*|*ab*|*bbbbb*|*b*|*bccccccccc*|
Indica um *i* que bombeie para fora: i =0 *aaaaaaaabbbbbbccccccccc* não pertence a L (*p*<*q*)	Perde, em qualquer decomposição com $\|vxy\| \leq 8$

Para outros valores de *m*, seguindo a mesma estratégia, chega-se à mesma conclusão: não existe um *m que* verifique o Lema.

A linguagem não é livre de contexto, *q.e.d.*

Há linguagens que não são livres de contexto e para as quais se verifica o Lema Livre de Contexto. Veja-se o exemplo 8.2.1.3.

Exemplo 8.2.1.3**.**

Seja $L = \{a^n b^n c^p d^p e^p,\ n \geq 2,\ p \geq 1\}$, em $\Sigma = \{a,b,c,d,e\}$

Esta linguagem não é livre de contexto devido à igualdade de *c*'s, *d*'s e *e*'s. No entanto é possível encontrar para ela um valor de *m* e uma decomposição *uvxyz*, em qualquer cadeia $|w| \geq m$, que verifica a bombagem para dentro.

A menor cadeia da linguagem é *w = aabbcde.*

Vejamos o que se passa com *m*=7.

Decompondo $w=uvxyz=|a|a||b|bcde|$, $x=\lambda$, bombeando $v=a$ e $y=b$ obêm-se sempre cadeias de *L.*

Para cadeias maiores que 7, usando sempre uma decomposição $|vxy|=|a||b|$, o primeiro par *ab* da cadeia, obtém-se resultado igual.

Mas não se pode concluir que a linguagem seja livre de contexto. De facto ela não o é devido à segunda parte dos *c,d,e*. Por isso verifica-se o ilustrado na Fig. 8.2.1.6.

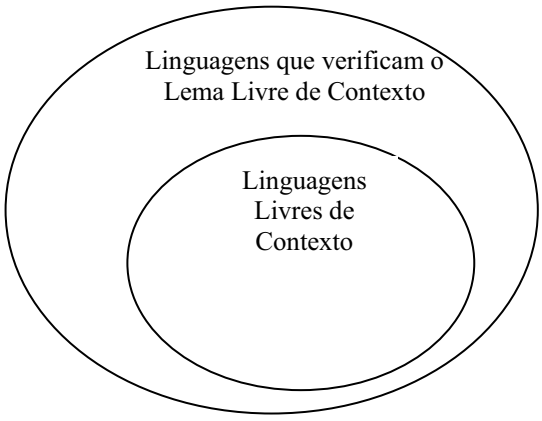


Figura 8.2.1.6. Nem todas as linguagens que verificam o Lema Livre de Contexto são livres de contexto. Mas se uma linguagem não o verifica, ela não é livre de contaxto.

8.2.2. Um lema de bombagem para linguagens lineares

Uma linguagem livre de contexto diz-se linear se existir uma gramática livre de contexto linear tal que L = $L(G)$. Sabemos que uma gramática é linear se do lado direito das produções aparece no máximo uma variável. Note-se que aqui uma produção pode ser linear à esquerda e outra produção linear à direita e outra ainda linear ao centro.

Teorema 8.2.2.1

Seja L uma linguagem infinita linear. Então existe algum inteiro positivo m, tal que toda a cadeia $w \in L$, com $|w| \geq m$, se pode decompor em

$$w = uvxyz$$

com

$$|uvyz| \leq m$$

$$|vy| \geq 1$$

de modo que bombeando simultaneamente v e y se obtém

$$uv^ixy^iz \in L$$

para todo o i=0, 1, 2, ...

A figura 8.2.2.1 ilustra graficamente onde se localizam as quatro sub-cadeias. Note-se que $|uvyz| \leq m$ implica que $|uv| \leq m$ e $|yz| \leq m$.

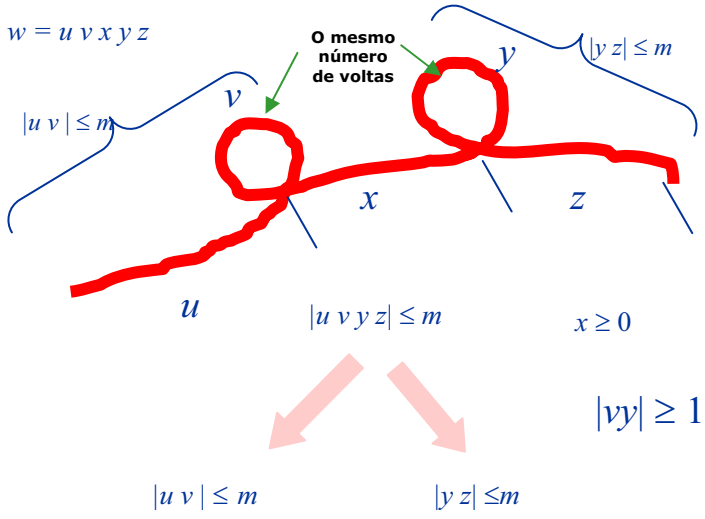


Figura 8.2.2.1 Lema de bombagem para linguagens lineares.

A grande diferença entre este lema e o anterior reside em *w*. Neste, as sub-cadeias a serem bombeadas (*v* e *y*) devem-se localizar dentro de *m* símbolos dos extremos esquerdo e direito de *w*, respectivamente; a cadeia do meio, *x*, pode ter um tamanho arbitrário. No lema anterior, *u* e *z* podem ser arbitrariamente grandes. Este facto deriva da linearidade: havendo apenas uma variável em cada derivação, chega-se mais rapidamente a um ciclo do que no caso em que existam muitas variáveis em cada derivação.

Demonstração:

Sendo a linguagem linear, existe alguma gramática linear para ela, que não contenha nem produções unitárias nem produções - λ (a eliminação deste tipo de produções não altera a linearidade da gramática). Consideremos a figura seguinte 8.2.2.2.

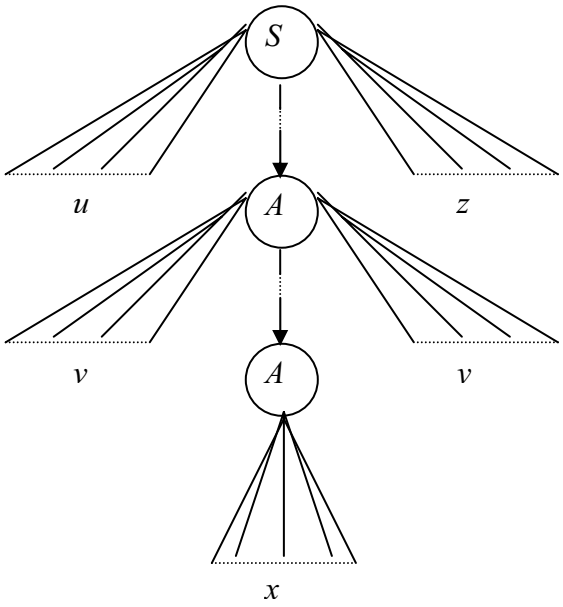


Figura 8.2.2.2 Árvore de derivação de uma linguagem linear.

Se a gramática é linear (contém no máximo uma só variável no corpo de cada derivação), as variáveis só podem aparecer no caminho central *S-A-A* ....(efeito conjugado de produções lineares à esquerda, à direita e ao centro). Havendo um número finito de variáveis no caminho de *S* até ao primeiro *A*, e gerando cada uma delas um número finito de símbolos terminais, quer *u* quer *z* são limitados. Por razões semelhantes *v* e y são também limitados e por isso existe um *m* tal que $|uvyz| \leq m$ (mesmo que o *m* tenha que ser grande, é limitado).

Da figura poderemos escrever a sequência de derivações

$$S \overset{*}{\Rightarrow} uAz \overset{*}{\Rightarrow} uvAyz \overset{*}{\Rightarrow} uvxyz \qquad uvxyz \in L$$

sendo *u, v, x, y, z* cadeias de símbolos terminais. Isto implica que

$$A \overset{*}{\Rightarrow} vAy.$$

Continuando a derivar a partir do último *A*, até encontrar novamente *A*, pode-se desenhar uma segunda figura incluindo a produção adicional $A \Rightarrow vAy$, Fig.8.2.2.3

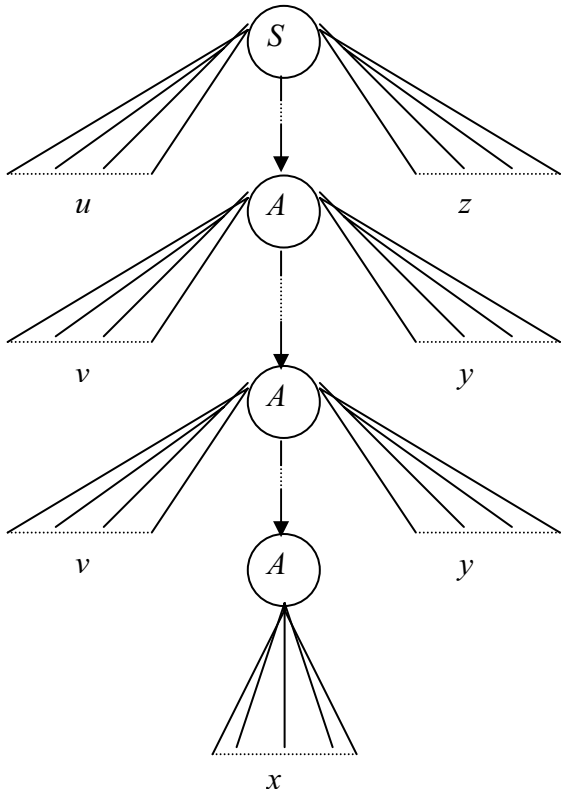


Figura 8.2.2.3 Mais um ciclo a partir da figura anterior.

Agora temos

$$S \overset{*}{\Rightarrow} uAz \overset{*}{\Rightarrow} uvAyz \overset{*}{\Rightarrow} uvvAyyz \overset{*}{\Rightarrow} uvvxyyz$$

e portanto a cadeia $uvvxyyz = uv^2xy^2z \in L$.

Continuando o processo, conclui-se que $uv^ixy^iz \in L$, para qualquer *i*.

As linguagens lineares são um subconjunto próprio da família das linguagens livres de contexto. Por isso se pode afirmar que o lema das linguagens lineares é um caso particular do lema livre de contexto.

Essa característica dá-nos já alguma luz sobre a estratégia da sua utilização: bloquear o adversário dando-lhe uma cadeia em que bombeando nos primeiros *m* caracteres e nos últimos *m* caracteres se destrua a característica da linguagem em causa.

Exemplo 8.2.2.1

Prove-se que a linguagem $L=\{a^n b^{2n} c^p d^p,\ n\geq 0,\ p\geq 0\}$ não é linear.

Para um valor de *m* qualquer, tem que se escolher uma cadeia segundo alguma estratégia. Aqui, trata-se de obrigar o adversário a não conseguir bombear sem destruir a característica da linguagem: a relação entre o número de *a*'s e de *b*'s **e** a relação entre o número de *c'*s e de *d*'s.

Sabendo nós que a zona de bombagem tem que estar próxima do início (à distância máxima de *m* caracteres) e próxima do fim (à distância máxima de *m* caracteres do fim), é suficiente apresentar uma cadeia que tenha

- *m a´*s no início, e assim bombeia *a*'s e não bombeia *b*'s
- e *m d*'s no fim, e assim bombeia *d'*s e não bombeia *c'*s

Tabela Exemplo 8.2.2.1

Jogador 1	Jogador 2
Define a linguagem: $a^n b^{2n} c^p d^p$,	
	Escolhe um *m*: 5
Indica uma cadeia: *aaaaabbbbbbbbbbccccddddd* estratégia: *m a*'s, 2*m b*'s, *m c*'s, *m d*'s	
	Decomposição: $\|uvyz\|\leq 5$ *uvxyz*= |*aa*|*a*|*aabbbbbbbbbbccccddd*|*d*|*d*|
Indica um *i* que bombeie para fora: i = 0 *aaaabbbbbbbbbbccccddddd* não pertence a L	Perde, em qualquer decomposição com $\|uvyz\|\leq 5$
	Escolhe outro *m*: 2

Indica uma cadeia: *aabbbbccdd* (estratégia, *m+1 a*'s, *m b*'s, *m c*'s)	
	Decomposição $uvxyz = \|\|a\|abbbbccd\|d\|\| \ , \ u=z) \lambda$
Indica um *i* que bombeie para fora: i =2 *aaabbbbccddd* não pertence a L	Perde, em qualquer decomposição com $\|uvyz\| \leq 2$

A linguagem não é linear.

Será livre de contexto não-linear ?

8.2.3. Relação entre os três Lemas de Bombagem

Pela hierarquia das linguagens sabemos que

- a família das linguagens lineares é um subconjunto próprio das linguagens livres de contexto
- a família das linguagens regulares é um subconjunto próprio das lineares e portanto também das livres de contexto

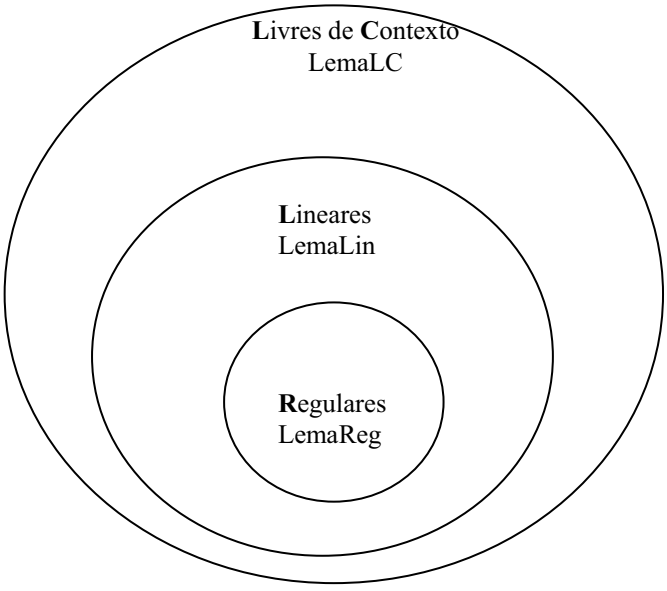


Figura 8.3.1. Relação entre os três lemas e as três famílias de linguagens em causa.

Assim sendo, quer dizer que

- uma linguagem linear deve satisfazer o Lema Livre de Contexto ? Sim

- uma linguagem regular deve satisfazer o Lema Livre de Contexto ? Sim

- uma linguagem regular deve satisfazer o Lema Linear ? Sim

De facto:

- o Lema Regular é um caso particular do Lema Linear

    - se na decomposição *uvxyz* (Linear) uma entre *v* e *y* é vazia (uma só "bomba") cai-se no caso Regular.

- o Lema Linear é um caso particular do Lema Livre de Contexto

    - se uma linguagem verifica o Lema Linear, ela também verifica o Lema Livre de Contexto. No entanto o valor de *m* não é necessariamente o mesmo para os dois lemas na mesma linguagem.

- o Lema Regular é um caso particular do Lema Livre de Contexto.

    - Se no Livre de Contexto *v* for vazia, temos o caso Regular, ou se *y* for vazia temos de igual modo o caso Regular.

Mas para provar que uma linguagem **não** pertence a uma das famílias, aplica-se o lema dessa família (pela prova por contradição).

O Lema **nunca** se aplica para provar que **sim, pertence**, qualquer que seja o caso.

8.3. Propriedades de fecho e algoritmos de decisão para linguagens livres de contexto.

8.3.1. Propriedades de fecho de linguagens livres de contexto

As propriedades de fecho referem-se a operações de conjuntos feitas sobre as linguagens. Por exemplo: se unirmos duas CFL, a linguagem obtida é livre de contexto ou não ?

Teorema 8.3.1.1. A família das linguagens livres de contexto **é fechada** em relação a

a) união

b) concatenação

c) fecho-estrela e fecho-positivo

d) homomorfismos

e) reversão

Demonstração

No caso das CFL, a prova é mais fácil se usarmos as suas gramáticas (relembre-se que no caso das linguagens regulares se construíam autómatos para a prova).

a) Prova em relação à união:

Exemplo 8.3.1.1Considerem-se as gramáticas G_1 e G_2 seguintes, a que correspondem as linguagens L_1 e L_2:

$G_1 = (V_1, T_1, S_1, P_1\}$ $S_1 \rightarrow aS_1 \mid bA$ $A \rightarrow bB$ $B \rightarrow aC$ $C \rightarrow bbC \mid c$	$G_2 = (V_2, T_2, S_2, P_2\}$ $S_2 \rightarrow 0S_1 \mid 01L$ $L \rightarrow 1M$ $M \rightarrow 00N$ $P \rightarrow 11P \mid 1$

Se quisermos a linguagem

$$L_3 = L_1 \cup L_2$$

poderemos usar as duas gramáticas para gerarem as cadeias da união, usando ora uma ora outra. Isto faz-se criando uma nova variável de inicialização (um novo axioma) que deriva ou a inicialização de G_1 ou a inicialização de G_2. Teremos assim a gramática G_3

$$G_3 = (V_3, T_3, S_3, P_3\}$$

$$\boldsymbol{S_3 \rightarrow S_1 \mid S_2}$$
$$S_1 \rightarrow aS_1 \mid bA$$
$$S_2 \rightarrow 0S_1 \mid 01L$$
$$A \rightarrow bB$$
$$B \rightarrow aC$$
$$C \rightarrow bbC \mid c$$
$$L \rightarrow 1M$$
$$M \rightarrow 00N$$
$$P \rightarrow 11P \mid 1$$

Esta forma de especificar a união é muito prática. Note-se que os alfabetos das linguagens L_1 e L_2 são bem diferentes mas isso não tira qualquer coerência à gramática G_3 que nunca mistura letras com números numa cadeia. Este procedimento é possível para quaisquer gramáticas livres de contexto.

b) Prova da concatenação

Considerando ainda o mesmo exemplo, se quisermos L_1L_2 cria-se de modo análogo uma gramática G_4 do modo seguinte

$G_4 = (V_4, T_4, S_4, P_4\}$

$$\boldsymbol{S_4 \rightarrow S_1\, S_2}$$
$$S_1 \rightarrow aS_1 \mid bAA$$
$$S_2 \rightarrow 0S_2 \mid 01L$$
$$A \rightarrow bBB$$
$$B \rightarrow aC$$
$$C \rightarrow bbC \mid c$$
$$L \rightarrow 1MP$$
$$M \rightarrow 00N$$
$$P \rightarrow 11P \mid 1$$

Verifique o leitor que G_4 nunca coloca letras depois de números.

c) Prova do fecho estrela e do fecho-positivo

O fecho estrela L_1^* resulta da concatenação de sucessivas potências de L_1. Ora tal pode fazer-se criando uma gramática G_5 que tenha um ciclo em torno de S_1, que possa ser feito um número qualquer de vezes, incluindo zero vezes:

$G_5 = (V_5, T_5, S_5, P_5\}$

$\boldsymbol{S_5 \rightarrow S_1 S_5 \mid \lambda}$ (ciclo em torno de S_1)

$S_1 \rightarrow aS_1 \mid bAA$

$A \rightarrow bBB$

$B \rightarrow aC$

$C \rightarrow bbC \mid c$

Se eliminarmos λ na produção S_5 elimina-se o carácter vazio, ou seja, teremos L_1^+.

d) prova em relação aos homomorfismos: ver Hopcroft , p.285

e) prova em relação à reversão

Queremos provar que se L é uma CFL, também o é L^R

Seja G a gramática de L

$$G = (\ V, T, S, P\ ), L = L\ (G)$$

Com por exemplo a produção P

$$A \rightarrow baABCba$$

Se construirmos uma gramática a partir de G revertendo todos os corpos das produções de G, no caso de P virá P^R

$$A \rightarrow abCBAab$$

obteremos a gramática G^R que gera L^R

$$G^R = (\ V, T, S, P\ R\ )$$

$$L^R = L\ (G^R)$$

Assim sendo L^R é livre de contexto (porquê?).

Teorema 8.3.1.2. A família das linguagens livres de contexto **não é fechada** em relação à

f) intersecção

g) complementação

h) diferença

Vejamos um exemplo de intersecção que demonstra a propriedade**.**

Exemplo 8.3.1.2 (Hopcroft, 285)

Pela aplicação do lema da bombagem verifica-se que

$$L=\{0^n1^n2^n,\ n\geq 1\}$$

é uma linguagem não livre de contexto (é dependente de contexto).

Considerem-se agora as duas linguagens L_1 e L_2 da tabela onde também estão as produções de uma sua gramática, que são livres de contexto.

Tabela Exemplo 8.3.1.2

$L_1=\{0^n1^n2^i,\ n\geq 1,\ i\geq 1\}$	$L_2=\{0^i1^n2^n,\ n\geq 1,\ i\geq 1\}$	$L_1\cap L_2=\{0^n1^n2^n,\ n\geq 1\}$
Produções: $S\rightarrow AB$ $A\rightarrow 0A1\mid 01$ (gera 0^n1^n) $B\rightarrow 2B\mid 2$ (gera 2^i)	Produções: $S\rightarrow AB$ $A\rightarrow 0\mid 0$ (gera 0^i) $B\rightarrow 1B2\mid 12$ (gera 1^n2^n)	Produções ????

A intersecção das duas linguagens dá de facto L. Por um lado em L_1 o número de 0's iguala o número de 1's. Por outro lado em L_2 o número de 1's iguala o número de 0's. Só pertencem à intersecção das duas as cadeias em que se verifiquem simultaneamente as duas igualdades, ou seja, em que o número de 0's iguale o número de 1's e o número de 2's, portanto as cadeias de L. Ora L_1 e L_2 são livres de contexto e L não o é.

Basta este exemplo para provar que a família das CFLs não é fechada em relação à intersecção.

A prova em relação à complementação e diferença pode ver-se em Hopcroft 289.

Teorema 8.3.1.3. A família das linguagens livres de contexto é fechada em relação à intersecção regular.

Se L é uma gramática livre de contexto e R é uma linguagem regular, então

$$L \cap R \text{ é livre de contexto.}$$

(propriedade de fecho em relação à **intersecção regular).**

Ver a demonstração em Hopcroft 286.

Em consequência, L-R uma CFL. De facto

$$L - R = L \cap \overline{R}$$

Ora se R é regular, também o é $\overline{R}$ e temos uma intersecção regular.

8.3.2. Algumas propriedades decidíveis de linguagens livres de contexto

Teorema 8.3.2.1 Existe um algoritmo para decidir se uma dada cadeia pertence ou não a $L(G)$:

- 1º remover as produções unitárias e as produções- λ

- 2º fazer o *parsing* exaustivo

O parsing termina após um número finito de etapas, encontrando a cadeia se ela pertence á linguagem ou não a encontrando se ela não pertence à linguagem.

Teorema 8.3.2.2. Dada uma linguagem livre de contexto $G = (V, T, S, P)$ existe um algoritmo para decidir de $L(G)$ é ou não vazia:

- 1º remover as produções unitárias, as produções- λ, os símbolos e produções inúteis.

- 2º Se o símbolo inicial S da gramática G for inútil, a gramática é vazia. Caso contrário não o é.

Teorema 8.3.2.3. Dada uma gramática livre de contexto $G= (V, T, S, P)$ existe um algoritmo para decidir de $L(G)$ é ou não infinita, sendo G sem produções-λ, sem produções unitárias e sem símbolos inúteis.

Verifica-se se nas derivações da gramática há alguma variável que se repita.

Por exemplo se numa derivação a variável A se repetir,

$$A \overset{*}{\Rightarrow} xAv \overset{*}{\Rightarrow} uzv$$

é possível gerar um número infinito de cadeias:

$$S \overset{*}{\Rightarrow} uAv \overset{*}{\Rightarrow} uxAvv \overset{*}{\Rightarrow} uxxAvv \overset{*}{\Rightarrow} \ldots$$

$$\overset{*}{\Rightarrow} ux^n A v^n v \overset{*}{\Rightarrow} ux^n zv^n v$$

Se nenhuma variável se repetir, a linguagem é finita.

O grafo de dependência das variáveis também permite decidir. Se ele contiver algum ciclo fechado, a linguagem é infinita. Caso contrário é finita.

Por exemplo o grafo de dependência da gramática seguinte, na figura 8.3.2.1 ,

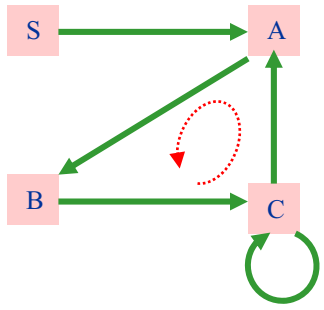
$S \rightarrow bAAa$ $A \rightarrow bBB$ $B \rightarrow aC$ $C \rightarrow bbCA \mid cA \mid c$	Grafo de dependências 

Figura 8.3.2.2. Uma linguagem infinita.

apresenta dois ciclos, identificados na figura: C, C, C, … ABC, ABC, ABC…. Logo a linguagem é infinita.

Já no exemplo seguinte teremos uma linguagem finita. Fica o leitor desafiado a enumerar as suas cadeias.

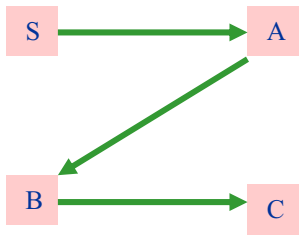
$S \rightarrow bAAa$ $A \rightarrow bBB$ $B \rightarrow aC$ $C \rightarrow c$	Grafo de dependências 

Figura 8.3.2.2. Uma linguagem finita

8.3.3. Algumas propriedades indecidíveis de CFL's (Hopcroft, 302).

Não existe nenhum algoritmo (não é possível resolver com um computador) algumas questões que são aparentemente simples, com por exemplo:

a) é uma dada gramática CFG ambígua ?

b) é uma dada linguagem CFL inerentemente ambígua ?

c) é vazia a intersecção de duas linguagens CFL's ?

d) são duas linguagens CFL's iguais ?

e) é uma linguagem CFL igual a Σ^* ?

Isto não quer dizer que para uma linguagem concreta não se possa obter resposta a qualquer daquelas questões. Quer dizer sim que não há nenhum algoritmo geral que dê resposta (em tempo útil) para toda e qualquer linguagem livre de contexto.

Bibliografia

An Introduction to Formal Languages and Automata, Peter Linz, 3rd Ed., Jones and Bartelett Computer Science, 2001.

Models of Computation and Formal Languages, R. Gregory Taylor, Oxford University Press, 1998.

Introduction to Automata Theory, Languages and Computation, 2nd Ed., John Hopcroft, Rajeev Motwani, Jeffrey Ullman, Addison Wesley, 2001.

Elements for the Theory of Computation, Harry Lewis and Christos Papadimitriou, 2nd Ed., Prentice Hall, 1998.

Introduction to the Theory of Computation, Michael Sipser, PWS Publishing Co, 1997.