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

Universidade do Porto
Faculdade de Engenharia
FEUP

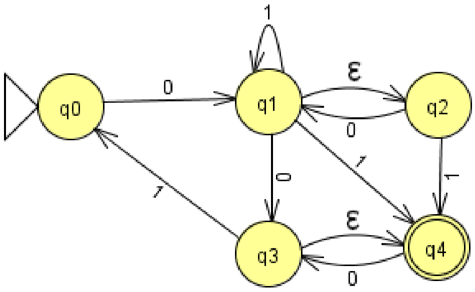
Teoria da Computação

Exame de Época Normal, 13 de Janeiro de 2011

DURAÇÃO MÁXIMA: 2 horas e 30 minutos

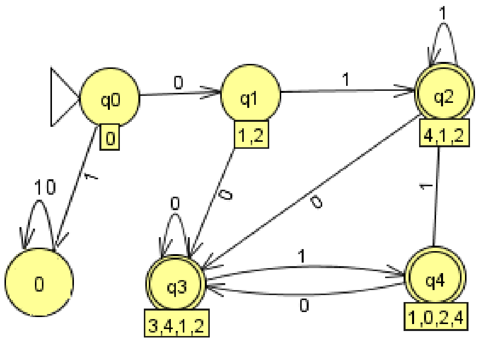
Problema 1: Autómatos Finitos e Expressões Regulares (5 valores)

Considere o autómato finito seguinte:



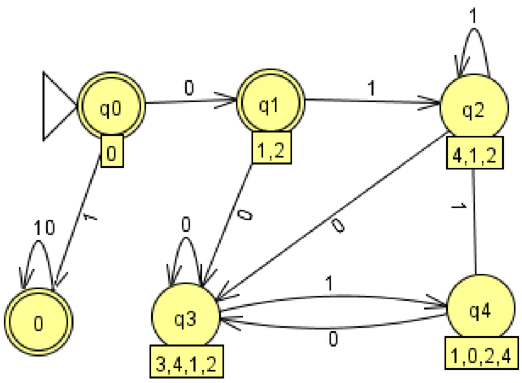
1.a) Converta-o para um DFA e desenhe o DFA completo resultante.

R:

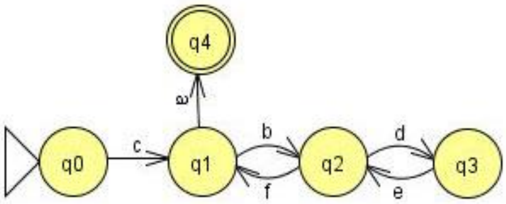


1.b) Apresente o diagrama do DFA que represente o complemento da linguagem representada pelo autómato obtido em 1a).

R:

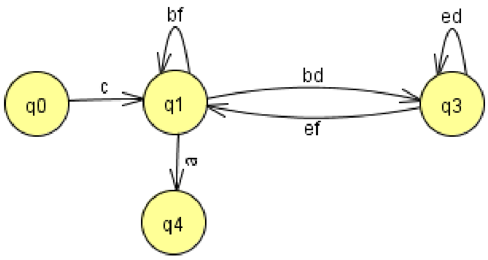


1.c) Converta o DFA seguinte para uma expressão regular usando o método de eliminação de estados pela ordem de eliminação de estados q2,q3, e q1. Simplifique a expressão regular resultante usando propriedades de equivalência.

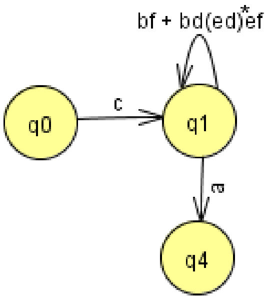


R:[nos autómatos seguintes q0 é o estado inicial e q4 o estado final]

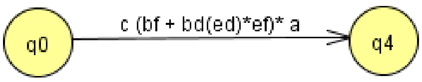
Após eliminação de q2:



Após eliminação de q3:



Após eliminação de q1:

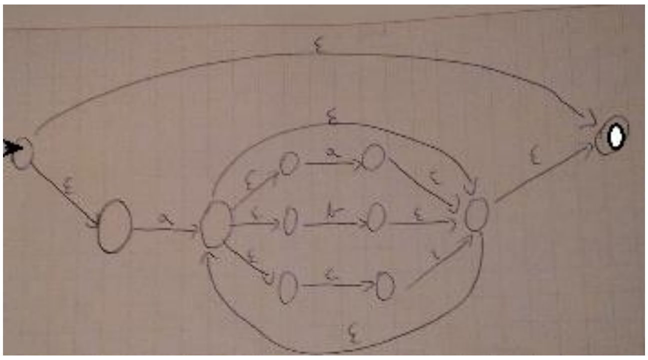


Obtém-se a expressão regular:
c(bf + bd(ed)*ef)*a

Que pode ser simplificada usando os passos seguintes:
c(b(f+d(ed)*ef))*a
c(b(f+(de)*def))*a
c(b(f+(de)*(de)f))*a
c(b((ε+(de)*(de))f))*a
c(b(de)*f)*a

1.d) Aplicando templates de ε-NFA associados a cada operador das expressões regulares, converta a expressão regular ε+a(a+b+c)* em ε-NFA.

R:



Problema 2: Linguagens (3 valores)

Prove que a linguagem L = $\{b^t a^k \mid t \geq 0 \text{ e } k = 2^n \text{ com } n \geq 0\}$ é uma linguagem não regular.

R:

A concatenação de duas linguagens regulares produz sempre uma linguagem regular. A concatenação[1] da linguagem L com L1= b que é uma linguagem regular resulta na linguagem* $L2=\{a^k \mid k = 2^n \text{ com } n \geq 0\}$*. Se L2 não for regular então L não pode ser regular.*

Vamos assumir que L2 é uma linguagem regular. Nesse caso L2 tem de satisfazer o lema da bombagem para linguagens regulares. Seja w= a^{2^n} *, com* $n \geq 0$*. Como se pode ver* $|w| = 2^n \geq n$

Sendo w=xyz temos $|xyz| = 2^n$ *e considerando* $|y|>0$ *temos* $1 \leq |y| \leq 2^n$ *(y pode ter de 1 'a' a* 2^n *'a's)*

Segundo o lema $xy^iz \in L2, \ \forall i \geq 0$*.*

Assim, para i=3 temos xyyyz e $2^n + 2 \leq |xyyyz| \leq 3 \times 2^n$ *.*

Quando i=3 estamos perante a segunda palavra que é aceite pela linguagem considerada a seguir a uma palavra de comprimento 2^n*. Essa palavra tem de ter comprimento* $2^{n+2} = 4 \times 2^n$*. Como* $3 \times 2^n < 4 \times 2^n$ *temos que a palavra formada ao bombearmos y 3 vezes (i=3) não pode pertencer à linguagem L2 pois não tem comprimento* $k = 2^m$ *com* $m \geq 0$*.*

Logo, L2 não satisfaz o lema da bombagem para linguagens regulares e por isso não pode ser uma linguagem regular.

*Conclui-se assim que, dado que a concatenação é uma operação fechada para as linguagens regulares, o facto de L2 não ser uma linguagem regular implica que L (L2=b*L) também não pode ser uma linguagem regular.*

[Nota: o que acontece para i=2?]

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

Dada a CFG G = (V, Σ, R, S), com V={S, R, U}, Σ={s, d, b, c, t}, e R o conjunto de regras:

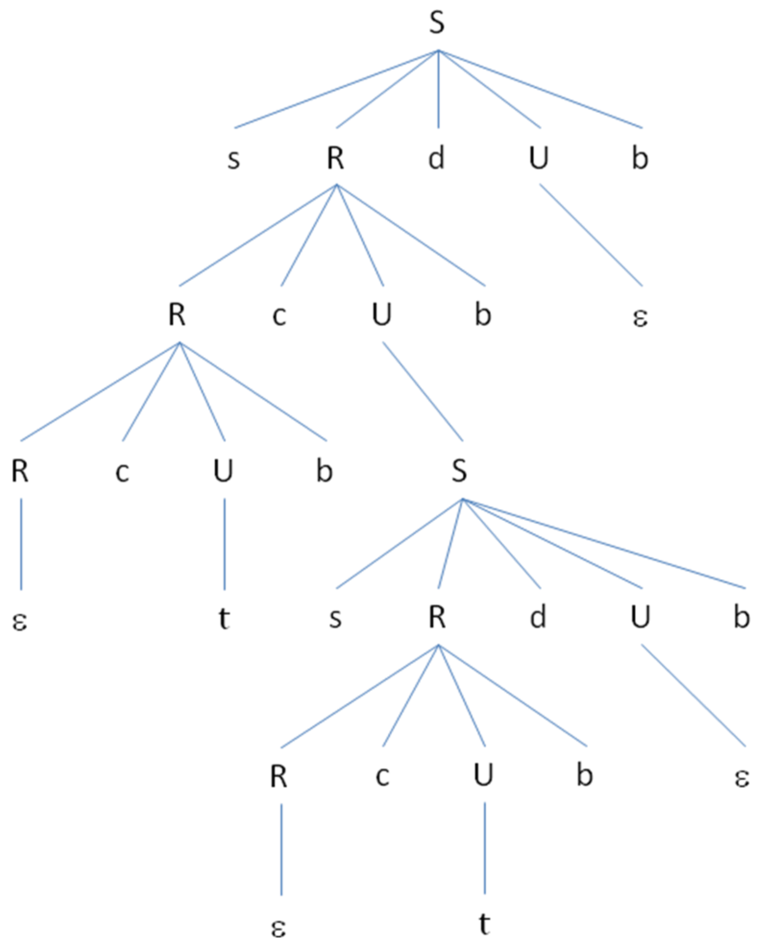
$S \rightarrow$ s R d U b

[1] uma outra possibilidade seria iniciarmos a prova com a utilização da propriedade de fecho da operação de intersecção .

$R \rightarrow R\ \mathrm{c}\ U\ \mathrm{b} \mid \varepsilon$

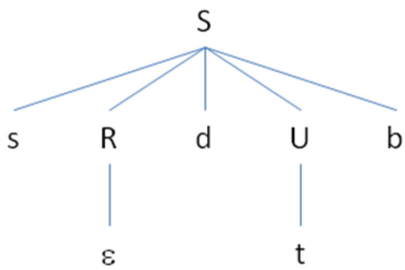
$U \rightarrow U\ U \mid S \mid \mathrm{t} \mid \varepsilon$

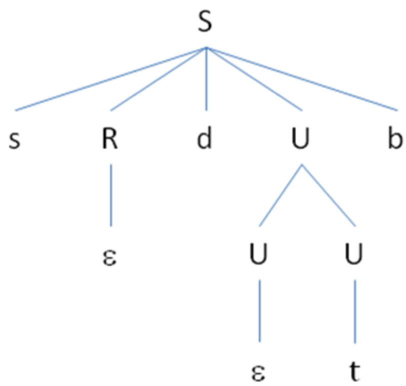
3.a) Desenhe a árvore de análise para a *string* "**sctbcsctbdbbdb**".



3.b) A gramática apresentada é ambígua? Justifique a resposta. Caso seja ambígua, apresente uma gramática não-ambígua para a mesma linguagem.

R: Sim, é ambígua, uma vez que podem ser obtidas mais de uma árvore de análise para determinadas cadeias (devido à regra U → UU). Por exemplo, para a cadeia sdtb:





Possível gramática não ambígua equivalente:

$S \rightarrow s\ R\ d\ U\ b$

$R \rightarrow R\ c\ U\ b \mid \varepsilon$

$U \rightarrow U\,S \mid U\,t \mid \varepsilon$

3.c) Converta a gramática para um PDA que aceita por pilha vazia e desenhe o PDA resultante.

R:

$P= (Q, \Sigma, \Gamma, \delta, q_0, S, F)$

$Q= \{q_0\}$, $\Sigma=\{s,c,t,b,d\}$, $\Gamma=\{s,c,t,b,d,S,R,U\}$, $F=\{q_0\}$

$\delta(q_0, \varepsilon, S) = \{(q_0, sRdUb)\}$

$\delta(q_0, \varepsilon, R) = \{(q_0, RcUb), (q_0, \varepsilon)\}$

$\delta(q_0, \varepsilon, U) = \{(q_0,UU), (q_0,S), (q_0,t), (q_0, \varepsilon)\}$

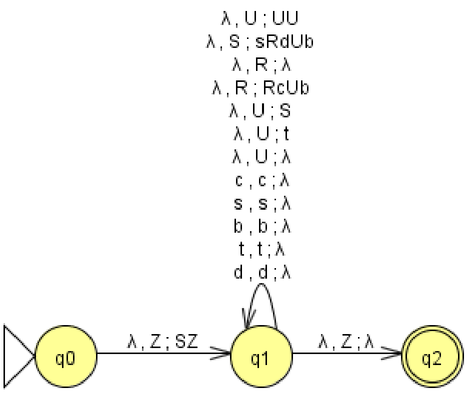
$\delta(q_0, s, s) = \delta(q_0, c, c) = \delta(q_0, t, t) = \delta(q_0, b, b) = \delta(q_0, d, d) = \{(q_0, \varepsilon)\}$

[Nota: na figura do PDA, o símbolo λ tem o mesmo significado do que o símbolo ε]

3.d) Mostre uma sequência de descrições instantâneas que conduz à aceitação quando o PDA obtido processa a *string* "**scbdtb**".

R: (q0, scbdtb, S) $\vdash$ (q0, scbdtb, sRdUb) $\vdash$ (q0, cbdtb, RdUb) $\vdash$ (q0, cbdtb, RcUbdUb) $\vdash$ (q0, cbdtb, cUbdUb) $\vdash$ (q0, bdtb, UbdUb) $\vdash$ (q0, bdtb, bdUb) $\vdash$ (q0, dtb, dUb) $\vdash$ (q0, tb, Ub) $\vdash$ (q0, tb, tb) $\vdash$ (q0, b, b) $\vdash$ (q0, ε, ε)

3.e) Apresente, para a gramática original, um PDA que aceita por estado de aceitação.



[Nota: na figura do PDA, o símbolo λ tem o mesmo significado do que o símbolo ε]

Problema 4: Máquina de Turing (4 valores)

4.a) Desenhe o diagrama de transições de estado de uma Máquina de Turing que calcule a subtracção de uma unidade de um número positivo não nulo representado em binário dado como input na fita. A Máquina de Turing deve respeitar a seguintes restrições:

- Manter o resultado com o mesmo número de bits ainda que os bits mais significativos sejam 0.
  Exemplos:

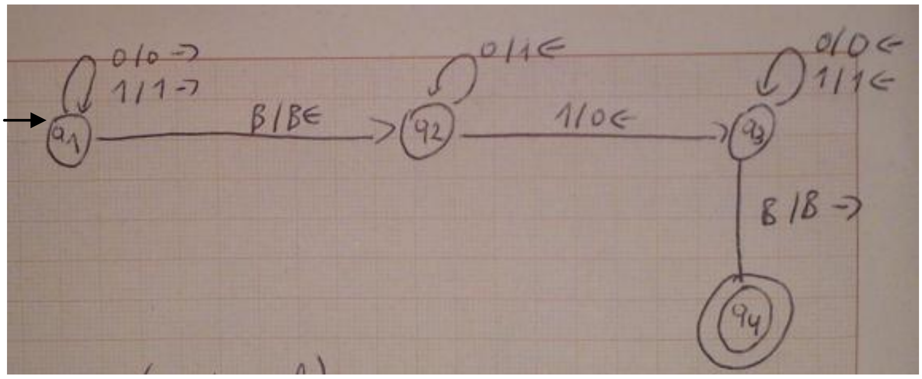
Input	Output
100	011
000101	000100

- No estado final que indica o fim do processamento, a cabeça da fita deverá estar colocada no bit mais à esquerda.
  Exemplo: (↓ indica cabeça da Máquina de Turing)

Input	Output
↓ 100	↓ 011

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

R: [nota: a explicação da estratégia a utilizar não é incluída nesta resolução. Descrevam a estratégia tendo em conta a máquina de Turing apresentada em baixo.]



4.b) Apresente o traço de computação da sua Máquina de Turing quando a entrada na fita é **0110**.

R:

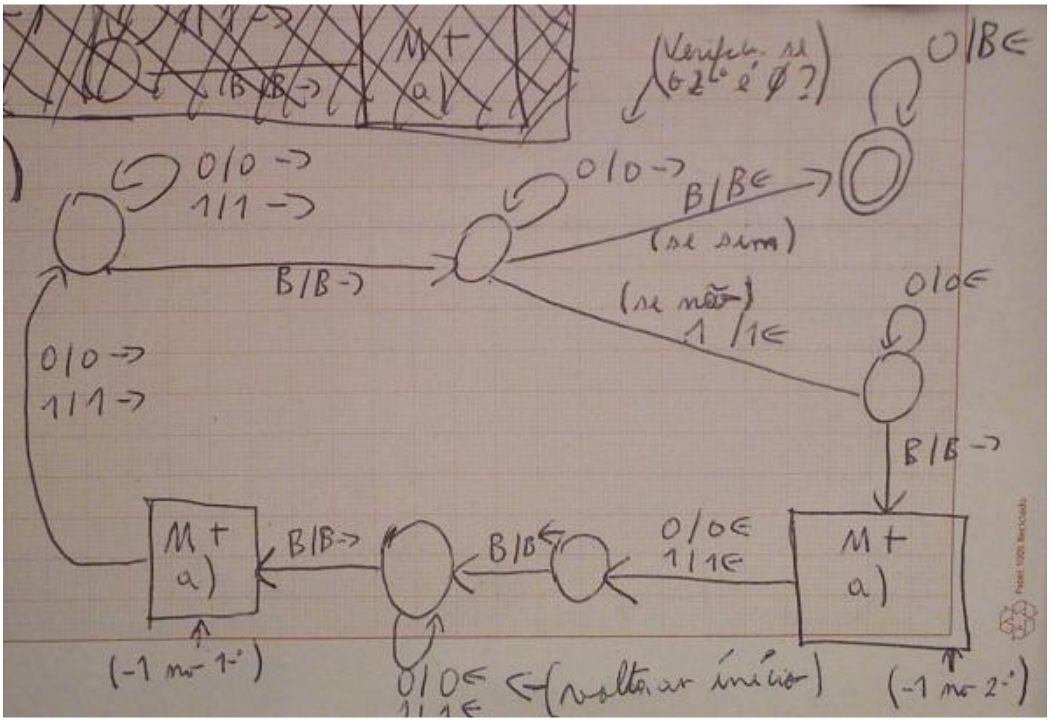
$q_1 0110 \square 0q_1 110 \square 01q_1 10 \square 011q_1 0 \square 0110q_1 B \square 011\, q_2 0 \square 01q_2 11 \square 0q_3 101 \square q_3 0101 \square q_3 B0101 \square q_4 0101$

4.c) Usando a Máquina de Turing desenvolvida em a) como um módulo, desenhe uma Máquina de Turing que subtraia dois números (primeiro pelo segundo) representados em binário. Os dois números a subtrair encontram-se na fita separados por uma (e só uma) posição sem nada (símbolo **B**). Assuma que o primeiro número é sempre maior do que o segundo e ambos são positivos não nulos.

Exemplos:

Input	Output
111B001	110
011001B011	010110

R:



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) Um PDA é não-determinista se a partir do mesmo estado existirem um ou mais símbolos na *string* de entrada, que podem levar a transições diferentes.

R:

Falsa. Um PDA é não-determinista se existir pelo menos um estado em que existam duas ou mais possibilidades de transição entre estados para o mesmo símbolo de entrada. Isso pode acontecer quando a partir de um estado existem duas ou mais transições espontâneas (ε) ou quando existem várias transições com o mesmo símbolo, mas em ambos os casos terá de estar o mesmo símbolo no topo da pilha.

5.b) As linguagens de programação Java e C/C++ são linguagens regulares.

R:

Falsa. Estas linguagens incluem regras que terão de verificar se por cada parêntesis aberto existe um parêntesis a fechar (ocorre o mesmo com as chavetas). Este tipo de característica torna essas linguagens não regulares.

5.c) A reunião de uma linguagem regular com uma linguagem não-regular origina sempre uma linguagem não-regular.

R:

Falsa. A reunião pode originar uma linguagem regular quando, por exemplo, a linguagem regular contém as palavras da linguagem não-regular. Exemplo: a linguagem resultante de $a^*b^* \cup a^nb^n$ *é uma linguagem regular.*

5.d) Qualquer que seja a linguagem regular que tenhamos existem sempre representações da mesma usando expressões regulares, DFAs, e ε-NFAs.

R:

Verdadeira. Qualquer expressão regular define por natureza uma linguagem regular. Qualquer linguagem regular pode ser representada por um ε-NFA e qualquer ε-NFA pode ser convertido num DFA.

5.e) O não determinismo nos NFAs é devido ao facto dos mesmos tanto poderem aceitar como não aceitar a mesma *string* de entrada, e por isso temos de os executar várias vezes para que o resultado seja o correcto.

R:

Falsa. Se a string de entrada pertencer à linguagem o NFA que representa essa linguagem aceita sempre essa string de entrada. O não-determinismo é relativo ao facto de a partir do mesmo estado poder haver pelo menos duas transições com o mesmo símbolo para estados diferentes.

5.f) Todas as linguagens livres de contexto podem ser processadas por uma Máquina de Turing.

R:

Verdadeira. As linguagens livres de contexto podem ser processadas por PDAs e esses PDAs podem ser implementados com uma Máquina de Turing (usando a fita para implementar a pilha).

(Fim.)