UNIVERSIDADE DE ÉVORA
DEPARTAMENTO DE INFORMÁTICA

Linguagens Formais e Autómatos
Resolução da Primeira Frequência
11 de novembro de 2015

Exercício 1 Seja L a linguagem das palavras sobre $\{a, b\}$ com, pelo menos dois símbolos, em que o primeiro símbolo e o segundo símbolo são diferentes. Por exemplo ba, aba estão em L enquanto que bb, aaa não estão.

1. **[1,5 valores]** Escreva uma expressão regular que represente L;
2. **[2,5 valores]** Defina um autómato finito determinista que reconheça L;

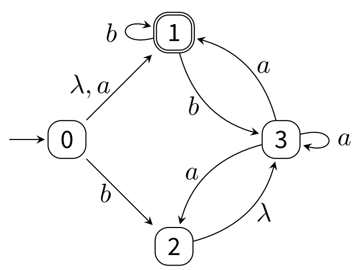
Uma expressão regular para a linguagem indicada é

$$(ab \cup ba)\,(a \cup b)^*$$

Um AFD que reconhece L é $A = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \delta, q_0, \{q_3\})$ em que a transição δ é dada por

q	a	b
q_0	q_1	q_2
q_1	q_4	q_3
q_2	q_3	q_4
q_3	q_3	q_3
q_4	q_4	q_4

Exercício 2 Considere o AFND definido pelo diagrama seguinte:



1. **[2,5 valores]** Aplique o algoritmo dado nas aulas para encontrar um AFD equivalente ao AFND definido acima;
2. **[1 valor]** Aplique algum dos algoritmos dados nas aulas para minimizar o AFD que encontrou na alínea anterior;
3. **[1 valor]** Diga (verdade ou falso), justificando, se a linguagem reconhecida por este autómato é o conjunto das palavras que não começam por aa;

A transição δ' do AFD equivalente obtém-se construindo a tabela

q	a	b	F
$q_0 = \lambda$-fecho $(0) = \{0, 1\}$	$\{1\}$	$\{1, 2, 3\}$	✓
$q_1 = \{1\}$	$\emptyset$	$\{1, 3\}$	✓
$q_2 = \{1, 2, 3\}$	$\{1, 2, 3\}$	$\{1, 3\}$	✓
$q_3 = \emptyset$	$\emptyset$	$\emptyset$	
$q_4 = \{1, 3\}$	$\{1, 2, 3\}$	$\{1, 3\}$	✓

onde o ✓ na última coluna significa que esse é um estado final. Portanto o AFD equivalente é

$$A' = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, q_0, \delta', \{q_0, q_1, q_2, q_4\}).$$

As partições iniciais são

$$I = \{q_0, q_1, q_2, q_4\} \qquad II = \{q_3\}$$

Como o único estado de I que transita para II é q_1, é preciso dividir

$$I = \{q_0, q_2, q_4\} \qquad II = \{q_3\} \qquad III = \{q_1\}$$

Agora, em I, apenas q_0 transita para III:

$$I = \{q_2, q_4\} \qquad II = \{q_3\} \qquad III = \{q_1\} \qquad IV = \{q_0\}$$

O AFD mínimo que se obtém é $A'' = (\{I, II, III, IV\}, \{a, b\}, IV, \delta'', \{I, III, IV\})$ em que a transição δ'' está definida na tabela

	q	a	b	F
	$I = \{q_2, q_4\}$	I	I	✓
	$II = \{q_3\}$	II	II	
	$III = \{q_1\}$	II	I	✓
$\rightarrow$	$IV = \{q_0\}$	III	I	✓

onde o estado inicial está indicado com $\rightarrow$.

Verdade: a única forma de rejeitar uma palavra é começar por $\rightarrow IV \xrightarrow{a} III \xrightarrow{a} II$.

Exercício 3 [2 valores] Seja $A = (Q, \{a, b\}, \delta, q_I, F)$ um AFD e considere $A' = (Q, \{a, b\}, \delta', q_I, F)$ em que a transição δ' está definida por

$$\delta'(q, a) = \delta(q, b) \qquad \delta'(q, b) = \delta(q, a) \qquad \forall q \in Q$$

Supondo que $L = \mathcal{L}(A)$ que linguagem é $\mathcal{L}(A')$? [*Sugestão*: considere A um autómato simples, por exemplo para reconhecer $\mathcal{L}\left((ab)^*\right)$, e encontre o respetivo A'.]

A linguagem $\mathcal{L}(A')$ é obtida de $\mathcal{L}(A)$ trocando a por b e vice-versa.

Exercício 4 [2,5 valores] Mostre que a GIC $G = (\{S, X\}, \{a, b\}, \{S \rightarrow bbX \mid Xaa, X \rightarrow bX \mid aX \mid \lambda\}, S)$ é ambígua.

Duas derivações direitas da palavra $bbaa$:

$$S \xrightarrow{S \rightarrow bbX} bbX \xrightarrow{X \rightarrow aX} bbaX \xrightarrow{X \rightarrow aX} bbaaX \xrightarrow{X \rightarrow \lambda} bbaa$$
$$S \xrightarrow{S \rightarrow Xaa} Xaa \xrightarrow{X \rightarrow bX} bXaa \xrightarrow{X \rightarrow bX} bbXaa \xrightarrow{X \rightarrow \lambda} bbaa$$

Exercício 5 [3 valores] Muitas linguagens de programação incluem testes da forma `if C { X }`. Estamos interessados na *guarda* do teste, `C`, que é uma *expressão booleana*. Para construir uma expressão booleana usam-se as operações `and`, `or`, `not` e predicados. Exemplos de predicados são `x == 3`, `y <= z` e `false`. A linguagem formal das expressões booleanas é definida pelas seguintes regras:

- qualquer predicado é uma expressão booleana;
- se a for uma expressão booleana então `(not` a`)` é uma expressão booleana;
- se a e b forem expressões booleanas então `(`a `and` b`)` e `(`a `or` b`)` são expressões booleanas;

Note que os parênteses "`(`" e "`)`" fazem parte das expressões.

As expressões booleanas *simplificadas* usam apenas o símbolo `p` para representar qualquer predicado e, para evitar escrever muitos parênteses, as seguintes regras de precedência: `not` tem precedência sobre `and` que tem precedência sobre `or`.

Por exemplo `not p or p and p` simplifica `((not p) or (p and p))` enquanto que `not (p and p or p)` simplifica `(not (p and (p or p)))`.

Defina uma GIC para as expressões booleanas simplificadas.

Uma GIC pretendida é $G = (\{B, C, N, A\}, \{\texttt{p}, \texttt{(}, \texttt{)}, \texttt{or}, \texttt{and}, \texttt{not}\}, \cdots, B)$ em que o conjunto das produções é

$$\begin{aligned} B &\to B \;\texttt{or}\; C \mid C \\ C &\to C \;\texttt{and}\; N \mid N \\ N &\to \texttt{not}\; A \mid A \\ A &\to \texttt{p} \mid \texttt{(}\; N \;\texttt{)} \end{aligned}$$

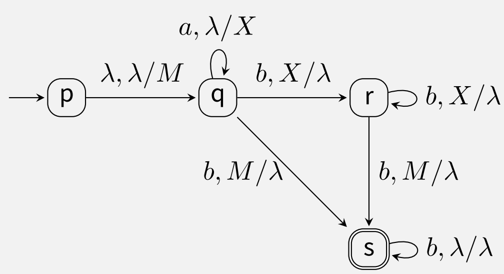
Exercício 6 Seja L a linguagem $\{a^i b^j \;:\; 0 \leq i < j\}$.

1. **[3 valores]** Defina um autómato de pilha para reconhecer L;
2. **[1 valor]** Para a demonstração de que L não é regular…
   - que palavra p usaria?
   - que decomposição (ou decomposições) de $p = uvw$ consideraria?
   - como, a partir daí, concluíria que L não é regular?

Um AP que reconhece L pode ser construído seguindo os seguintes pontos:

- antes de ser lido qualquer símbolo da palavra, é adicionado à pilha um “marcador”, M;
- se o primeiro símbolo da palavra for um b, ir diretamente para um estado de aceitação, s;
- se o primeiro símbolo da palavra for um a, acrescentar um “marcador” X à pilha;
- repetir o passo anterior até ser detetado o primeiro b; aí, transitar para um estado, r, em que os X’s são “cancelados” pelos b’s;
- se no estado anterior é detetado o marcador M e ainda existem b para ler, transitar para o estado de aceitação s;

Seguindo estes pontos obtemos



Supondo que L é regular, e sendo k o número de estados de controlo de um AFD que reconheça L,

$$p = a^k b^{k+1} \in L.$$

Pelo Lema de *Pumping*, na decomposição $p = uvw$ com $|uv| \leq k$ e $|v| > 0$, teria de ser

$$\begin{aligned} u &= a^x, & x &\geq 0 \\ v &= a^y, & y &> 0 \text{ e } x + y \leq k \\ w &= a^z b^{k+1}, & z &= k - (x + y) \end{aligned}$$

Ainda pelo Lema de *Pumping*, qualquer palavra $uv^n w \in L, n \geq 0$. Fazendo, por exemplo, $n = 2$ obtemos

$$uv^2 w = a^x a^{2y} a^z b^{k+1}$$

onde o número de a’s é $x + 2y + z > x + y + z = k$. Portanto, o número de a’s em $uv^2 w$ é, pelo menos, $k + 1$. Isto é, o número de a’s é igual ou maior que o número de b’s. Mas nas palavras de L isso não pode acontecer.
A linguagem não pode ser regular porque a suposição que é regular leva a uma contradição no Lema de *Pumping*.