Prova sem consulta. Duração: 2h30m. Exame de Época Normal
Nome: _____ Número: _____

[Possível Resolução]

1 Expressões Regulares e Autómatos Finitos (5 valores)

Considere a expressão regular: (0+1)*10

1.a) Apresente um DFA (autómato finito determinista) que aceite a linguagem representada pela expressão regular.

[Nota: a resolução abaixo considera um NFA a partir da expressão regular e posterior conversão para DFA.
NFA para a expressão regular:

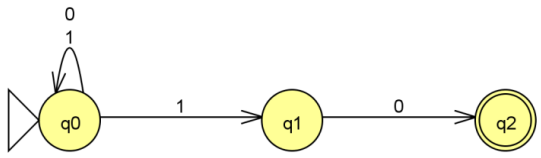


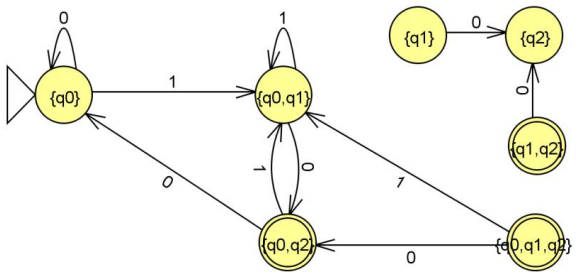
Tabela de transições NFA:

		0	1
→	{q0}	{q0}	{q0, q1}
	{q1}	{q2}	∅
*	{q2}	∅	∅

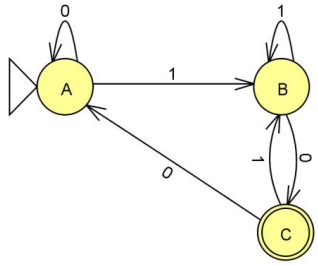
Conversão de NFA para DFA:

		0	1
→	{q0}	{q0}	{q0, q1}
	{q1}	{q2}	∅
*	{q2}	∅	∅
	{q0, q1}	{q0, q2}	{q0, q1}
*	{q0, q2}	{q0}	{q0, q1}
*	{q1, q2}	{q2}	∅
*	{q0, q1, q2}	{q0, q2}	{q0, q1}
	∅	∅	∅

DFA após conversão:



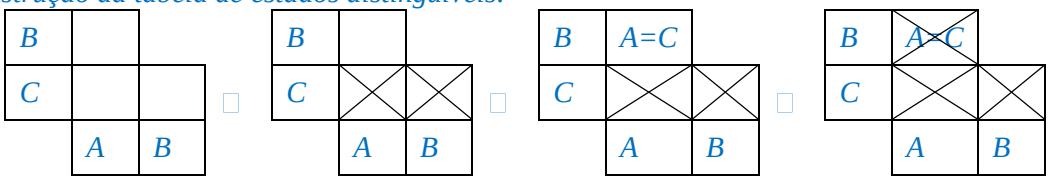
Os estados {q1}, {q2}, {q1, q2} e {q0, q1, q2} podem ser eliminados pois não são alcançáveis a partir do estado inicial. Retirando esses estados e assumindo que A = {q0}, B = {q0, q1} e C = {q0, q2} o DFA fica:



■]

1.b) Apresente o DFA minimizado, inclua a tabela de estados distinguíveis, e indique os estados que sejam equivalentes.

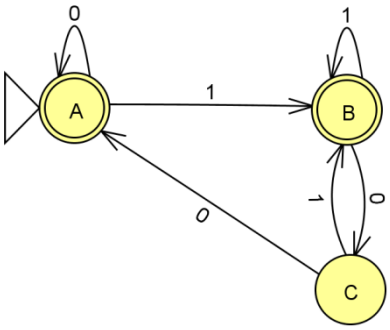
[Construção da tabela de estados distinguíveis:



Não há equivalência entre estados. O DFA já se encontrava minimizado.
■]

1.c) A partir do DFA minimizado, apresente um DFA que represente o complemento da linguagem dada pela expressão regular, considerando o alfabeto {0,1}.

[



São aceites todas as cadeias que não terminam em 10■]

Considere a linguagem L = {w ∈ {0,1}* | w não contém a *substring* 011}:

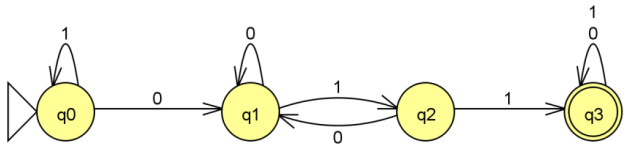
1.d) Apresente uma expressão regular que represente a linguagem L.

[Para resolução deste exercício podemos efectuar os seguintes passos:

- *Obter um DFA que só aceita as cadeias que contém a substring 011;*
- *Efectuar o complemento;*

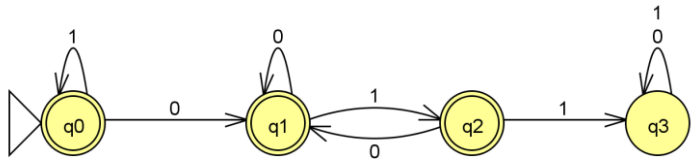
- *Obter uma expressão regular no DFA minimizado.*

O seguinte DFA só aceita cadeias que contêm a substring 011:

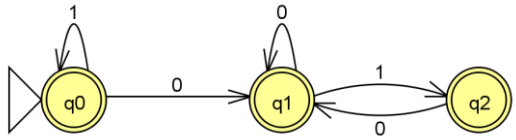


Descrição dos estados:
q0: *o autómato permanece neste estado até não aparecer o primeiro '0', só transita para q1 após receber um '0';*
q1: *o autómato permanece neste estado enquanto não recebe um '1';*
q2: *o autómato encontra-se neste estado após receber um '1' seguido a um '0': "01". No caso de receber outro '1' ("011") transita para q3 e aceita a partir daí qualquer sequência de 0's e 1's. Pelo contrário, se receber um '0' ("010"), retorna ao estado q1 e espera até ser recebido um '1';*
q3: *o autómato só se entra neste estado após receber a subsequência "011", sendo que deverá permanecer neste estado independentemente do símbolo recebido '0' ou '1'.*

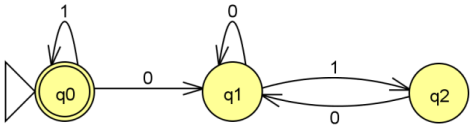
Para obter um DFA aceita todas as cadeias que não contém a substring 011, basta efectuar o complemento ao DFA anterior:



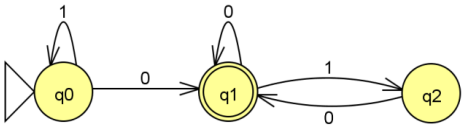
Uma vez que δ(q2, '1') = q3 e como q3 não tem transições para nenhum outro dos estados, podemos assumir que q3 é um estado morto:



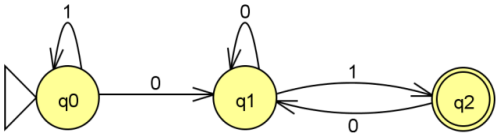
Pelo método de eliminação de estados:



*1**



*1*0(0+10)**



*1*00*1(00*1)**
A RE final e a união das 3 RE anteriores:
1+ 1*0(0+10)* + 1*00*1(00*1)**
■]

2 Propriedades de Linguagens Regulares (4 valores)

2.a) Prove usando o lema da bombagem para linguagens regulares que a linguagem L = {w ∈ {0,1}* | o comprimento de w é ímpar e o símbolo no meio de w é o 1} é não-regular.

[A linguagem L representa cadeias da forma: $(0+1)^n1(0+1)^n$, com $n \geq 0$.
Vamos considerar: $L \cap L(0^*10^*) = L(0^n10^n)$. Dado que a intersecção é uma operação fechada para linguagens regulares e como $L(0^*10^*)$ é uma linguagem regular, para que L fosse regular então $L(0^n10^n)$ teria de ser uma linguagem regular. Podemos por isso provar que L é uma linguagem não regular provando que $L(0^n10^n)$ é uma linguagem não regular.
Vamos provar por contradição. Assumimos que L é uma linguagem regular e se for uma linguagem regular terá de satisfazer o lema da bombagem para linguagens regulares.
Seja $w=xyz=0^n10^n$, então: $|w| = 2n+1$ que é $\geq n$ e por isso respeita a condição do lema $|w| \geq n$
Como $|xy| \leq n$, então xy só podem ter 0's (à esquerda do 1 na palavra w)
Como $|y| \geq 1$, então y tem pelo menos um zero. De facto y pode ter de 1 a n zeros, ou seja $1 \leq |y| \leq n$[1].
Se bombearmos segundo o lema, e de acordo com xy^kz, para k=0 teremos xz que representará uma palavra com um número de 0's à esquerda do 1 menor do que o número de 0's à direita do 1. De facto xz é do tipo: 0^j10^n, com $0 \leq j \leq n-1$. Logo $xz \notin L$ e por isso a linguagem não satisfaz o lema e por isso é uma linguagem não regular. qed

NOTA: Poderíamos realizar a prova usando $w=(0+1)^n1(0+1)^n$
■]

2.b) Prove sem usar o lema da bombagem que a linguagem L = $\{0^n1^k2^{n-k} \mid n \geq k \geq 0\}$ é uma linguagem não regular.

[Podemos basear a prova tendo em conta que a linguagem $L(0^n1^n)$, com $n \geq 0$, é uma linguagem não regular.
$L \cap L(0^*1^*) = L(0^n1^n)$. Dado que a intersecção uma operação fechada para linguagens regulares e como $L(0^*1^*)$ é uma linguagem regular, para que L fosse regular então $L(0^n1^n)$ teria de ser uma linguagem regular. Como sabemos que $L(0^n1^n)$ é uma linguagem não regular então L não pode ser uma linguagem regular. Conclui-se por isso que L é uma linguagem não regular. qed

Nota: poderíamos provar da mesma forma mas usando homomorfismo.
■]

2.c) Para provarmos que uma linguagem L é regular temos de indicar para L um autómato (DFA, NFA, ou ε-NFA), uma expressão regular, ou usar as propriedades de fecho das linguagens regulares para mostrar que podemos obter a linguagem L a partir de uma ou mais linguagens regulares. Por que é que não podemos usar o lema da bombagem para provar que uma linguagem é regular?

[O lema da bombagem para linguagens regulares diz que se L for uma linguagem regular então L satisfaz o lema, i.e., L é regular $\rightarrow$ L satisfaz o lema. O lema não postula uma relação sse do tipo: L é regular $\leftrightarrow$ L satisfaz o lema. Ou seja o lema estabelece uma condição necessária para uma linguagem ser regular mas não uma condição suficiente. A preposição L satisfaz o lema $\rightarrow$ L é regular não é válida. Existem linguagens regulares que satisfazem o lema. Não podemos por isso usar o lema para provar que uma linguagem é regular.
É de notar que o lema da bombagem para linguagens regulares só é válido para linguagens regulares infinitas e por isso nunca poderia ser usado para provar que uma linguagem (finita ou infinita) é regular.
■]

3 Gramáticas sem Contexto (4 valores)

Considere a seguinte gramática livre de contexto (CFG), na qual S representa a variável de início:

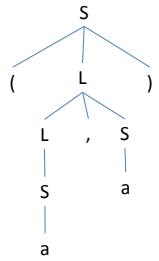
---

[1] *Ou seja $y=0^i$, $1 \leq i \leq n$. Quando k=0, $xy^kz=xz$ e em xz teremos n-i zeros seguidos de um 1 e de n zeros. Logo, $xz \notin L$ e por isso Lé uma linguagem não regular.*

$S \rightarrow (L) \mid a$
$L \rightarrow L,S \mid S$

3.a) Desenhe uma árvore de análise para a cadeia (a,a), e indique os passos da derivação o mais à direita.

*[árvore de análise para a cadeia **(a,a):***



Passos da derivação o mais à direita: $S \Rightarrow (L) \Rightarrow (L,S) \Rightarrow (L,a) \Rightarrow (S,a) \Rightarrow (a,a)$
■]

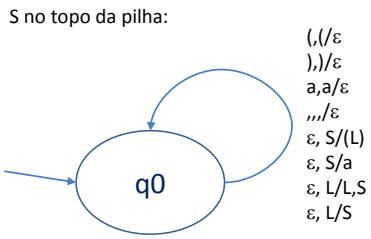
3.b) Esta gramática é ambígua? Justifique a resposta dada e caso seja ambígua elimine a ambiguidade da gramática.

[A gramática apresentada não é ambígua.
Justificação, 85%: A gramática não é ambígua pois não existe nenhuma palavra aceite pela gramática que implique duas ou mais árvores de análise diferentes.
Pelas regras da gramática pode-se ver que não existem duas ou mais formas de aceitar cadeias.
A variável e início, S, deriva "a" ou (L) e não há mais nenhuma forma de fazer estas duas derivações: $S \rightarrow (L) \mid a$
A variável L pode derivar L,S ou S:
$L \rightarrow L,S \mid S$
No caso da derivação S será sempre para terminar com "a" ou para considerar um novo encadeamento de () via S.
No caso da derivação L,S será sempre para formar duas subcadeias separadas por "," e a única forma de derivar subcadeias separadas por vírgulas é através de L.
■]

3.c) Converta a gramática para um PDA que aceite por pilha vazia.

[PDA) $P = (\{q0\}, \{', ',(,),a\}, \{', ',(,),a, S, L\}, \delta, q_0, S, \{\})$
$\delta(q0, (, () = (q0, \varepsilon)$
$\delta(q0,),)) = (q0, \varepsilon)$
$\delta(q0, a, a) = (q0, \varepsilon)$
$\delta(q0, ', ', ', ') = (q0, \varepsilon)$
$\delta(q0, \varepsilon, S) = \{(q0, (L)), (q0, a)\}$
$\delta(q0, \varepsilon, L) = \{(q0, L', 'S), (q0, S)\}$

Ou graficamente:



■]

3.d) O PDA que obteve é determinista ou não-determinista? Justifique.

[O PDA anterior é não determinista.

Existem transições que para a mesma palavra na entrada conduzirão a mais do que uma sequência de transições. Por exemplo, para qualquer que seja o símbolo na entrada e o símbolo L no topo da pilha o PDA pode transitar de q0 para q0 substituindo o topo da pilha por (L) ou por S.
■]

4 Máquina de Turing (4 valores)

Pretende-se uma Máquina de Turing que duplique uma sequência de símbolos no alfabeto ∑={0,1} existente na fita.

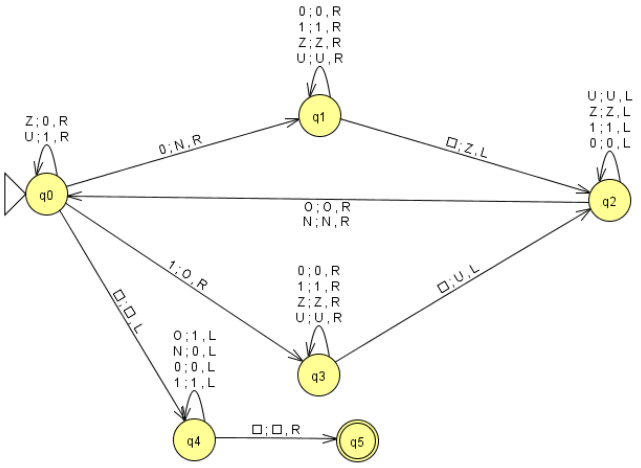
Exemplo: 01100 → 0110001100

4.a) Descreva uma estratégia para resolver o problema com uma Máquina de Turing.

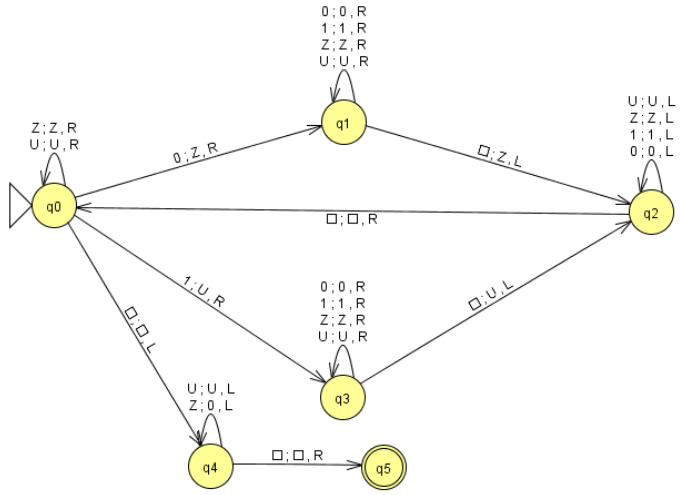
[Percorre-se a entrada na fita da esquerda para a direita e:
- se encontrarmos um símbolo 1 substituímo-lo por O e vamos até ao primeiro B à direita e substituímo-lo por U. Depois deslocamo-nos para a esquerda até encontrarmos um O ou um N;
- se encontrarmos um símbolo 0 substituímo-lo por N e vamos até ao primeiro B à direita e substituímo-lo por Z; Depois deslocamo-nos para a esquerda até encontrarmos um O ou um N;
Repetimos estes passos até não haver mais símbolos 1 e 0 e depois disso substituímos todos os O's e U's por 1 e todos os N's e Z's por 0.■]

4.b) Desenhe a Máquina de Turing correspondente à estratégia anterior.

[
Máquina de Turing M=({q0, q1, q2, q3, q4, q5}, {0, 1}, {0, 1, Z, U, O, N, □}, q0, □, {q5})



Uma outra máquina de Turing que segue uma estratégia ligeiramente diferente (usa apenas dois símbolos auxiliares, U e Z):



■]

4.c) Apresente o traço de computação quando a entrada na fita é *011*.

[q_0011 ⊢ Nq_111 ⊢ $N1q_11$ ⊢ $N11q_1$ ⊢ $N1q_21Z$ ⊢ Nq_211Z ⊢ q_2N11Z ⊢
Nq_011Z ⊢ NOq_31Z ⊢ $NO1q_3Z$ ⊢$_M$ $NO1Zq_3$ ⊢ $NO1q_2ZU$ ⊢ NOq_21ZU ⊢ Nq_2O1ZU ⊢
NOq_01ZU ⊢ $NOOq_3ZU$ ⊢ $NOOZq_3U$ ⊢ $NOOZUq_3$ ⊢ $NOOZq_2UU$ ⊢ NOq_2OZUU ⊢
$NOOq_0ZUU$ ⊢
$NOO0q_0UU$ ⊢ $NOO01q_0U$ ⊢ $NOO011q_0$ ⊢
$NOO01q_41$ ⊢ $NOO0q_411$ ⊢ $NOOq_4011$ ⊢ NOq_4O011 ⊢ Nq_4O1011 ⊢ $q_4N11011$ ⊢ $q_4B011011$ ⊢ $q_5011011$
■]

5 Afirmações sobre linguagens (3 valores)

Indique se cada uma das seguintes afirmações é verdadeira ou falsa e justifique sucintamente a resposta dada.

5.a) Uma Máquina de Turing para implementar um reconhecedor de palíndromos tem de ser não-determinista.

[F. As máquinas de Turing não-deterministas não acrescentam poder computacional às máquinas de Turing deterministas. De facto a máquina de Turing não-determinista pode ser implementada por uma máquina de Turing determinista. ■]

5.b) Um PDA (autómato de pilha) é não-determinista se a partir de um estado existirem duas ou mais transições com o mesmo símbolo de entrada.

[F. Um PDA não-determinista é um PDA que para uma dada entrada tem várias sequências de transições possíveis. Seria não-determinista se "a partir de um estado existirem duas ou mais transições com o mesmo símbolo de entrada (ou ε) e com o mesmo símbolo no topo da pilha." ■]

5.c) O reverso de uma linguagem regular é uma linguagem regular.

[V. Se é uma linguagem regular então podemos representá-la por um DFA. O reverso de uma linguagem regular pode ser obtido modificando o DFA para que os estados de aceitação passem a ser estados normais e alcançáveis por um novo estado de início (no caso de ser apenas um estado de aceitação basta converter esse estado para estado de início), e o estado de início passe a ser o estado de aceitação. Estas modificações podem originar um DFA ou um ε-NFA e por isso a linguagem resultante é uma linguagem regular. ■]

5.d) A intersecção de duas linguagens não-regulares é uma linguagem não-regular.

[F. Contra-exemplo: basta que ambas as linguagens não regulares não tenham palavras em comum para que a intersecção das mesmas forneça a linguagem vazia, que não é uma linguagem não regular. ■]

5.e) Dadas duas linguagens regulares L1 e L2, uma forma de verificar que L1 = L2 é testar se:
$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \varnothing$

[V. A verificação de que as duas linguagens são iguais implica que se tenha de provar que não há palavras numa das linguagens que não faça parte da outra.
Para isso teremos de fazer a união de dois termos:
$(L_1 \cap \overline{L_2})$*: o conjunto de palavras de L1 intersectado com o conjunto de palavras não contidas em L2 tem de ser vazio. Isto garante que não há palavras em L1 que não estejam em L2.*
$(\overline{L_1} \cap L_2)$*: o conjunto de palavras de L2 intersectado com o conjunto de palavras não contidas em L1 tem de ser vazio. Isto garante que não há palavras em L2 que não estejam em L1. ■]*

5.f) Uma linguagem é ambígua quando conseguimos arranjar pelo menos uma gramática ambígua para representar essa linguagem.

[F. Por definição, uma linguagem ambígua é uma linguagem para a qual não conseguimos arranjar uma CFG não ambígua. Por isso, o facto de conseguirmos arranjar uma gramática ambígua para representar essa linguagem não quer dizer que a linguagem seja ambígua! ■]

(Fim.)