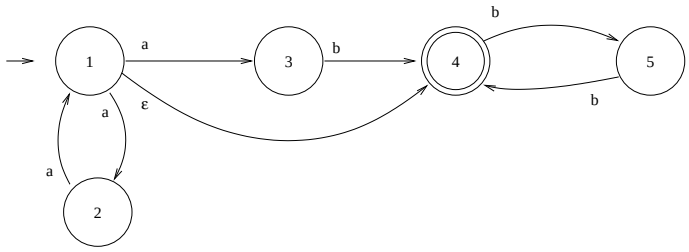
Universidad de Valladolid
Departamento de Informática

Teoría de autómatas y lenguajes formales . 2º I.T.Informática. Gestión.

Soluciones al examen de segunda convocatoria, 4 de septiembre de 2008

---

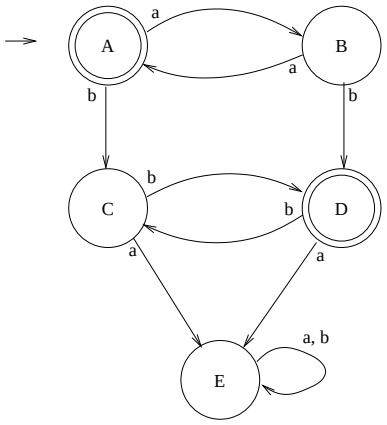
1 (a) Se puede partir de un RFN obtenido mediante el método de Thompson, o algo reducido, como



	a	b	ϵ
$\rightarrow 1$	$2,3$		4
2	1		
3		4	
(4)		5	
5		4	

para obtener el RFD

	a	b	
$\rightarrow (A)$	B	C	$1,4$
B	A	D	$2,3$
C	E	D	5
(D)	E	C	4
E	E	E	



que es mínimo, lo que puede comprobarse mediante el algoritmo de minimización:

$$\begin{array}{rcl} \mathcal{P}_0 & = & \{\ \{A,D\},\ \{B,C,E\}\ \} \\ \mathcal{P}_1 & = & \{\ \{A,D\},\ \{B\},\ \{C\},\ \{E\}\ \} \\ \mathcal{P}_2 & = & \{\ \{A\},\ \{D\},\ \{B\},\ \{C\},\ \{E\}\ \} \end{array}$$

o sencillamente obteniendo cadenas que distingan cada par de estados:

B	ϵ			
C	ϵ	a		
D	aa	ϵ	ϵ	
E	ϵ	b	b	ϵ
	A	B	C	D

Otra opción de solución es trabajar un poco con la expresión regular:
$(aa)^*(\epsilon|ab)(bb)^* \ = \ (aa)^*(bb)^* \mid (aa)^*ab(bb)^* \ = \ (aa)^* \mid (aa)^*bb(bb)^* \mid (aa)^*ab(bb)^*$
de donde podría obtenerse directamente el RFD con fácil justificación,
o también interpretar la expresión regular en su forma $(aa)^*(bb)^* \mid (aa)^*ab(bb)^*$, como cadenas de la forma a^nb^m donde n y m son ambos pares o ambos impares para explicar el RFD (A lleva la cuenta de número par de aes, B de número impar de aes, etc).

(b) Una forma de resolverlo es partir del RFD del apartado anterior y añadir arcos ϵ a cada arco existente, con el significado de poder borrar ese símbolo o no, es decir, obtener subsecuencias. (Realmente sólo hace falta

poner esos arcos en los que puedan pertenecer a un camino que lleve a un estado final). El reconocedor es entonces:

	a	b	ϵ
$\rightarrow (A)$	B	C	B, C
B	A	D	A, D
C	E	D	E, D
(D)	E	C	E, C
E	E	E	E

de donde se obtiene el RFD

	a	b	
$\rightarrow (1)$	1	2	$\underline{A}, B, C, D, E$
(2)	3	2	$\underline{C}, \underline{D}, \underline{E}$
3	3	3	$\underline{E}$

y de ahí, el lenguaje pedido

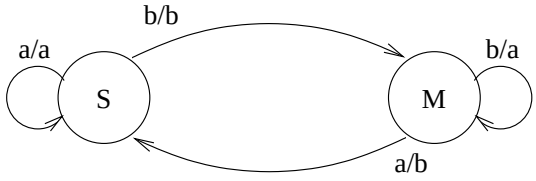
$$a^*b^*$$

Otra forma de resolverlo es partir de la expresión y sustituir en ella cada símbolo e por $e|\epsilon$, nuevamente porque las subsecuencias se obtienen borrando o no cada símbolo que aparezca:

$$((a|\epsilon)(a|\epsilon))^*(\epsilon|(a|\epsilon)(b|\epsilon))((b|\epsilon)(b|\epsilon))^*$$

Para simplificar la expresión, se usa que:
$(a|\epsilon)(b|\epsilon) = \epsilon|a|b|ab \qquad (a|\epsilon)(a|\epsilon) = \epsilon|a|aa \qquad (\epsilon|a|aa)^* = (a|aa)^* = a^*$
para obtener las equivalencias:
$a^*(\epsilon|a|b|ab)b^* \;=\; a^*b^* \mid a^*ab^* \mid a^*bb^* \mid a^*abb^* \;= a^*b^*$

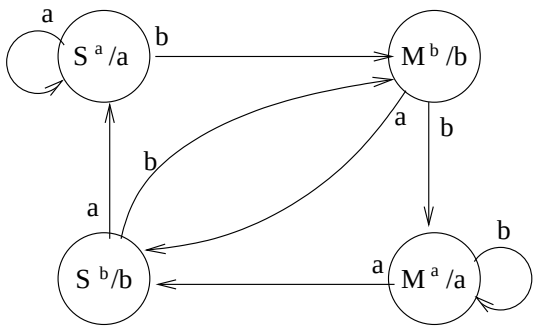
2 (a) Máquina de Mealy:



En forma de tabla:

	a	b
S	S/a	M/b
M	S/b	M/a

(b) Tranfomado a máquina de Moore:



En forma de tabla:

	a	b
S^a/a	S^a	M^b
S^b/a	S^a	M^b
M^a/a	S^b	M^a
M^b/b	S^b	M^a

(c) Considerando la máquina del primer apartado, basta quitar los arcos que llevan asociada salida a y encontrar la expresión regular de las cadenas no vacías que lo recorren, partiendo del estado S (y llegando a cualquiera de los dos estados):
$b(ab)^*(\epsilon|a) \;=\; (ba)^*(b|ba) \;=\; (ba)^+ \,|\, (ba)^*b$

3 (a) Si la diferencia es múltiplo de 3, tiene que suceder que

- o ambos son múltiplos de 3
- o ambos son múltiplos de 3 más 1
- o ambos son múltiplos de 3 más 2

Por lo tanto, las cadenas deben tener una de las formas

- $a^{3p}b^{3q}$
- $a^{3p+1}b^{3q+1}$
- $a^{3p+2}b^{3q+2}$

es decir: constituyen el lenguaje regular

$$(aaa)^*(bbb)^* \mid a(aaa)^*b(bbb)^* \mid aa(aaa)^*bb(bbb)^*$$

Otra forma de abordar el problema: cara al resto módulo 3, se puede prescindir de tríos de aes o de tríos de bes (de los extremos), de forma que queden un número de aes y bes menores que 3. La parte central entonces debe tener 0, 1 ó 2 aes y 0, 1 ó 2 bes. Para que la diferencia sea múltiplo de tres, las únicas posibilidades son ϵ, ab y $aabb$, de donde se obtiene la expresión regular

$$(aaa)^*(\epsilon \,|\, ab \,|\, aabb)(bbb)^*$$

(b) Demostrar que $L_2 = \{a^ib^j \ / \ i \geq j \text{ y } i-j \text{ es múltiplo de 3 }\}$ es independiente de contexto no regular
Para probar que es independiente de contexto puede partirse de que

$$L_2 = L_3 \cap \{a^nb^m \ / \ n \geq m\}$$

y por lo tanto, como intersección de un independiente de contexto con un regular, L_2 es independiente de contexto.
Otra forma de verlo (\$):
si $i \geq j$ y $i-j = 3k$ para cierto $k \geq 0$, se tendrá que
$i = (i-j)+j = 3k+j$, donde $k \geq 0$ y $j \geq 0$
luego el lenguaje puede reescribirse:
$L_2 = \{a^{3k}a^jb^j \ / \ j,k \geq 0\} = (aaa)^*\{a^jb^j \ / \ j \geq 0\}$
y L_2 aparece entonces como concatenación de dos lenguajes independientes de contexto, así que es independiente de contexto.
También puede construirse la gramática generadora de L_2 basada en esta última consideración:

$$\begin{cases} S & \rightarrow & aaaS \mid R \\ R & \rightarrow & aRb \mid \epsilon \end{cases}$$

Para demostrar que no es regular, se puede probar que no cumple el lema de bombeo de los regulares:
dado un $N > 0$ cualquiera, la cadena a^Nb^N está evidentemente en L_2 y tiene longitud $2N \geq N$. Debería admitir una subcadena no nula bombeable entre los primeros N caracteres, es decir, una cadena de aes, $v = a^p$ con $p > 0$. Pero (aunque p fuera un múltiplo de 3), al quitarla, quedaría $a^{N-p}b^N$ que no está en L_2 (porque el número de aes es menor que el de bes).

NOTA: Se puede resolver este ejercicio en orden inverso: demostrar primero que L_2 es independiente de contexto por el segundo razonamiento (marcado con \$) y expresar después L_3
como la unión de L_2 y $L_2' = \{a^ib^j \ / \ i \leq j \text{ y } j-i \text{ es múltiplo de 3 }\}$
que también es independiente de contexto por un razonamiento similar al de L_2.

4 El AP es el que obtendría desde la gramática en FNG siguiente el algoritmo visto en la asignatura:

$$\begin{cases} S & \rightarrow & aSA \mid a \\ S & \rightarrow & bSB \mid b \\ A & \rightarrow & a \\ B & \rightarrow & b \end{cases}$$

Otra posibilidad para hallarla es utilizar el algoritmo que permite obtener a partir de un AP la gramática que genera el lenguaje que acepta por vaciado de pila:

$$Z \rightarrow [qSq] \qquad \begin{matrix} [qSq] \rightarrow a[qSq][qAq] \\ [qSq] \rightarrow a \end{matrix} \qquad \begin{matrix} [qSq] \rightarrow b[qSq][qBq] \\ [qSq] \rightarrow b \end{matrix} \qquad \begin{matrix} [qAq] \rightarrow a \\ [qBq] \rightarrow b \end{matrix}$$

(renombrando se obtiene la misma gramática).

La gramática es evidentemente equivalente a

$$S \rightarrow aSa \mid bSb \mid a \mid b$$

de donde puede deducirse que el lenguaje aceptado por vaciado de pila es

$$\{waw^R \ / \ w \in (a|b)^*\} \cup \{wbw^R \ / \ w \in (a|b)^*\}$$

es decir, cadenas de palíndromos impares sobre $\{a, b\}$.

5 El algoritmo de eliminación de símbolos inútiles transforma la gramática en

$$\begin{array}{lcl} S & \rightarrow & AA \mid b \\ A & \rightarrow & aSb \end{array}$$

(B y C son inútiles por no terminables, y una vez eliminados,

$$\begin{array}{lcl} S & \rightarrow & AA \mid b \\ A & \rightarrow & aSb \\ D & \rightarrow & a \end{array}$$

D resulta inaccesible).

Se ha obtenido una gramática equivalente (que no tiene recursión por la izquierda). Se puede aplicar el algoritmo estándar (pasar a FNC, numerar los símbolos, etc.) o directamente añadir un nuevo auxiliar N:

$$\begin{array}{lcl} S & \rightarrow & AA \mid b \\ A & \rightarrow & aSN \\ N & \rightarrow & b \end{array} \qquad \text{y de ahí :} \qquad \begin{array}{lcl} S & \rightarrow & aSNA \mid b \\ A & \rightarrow & aSN \\ N & \rightarrow & b \end{array}$$

6 (a) CIERTO. La intersección de 2 lenguajes recursivamente numerables lo es. (Basta considerar a 2 máquinas reconocedoras de los dos lenguajes trabajando a pasos alternativos. Si en algún momento ambas están paradas en un estado de aceptación, se acepta. Esta combinación parará aceptando ante una cadena si y sólo si la cadena es aceptada por ambas máquinas, es decir, si y sólo si está en la intersección de los lenguajes).

(b) FALSO. No es posible que AMBOS lo sean: de serlo, su unión también lo sería, pero su unión es

$$(L \cap R) \cup (L \cap \overline{R}) = L \cap (R \cup \overline{R}) = L$$

así que L sería recursivo, y sabemos que no lo es.

(c) CIERTO: uno de ellos podría serlo. Por ejemplo, si $L \subset (a|b)^*$ y $R = c^*$, entonces $L \cap R = \emptyset$ que es recursivo.

(d) FALSO: si ambos fueran finitos, su unión, L, sería finita y por lo tanto recursiva.