Uma Possível Resolução

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
Mestrado Integrado em Engenharia Informática e Computação

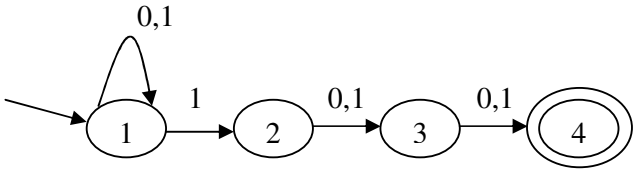
Teoria da Computação

Exame, 22 de Junho de 2009

DURAÇÃO MÁXIMA: 2 horas e 30 minutos

Problema 1: Autómatos Finitos (5 valores)

Considere o seguinte NFA sobre o alfabeto Σ={0,1}:



1.a) Explique informalmente que linguagem é aceite por este NFA.

R: Cadeias sobre o alfabeto 0, 1, de tamanho maior ou igual a 3, e cujo antepenúltimo símbolo é um 1.

1.b) Converta o autómato para um DFA que aceite a mesma linguagem. Inclua os passos necessários para a conversão. Desenhe o DFA completo resultante.

R:

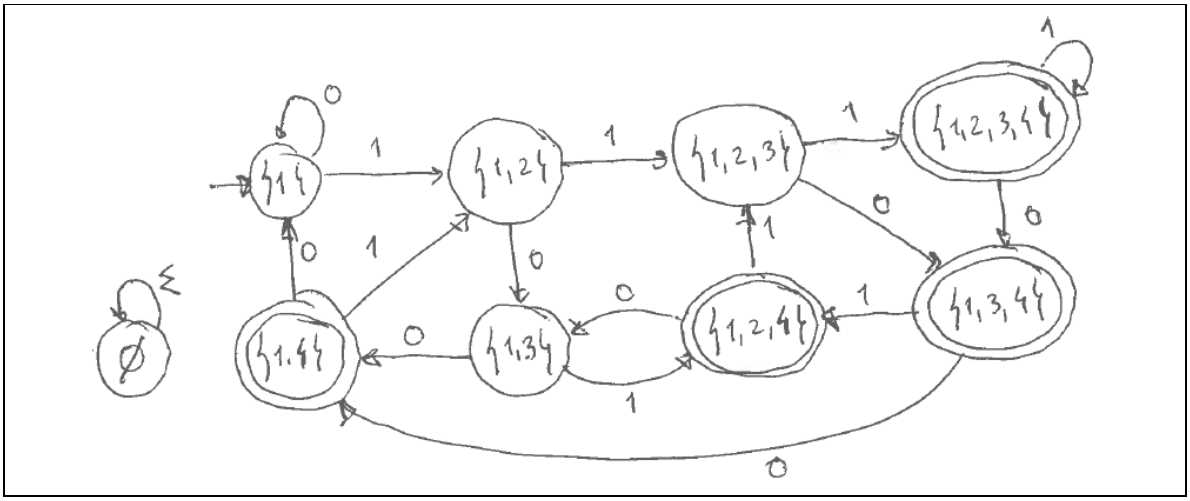
Tabela de transições do NFA:

	0	1
→1	{1}	{1,2}
2	{3}	{3}
3	{4}	{4}
*4	{∅}	{∅}

Tabela de transições do DFA:

	0	1
→{1}	{1}	{1,2}
{1,2}	{1,3}	{1,2,3}
{1,3}	{1,4}	{1,2,4}
{1,2,3}	{1,3,4}	{1,2,3,4}
*{1,4}	{1}	{1,2}
*{1,2,4}	{1,3}	{1,2,3}
*{1,3,4}	{1,4}	{1,2,4}
*{1,2,3,4}	{1,3,4}	{1,2,3,4}

Desenho do DFA resultante:



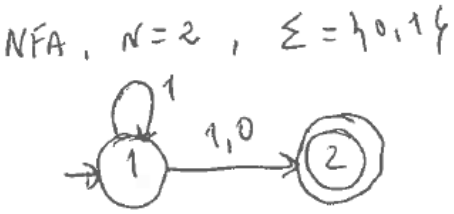
1.c) Indique a expressão regular que represente a linguagem que o autómato aceita.

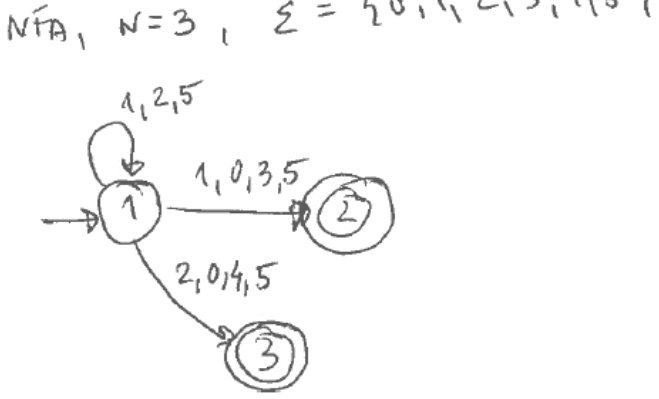
*R: (0+1)*1(0+1)(0+1)*

1.d) Explique por que motivo os DFAs resultantes de autómatos finitos não deterministas com N estados podem ter 2^N estados. Apresente um NFA com 2 estados e um NFA com 3 estados, cujos DFA equivalentes, sem uma eventual minimização de estados, têm 4 e 8 estados, respectivamente.

R: Para transformar um NFA num DFA é necessário considerar os 2^N-1 agrupamentos de estados possíveis de formar a partir dos N estados do NFA original. No pior dos casos, o DFA resultante inclui (i.e., são todos alcançaceis) esses 2^N-1 estados mais o estado de erro (∅), o que perfaz os 2^N estados.

Dois exemplos de NFAs de 2 e 3 estados que original DFAs com 4 e 8 estados, respectivamente:





Problema 2: Linguagens (3 valores)

Mostre, recorrendo ao Lema da Bombagem, que a linguagem das cadeias palíndromo sobre o alfabeto {0, 1} não é uma linguagem regular.

R: Vamos considerar a linguagem de palíndromos L1 = $\{1^n01^n, n\geq 1\}$. Esta linguagem representa um subconunto de palíndromos sobre o alfabeto {0, 1}.

Sabemos que se a linguagem que representa um subconjunto dos palíndromos não for uma linguagem regular então L, a linguagem de todos os palíndromos, não é uma linguagem regular.

Vamos tentar provar por contradição que L1 é uma linguagem regular. O lema da bombagem diz-nos:

Lema: Seja L uma linguagem regular. Então existe uma constante n (dependente de L) tal que para todas as cadeias w em L com $|w| \geq n$ se pode partir w em 3 subcadeias w=xyz tais que:

- $y \neq \varepsilon$

- $|xy| \leq n$

Para todo o $k \geq 0$, a cadeia xy^kz também está em L.

De forma a respresentar todas as cadeias em L1 escolhemos $w=1^n01^n$, que verifica a condição $|w| \geq n$, pois $|w| = 2n+1$

Para partirmos w em 3 subcadeias xyz e atendendo a que $|xy| \leq n$ só poderemos ter apenas 1's nas subcadeias xy:

a) temos todos os n primeiros 1's em xy

b) temos uma subcadeia de 1's em xy

Em qualquer dos dois casos teremos que ter pelo menos um 1 em y e por isso quando bombeamos xy^kz vamos ter sempre possibilidade de produzir cadeias que não têm o mesmo número de 1's à esquerda e à direita de 0 e que por isso não são palíndromos.

Com k=0 ficamos com algo da forma $1^{n'}01^n$, em que n'=n-|y|. I.e., com k=0 retiramos os 1's que há em y (pelo menos 1 e no máximo n).

Podemos assim concluir que L1 não é uma linguagem regular e genericamente que a linguagem de palíndromos sobre o alfabeto {0,1} não é uma linguagem regular.

Problema 3: Gramáticas e Autómatos de Pilha (5 valores)

Seja G = (V, Σ, R, S) a seguinte CFG. V={S, T, U}; Σ={0, #}; e R o conjunto de regras:

$S \rightarrow TT \mid U$

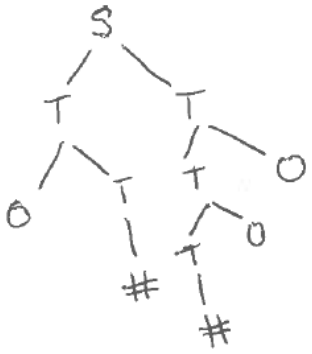
$T \rightarrow 0T \mid T0 \mid \#$

$U \rightarrow 0U00 \mid \#$

3.a) Indique a string de menor tamanho aceite pela gramática.

R: #

3.b) Desenhe a árvore de análise para a cadeia **0##00**.



3.c) Converta a gramática para um PDA que aceita por pilha vazia e desenhe o PDA resultante. Mostre a sequência de descrições instantâneas quando o PDA obtido processa a string **0##00**.

PDA

$P_N = \{\{q\}, \{0,\#\}, \{0,\#,T,U,S\}, \delta, q, S\}$

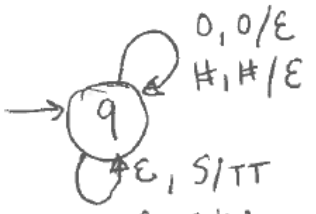
$\delta(q,0,0) = \{(q,\varepsilon)\}$

$\delta(q,\#,\#) = \{(q,\varepsilon)\}$

$\delta(q,\varepsilon,S) = \{(q,TT), (q,U)\}$

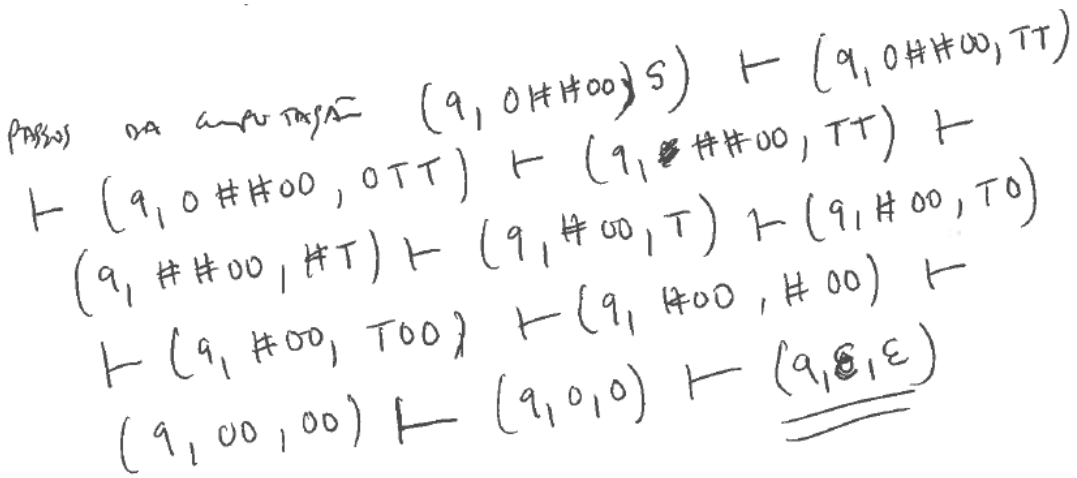
$\delta(q,\varepsilon,T) = \{(q,0T), (q,T0), (q,\#)\}$

$\delta(q,\varepsilon,U) = \{(q,0U00), (q,\#)\}$

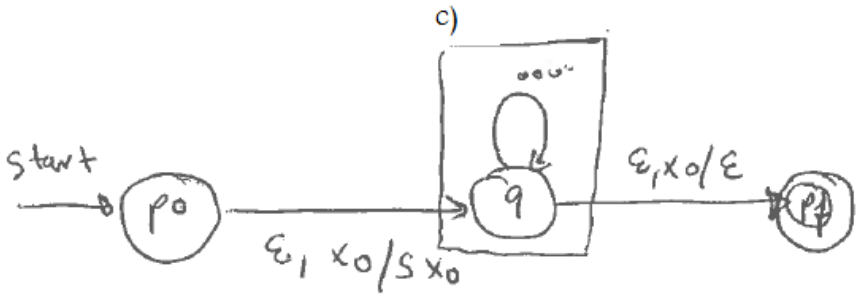


NO INÍCIO ESTÁ S NA PILHA

Sequência de descrições instantâneas quando o PDA obtido processa a string **0##00** (representa-se aqui a sequência que origina a aceitação da string):

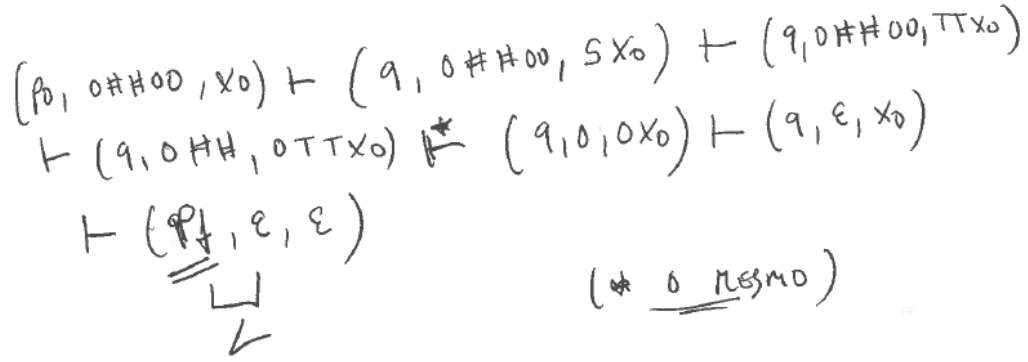


3.d) Converta o PDA anterior para um PDA que aceita por estado de aceitação. Mostre a sequência de descrições instantâneas quando o PDA obtido processa a string **0##00**. [neste caso pode omitir passos de computação intermédios]



$P_F = (\{q, p_0, p_f\}, \{0, \#\}, \{0, \#, T, U, S\}, \delta, q, x_0, \{p_f\})$

Sequência de descrições instantâneas quando o PDA obtido processa a string **0##00**. [neste caso omitindo passos de computação intermédios]



Problema 4: Máquina de Turing (4 valores)

4.a) Desenhe o diagrama de transições de estado de uma Máquina de Turing que converta uma cadeia de letras do alfabeto ∑={A,C} numa cadeia constituída unicamente pelas letras A existentes na cadeia de entrada. Os A's na cadeia resultante terão de estar em posições contíguas da fita. Exemplos:

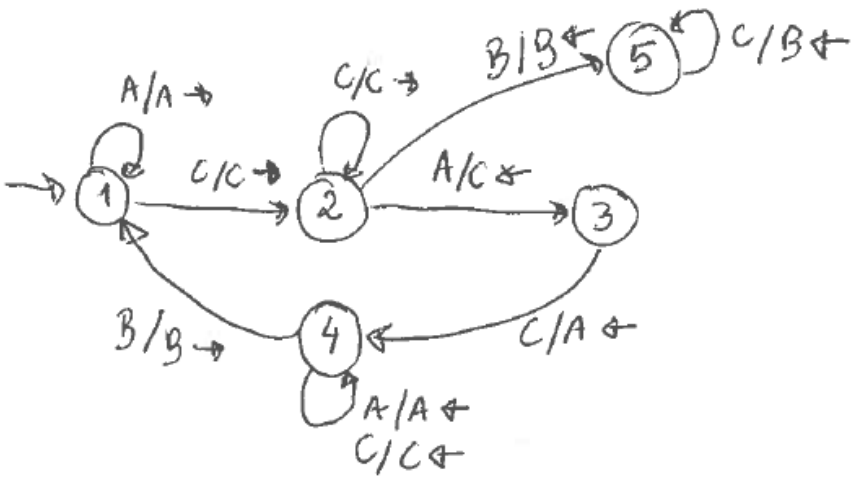
ACA	→	AA
CAACA	→	AAA
CACC	→	A
C	→	B

Não se esqueça de começar por **descrever sucintamente a estratégia** que vai adoptar.

R:

Começar por descrever sucintamente a estratégia adoptada [não incluída na resolução]

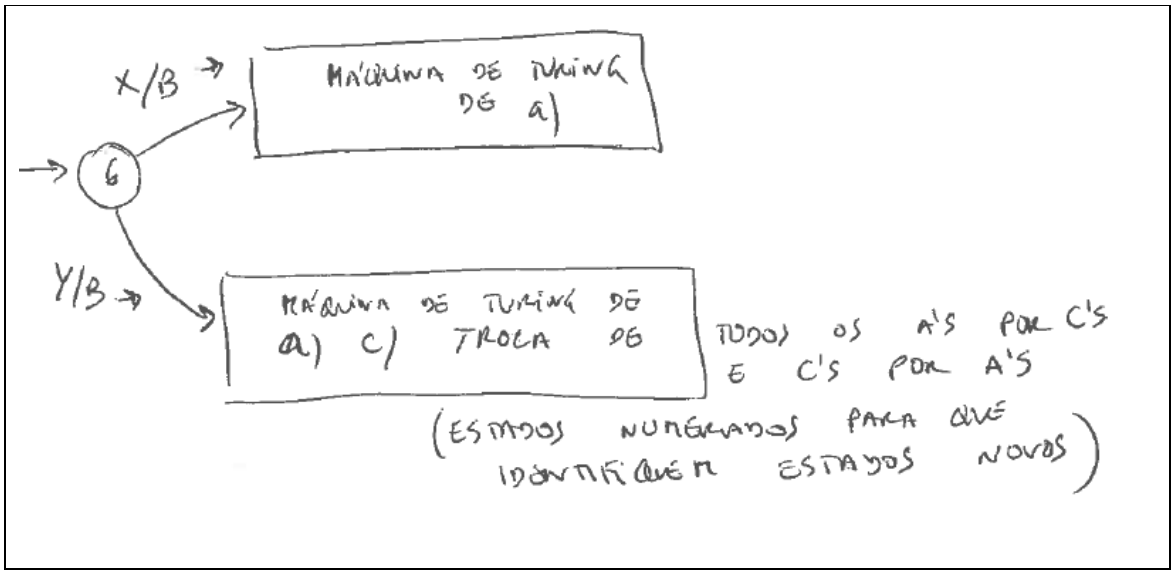
Máquina de Turing:



4.b) Apresente o traço de computação da sua Máquina de Turing quando a entrada na fita é ACA.

R: 1ACA ├ A1CA ├ AC2A ├ A3CC ├ 4AAC ├ 4BAAC ├ 1AAC ├ A1AC ├ AA1C ├
├ AAC2 ├ AA5C ├ A5A

4.c) Tendo por base a máquina de Turing obtida em a), indique uma Máquina de Turing que inicia com a leitura de um símbolo (S) na fita que indica se deve modificar a cadeia de letras do alfabeto ∑={A,C} numa cadeia constituída unicamente por letras A (S=X) ou por letras C (S=Y).



Problema 5: Afirmações sobre Linguagens (3 valores)

Para cada uma das afirmações seguintes, diga se é verdadeira ou falsa e dê uma justificação sucinta.

5.a) A união de uma linguagem regular com uma linguagem não regular é sempre uma linguagem não regular.

*R: **Falso.** A linguagem resultante pode ser regular ou não regular. A linguagem regular pode por exemplo incluir as strings representadas pela linguagem não regular e nesse caso a união das duas é representada pela linguagem regular. A união da linguagem dos palíndromos no alfabeto {0,1} (linguagem não regular) com a linguagem das cadeias de 0s e 1s, (0+1)*, que é uma linguagem regular, forma a linguagem regular (0+1)*.*

Outro exemplo: $0^*1^* \cup 0^n1^n = 0^*1^*$

5.b) Uma gramática é ambígua se conseguirmos arranjar uma string aceite pela gramática que possa produzir duas árvores de análise, uma por derivação o mais à esquerda, e a outra por derivação o mais à direita.

*R: **Falso.** Uma gramática á ambígua se conseguirmos arranjar uma string aceite pela gramática que possa produzir duas árvores de análise distintas, por derivação o mais à esquerda, ou por derivação o mais à direita*

5.c) A linguagem das cadeias que não ocorrem no enunciado deste exame é uma linguagem regular.

*R: **Verdadeira.** Podemos representar a linguagem das cadeias que ocorrem neste exame (que é um conjunto finito) por um DFA e fazer o complemento do mesmo (a classe de linguagens regulares é fechada sobre o complemento, i.e., o complemento de uma linguagem regular continua a ser uma linguagem regular).*

5.d) A linguagem $a^nb^ma^nb^m$ não é uma linguagem sem contexto, mas pode ser reconhecida por uma Máquina de Turing.

*R: **Verdadeira.** Esta linguagem não verifica o lema da bombagem para CFLs pois não conseguimos bombear a's e b's de forma a continuar a ter balanceamento entre a's e entre b's na cadeia. Pode ser reconhecida por uma máquina de Turing pois neste caso podemos por cada a no inicio verificar se existe um a na segunda sequência de a's e ir apagando os a's que emparelharem. Fazemos o mesmo para os b's e no final verificamos se sobraram a's ou b's.*

(Fim.)