Departamento de Ciência de Computadores — FCUP

Modelos de Computação (CC1004) — 2014/15

Exame (15.06.2015) — Cotação: 4×2.5, 1.5+1, 0.5+2, 1+2.5, 1.5 — *duração: 3h*

Resolução de questões selecionadas (v1)

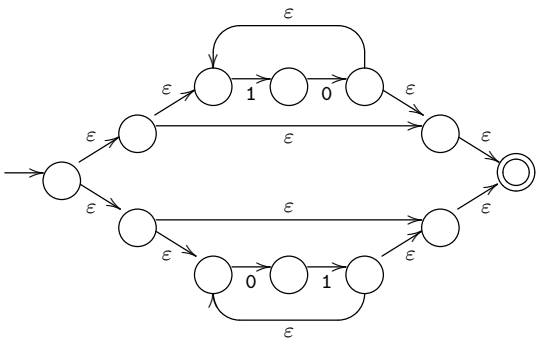
O alfabeto Σ é $\{0, 1\}$, exceto em **6.**.

Grupo I – Resolva quatro dos cinco problemas

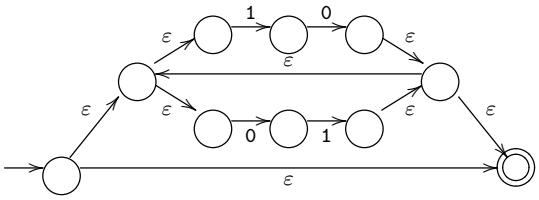
1. Sejam $r = (((10)^\star) + ((01)^\star))$ e $s = (((10) + (01))^\star)$ expressões regulares sobre Σ. Desenhe os diagramas de transição dos AFNDs-ε que se obtêm por aplicação do método de Thompson a essas expressões, de acordo com a construção definida nas aulas.

Resposta:

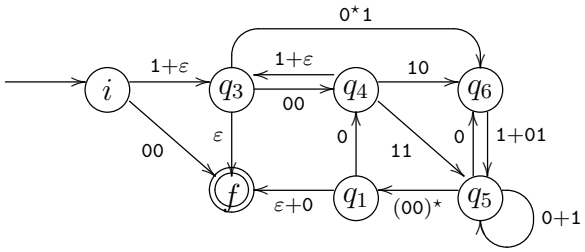
Para $r = (((10)^\star) + ((01)^\star))$:



Para $s = (((10) + (01))^\star)$:



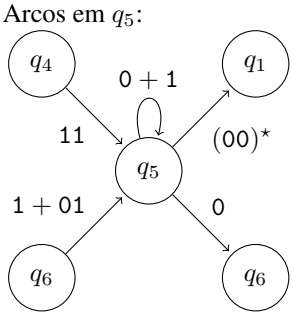
2. Assuma que o diagrama seguinte foi obtido de um automáto finito, de alfabeto Σ, após algumas iterações do método de eliminação de estados. Desenhe o diagrama que se obtém no **passo seguinte** se se eliminar q_5.



Não simplifique as expressões que obtiver e ilustre como efetuou a eliminação de q_5.

Resposta:

Arcos em q_5:



Novas transições:

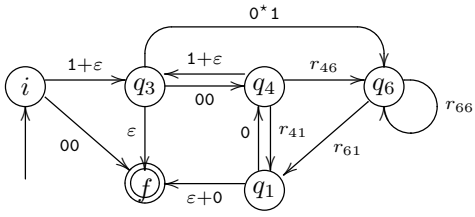
$q_4 \xrightarrow{r_{41}=11(0+1)^\star(00)^\star} q_1$

$q_4 \xrightarrow{r_{46}=11(0+1)^\star 0 \ + \ 10} q_6$

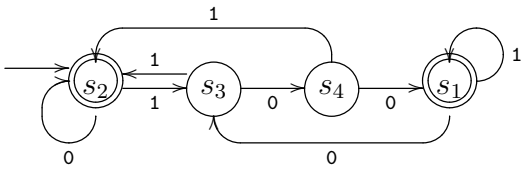
$q_6 \xrightarrow{r_{61}=(1+01)(0+1)^\star(00)^\star} q_1$

$q_6 \xrightarrow{r_{66}=(1+01)(0+1)^\star 0} q_6$

Novo diagrama (r_{41}, r_{46}, r_{61} e r_{66} à esquerda):

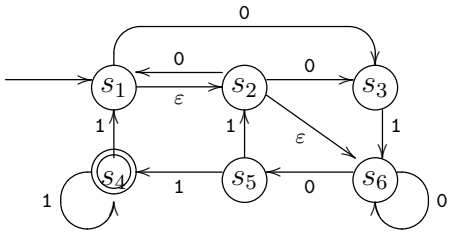


3. Seja A o AFD representado a seguir. Determine um AFD que reconheça $\mathcal{L}(A)^R$, isto é a linguagem reversa de $\mathcal{L}(A)$, por aplicação de métodos de conversão dados. Apresente os passos principais.



Sugestão: Aplicar o método dado nas aulas para transformar o autómato A de modo a obter um AFND-ε que reconhece $\mathcal{L}(A)^R$. A seguir, por aplicação da construção baseada em subconjuntos, converter esse AFND-ε num AFD equivalente. (o AFD vai ter cerca de 16 estados)

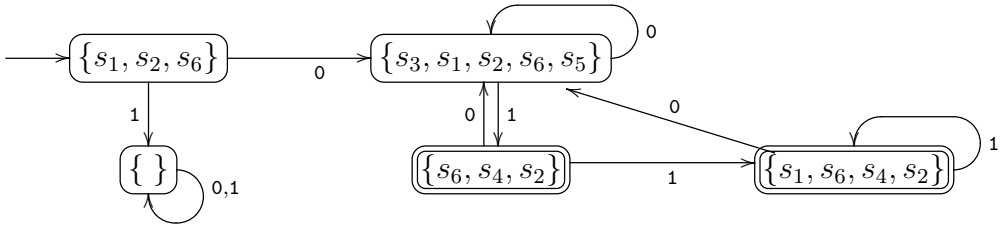
4. Usando a construção baseada em subconjuntos, converta o AFND-ε seguinte num AFD equivalente.



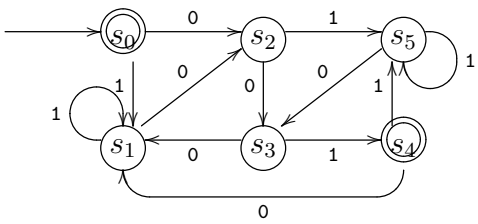
Indique apenas estados acessíveis do *estado inicial do AFD* e use **conjuntos** para designar os estados.

Resposta:

O estado inicial do AFD equivalente é $Fecho_\varepsilon(s_1) = \{s_1, s_2, s_6\}$ e o diagrama de transição é:



5. Aplicando o algoritmo de Moore, determine o AFD mínimo equivalente ao AFD seguinte.

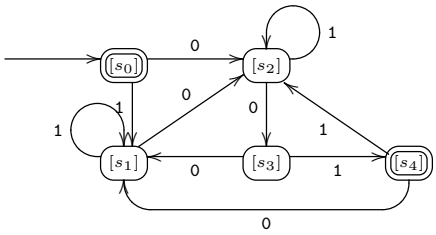


Não desenhe duas tabelas. Use $\boxed{\equiv}$ e $\boxed{\text{x}}$ para assinalar as entradas na *segunda* fase. Inclua as anotações intermédias.

Resposta:

s_0	$\equiv$					
s_1	**X**	$\equiv$				
s_2	**X**	(s_0, s_4) ?/ X	$\equiv$			
s_3	**X**	X	(s_1, s_2) (s_1, s_5) X	$\equiv$		
s_4	?/ X	**X**	**X**	**X**	$\equiv$	
s_5	**X**	(s_0, s_4) (s_1, s_2) ?/ X	$\equiv$	X	**X**	$\equiv$
	s_0	s_1	s_2	s_3	s_4	s_5

O AFD mínimo equivalente é:



1ª fase: Assinalar com $\equiv$ os pares (s_k, s_k), com $k \in \{0, 1, 2, 3, 4, 5\}$ (qualquer estado é equivalente a si mesmo); e com **X** os pares (s_i, s_j), com $i \in \{0, 4\}$ e $j \in \{1, 2, 3, 5\}$ (os estados finais não são equivalentes a estados não finais).

2ª fase: analisar as restantes entradas, começando pela coluna mais à esquerda:

- (s_0, s_4)
  - $\delta(s_0, 0) = s_2$ e $\delta(s_4, 0) = s_1$; nada se conclui pois ainda não há decisão sobre (s_2, s_1).
  - $\delta(s_0, 1) = s_1$ e $\delta(s_4, 1) = s_5$; nada se conclui pois ainda não há decisão sobre (s_1, s_5).
  - $s_0 \not\equiv s_4$ se e só se $s_2 \not\equiv s_1$ ou $s_1 \not\equiv s_5$; (s_0, s_4) fica pendente (?). Registar (s_0, s_4) em (s_1, s_2) e (s_1, s_5).
- (s_1, s_2)
  - $\delta(s_1, 0) = s_2$ e $\delta(s_2, 0) = s_3$; nada se conclui pois ainda não há decisão sobre (s_2, s_3).
  - $\delta(s_1, 1) = s_1$ e $\delta(s_2, 1) = s_5$; nada se conclui pois ainda não há decisão sobre (s_1, s_5).
  - $s_1 \not\equiv s_2$ se e só se $s_2 \not\equiv s_3$ ou $s_1 \not\equiv s_5$; (s_1, s_2) pendente (?). Registar (s_1, s_2) em (s_2, s_3) e (s_1, s_5).
- (s_1, s_3)
  - $\delta(s_1, 0) = s_2$ e $\delta(s_3, 0) = s_1$; nada se conclui pois ainda não há decisão sobre (s_1, s_2).
  - $\delta(s_1, 1) = s_1$ e $\delta(s_3, 1) = s_4$; concluimos que $s_1 \not\equiv s_3$ pois $s_1 \not\equiv s_4$. Colocar X em (s_1, s_3).
- (s_1, s_5)
  - $\delta(s_1, 0) = s_2$ e $\delta(s_5, 0) = s_3$; nada se conclui pois ainda não há decisão sobre (s_2, s_3).
  - $\delta(s_1, 1) = s_1$ e $\delta(s_5, 1) = s_5$; Logo, (s_1, s_5) fica pendente (?). Registar (s_1, s_5) em (s_2, s_3).
- (s_2, s_3)
  - $\delta(s_2, 0) = s_3$ e $\delta(s_3, 0) = s_1$; como $s_3 \not\equiv s_1$, concluimos que $s_2 \not\equiv s_3$. Colocar X em (s_2, s_3).
  - Propagar: assinalar (s_1, s_5) e (s_1, s_2) com X e, consequentemente, também (s_0, s_4).
- (s_2, s_5)
  - $\delta(s_2, 0) = s_3$ e $\delta(s_5, 0) = s_3$; $\delta(s_2, 1) = s_5$ e $\delta(s_5, 1) = s_5$; Logo, $s_2 \equiv s_5$. Colocar $\equiv$ em (s_2, s_5).
- (s_3, s_5)
  - $\delta(s_3, 0) = s_1$ e $\delta(s_5, 0) = s_3$; Logo, $s_3 \not\equiv s_5$ pois $s_1 \not\equiv s_3$. Colocar X em (s_3, s_5).
- Conclusão: $[s_2] = [s_5] = \{s_2, s_5\}$ e $[s_i] = \{s_i\}$, para $i = 0, 1, 3, 4$. A função de transição do AFD mínimo equivalente é dada por: $\delta'([s], a) = [\delta(s, a)]$, para $a \in \Sigma$, $s \in S$, sendo δ a função de transição do AFD original.

Grupo II

6. Seja $G = (\{X, Y\},\ \{\mathtt{a}, \mathtt{b}, \mathtt{c}\},\ \{Y \to \mathtt{a}Y\mathtt{a},\ Y \to \mathtt{a},\ Y \to XX,\ X \to \mathtt{bb}X,\ X \to \mathtt{cc}X,\ X \to \mathtt{b}\},\ Y)$.

a) Prove que a palavra aaccbbaa de $\mathcal{L}(G)$ tem pelo menos duas derivações distintas mas não pode ser usada para mostrar que G é ambígua. Apresente todos os passos das derivações que analisar.

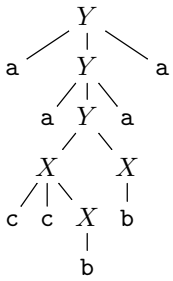
Resposta:

Existem exatamente três derivações distintas para tal palavra mas correspondem à mesma árvore de derivação.

$Y \Rightarrow \mathtt{a}Y\mathtt{a} \Rightarrow \mathtt{aa}Y\mathtt{aa} \Rightarrow \mathtt{aa}XX\mathtt{aa} \Rightarrow \mathtt{aacc}XX\mathtt{aa} \Rightarrow \mathtt{aaccb}X\mathtt{aa} \Rightarrow \mathtt{aaccbbaa}$

$Y \Rightarrow \mathtt{a}Y\mathtt{a} \Rightarrow \mathtt{aa}Y\mathtt{aa} \Rightarrow \mathtt{aa}XX\mathtt{aa} \Rightarrow \mathtt{aacc}XX\mathtt{aa} \Rightarrow \mathtt{aacc}X\mathtt{baa} \Rightarrow \mathtt{aaccbbaa}$

$Y \Rightarrow \mathtt{a}Y\mathtt{a} \Rightarrow \mathtt{aa}Y\mathtt{aa} \Rightarrow \mathtt{aa}XX\mathtt{aa} \Rightarrow \mathtt{aa}X\mathtt{baa} \Rightarrow \mathtt{aacc}X\mathtt{baa} \Rightarrow \mathtt{aaccbbaa}$



b) Indique uma gramática não ambígua que seja equivalente a G. Explique como resolveu a ambiguidade.

Resposta:

$$\mathcal{L}(G) = \{\mathtt{a}^{2n+1} \mid n \in \mathbb{N}\} \cup \{\mathtt{a}^n w \mathtt{a}^n \ \mid\ n \in \mathbb{N},\ \ w \in \mathcal{L}((\mathtt{bb} + \mathtt{cc})^\star \mathtt{b} (\mathtt{bb} + \mathtt{cc})^\star \mathtt{b})\}$$

A gramática é ambígua porque algumas palavras da forma $\mathtt{a}^n w \mathtt{a}^n$, com $w \in \mathcal{L}((\mathtt{bb} + \mathtt{cc})^\star \mathtt{b} (\mathtt{bb} + \mathtt{cc})^\star \mathtt{b})$, admitem mais do que uma árvore de derivação em G. A regra $Y \to XX$ não fixa uma decomposição única para w. Por exemplo, para bbbb, a sub-árvore esquerda (da raíz) pode ter b e a direita bbb, ou a sub-árvore esquerda pode ter bbb e a direita b.

A ambiguidade advém de G permitir várias decomposições para alguns blocos de b's. Podemos abordar diretamente o problema e fixar uma única decomposição para w, tendo em conta que w pode ter uma das quatro formas seguintes:

- $w = \mathtt{b}\mathtt{b}^{2k}\mathtt{b}$, com $k \geq 0$
- $w = \mathtt{b}\mathtt{b}^{2k}\mathtt{cc}w_2\mathtt{b}$, com $k \geq 0$ e $w_2 \in \mathcal{L}((\mathtt{bb} + \mathtt{cc})^\star)$
- $w = w_1\mathtt{ccb}\mathtt{b}^{2k}\mathtt{b}$, com $k \geq 0$ e $w_1 \in \mathcal{L}((\mathtt{bb} + \mathtt{cc})^\star)$
- $w = w_1\mathtt{ccb}\mathtt{b}^{2k}\mathtt{cc}w_2\mathtt{b}$, com $k \geq 0$ e $w_1, w_2 \in \mathcal{L}((\mathtt{bb} + \mathtt{cc})^\star)$.

Assim, a nova gramática (não ambígua) define w, nos dois primeiros casos, como $w = \mathtt{b}z\mathtt{b}$, com $z \in L((\mathtt{bb} + \mathtt{cc})^\star)$ e, nos dois últimos, como $w = z_1\mathtt{ccb}z_2\mathtt{b}$, com $z_1, z_2 \in L((\mathtt{bb} + \mathtt{cc})^\star)$. O símbolo inicial é Y e as regras são:

$$\begin{array}{lcl} Y & \to & \mathtt{a}Y\mathtt{a} \mid \mathtt{a} \\ Y & \to & \mathtt{b}Z\mathtt{b} \mid Z\mathtt{ccb}Z\mathtt{b} \\ Z & \to & \mathtt{bb}Z \mid \mathtt{cc}Z \mid \varepsilon \end{array}$$

Em alternativa, note que $\mathcal{L}((\mathtt{bb} + \mathtt{cc})^\star \mathtt{b} (\mathtt{bb} + \mathtt{cc})^\star \mathtt{b}) = \{w \mid w \in \Sigma^\star \text{ e } XX \Rightarrow_G^\star w\}$ é não ambígua pois é regular. Para encontrar uma GIC não ambígua que a gere, basta definir um AFD A que a reconheça e aplicar o método de conversão dado para determinar uma GIC linear à direita G' que gere $\mathcal{L}(A)$. Se essa GIC tiver símbolo inicial S, bastará depois substituir a regra $Y \to XX$ de G por $Y \to S$, e acrescentar a G as regras e variáveis de G'. (Em caso de dúvida, deve começar por rever a secção 6.1.3. dos Apontamentos, págs 116–120.)

7. Seja L a linguagem das palavras de $\Sigma^\star$ que têm número ímpar de 1's e terminam em 0 ou em 01.

a) Prove que as palavras 01 e 10 são equivalentes segundo a relação de equivalência R_L, referida no teorema de Myhill-Nerode.

Resposta:

$(10, 01) \in R_L$ porque vamos mostrar que, qualquer que seja $z \in \Sigma^\star$, se tem $01z \in L$ se e só se $10z \in L$.

Da definição de L, podemos concluir que $01z \in L$ se e só se $z = \varepsilon$ ou z tem número par de 1's e termina em 0 ou 01. De facto, se z não tiver 1's, então $z = 0^k$, para algum $k \in \mathbb{N}$. Assim, $01z = 010^k$, e, portanto, $010^k \in L$. Se z tiver algum 1, então z tem de ter número par (≥ 2) de 1's para que $01z \in L$ e, nessas condições, z terá ainda de terminar em 0 ou em 01, pois se $01z$ não terminar em 0 tem de ter 0 imediatamente antes do 1 em que termina (se não, $01z \notin L$). Finalmente, note-se que se $z = \varepsilon$ ou se z tiver um número par de 1's e terminar em 0 ou 01 então $01z \in L$. Em suma, $01z \in L$ se e só se $z = \varepsilon$ ou z tem número par de 1's e termina em 0 ou 01.

Analogamente se prova que $10z \in L$ se e só se $z = \varepsilon$ ou z tem número par de 1's e termina em 0 ou 01.

Portanto, por definição de L e de R_L, tem-se $(10, 01) \in R_L$.

b) Apresente o AFD mínimo que reconhece L. Justifique a correção da resposta. Para obter tal AFD, se preferir, pode não seguir a caraterização dada pelo teorema de Myhill-Nerode, mas deve indicar o que memoriza cada estado e a necessidade de cada estado que definiu.

Resposta:

O estado inicial do AFD mínimo é $[\varepsilon]$ e, como $\varepsilon \notin L$, não é um estado final.

$\delta([\varepsilon], 0) \stackrel{\text{def}}{=} [0] \neq [\varepsilon]$, pois para $z = 1$ tem-se $0z = 01 \in L$ e $\varepsilon z = 1 \notin L$. Logo, $[0]$ é um novo estado não final ($0 \notin L$).

$\delta([\varepsilon], 1) \stackrel{\text{def}}{=} [1]$ e $[1]$ é um novo estado não final porque $1 \notin L$ e para $z = 0$ tem-se $1z \in L$ mas $\varepsilon z = 0 \notin L$ e $0z = 00 \notin L$.

$\delta([0], 0) \stackrel{\text{def}}{=} [00] = [0]$, porque $(0, 00) \in R_L$ pois $00z \in L$ sse $z = 1$ ou $z \in L$. Também, $0z \in L$ sse $z = 1$ ou $z \in L$.

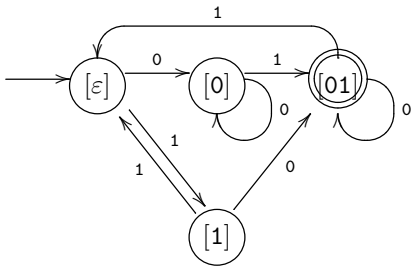
$\delta([0], 1) \stackrel{\text{def}}{=} [01]$ e como $01 \in L$ e $0 \notin L$, $1 \notin L$ e $\varepsilon \notin L$, a classe $[01]$ corresponde a um novo estado (que é um estado final).

$\delta([1], 1) \stackrel{\text{def}}{=} [11] = [\varepsilon]$, pois $11z \in L$ se e só se $z \in L$. Do mesmo modo, $\varepsilon z \in L$ sse $z \in L$.

$\delta([1], 0) \stackrel{\text{def}}{=} [10] = [01]$, por **7a)**.

$\delta([01], 0) \stackrel{\text{def}}{=} [010] = [01]$, porque $010z \in L$ se e só se $z = \varepsilon$ ou z tem número par de 1's e termina em 0 ou 01.

$\delta([01], 1) \stackrel{\text{def}}{=} [011] = [\varepsilon]$, porque $011z \in L$ se e só se $z \in L$.



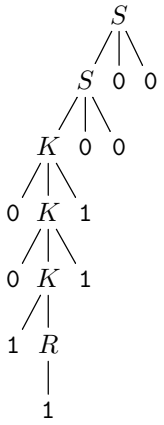
8. Considere a gramática independente de contexto $G = (\{S, K, R\}, \Sigma, P, S)$, com P dado por

$$\begin{array}{lcl} S & \to & S\mathtt{00} \mid \mathtt{00} \mid K\mathtt{00} \\ K & \to & \mathtt{0}K\mathtt{1} \mid \mathtt{1} \mid \mathtt{1}R \\ R & \to & \mathtt{1}R \mid \mathtt{1} \end{array}$$

a) Desenhe uma árvore de derivação da palavra 0011110000 de $\mathcal{L}(G)$ e indique a forma genérica das palavras de $\mathcal{L}(G)$. Explique como chegou a essa conclusão, usando $\Rightarrow_G^\star$, $\Rightarrow_G$, e/ou $\Rightarrow_G^n$, com $n \in \mathbb{N}$.

Resposta:

A árvore de derivação de 0011110000:



As palavras de $\mathcal{L}(G)$ são da forma $\mathtt{0}^{2k}$ ou $\mathtt{0}^n\mathtt{1}^m\mathtt{0}^{2k}$, com $k \geq 1$, $m > n \geq 0$.

Explicação:

A linguagem gerada a partir de R é $\mathcal{L}(\mathtt{1}^\star)$.
$R \Rightarrow^{p+1} \mathtt{1}^p$, com $p \in \mathbb{N}$.

A linguagem gerada a partir de K é $\{\mathtt{0}^n\mathtt{1}^m \mid m > n \geq 0\}$.
$K \Rightarrow^n \mathtt{0}^n K \mathtt{1}^n \Rightarrow \mathtt{0}^n\mathtt{1}\mathtt{1}^n$, com $n \geq 0$
$K \Rightarrow^n \mathtt{0}^n K \mathtt{1}^n \Rightarrow \mathtt{0}^n\mathtt{1}R\mathtt{1}^n \Rightarrow^{p+1} \mathtt{0}^n\mathtt{1}\mathtt{1}\mathtt{1}^p\mathtt{1}^n$, com $n \geq 0, p \geq 0$
Assim, $K \Rightarrow^\star \mathtt{0}^n\mathtt{1}^m$, com $m > n \geq 0$.

$\mathcal{L}(G) = \{\mathtt{0}^{2k} \mid k \geq 1\} \cup \{\mathtt{0}^n\mathtt{1}^m\mathtt{0}^{2k} \mid m > n \geq 0,\ k \geq 1\}$, por:
$S \Rightarrow^q S\mathtt{0}^{2q} \Rightarrow \mathtt{0}\mathtt{0}\mathtt{0}^{2q}$, com $q \geq 0$.
$S \Rightarrow^q S\mathtt{0}^{2q} \Rightarrow K\mathtt{0}\mathtt{0}\mathtt{0}^{2q} \Rightarrow^\star \mathtt{0}^n\mathtt{1}^m\mathtt{0}\mathtt{0}\mathtt{0}^{2q}$, com $m > n \geq 0$ e $q \geq 0$.

b) Converta G à forma normal de Chomsky, por aplicação do método de conversão e, a seguir, aplique o algoritmo CYK para mostrar que 01100 pertence à linguagem gerada por essa gramática. Explique sucintamente o significado das entradas da tabela construída e apresente detalhadamente a construção da primeira e da última linha da tabela.

Sugestão:

A gramática G não tem regras unitárias nem produções-ε. Por isso, a conversão é simples. Em caso de dúvida, deve começar por rever os Apontamentos (secção 6.3.5., págs. 135–138) e, por exemplo, a resolução do 2ºTeste (2014/15) ou a do Exame da época normal de 2013/14.

A tabela que é construída pelo algoritmo CYK para uma palavra $x_1 \dots x_n$ tem na entrada (i, j) o *conjunto das variáveis* da gramática (na forma normal de Chomsky) que produzem a subpalavra $x_j \dots x_{j+i-1}$, para $1 \leq j \leq j + i - 1 \leq n$. Esse invariante é válido para a primeira linha da tabela, a qual carateriza as variáveis que produzem cada terminal, e é preservado pelo algoritmo. O modo como as diversas entradas são preenchidas corresponde à aplicação de uma estratégia "bottom-up"(de baixo para cima) para construção de árvores de derivação para as subpalavras (tendo como raíz as possíveis categorias gramaticais). Para gramáticas na forma normal de Chomsky, as árvores de derivação são *árvores binárias*, já que cada nó interno tem no máximo dois filhos. Se um nó A só tem um filho, então o filho é um terminal a, traduzindo a aplicação de uma regra do tipo $A \to a$, com $a \in \Sigma$; se A tem dois filhos, então os filhos são duas variáveis, digamos B e C, resultando da aplicação de uma regra do tipo $A \to BC$.

Importa salientar que o algoritmo CYK não pode ser aplicado a gramáticas que não estão na F. N. Chomsky.

9. Seja G a gramática definida no problema **8.** e sejam L_1 e L_2 as linguagens definidas por:

$$\begin{aligned} L_1 &= \mathcal{L}(G) \cap \{x \mid x \in \Sigma^\star \text{ e o número de } \mathtt{0}\text{'s ém } x \text{ é o dobro número de } \mathtt{1}\text{'s}\} \\ L_2 &= \mathcal{L}(G) \cap \{x \mid x \in \Sigma^\star \text{ e o número de } \mathtt{0}\text{'s em } x \text{ é igual ao número de } \mathtt{1}\text{'s}\} \end{aligned}$$

Resolva apenas uma das três alíneas seguintes:

a) Apresente um autómato de pilha que aceite L_2 por pilha vazia. Descreva a ideia do algoritmo subjacente e a interpretação de cada estado de forma a permitir aferir a correção do autómato.

Sugestão:

De **8a)**, conclui-se que $L_2 = \{\mathtt{0}^n\mathtt{1}^m\mathtt{0}^{2k} \mid k \geq 1, m > n \geq 0, n + 2k = m\}$.

Para definir o AP, é importante começar por observar que $L_2 = \{\mathtt{0}^n\mathtt{1}^n\mathtt{1}^{2k}\mathtt{0}^{2k} \mid k \geq 1, n \geq 0\}$.

b) Apresente uma máquina de Turing que aceite L_1. Descreva a ideia do algoritmo e a interpretação de cada estado de forma a permitir aferir a correção do máquina.

Sugestão:

De **8a)**, conclui-se que $L_1 = \{\mathtt{0}^n\mathtt{1}^m\mathtt{0}^{2k} \mid k \geq 1, m > n \geq 0, n + 2k = 2m\}$.

A definição da MT é mais simples se se começar por verificar se a palavra é da forma $\mathtt{0}^n\mathtt{1}^m\mathtt{0}^{2k}$, com $k \geq 1, m > n \geq 0$ e, só depois, se verificar se o número de 0's é o dobro do número de 1's.

c) Prove que L_1 não é independente de contexto.

Resolução:

NB: Esta alínea é muito mais difícil do que **9a)** e **9b)**. De facto, é a questão mais difícil do exame. Para a resolver, não basta um conhecimento superficial do Lema da Repetição para LICs e é necessário também alguma intuição e sorte (que, em parte, se pode adquirir com o estudo, prática e abstração). É importante ter compreendido muito bem o modo como este lema foi usado nas aulas para, por exemplo, mostrar que a linguagem $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}$ não era LIC.

Lema da Repetição para LICs: Seja L uma linguagem independente de contexto. Então, existe uma constante $n \in \mathbb{N}\setminus\{0\}$, só dependente de L, tal que, qualquer que seja $z \in L$ com $|z| \geq n$, existe uma decomposição de z na forma $z = uvwxy$, com $|vx| \geq 1$, $|vwx| \leq n$, e para a qual se tem $uv^iwx^iy \in L$, para todo $i \in \mathbb{N}$.

Para mostrar que L_1 não é LIC, podemos tentar provar que não satisfaz a condição indicada neste lema.

Dado $n > 0$, a palavra que vamos usar para mostrar que a condição não se verifica é $z = \mathtt{0}^{2n}\mathtt{1}^{2n+1}\mathtt{0}^{2n+2}$.

Temos $z \in L_1$, porque pertence a $\mathcal{L}(G)$ e o número de 0's é o dobro do número de 1's. Tem-se também $|z| = 6n + 3 \geq n$.

Agora é necessário analisar **todas** as decomposições de z da forma $uvwxy$, com vwx nas condições indicadas e provar que **todas** essas decomposições têm problemas para algum i (podendo i variar de decomposição para decomposição). Para facilitar a análise, vamos identificar os blocos de 0's e de 1's que constituem z pelas letras A, B e C, assim:

$$z = \underbrace{\mathtt{0}^{2n}}_{A}\underbrace{\mathtt{1}^{2n+1}}_{B}\underbrace{\mathtt{0}^{2n+2}}_{C}$$

- Se vwx for subpalavra de A ou de C, tomamos $i = 0$. Tem-se $uv^0wx^0y \notin L_1$, pois cortamos 0's sem cortar 1's, deixando o número de 0's de ser o dobro do número de 1's.
- Analogamente, se vwx for subpalavra de B, tomamos $i = 0$, pois $uv^0wx^0y \notin L_1$ (nem pertence a $\mathcal{L}(G)$), pois cortamos 1's sem alterar o número de 0's.

(a resposta continua na pág. seguinte)

Resolução (9.c) cont.):

- Se vwx for subpalavra de AB e tiver 0's e 1's, então $vwx = 0^p1^q$, com $p+q \leq n$ e $p \geq 1$ e $q \geq 1$, sendo

$$z = \underbrace{0^{2n-p}}_{u}\underbrace{0^p1^q}_{vwx}\underbrace{1^{2n+1-q}0^{2n+2}}_{y}$$

  - Se v tem 0's e 1's, tomamos $i = 2$. A palavra $uv^2wx^2y \notin \mathcal{L}(G)$, pois fica com dois blocos de 1's. Portanto, também não é de L_1;
  - Analogamente, se x tem 0's e 1's, tomamos $i = 2$. A palavra $uv^2wx^2y \notin \mathcal{L}(G)$. Logo, também não é de L_1;
  - Se $v = \varepsilon$ e x só tem 1's, tomamos $i = 0$. Tem-se $uv^0wx^0y \notin \mathcal{L}(G)$, pois cortámos alguns 1's sem cortar 0's.
  - Se $x = \varepsilon$ e v só tem 0's, tomamos $i = 0$. A palavra $uv^0wx^0y \notin \mathcal{L}_1$ pois o número de 0's é inferior ao dobro do número de 1's;
  - Se v só tem 0's e x só tem 1's sendo ambos distintos de ε, então

$$vwx = \underbrace{0^{|v|}}_{v}\underbrace{0^{p-|v|}1^{q-|x|}}_{w}\underbrace{1^{|x|}}_{x}$$

    - se $|v| \leq |x|$, tomamos $i = 0$, e $uv^0wx^0y \notin L_1$ porque o número de 0's não é o dobro do número de 1's.
    - se $|v| > |x|$, então
      - se $|v| \neq 2|x|$, tomamos $i = 0$, e o número de 0's de uv^0wx^0y não é o dobro número de 1's;
      - se $|v| = 2|x|$, tomamos $i = 4$; nesse caso $uv^4wx^4y \notin \mathcal{L}(G)$ pois tem $2n + 6|x|$ símbolos 0 antes dos 1's mas só $2n + 3|x| + 1$ símbolos 1.

- Como vwx não pode abranger simultaneamente os três blocos A, B e C, resta analisar o caso em que vwx é subpalavra de BC e tem 0's e 1's. Nesse caso, para algum $p, q \geq 1$ com $p + q \leq n$, tem-se

$$z = \underbrace{0^{2n}1^{2n+1-p}}_{u}\underbrace{1^p0^q}_{vwx}\underbrace{0^{2n+2-q}}_{y}$$

  - Se vwx tem 1's em v ou em x, tomamos $i = 0$, pois dessa forma $uv^0wx^0y \notin \mathcal{L}(G)$, pois não o número de 0's do bloco A não é alterado e uv^0wx^0y teria menos 1's do que os necessários.
  - Se nem v nem x têm 1's então, como vwx é subpalavra de 1^p0^q, tem-se $v = \varepsilon$ e x só tem 0's. Logo, se tomarmos $i = 0$, a palavra uv^0wx^0y tem um número de 0's que é inferior ao dobro do seu número de 1's.

Nota adicional sobre a escolha de z: A palavra z foi escolhida de modo a garantir que vwx abrange no máximo dois dos blocos A, B e C. Além disso, z tem o número de 1's mínimo para pertencer a $\mathcal{L}(G)$, por começar por 0^{2n}. Por outro lado, se pensarmos no modo como um autómato de pilha trataria a subpalavra AB, concluimos que depois de descarregar a pilha na verificação do bloco B, perde a informação necessária para a análise do bloco C. Por isso, podemos suspeitar que um tal autómato não conseguiria verificar simultaneamente as restrições impostas às palavras de L_1 desta forma (e, portanto, suspeitar que L_1 não era LIC, pois a classe de linguagens reconhecidas por autómatos de pilha é a classe de LICs). O facto de termos provado que, para L_1, não existe n nas condições indicadas no Lema permite-nos concluir que L_1 não é LIC.

(Fim)