Capítulo 9

Propriedades das Linguagens Livres de Contexto

As Linguagens Livres de Contexto (CFL, do inglês *Context-Free Language*) gozam de um conjunto de propriedades que são aqui exploradas.

9.1 Simplificação de CFGs e Forma Normal de Chomsky

Existem formas de transformar as CFGs de forma a encontrar representações alternativas para a mesma linguagem que permitam facilitar o processamento ou a realização de provas ou operações sobre as gramáticas.

Uma das formas mais usadas é a Forma Normal de Chomsky (CNF, ou *Chomsky Normal Form*), em que todas as produções da gramática dão origem a duas variáveis ou a um terminal, ou seja, são produções na forma $A \to BC$ ou $A \to a$, em que A, B e C são variáveis e a um terminal. Note-se que esta forma normal não permite obter a cadeia vazia, pelo que a gramática G de uma linguagem L que inclua a cadeia vazia irá ter uma gramática correspondente em CNF para $L \backslash \{\varepsilon\}$.

Para chegar a esta forma, podem seguir-se um conjunto de passos, eliminando produções-ε, produções unitárias e símbolos inúteis.

Para eliminar produções-ε, é necessário primeiro identificar quais as variáveis anuláveis, que são aquelas que podem produzir a cadeia vazia. Se houver uma produção $A \to \varepsilon$, então a variável A é anulável. Se houver uma produção $B \to CD$, em que tanto C como D são variáveis anuláveis, então B será também anulável. Depois de identificar todas as variáveis anuláveis, é necessário identificar todas as produções em que essas variáveis aparecem no corpo, e criar novas produções em que se fornece a alternativa em que a

variável anulável não esteja presente. Por exemplo, se A for anulável, e houver uma produção $B \to AC$ então esta será transformada em $B \to AC \mid C$. Todas as produções $A \to \varepsilon$ são também obviamente eliminadas.

Uma forma simples de eliminar produções unitárias (aquelas no formato $A \to B$) seria iterativamente ir substituindo o corpo da produção unitária pelas produções da variável (ou seja, para o caso de $A \to B$, substituir por $A \to \alpha$, em que α são as produções de B). No entanto, se existirem ciclos, este método torna-se falível. Então, torna-se necessário usar um método mais robusto, identificando os pares unitários, e depois, para cada par unitário (A, B), incluir as produções não unitárias de B em A. Os pares (X, X) são considerados pares unitários. Um par (X, Y) é considerado um par unitário se houver uma sequência de produções unitárias que permita gerar Y a partir de X. Por exemplo, considerando as produções $A \to B$ e $B \to C$, temos (A, A), (B, B) e (C, C) como pares unitários; olhando para as produções existentes, temos que (A, B) e (B, C) são também pares unitários, assim como (A, C).

Para eliminar símbolos inúteis, é preciso identificar e eliminar símbolos não geradores, e símbolos não atingíveis. Símbolos não geradores são aqueles que não dão origem a terminais. Identificam-se os símbolos geradores tendo em conta que os terminais são geradores e se houver uma produção $A \to \alpha$ em que α é composto apenas por geradores (terminais e/ou variáveis), então A também é gerador. Em relação aos símbolos atingíveis, o símbolo de arranque é atingível por definição, e todos símbolos presentes no corpo das produções de um símbolo atingível são também atingíveis.

Vamos ver um exemplo, usando a seguinte gramática:
$S \to traX \mid X$
$X \to laX \mid \varepsilon$
$Y \to alY \mid \varepsilon$

Para obter a gramática equivalente na Forma Normal de Chomsky precisamos então de realizar os três passos de simplificação.

Para eliminar as produções-ε, começamos por identificar quais as produções que podem dar origem à cadeia vazia. Neste caso, originalmente apenas X e Y têm essa possibilidade (pelas produções $X \to \varepsilon$ e $Y \to \varepsilon$, respetivamente). Olhando depois para os corpos das restantes produções, podemos ver que S também é anulável (pela produção $S \to X$). Então, para obter a gramática resultante da aplicação deste primeiro passo, devemos retirar as produções-ε da gramática, e acrescentar novas produções para os casos em que X, Y ou S possam não existir. Ficamos então com a seguinte gramática:
$S \to traX \mid X \mid tra$
$X \to laX \mid la$

$Y \to alY \mid al$

No segundo passo, para eliminar as produções unitárias, podemos identificar apenas uma produção unitária: $S \to X$, pelo que temos apenas de a substituir pelas produções de X, ficando com:
$S \to traX \mid laX \mid la \mid tra$
$X \to laX \mid la$
$Y \to alY \mid al$

Finalmente, no terceiro passo eliminam-se símbolos inúteis. Apesar de todos os símbolos serem geradores, nem todos são atingíveis: Y não é atingível a partir de S. Então, no final, ficamos com apenas duas produções:
$S \to traX \mid laX \mid la \mid tra$
$X \to laX \mid la$

Com exceção da cadeia vazia (que não é contemplada por gramáticas na CNF), a linguagem aceite por esta gramática é a mesma que a aceite pela grmática original (isto é, cadeias iniciadas por 'tra', seguidas de um qualquer número de 'la's).

Falta agora apenas colocar a gramática no formato final, em que todas as produções dão origem a duas variáveis ou a um terminal. Podemos começar por definir novas variáveis para produção dos terminais:
$T \to t$
$R \to r$
$A \to a$
$L \to l$

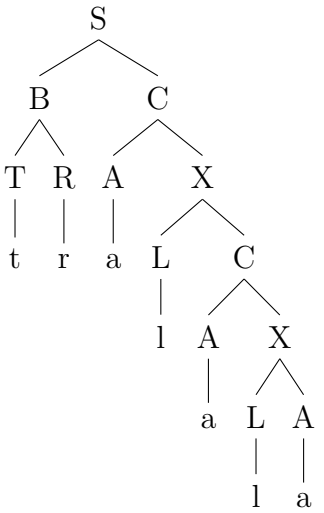
Fazendo a substituição, ficamos com as seguintes produções:
$S \to TRAX \mid LAX \mid LA \mid TRA$
$X \to LAX \mid LA$

Acrescentando as produções
$B \to TR$
$C \to AX$

Podemos substituir e ficar com a gramática final:
$S \to BC \mid LC \mid LA \mid BA$
$X \to LC \mid LA$
$B \to TR$
$C \to AX$
$T \to t$

$R \to r$
$A \to a$
$L \to l$

Podemos ver que a gramática não é tão simples de interpretar, mas tem a vantagem de produzir apenas árvores binárias: todos os nós derivam dois novos nós, ou uma folha. Por exemplo, para a cadeia *tralala* teremos a seguinte árvore:



Exercício 1 (Desafio 2014/15)

Converta a seguinte gramática para a Forma Normal de Chomsky e apresente árvores de análise para a cadeia 1001001 na gramática original e na CNF.
$S \to 1AB0 \mid 0BA1 \mid AA \mid BB \mid SS$
$A \to 0A \mid \varepsilon$
$B \to 1B \mid \varepsilon$

Para converter a gramática para CNF, necessitamos então de seguir os três passos de simplificação. Para eliminar as produções-ε, começamos por identificar que tanto A como B podem produzir ε, e por isso são anuláveis. S, por sua vez, tem pelo menos uma produção composta apenas por símbolos anuláveis ($S \to AA$), e será por isso também anulável. Então, é necessário eliminar as produções-ε e acrescentar novas produções, ficando a gramática nesta forma:

$S \to 1AB0 \mid 0BA1 \mid AA \mid BB \mid SS \mid 1B0 \mid 1A0 \mid 10 \mid 0B1 \mid 0A1 \mid 01 \mid A \mid B \mid S$
$A \to 0A \mid 0$
$B \to 1B \mid 1$

No próximo passo, é necessário identificar as produções unitárias, e vemos que existem em S três produções unitárias ($S \to A \mid B \mid S$). Como não há problemas de ciclos, podemos fazer diretamente a substituição pelas produções das variáveis respetivas, ficando assim com a seguinte gramática:
$S \to 1AB0 \mid 0BA1 \mid AA \mid BB \mid SS \mid 1B0 \mid 1A0 \mid 10 \mid 0B1 \mid 0A1 \mid 01 \mid 0A \mid 0 \mid 1B \mid 1$
$A \to 0A \mid 0$
$B \to 1B \mid 1$

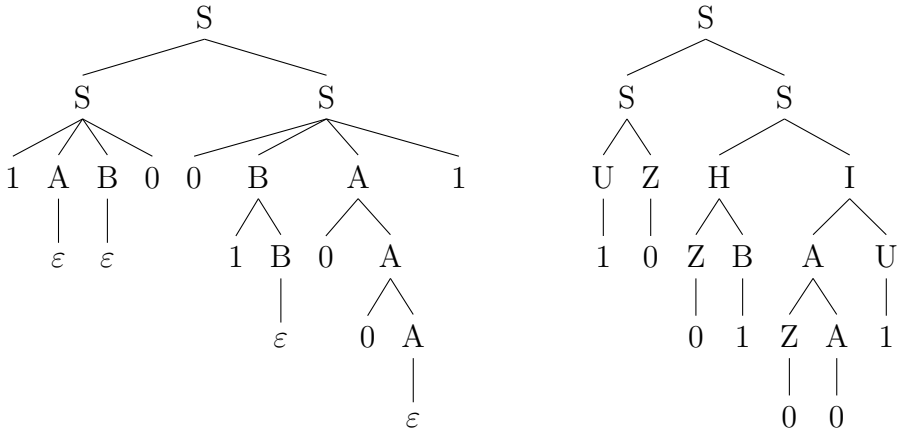
De seguida, temos de identificar símbolos não geradores, ou não atingíveis. Neste caso, todos os símbolos geram terminais, pelo que todos são geradores. Começando em S, é também possível atingir todos os restantes símbolos, pelo que não existem também símbolos não atingíveis.

Então, falta apenas o passo final de colocar todas as produções no formato da CNF. Começamos por criar novas variáveis para os terminais da linguagem:
$Z \to 0$
$U \to 1$

Substituímos agora na gramática as ocorrências de terminais não isolados pelas novas variáveis:
$S \to UABZ \mid ZBAU \mid AA \mid BB \mid SS \mid UBZ \mid UAZ \mid UZ \mid ZBU \mid ZAU \mid ZU \mid ZA \mid 0 \mid UB \mid 1$
$A \to ZA \mid 0$
$B \to UB \mid 1$
$Z \to 0$
$U \to 1$

Agora, falta apenas adicionar novas variáveis para reduzir produções de tamanho maior do que 2. Uma solução possível seria:
$S \to FG \mid HI \mid AA \mid BB \mid SS \mid UG \mid FZ \mid UZ \mid HU \mid ZI \mid ZU \mid ZA \mid 0 \mid UB \mid 1$
$A \to ZA \mid 0$
$B \to UB \mid 1$
$F \to UA$
$G \to BZ$
$H \to ZB$
$I \to AU$
$Z \to 0$
$U \to 1$

Em relação à cadeia 1001001, temos então as seguintes árvores:



A árvore da esquerda é obtida usando a gramática original, sendo uma possível árvore para a cadeia. A árvore da direita é obtida usando a gramática na CNF, sendo portanto uma árvore binária (note-se que cada nó interno ou tem dois filhos que são também nós, ou um filho que é folha).

Exercício 2

Converta a seguinte gramática para a CNF.
$S \to aXYb \mid bYXa$
$X \to XaaX \mid \varepsilon$
$Y \to YbY \mid YcT \mid \varepsilon$
$T \to XY \mid Z$
$Z \to cT$

Novamente, vamos seguir os vários passos de simplificação. Começamos por eliminar das produções-ε, identificando que tanto X como Y podem produzir ε, e por isso são anuláveis. Por sua vez, T também é anulável uma vez que tem uma produção $T \to XY$, que é composta apenas por símbolos anuláveis. Então, é necessário eliminar as produções-ε e acrescentar novas produções, ficando a gramática nesta forma:
$S \to aXYb \mid bYXa \mid aYb \mid aXb \mid ab \mid bYa \mid bXa \mid ba$
$X \to XaaX \mid aaX \mid Xaa \mid aa$
$Y \to YbY \mid YcT \mid bY \mid Yb \mid b \mid cT \mid Yc \mid c$
$T \to XY \mid Z \mid X \mid Y$

$Z \to cT \mid c$

Agora, identificamos as produções unitárias, e vemos que existem em T três produções unitárias ($T \to A|B|S$), sendo as únicas existentes. Como não há problemas de ciclos, fazemos diretamente a substituição pelas produções das variáveis respetivas, ficando assim com a seguinte gramática:
$S \to aXYb \mid bYXa \mid aYb \mid aXb \mid ab \mid bYa \mid bXa \mid ba$
$X \to XaaX \mid aaX \mid Xaa \mid aa$
$Y \to YbY \mid YcT \mid bY \mid Yb \mid b \mid cT \mid Yc \mid c$
$T \to XY \mid cT \mid c \mid XaaX \mid aaX \mid Xaa \mid aa \mid YbY \mid YcT \mid bY \mid Yb \mid b \mid Yc$
$Z \to cT \mid c$

Agora identificamos símbolos não geradores ou não atingíveis. Neste caso, todos os símbolos geram terminais, pelo que todos são geradores. Começando em S, consegue-se atingir X e Y, e a partir de Y atinge-se T, mas nenhuma destas variáveis permite atingir Z, pelo que Z pode ser eliminada por ser uma produção não-atingível e portanto inútil.
$S \to aXYb \mid bYXa \mid aYb \mid aXb \mid ab \mid bYa \mid bXa \mid ba$
$X \to XaaX \mid aaX \mid Xaa \mid aa$
$Y \to YbY \mid YcT \mid bY \mid Yb \mid b \mid cT \mid Yc \mid c$
$T \to XY \mid cT \mid c \mid XaaX \mid aaX \mid Xaa \mid aa \mid YbY \mid YcT \mid bY \mid Yb \mid b \mid Yc$

Falta agora apenas o passo final de colocar todas as produções no formato da CNF. Começamos por criar novas variáveis para os terminais da linguagem:
$A \to a$
$B \to b$
$C \to c$

Substituindo na gramática as ocorrências de terminais não isolados pelas novas variáveis:
$S \to AXYB \mid BYXA \mid AYB \mid AXB \mid AB \mid BYA \mid BXA \mid BA$
$X \to XAAX \mid AAX \mid XAA \mid AA$
$Y \to YBY \mid YCT \mid BY \mid YB \mid b \mid CT \mid YC \mid c$
$T \to XY \mid CT \mid c \mid XAAX \mid AAX \mid XAA \mid AA \mid YBY \mid YCT \mid BY \mid YB \mid b \mid YC$

Agora, falta apenas adicionar novas variáveis para reduzir produções de tamanho maior do que 2. Uma solução possível seria:
$S \to JK \mid LM \mid AK \mid JB \mid AB \mid LA \mid BM \mid BA$
$X \to MJ \mid AJ \mid MA \mid AA$
$Y \to KY \mid YN \mid BY \mid YB \mid b \mid CT \mid YC \mid c$

$T \to XY \mid CT \mid c \mid MJ \mid AJ \mid MA \mid AA \mid KY \mid YN \mid BY \mid YB \mid b \mid YC$
$J \to AX$
$K \to YB$
$L \to BY$
$M \to XA$
$N \to CT$
$A \to a$
$B \to b$
$C \to c$

9.2 Propriedades de Fecho das CFLs

Tal como as linguagens regulares, existe um conjunto de propriedades que se aplicam às linguagens livres de contexto, incluindo um conjunto de operações que produzem como resultado também uma linguagem livre de contexto.

Exemplos de operações fechadas para as linguagens livres de contexto são a união, a concatenação, o fecho, o reverso, homomorfismo e homomorfismo inverso. A interseção com uma linguagem regular é uma operação fechada, muito embora a interseção com uma linguagem livre de contexto não resulte garantidamente numa linguagem livre de contexto.

Um exemplo poderá ser visto olhando para as linguagens $A = \{a^i b^j c^k, i = j\}$ e $B = \{a^i b^j c^k, j = k\}$, ambas linguagens livres de contexto (embora nenhuma das duas seja uma linguagem regular). A interseção destas duas linguagens será a linguagem $C = A \cap B = \{a^i b^j c^k, i = j = k\}$, a qual não é uma linguagem livre de contexto, uma vez que não é possível encontrar uma gramática ou um autómato de pilha para esta linguagem (mais à frente poderemos ver uma prova forma em como esta linguagem não é uma CFL).

Exercício 1

Considere as linguagens L1 e L2, sendo L1 definida pela gramática
$A \to 0A1 \mid \varepsilon$
e L2 por
$B \to aBb \mid \varepsilon$.
Apresente gramáticas para as linguagens $L3 = L1 \cup L2$, $L4 = L2.L1$, $L5 = L1^*$ e $L6 = L2^R$.

A operação de união entre duas CFLs é relativamente simples de fazer: a linguagem resultado vai ter as mesmas produções que as duas CFLs a unir

(com o devido cuidado no caso de haver nomes de variáveis repetidas entre as duas), mais uma nova variável, que será a variável de arranque da linguagem resultado, que vai ter como produções as antigas variáveis de arranque das linguagens originais. Então, L3 poderá ser definida pela seguinte gramática (com C como variável de arranque):
$C \to A \mid B$
$A \to 0A1 \mid \varepsilon$
$B \to aBb \mid \varepsilon$

A concatenação de duas CFLs é também simples de realizar, bastando acrescentar uma nova variável que contenha uma produçao que concatene as variáveis de arranque das duas gramáticas originais. Assim, L4 poderá ser definida pela seguinte gramática (com D como variável de arranque):
$D \to AB$
$A \to 0A1 \mid \varepsilon$
$B \to aBb \mid \varepsilon$

O fecho de uma CFL é também simples de fazer, bastando ter uma nova variável de arranque que permita ter 0 ou mais repetições da antiga variável de arranque da linguagem original. Então, L5 poderá ser definida pela seguinte gramática (com E como variável de arranque):
$E \to AE \mid \varepsilon$
$A \to 0A1 \mid \varepsilon$

O reverso de uma gramática é um pouco mais trabalhoso, sendo necessário reverter a ordem dos símbolos em todas as produções da gramática. Então, L6 pode ser definida pela seguinte gramática:
$B \to bBa \mid \varepsilon$

Exercício 2

Partindo do conhecimento que L1 é uma CFL e L2 não é uma CFL, e que $L1 \cap L3 = L2$ o que pode dizer acerca de L3?

Ora, sabemos que a interseção não é uma operação fechada para as CFLs, mas também sabemos que a interseção entre uma CFL e uma linguagem regular resulta numa CFL. Então, se sabemos que L2 não é uma CFL, e que L1 é uma CFL, sabemos que L3 não pode ser uma linguagem regular (se o fosse, então o resultado da interseção com L1 seria também uma CFL). Por

outro lado, não podemos afirmar mais nada sobre L3 (isto é, sobre se esta é ou não uma CFL).

Exercício 3 (Desafio 2015/16)

Partindo do conhecimento de que a linguagem $L1 = \{a^p b^p c^p b^q a^q : p, q \geq 0\}$ não é uma linguagem livre de contexto (CFL) e que a linguagem $L2 = 0^n 1^n : n \geq 0$ é uma CFL, prove, usando as propriedades de fecho das CFLs, que a linguagem $L3 = a^n b^n c^n : n \geq 0$ não é uma CFL.

Para fazer a prova, é necessário usar um conjunto de operações fechadas para as linguagens livres de contexto. Para podemos fazer a prova, temos de começar por ter todas as linguagens no mesmo alfabeto. Usando o homomorfismo, podemos, em L2, substituir os 0's por b's e os 1's por a's, obtendo assim a linguagem $L2' = \{b^n a^n : n \geq 0\}$. Observando as linguagens $L1$, $L2'$ e $L3$, podemos constatar que existe uma relação entre elas: $L1 = L3.L2'$. Como a concatenação é uma operação fechada para as CFLs, sabemos que se ambas $L3$ e $L2'$ fossem CFLs então $L1$ seria também uma CFL. No entanto, como sabemos que $L1$ não é uma CFL, então pelo menos um dos operandos também não será uma CFL. Como sabemos que $L2'$ é uma CFL (uma vez que foi obtida a partir de $L2$ aplicando uma operação fechada para as CFLs), então $L3$ não poderá ser uma CFL.

9.3 Propriedades de Decisão das CFLs

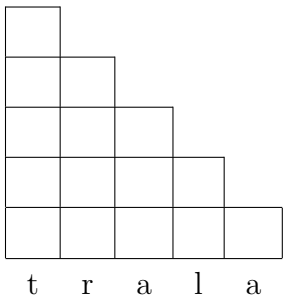
Entende-se por propriedades de decisão a capacidade de responder a um conjunto de questões relativamente a CFLs. Curiosamente, existem várias questões para as quais não são atualmente conhecidos algoritmos capazes de dar uma resposta. São exemplos disso questões relativas à ambiguidade de uma gramática ou de uma linguagem: não existe uma forma universal de provar que uma dada gramática (ou linguagem) é ambígua ou não. Da mesma forma, não existe também forma de provar se duas CFGs (ou PDAs) definem a mesma linguagem, ou se a interseção de duas CFGs (ou PDAs) resulta na linguagem vazia (ou seja, se não têm qualquer cadeia em comum).

Uma das questões respondidas aqui é relativamente à pertença de uma cadeia à linguagem definida por uma gramática. Um dos algoritmos usados para verificar essa pertença é o algoritmos de Cocke-Younger-Kasami (CYK),

o qual se baseia numa construção da árvore de análise da cadeia de baixo para cima. Este método necessita que a gramática da linguagem esteja na Forma Normal de Chomsky, algo que como foi visto anteriormente, é sempre possível de obter para qualquer CFG.

Começa-se por desenhar uma tabela triangular, tendo na base da tabela a cadeia a testar. Depois, essa tabela é preenchida de baixo para cima, verificando para cada posição da tabela qual ou quais as variáveis que podem dar origem às subcadeias respetivas. No final, caso a variável de arranque esteja presente no topo da tabela, significa que é possível obter a cadeia com a gramática. Vamos ver um exemplo, usando a gramática que foi já vista acima em CNF e testando a pertença da cadeia *trala*. Recordando a gramática já em CNF:
$S \to BC \mid LC \mid LA \mid BA$
$X \to LC \mid LA$
$B \to TR$
$C \to AX$
$T \to t$
$R \to r$
$A \to a$
$L \to l$

Começamos então por desenhar uma tabela triangular que na base tem 5 casas, tantas quantas o número de símbolos na cadeia.



Podemos interpretar o conteúdo de cada célula como expresso na seguinte tabela, em que a célula X_{ij} vai conter as variáveis que permitem gerar a subcadeia i-j. Por exemplo, a célula X_{24} será preenchida com as variáveis que permitam gerar a subcadeia 2-4, ou seja, *ral*. Então, a célula X_{15} irá conter as variáveis capazes de gerar a cadeia completa, daí que se esta célula contiver a variável de arranque da gramática, significa que a cadeia pertence à gramática.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
t	r	a	l	a

Começamos então por preencher a linha de baixo da tabela com a variável ou variáveis que permitem gerar a sub-cadeia respetiva, ou seja, o símbolo que está debaixo dessa célula. Neste exemplo, olhando para a gramática, temos apenas uma variável capaz de gerar cada um dos símbolos, pelo que preenchemos as células com esses símbolos respetivos (note-se que como podemos ter mais do que uma variável, devemos sempre usar conjuntos):

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
{T}	{R}	{A}	{L}	{A}
t	r	a	l	a

Passando agora para a segunda linha da tabela, vamos ver quais as variáveis que permitem gerar sub-sequências de comprimento dois. Para isso, vamos olhar para a célula imediatamente abaixo e para a célula em baixo à direita. Então, para X_{12} temos de olhar para as células X_{11} e X_{22}, e ver todas as variáveis que permitam gerar uma qualquer das possíveis combinações (sequências) de variáveis. Neste caso, temos apenas uma combinação, TR, e olhando para a gramática, apenas uma variável que permite produzir TR: B. X_{12} fica então preenchido com {B}. Da mesma forma, X_{23} irá ter as variáveis que permitem obter RA (não havendo nenhuma, a célula fica vazia). X_{34} irá conter as variáveis capazes de gerar AL (novamente, nenhuma), e finalmente X_{45} irá conter as variáveis capazes de gerar LA (existem duas, S e X). Ficamos então com a seguinte tabela:

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
$\{B\}$			$\{S, X\}$	
{T}	{R}	{A}	{L}	{A}
t	r	a	l	a

Nas linhas seguintes, já fica um pouco mais complexo: como as sub-cadeias já terão comprimento maior do que dois, temos de ver todas as hipóteses de gerar essa sub-cadeia. Isso implica que não iremos ver apenas duas células, mas vários pares de células capazes de gerar cada sub-cadeia. As células a consultar ficam sempre na vertical e na diagonal da célula a analisar, e uma está tanto mais próxima como a outra estará mais distante.

No caso da célula X_{13}, vamos olhar para o par constituído pela célula imediatamente abaixo (X_{12}) e pela mais distante na diagonal (X_{33}), e depois para o par constituído pela célula mais abaixo verticalmente (X_{11}), e mais próxima diagonalmente (X_{23}). No primeiro caso, temos apenas uma combinação possível, que resulta na sequência BA, a qual pode ser produzida pela variável S. No segundo caso, não temos nenhuma sequência possível, dado que a célula X_{23} está vazia.

No caso da célula X_{24}, vamos olhar para o par X_{23} e X_{44}, e depois para o par X_{22} e X_{34}. Tanto no primeiro como no segundo casos não existem quaisquer sequências possíveis, dado que as célula X_{23} e X_{34} estão vazias.

Finalmente, para a célula X_{35}, vamos olhar para o par X_{34} e X_{55} e depois para o par X_{33} e X_{45}. No primeiro caso temos uma célula vazia, pelo que não temos nenhuma possibilidade. No segundo caso, temos duas combinações possíveis, as sequências AS e AX, sendo que apenas a variável C é capaz de gerar a sequência AX. Ficamos então com a seguinte tabela:

X_{15}				
X_{14}	X_{25}			
$\{S\}$		$\{C\}$		
$\{B\}$			$\{S, X\}$	
{T}	{R}	{A}	{L}	{A}
t	r	a	l	a

Passando para a linha seguinte, vamos então olhar para as duas células.

No caso da célula X_{14}, temos de analisar o par X_{13} e X_{44}, seguido do par X_{12} e X_{34}, seguido do par X_{11} e X_{24}. No primeiro par, temos apenas uma combinação que gera a sequência SL. No entanto, não há nenhuma variável capaz de gerar esta sequência, pelo que não teremos aqui nenhuma variável. No segundo par, temos uma célula vazia, pelo que não teremos também nenhuma variável a acrescentar. Finalmente, no terceiro par, temos também uma célula vazia. Como nenhum dos pares tem nenhuma possibilidade de geração, esta célula irá ficar vazia.

No caso da célula X_{25}, temos de analisar o par X_{24} e X_{55}, seguido do par X_{23} e X_{45}, seguido do par X_{22} e X_{35}. Tanto no primeiro par como no segundo deparamo-nos com uma célula vazia, pelo que estes pares não irão contribuir com nenhuma variável. No terceiro par, temos uma combinação possível, que resulta na sequência RC, a qual, no entanto, não tem qualquer variável capaz de a gerar, ficando esta célula também vazia.

Não havendo alterações na tabela, passamos finalmente para a última linha, olhando para a célula X_{15}. Temos aqui de analisar o par X_{14} e X_{55}, seguido do par X_{13} e X_{45}, seguido do par X_{12} e X_{35}, seguido do par X_{11} e X_{25}. Para o primeiro caso, temos uma célula vazia (X_{14}), pelo que não resulta em nenhuma variável. No segundo caso, temos duas combinações possíveis, as sequências SS e SX, sendo, no entanto, que nenhuma delas é possível de gerar com a gramática. No terceiro caso, temos uma combinação, a sequência BC, a qual é possível de gerar com a variável S. Finalmente, no quarto caso, temos também uma célula vazia, pelo que não teremos mais nenhuma variável a acrescentar. Ficamos então com a seguinte tabela no final:

$\{S\}$				
$\{S\}$		$\{C\}$		
$\{B\}$			$\{S, X\}$	
{T}	{R}	{A}	{L}	{A}
t	r	a	l	a

Como a célula do topo da tabela contém a variável de arranque da gramática (S), podemos concluir que a cadeia efetivamente pertence à linguagem definida por esta gramática.

9.4 Lema da Bombagem para CFLs

Tal como acontece com o Lema da Bombagem para as linguagens regulares, também para as CFLs existe uma forma de provar que uma linguagem não é uma linguagem livre de contexto. O lema da bombagem para CFLs funciona de forma semelhante ao lema da bombagem para linguagens regulares, com as necessárias alterações.

À semelhança do que acontecia nas linguagens regulares, também aqui caso L seja uma linguagem livre de contexto, então L segue o Lema da Bombagem para CFLs. Novamente, aquilo que nos interessa é provar que uma linguagem não é uma CFL, o que acontece quando a linguagem não segue o Lema da Bombagem para CFLs.

O lema aqui baseia-se no facto de, numa gramática finita, cadeias muito longas necessitarem de provocar repetição de produções. Pensando em termos da árvore de análise de uma cadeia longa, haverá necessariamente partes internas da árvore com estrutura repetida, e é essa repetição que é explorada no lema.

Então, aquilo que o lema nos diz é que se L for uma CFL, então existe uma constante m (um tamanho mínimo, dependente da linguagem), tal que todas as cadeias z pertencentes a L e com um tamanho maior ou igual a m podem ser decompostas em cinco partes ($z = uvwxy$), em que $vx \neq \varepsilon$ (ou seja, pelo menos um dos elementos v e x será não vazio), e $|vwx| \leq m$ (ou seja, a parte do meio da cadeia não é demasiado comprida), tal que para $k \geq 0$, as cadeias geradas por repetição de v e x, no formato uv^kwx^ky, também pertencem a L. De uma forma simplificada, será algo como:

$$\exists m : \forall z \in L, |z| \geq m : \exists z = uvwxy, vx \neq \varepsilon, |vwx| \leq m : \forall k : uv^kwx^ky \in L$$

Para fazer a prova, temos de negar o lema, o que implica inverter todos os quantificadores. Então, temos que para uma linguagem L não seguir o lema, qualquer que seja o valor de m (por muito grande que seja), existe pelo menos uma cadeia z pertencente a L com comprimento maior ou igual a m, tal que qualquer que seja a divisão dessa cadeia em cinco partes, existem pelo menos um k tal que uv^kwx^ky não pertence a L. Novamente, de uma forma simplificada, será algo como:

$$\forall m : \exists z \in L, |z| \geq m : \forall z = uvwxy, vx \neq \varepsilon, |vwx| \leq m : \exists k : uv^kwx^ky \notin L$$

Vamos ver um exemplo para a linguagem L = $\{\ vv : v \in \{0,1\}^*\ \}$, ou seja cadeias binárias de comprimento par que podem ser divididas em duas metades iguais.

Se tentássemos construir um PDA ou uma CFG para a linguagem iríamos ver que não seria possível fazê-lo. Vamos então ver uma prova mais formal em como L não é uma CFG.

Começamos por escolher uma cadeia que pertença a L e cujo comprimento seja maior do que m, como por exemplo a cadeia $z = 0^m1^m0^m1^m$, com comprimento $4m > m$. Temos agora de ver as divisões possíveis da cadeia tal que $z = uvwxy$ em que $vx \neq \varepsilon$ e $|vwx| \leq m$:

- No caso de vwx abranger apenas a primeira ou apenas a segunda metade da cadeia, então bombeamentos com $k \neq 1$ irão originar cadeias não pertencentes a L (a outra metade da cadeia irá permanecer inalterada, e não acompanha a parte bombeada).

- No caso de vwx abranger parte da primeira metade da cadeia e parte da segunda metade, então um bombeamento também causará cadeias não pertencentes à linguagem (poderão ser bombeados 1's da primeira metade da cadeia e/ou 0's da segunda metade da cadeia, não sendo acompanhados pelas partes que se mantém inalteradas).

Na realidade a única hipótese de continuar a gerar cadeias pertencentes à linguagem seria se v contivesse um ou mais zeros (ou uns) da primeira metade da cadeia e x o mesmo número de zeros (ou uns) da segunda metade da cadeia. No entanto, para isso acontecer, vwx seria necessariamente maior do que m (comprimento mínimo de $m + 2$ para abranger um mínimo de um dígito de cada lado), pelo que tal divisão não é possível.

Então, como vimos que para qualquer divisão possível da cadeia, existe sempre pelo menos um valor de k que gera cadeias não pertencentes à linguagem, provamos que a linguagem L não é uma CFL.

Exercício 1 (TPC 2011/12)

Considere a linguagem $L1 = \{0^n10^n1, n \geq 0\}$. Mostre que L1 obedece ao Lema da Bombagem para CFLs. L1 é uma CFL?

Para provar que a linguagem segue o Lema, devemos encontrar um valor de m, a partir do qual todas as cadeias da linguagem tenham uma divisão que respeite as condições impostas pelo Lema, e que resulte em cadeias aceites ao bombear as partes repetíveis da cadeia. Neste caso concreto, podemos fazer a prova para m = 4. Cadeias com tamanho superior ou igual a 4 terão o formato $0^a10^a1, a \geq 1$, as quais podem ser divididas da seguinte forma: $u = 0^{a-1}$; $v = 0$; $w = 1$; $x = 0$; $y = 0^{a-1}1$. Com esta divisão, válida para qualquer

cadeia com tamanho igual ou superior a 4, qualquer que seja o valor de k, a cadeia uv^kwx^ky continua a pertencer à linguagem L1. Para $k = 0$, teremos a cadeia $uwy = 0^{a-1}10^{a-1}1$; como $a \geq 1$, resulta numa cadeia pertencente a L1. Para $k > 1$, teremos cadeias no formato $0^{a-1}0^k10^k0^{a-1}1 = 0^{a+k-1}10^{a+k-1}1$, as quais pertencem também a L1.

Embora esta prova em como a linguagem segue o Lema da Bombagem não seja prova de que a linguagem é uma CFL, podemos tentar criar uma CFG ou um PDA para a linguagem, provando assim que a linguagem é uma CFL. Podemos ter então a seguinte gramática:

$S \to A1$

$A \to 0A0 \mid 1$

Exercício 2 (TPC 2011/12)

Prove, usando o Lema da Bombagem para CFLs, que a linguagem $L = \{0^n1^n0^n1^n, n \geq 0\}$ não é uma CFL.

Ora para provar que L não é uma CFL usando o Lema da Bombagem começamos por escolher uma cadeia da linguagem. Vamos escolher $z = 0^m1^m0^m1^m$, a qual tem comprimento $4m$, e por isso respeita as condições do Lema. Temos agora de verificar todas as divisões possíveis da cadeia em cinco partes. Dadas as limitações da divisão, nomeadamente a condição de que $|vwx| \leq m$, a parte 'do meio' da cadeia (vwx) não pode conter mais do que duas 'partes' consecutivas da cadeia (considerando cada dígito como uma parte, a cadeia terá então 4 partes), podendo conter apenas uma ou duas delas.

- No caso de vwx incluir apenas um dos dígitos, repetições de v e x irão gerar cadeias em que esse dígito se repete mais vezes do que os restantes, gerando assim cadeias que não pertencem a L.

- No caso de vwx incluir dois dígitos, repetições de v e x irão gerar cadeias em que essas duas partes irão ter tamanho superior às restantes duas partes.

Estes dois casos abrangem então todas as divisões possíveis da cadeia, pelo que conseguimos provar que L não segue eo Lema da Bombagem, e não é portanto uma CFL.

Exercício 3

Tente provar, usando o Lema da Bombagem para CFLs, que a linguagem

L=$\{a^n b^n : n \geq 0\}$ não é uma CFL.

Começamos novamente por escolher uma cadeia pertencente à linguagem, como por exemplo $z = a^m b^m$, que tem comprimento $2m$. De seguida, temos de ver todas as divisões possíveis da cadeia em cinco partes.

Uma das divisões possíveis é $z = uvwxy$ em que $u = a^{n-1}$, $v = a$, $w = \varepsilon$, $x = b$ e $y = b^{n-1}$. Para $k = 0$, a nova cadeia será $z' = uwy = a^{n-1}b^{n-1}$, que pertence à linguagem; para $k \geq 1$, teremos as novas cadeias como $z' = uv^k wx^k y$, o que resulta em $a^{n-1+k}b^{n-1+k}$, que também pertencem à linguagem. Então, havendo uma divisão para a qual todos os valores de k resultam em cadeias pertencentes à linguagem, significa que o Lema é cumprido no caso da cadeia escolhida.

Poder-se-ia tentar escolher outra cadeia com a qual fazer a prova, mas qualquer todas as cadeias terão o mesmo formato que a escolhida, pelo que será sempre possível realizar a divisão da cadeia como feito acima, e mostrar que o Lema é seguido. Não conseguimos então provar que a linguagem não é uma CFL (efetivamente, se tentarmos obter um PDA ou uma CFG para a linguagem, facilmente podemos ver que é possível fazê-lo e que a linguagem é, de facto, uma CFL).

Exercício Proposto 1 Encontre árvores de análise alternativas (gramática original e na CNF) às obtidas no **Exercício 1** da secção Simplificação de CFGs e Forma Normal de Chomsky para a mesma cadeia *1001001*.

Exercício Proposto 2 Considerando a gramática do mesmo **Exercício 1** da secção Simplificação de CFGs e Forma Normal de Chomsky, verifique, usando o algoritmo CYK, se as cadeias 10110 e 0011 pertencem à linguagem definida pela gramática.