Exame de Época Normal (2016/01/14)

Nome: _____ **Número:** _____

Duração: 2h30 **Versão A**

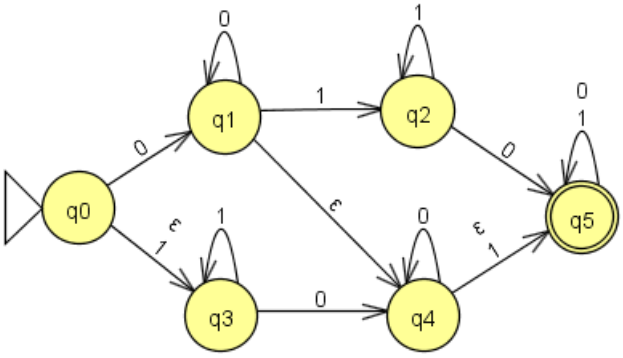
Prova sem consulta, para além do documento fornecido.
Não são permitidos meios eletrónicos (computador, telemóvel, …).
Tentativas de fraude conduzem à anulação da prova para todos os intervenientes.

Responda a cada grupo em folhas separadas!

Coloque o seu nome completo e a versão do exame em todas as folhas!

Grupo I: [4.5 Val] Autómatos Finitos e Expressões Regulares

Considere o ε-NFA à direita.

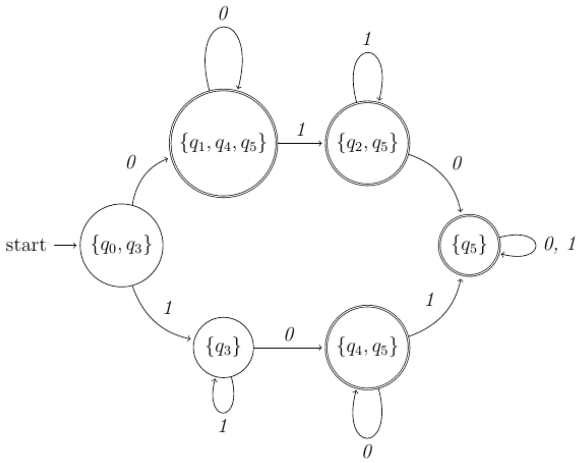


a) Determine o fecho-ε de cada um dos estados do ε-NFA.

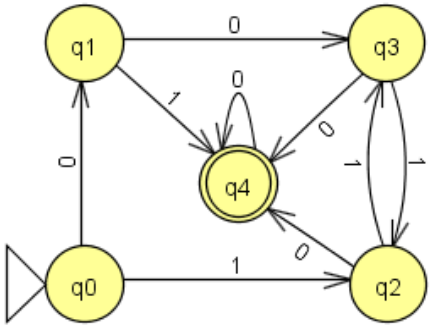
ε-Close(q0) = {q0, q3}
ε-Close(q1) = {q1, q4, q5}
ε-Close(q2) = {q2}
ε-Close(q3) = {q3}
ε-Close(q4) = {q4, q5}
ε-Close(q5) = {q5}

b) Obtenha o DFA equivalente ao ε-NFA da direita. Apresente a tabela de transições e o diagrama de estados do DFA.

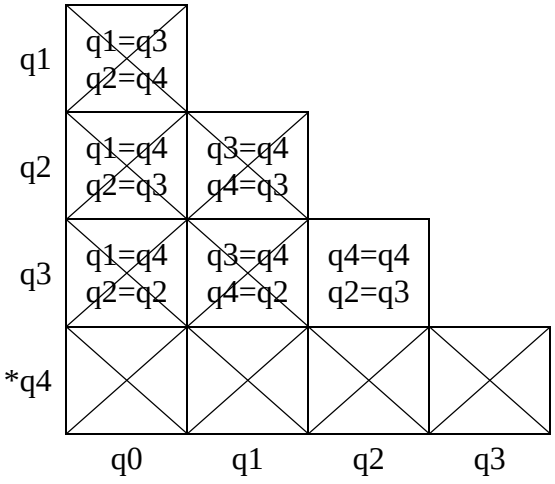
	0	1
➔ {q0, q3}	{q1, q4, q5}	{q3}
*{q1, q4, q5}	{q1, q4, q5}	{q2, q5}
{q3}	{q4, q5}	{q3}
*{q2, q5}	{q5}	{q2, q5}
*{q4, q5}	{q4, q5}	{q5}
*{q5}	{q5}	{q5}

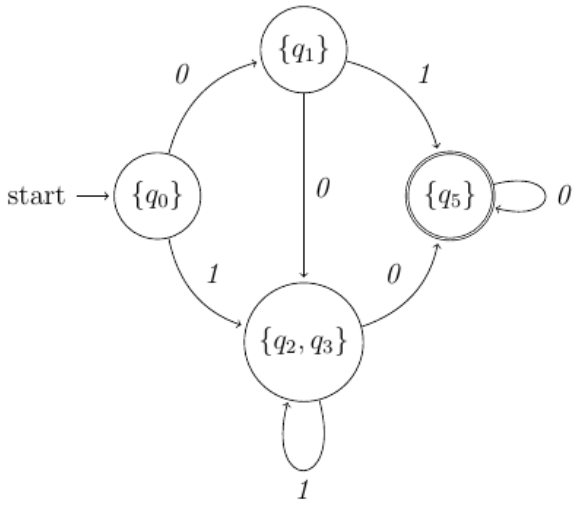


Considere o DFA em baixo.

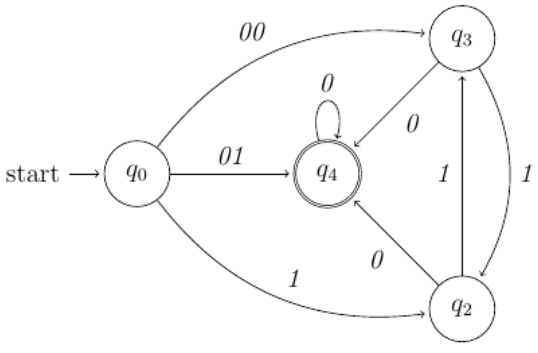


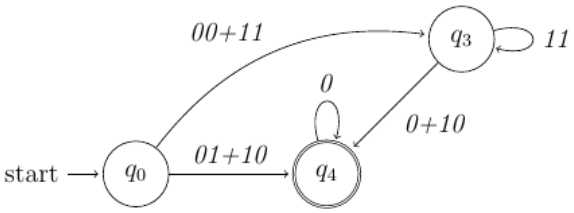
c) Minimize o DFA da esquerda. Apresente a tabela de estados distinguíveis, e o diagrama de estados para o DFA minimizado.

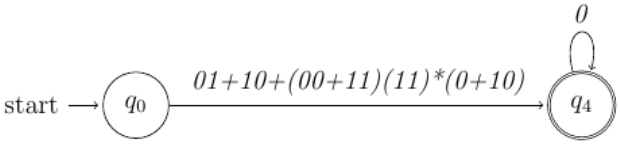




d) Obtenha uma expressão regular para a linguagem definida pelo DFA da esquerda usando o método de eliminação de estados, considerando a ordem de eliminação 1-2-3 (eliminar primeiro o estado q1, depois q2 e depois q3). Mostre todos os passos intermédios.







RE: (01 + 10 + (00+11) (11)* (1+10)) 0*

e) Apresente os valores dos termos $R_{01}^{(0)}$, $R_{02}^{(0)}$, $R_{23}^{(0)}$, $R_{12}^{(0)}$, $R_{23}^{(1)}$, $R_{12}^{(1)}$ e $R_{14}^{(1)}$ obtidos pelo método de construção de caminhos para conversão do DFA para expressão regular.

$R_{01}^{(0)} = 0$ $R_{02}^{(0)} = 1$ $R_{23}^{(0)} = 1$ $R_{12}^{(0)} = \emptyset$

$R_{23}^{(1)} = R_{23}^{(0)} + R_{21}^{(0)} (R_{11}^{(0)})^* R_{13}^{(0)} = 1 + \emptyset\ \varepsilon^*\ 0 = 1$

$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)} (R_{11}^{(0)})^* R_{12}^{(0)} = \emptyset + \varepsilon\ \varepsilon^*\ \emptyset = \emptyset$

$R_{14}^{(1)} = R_{14}^{(0)} + R_{11}^{(0)} (R_{11}^{(0)})^* R_{14}^{(0)} = 1 + \varepsilon\ \varepsilon^*\ 1 = 1$

f) Apresente uma expressão regular para reconhecer cadeias que contenham o código, o indicativo telefónico e o nome de um país no formato apresentado nos exemplos abaixo. O código de um país é composto por duas maiúsculas, e o indicativo telefónico por um a quatro dígitos (sendo o primeiro dígito diferente de 0). Considere o símbolo D como representando qualquer dígito (0 a 9), M como uma letra maiúscula, m como uma letra minúscula, E como um espaço, T como um traço, A como '(' e F como ')'. Exemplos de cadeias: 'PT (351) – Portugal', 'US (1) – United States of America', 'ES (34) – Spain', 'TV (688) – Tuvalu', 'AG (1268) – Antigua and Barbuda'.

RE: MMEA(1+2+3+4+5+6+7+8+9)(ε+D+DD+DDD)FETEMm*(E(M+m)m*)*

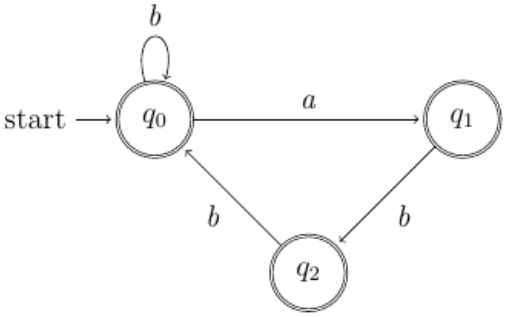
Grupo II: [2 Val] Propriedades de Linguagens Regulares

Prove, para cada uma das seguintes linguagens, se esta é regular ou não. Caso seja, apresente um autómato para a reconhecer. Caso não seja, prove-o usando o lema da bombagem para linguagens regulares.

a) Cadeias sobre o alfabeto {a, b} em que o número de a's na cadeia é ímpar e o número de b's na cadeia é igual ao dobro do número de a's.

Linguagem não regular.
Começamos por assumir que a linguagem é regular e usamos o Lema da Bombagem para provar por contradição que não é regular (se fosse regular, seguiria o lema da bombagem).
Seja n a constante para esta linguagem. Seja w = $a^{2n+1}b^{4n+2}$ a cadeia escolhida (pertence à linguagem)
|w| = 6n+3 (logo ≥ n, e portanto respeita essa condição)
Para que restantes as condições (|xy| ≤ n, |y| ≥ 1) sejam cumpridas, x e y podem ter apenas a's. As divisões possíveis da cadeia w podem ser sumariadas da seguinte forma:
x – a^i, i ≥ 0
y – a^j, j ≥ 1
z – $a^{2n+1-i-j}b^{4n+2}$
Para k=0, a cadeia bombeada w' será $xy^0z = xz = a^i\, a^{2n+1-i-j}b^{4n+2} = a^{2n+1-j}b^{4n+2}$
Como j ≥ 1, então ficamos com menos a's na primeira parte da cadeia, e por isso o número de b's deixa necessariamente de ser igual ao dobro do número de a's, o que faz com que a cadeia não pertença à linguagem. Como a divisão apresentada corresponde a todas as divisões possíveis da cadeia, fica provado que a linguagem não segue o lema, e portanto não é regular.

b) Cadeias sobre o alfabeto {a, b} em que existem pelo menos dois b's a separar cada par de a's.



Esta linguagem é regular.
Exemplo de autómato para reconhecer a linguagem apresentado à direita.

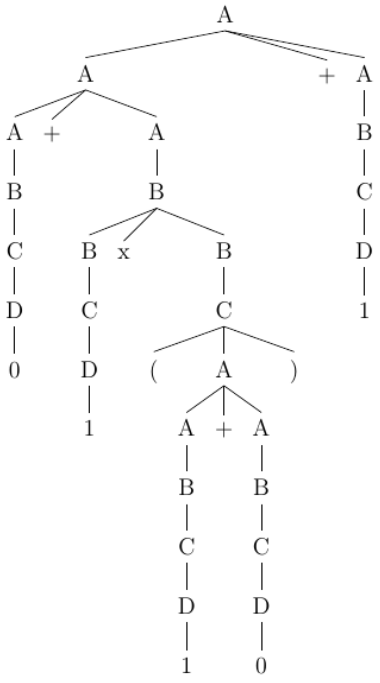
Grupo III: [4.5 Val] Gramáticas Livres de Contexto (CFG) e Autómatos de Pilha (PDA)

$A \rightarrow A + A \mid B$
$B \rightarrow B\ x\ B \mid C$
$C \rightarrow (A) \mid D$
$D \rightarrow 0 \mid 1$

Considere a CFG da esquerda, em que A é a variável de início.

a) Apresente uma árvore de análise e uma derivação mais à esquerda para a cadeia 0+1x(1+0)+1.

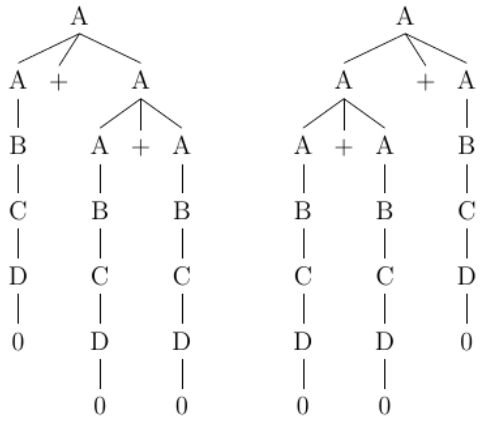
Possível árvore à esquerda.

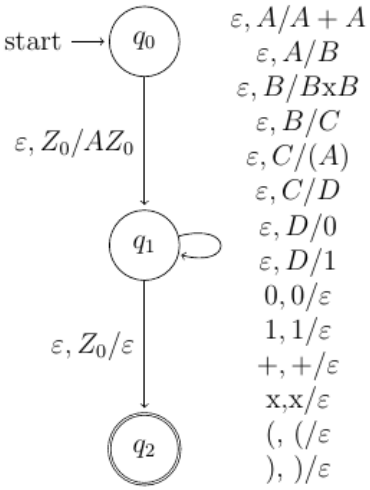


Possível derivação: $A \Rightarrow A + A \Rightarrow A + A + A \Rightarrow B + A + A \Rightarrow C + A + A \Rightarrow D + A + A \Rightarrow 0 + A + A \Rightarrow 0 + B + A \Rightarrow 0 + B\ x\ B + A \Rightarrow 0 + C\ x\ B + A \Rightarrow 0 + D\ x\ B + A \Rightarrow 0 + 1\ x\ B + A \Rightarrow 0 + 1\ x\ C + A \Rightarrow 0 + 1\ x\ (A) + A \Rightarrow 0 + 1\ x\ (A + A) + A \Rightarrow 0 + 1\ x\ (B + A) + A \Rightarrow 0 + 1\ x\ (C + A) + A \Rightarrow 0 + 1\ x\ (D + A) + A \Rightarrow 0 + 1\ x\ (1 + A) + A \Rightarrow 0 + 1\ x\ (1 + B) + A \Rightarrow 0 + 1\ x\ (1 + C) + A \Rightarrow 0 + 1\ x\ (1 + D) + A \Rightarrow 0 + 1\ x\ (1 + 0) + A \Rightarrow 0 + 1\ x\ (1 + 0) + B \Rightarrow 0 + 1\ x\ (1 + 0) + C \Rightarrow 0 + 1\ x\ (1 + 0) + D \Rightarrow 0 + 1\ x\ (1 + 0) + 1$

b) A gramática apresentada é ambígua? Justifique.

Sim, é ambígua, dado que podem existir árvores de análise diferentes para a mesma cadeia. Por exemplo, para a cadeia 0+0+0 podemos ter as árvores apresentadas à direita.





Converta a CFG para um PDA com aceitação por estado final, apresentando o diagrama do PDA resultante.

PDA à esquerda.

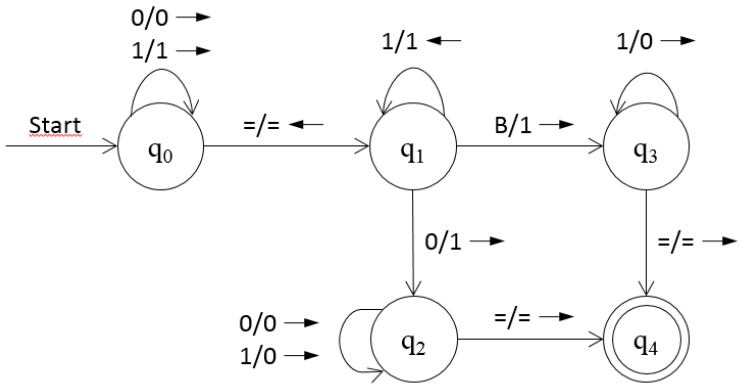
c) O PDA obtido da alínea anterior é determinista? Justifique.

Não. Existem estados com mais do que uma possível transição para o mesmo par (símbolo na entrada, símbolo na pilha). Por exemplo, e independentemente do símbolo na entrada, se houver um D no topo da pilha, este pode ser substituído por um 0 ou por um 1.

d) Apresente utilizando o PDA a sequência de descrições instantâneas que aceita a cadeia 0+1x(1+0)+1.

$(q_0, 0+1x(1+0)+1, Z_0) \vdash (q_1, 0+1x(1+0)+1, AZ_0) \vdash$
$(q_1, 0+1x(1+0)+1, A+AZ_0) \vdash (q_1, 0+1x(1+0)+1, A+A+AZ_0) \vdash$
$(q_1, 0+1x(1+0)+1, B+A+AZ_0) \vdash (q_1, 0+1x(1+0)+1, C+A+AZ_0) \vdash (q_1, 0+1x(1+0)+1, D+A+AZ_0) \vdash$
$(q_1, 0+1x(1+0)+1, 0+A+AZ_0) \vdash (q_1, +1x(1+0)+1, +A+AZ_0) \vdash (q_1, 1x(1+0)+1, A+AZ_0) \vdash$
$(q_1, 1x(1+0)+1, B+AZ_0) \vdash (q_1, 1x(1+0)+1, BxB+AZ_0) \vdash (q_1, 1x(1+0)+1, CxB+AZ_0) \vdash$
$(q_1, 1x(1+0)+1, DxB+AZ_0) \vdash (q_1, 1x(1+0)+1, 1xB+AZ_0) \vdash (q_1, x(1+0)+1, xB+AZ_0) \vdash$
$(q_1, (1+0)+1, B+AZ_0) \vdash (q_1, (1+0)+1, C+AZ_0) \vdash (q_1, (1+0)+1, (A)+AZ_0) \vdash$
$(q_1, 1+0)+1, A)+AZ_0) \vdash (q_1, 1+0)+1, A)+AZ_0) \vdash (q_1, 1+0)+1, A+A)+AZ_0) \vdash$
$(q_1, 1+0)+1, B+A)+AZ_0) \vdash (q_1, 1+0)+1, C+A)+AZ_0) \vdash (q_1, 1+0)+1, D+A)+AZ_0) \vdash$
$(q_1, 1+0)+1, 1+A)+AZ_0) \vdash (q_1, +0)+1, +A)+AZ_0) \vdash (q_1, 0)+1, A)+AZ_0) \vdash$
$(q_1, 0)+1, B)+AZ_0) \vdash (q_1, 0)+1, C)+AZ_0) \vdash (q_1, 0)+1, D)+AZ_0) \vdash (q_1, 0)+1, 0)+AZ_0) \vdash$
$(q_1,)+1,)+AZ_0) \vdash (q_1, +1, +AZ_0) \vdash (q_1, 1, AZ_0) \vdash (q_1, 1, BZ_0) \vdash$
$(q_1, 1, CZ_0) \vdash (q_1, 1, DZ_0) \vdash (q_1, 1, 1Z_0) \vdash (q_1, \varepsilon, Z_0) \vdash (q_1, \varepsilon, \varepsilon)$

Grupo IV: [4 Val] Máquina de Turing



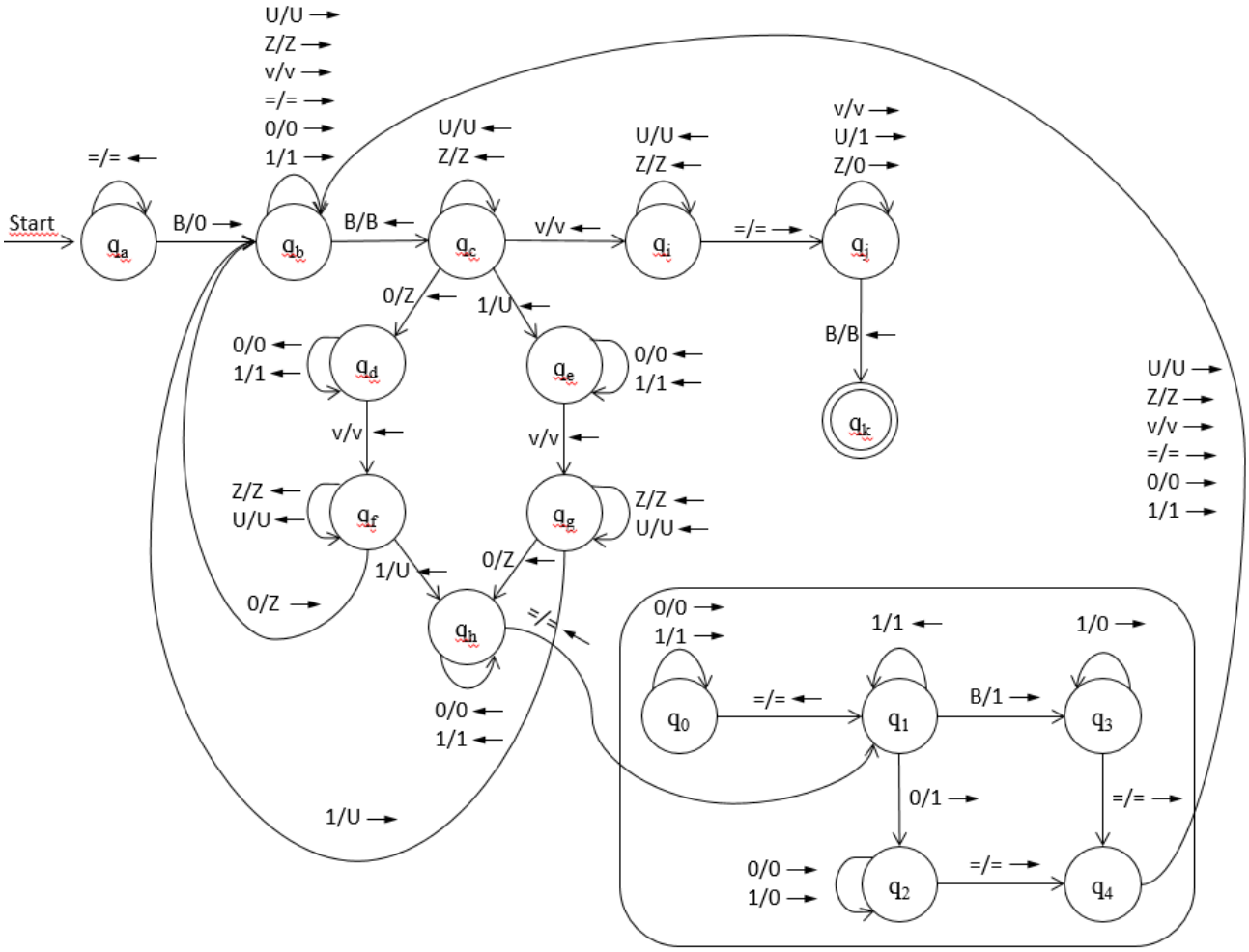
a) Considere a máquina de Turing apresentada. Descreva o que esta faz, e exemplifique, indicando a sequência de descrições instantâneas quando a entrada na fita é 101=.

A MT apresentada soma 1 a um número representado em binário à esquerda de um símbolo de igualdade.
Mais em pormenor, em q0 é percorrido o número para a direita até atingir o símbolo =, altura em que transita para q1, analisando o símbolo à esquerda do =. Caso seja branco (ou seja, não exista ainda número), ou caso o número seja constituído apenas por 1's transita para q3, adicionando um 1 à esquerda dos já existentes. Caso o número contenha um 0, converte-o para 1 e transita para q2. Em q3 o número é novamente percorrido para a direita até o símbolo =, passando os 1's para 0's. Em q2 o número é igualmente percorrido para a direita até o símbolo =, passando também os 1's para 0's.

Sequência de descrições instantâneas quando a entrada na fita é 101=:
$q_0101= \vdash 1q_001= \vdash 10q_01= \vdash 101q_0= \vdash 10q_11= \vdash 1q_101= \vdash 11q_21= \vdash 110q_2= \vdash 110=q_4$

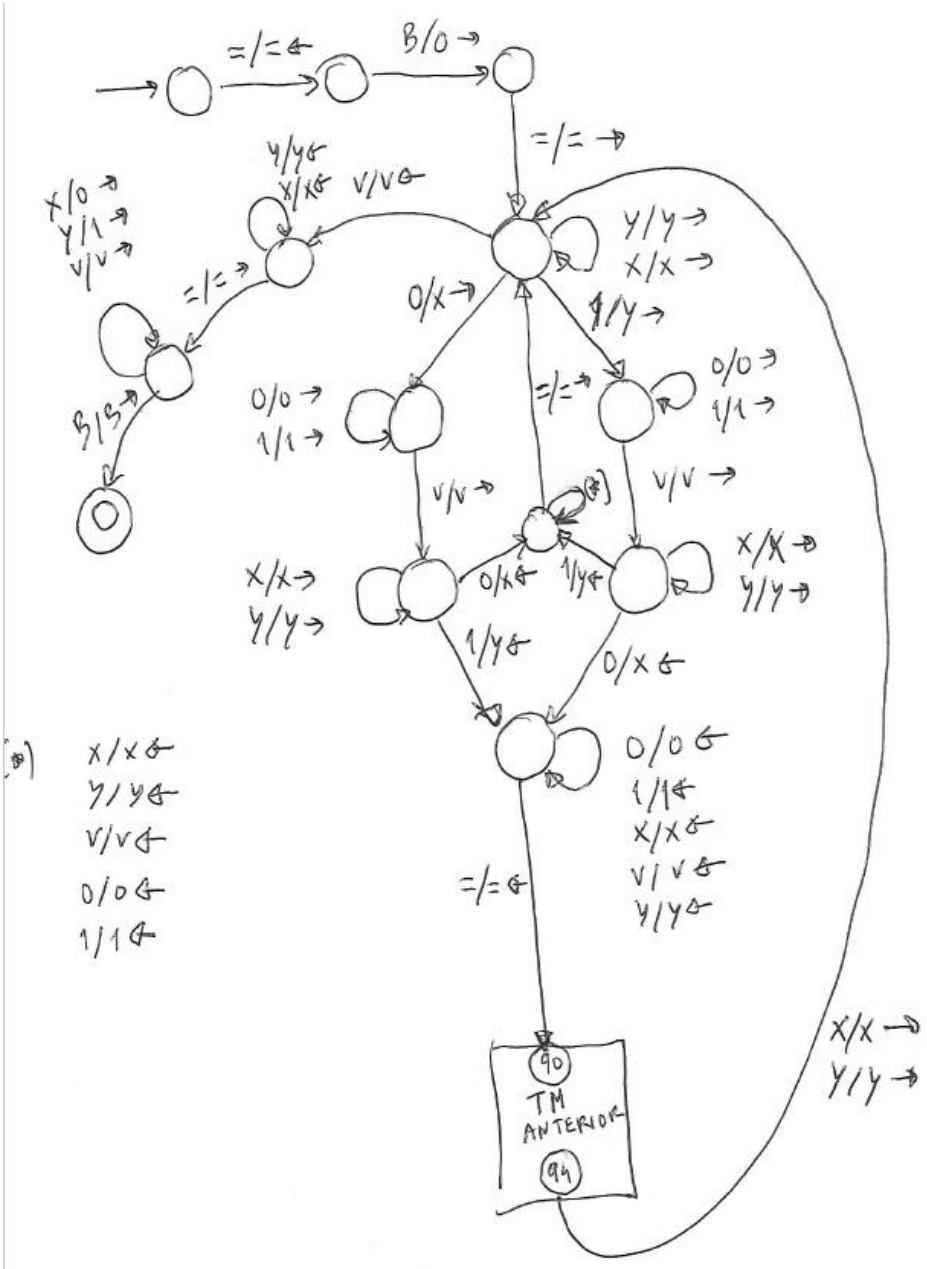
b) Pretende-se usar a TM apresentada como parte da implementação de uma TM que compare o conteúdo de dois operandos em binário (é garantido que o tamanho dos dois operandos é igual), e apresente uma contagem do número de bits diferentes entre os dois operandos. Ex.: =0110v1001 deve resultar em 100=0110v1001; =00110101v00110110 deve resultar em 10=00110101v00110110. Note que a contagem do número de bits diferentes é feita também em binário, à esquerda dos operandos, e que os operandos mantêm o seu valor no final da execução.

Solução possível:



As transições iniciais em q_a e para q_b asseguram que quando os dois operandos são iguais o resultado será 0. Em q_b, percorremos os dois operandos até ao final, e ao encontrar um Branco, volta-se atrás, analisando o segundo operando. Encontrando-se um 0 ou 1 marca-se com o símbolo Z ou U respetivamente (para depois poder restaurar os operandos no final), e avança-se para q_d ou q_e respetivamente. Nesses estados, percorre-se o resto do segundo operando até à separação com o primeiro operando, transitando para q_f ou q_g, onde se continua a percorrer a parte já processada, até encontrar o 0 ou 1 do primeiro operando na posição equivalente ao que se está a processar. Caso se encontre o bit igual, substitui-se novamente pelo Z ou U, e volta-se a q_b, para processar o próximo bit. Caso o bit seja diferente, passa-se para a MT da soma (aqui passando primeiro por q_h de forma a colocar a cabeça no local certo; por uma questão de eficiência, passa-se diretamente para q_1 quando se encontra o =). No final da soma, volta-se a q_b, para processar o próximo bit (as transições poderiam ser feitas diretamente de q_2 e q_3, sem passar por q_4). Quando todos os bits tiverem sido processados, em q_c há uma transição para q_i, que vai colocar a cabeça no início do primeiro operando para que em q_j os Zs e Us sejam novamente substituídos por 0s e 1s respetivamente.

Solução alternativa:
Nota: para simplificar não se rotulou os estados da máquina.


=/=
B/0
=/=
Y/Y
X/X
V/V
X/0
Y/1
V/V
=/=
Y/Y
X/X
0/X
1/Y
B/B
0/0
1/1
=/=
0/0
1/1
V/V
V/V
(*)
X/X
Y/Y
0/X
1/Y
1/Y
0/X
X/X
Y/Y
(*)
X/X
Y/Y
V/V
0/0
1/1
0/0
1/1
X/X
V/V
Y/Y
=/=
90
TM
ANTERIOR
94
X/X
Y/Y

Grupo V: [5 Val] Afirmações sobre Linguagens ( )

Indique, justificando sucintamente, se cada uma das seguintes afirmações é Verdadeira ou Falsa.

a) Nem todas as linguagens expressas por DFAs podem ser expressas por expressões regulares.

Falso. Qualquer que seja a linguagem expressa por um DFA este pode ser sempre convertido para uma expressão regular (utilizando, por exemplo, o método de eliminação de estados)

b) Uma linguagem regular só pode ser expressa por um DFA, um NFA, um ε-NFA ou por uma expressão regular.

Falso. Qualquer linguagem regular também pode ser expressa por um PDA, por uma CFG, por uma Máquina de Turing, etc.

c) Seja B uma Gramática Livre de Contexto (CFG) ambígua. Existe sempre uma CFG A não ambígua que define a mesma linguagem que B.

Falso. Se a linguagem for ambígua não pode ser representada por uma CFG não ambígua.

d) Seja L uma Linguagem Livre de Contexto (CFL) expressa na Forma Normal de Chomsky (CNF). A árvore de análise de qualquer cadeia w pertencente a L será sempre uma árvore binária.

Verdadeiro. Dado que numa gramática em CNF as produções só podem ter um símbolo terminal do lado direito ou dois símbolos não-terminais (variáveis), cada nó da árvore de análise só poderá ter um ou dois filhos e por isso essa árvore é sempre uma árvore binária.

e) Sejam L1, L2 e L3 CFLs. A linguagem L=(L1 L2) $\cap$ (L1 L3) é também uma CFL.

Falso. A operação de intersecção de CFLs pode produzir uma linguagem que não é uma CFL. Por exemplo $\{0^n1^n2^i \mid n\geq1, i\geq1\} \cap \{0^i1^n2^n \mid n\geq1, i\geq1\} = \{0^n1^n2^n \mid n\geq1\}$ que sabemos não ser uma CFL.

f) Nem todos os PDAs podem ser transformados em DFAs.

Verdadeiro. Os PDAs conseguem representar qualquer CFL, mas todas as CFLs que não sejam linguagens regulares (e.g., $\{0^n1^n \mid n\geq1\}$) não podem ser representadas por DFAs.

g) A linguagem $L = \{xyz \mid x,y,z \in \{0,1\}^* \wedge |x| = |z| \wedge$ o número de zeros em x e em y é igual$\}$ é uma linguagem regular.

Falso. Garantir que o número de zeros em x e em y é igual pode ser conseguido aceitando palavras com um número par de zeros. O problema está em identificar o comprimento da substring identificada por x e verificar a existência da substring identificada por y com comprimento igual ao comprimento de x. Sendo a linguagem infinita, um DFA não tem a capacidade para considerar as várias possibilidades para x e y e depois verificar que o comprimento de y é igual ao comprimento de x.

h) Nem todos os PDAs podem ser transformados em CFGs.

Falso. É possível transformar qualquer PDA em CFG. Por exemplo nas aulas foi apresentado um método que converte qualquer PDA que aceite por pilha vazia numa CFG e qualquer PDA que aceite por estado de aceitação pode ser transformado num PDA que aceita por pilha vazia.

i) Pode provar-se que $L=\{a^nb^mc^rd^se^t \mid n+m-r=s+t\}$ não é uma linguagem regular sabendo que $\{a^nb^n \mid n \geq0\}$ é uma linguagem não regular.

Verdadeiro. A operação de homomorfismo é fechada para LRs. Se aplicarmos h(a)= a, h(b) = a, h(c)=b, h(d)=b, e h(e)=b a L, obtemos a^nb^n que não é uma LR. Logo, se L fosse uma LR o resultado seria uma LR e não é.