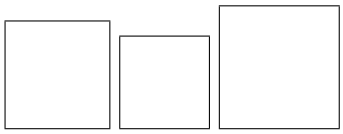
Universidad de Valladolid
Departamento de Informática
Teoría de autómatas y lenguajes formales . 2º I.T.Informática. Gestión.

Examen de primera convocatoria. 18 de junio de 2009

Apellidos, Nombre..... **Grupo:** .....

Notas para la resolución del examen: Responder a los ejercicios 5 a 8 en la misma página del enunciado. Responder y entregar por separado los ejercicios 9 y 10.



---

1 **(1 p.)** Construya una máquina de Mealy que obtenga el cociente entero entre 3 de un entero binario. Por ejemplo, para la entrada 101000101(325) la salida debe ser 001101100(108)

Indicación: son situaciones distintas que la cadena de entrada represente un múltiplo de 3, un múltiplo de 3 + 1, o un múltiplo de 3 + 2. Añadir un 0 a un número binario supone multiplicarlo por 2, mientras que añadir un 1 supone multiplicarlo por 2 y sumarle 1; en ambos casos el resto y el cociente se modifican en consecuencia.

Solución:

Si un número (binario) es múltiplo de 3, o sea $3k$, su cociente respecto a 3 es k. Si se le añade un 0, su valor pasa a ser $6k$, sigue siendo múltiplo de 3, y su cociente respecto a 3 es $2k$, que se obtiene a partir de la respresentación binaria de k añadiendo un 0.

Si se le añade un uno, su valor pasa a ser $6k+1$, múltiplo de 3 más 1, y cuyo cociente respecto a 3 es $2k$, que se obtiene añadiendo un 0 igual que en el caso anterior.

El mismo tipo de razonamiento se aplica en los otros casos. Por ejemplo, si el número es múltiplo de 3 más 1, o sea $3k+1$, su cociente es k, y al añadir un 1 se transforma en $6k+3$, cuyo cociente es $2k+1$ que se obtiene de k añadiendo un 1.

El resumen se ve en la siguiente tabla:

cadena binaria	valor	resto	cociente	cadena binaria	valor	resto	cociente
x	$3k$	0	k	$x0$	$6k$	0	$2k$
				$x1$	$6k+1$	1	$2k$
y	$3k+1$	1	k	$y0$	$6k+2$	2	$2k$
				$y1$	$6k+3$	0	$2k+1$
z	$3k+2$	2	k	$z0$	$6k+4$	1	$2k+1$
				$z1$	$6k+5$	2	$2k+1$

Se aquí se puede deducir la máquina de Mealy siguiente:

	0	1	
0	3/0	1/0	inicio
1	2/0	3/1	múltiplo de 3 más 1
2	1/1	2/1	múltiplo de 3 más 2
3	3/0	1/0	múltiplo de 3

que tiene el comportamiento pedido si comienza en el estado 0. (También sería válida la máquina sin el estado 0, comenzando en el estado 3).

2 **(2 p.)** 1. Obtenga un reconocedor finito determinista mínimo equivalente al dado por la tabla siguiente.

	0	1
$\rightarrow p$	$\{p,q\}$	$\{p\}$
q	$\{r,s\}$	$\{t\}$
r	$\{p,r\}$	$\{t\}$
(s)		
(t)		

Solución:

	0	1	
$\to A$	B	A	p
B	C	D	p, q
(C)	C	D	p, q, r, s
(D)	B	A	p, t

Es mínimo:

cadenas que distinguen

B	$0, 1$		
C	ϵ	ϵ	
D	ϵ	ϵ	$0, 1$
	A	B	C

ó alg. de minimización:
$\mathcal{P}_0 = \{ \{A, B\}, \{C, D\} \}$
$\mathcal{P}_1 = \{ \{A\}, \{B\}, \{C\}, \{D\} \}$

2. Obtenga una gramática regular por la izquierda para el lenguaje

Solución:

Del reconocedor anterior se obtiene la gramática regular por la izquierda

$$\begin{array}{l} S \to D \mid C \\ A \to A1 \mid D1 \mid \epsilon \\ B \to A0 \mid D0 \\ C \to B0 \mid C0 \\ D \to B1 \mid C1 \end{array} \quad \text{eliminando reglas simples:} \quad \begin{array}{l} S \to B1 \mid C1 \mid B0 \mid C0 \\ A \to A1 \mid D1 \mid \epsilon \\ B \to A0 \mid D0 \\ C \to B0 \mid C0 \\ D \to B1 \mid C1 \end{array}$$

3. Obtenga una gramática regular por la derecha para el lenguaje

Solución:

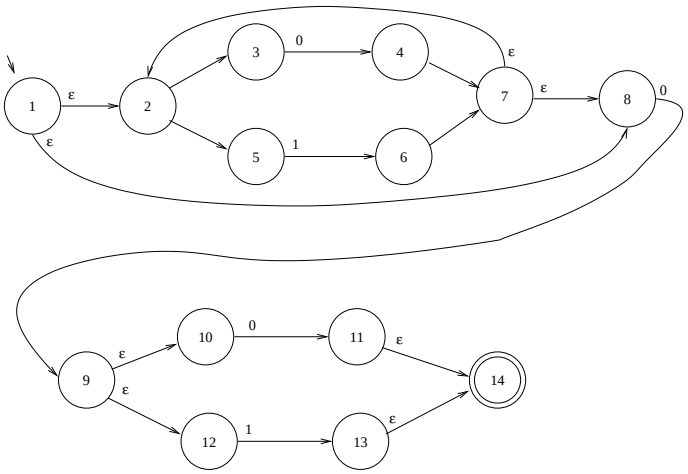
También del reconocedor anterior:

$$\begin{array}{l} A \to 0B \mid 1A \\ B \to 0C \mid 1D \\ C \to 0C \mid 1D \mid \epsilon \\ D \to 0B \mid 1A \mid \epsilon \end{array}$$

4. Justifique si el lenguaje reconocido es o no el de las cadenas binarias cuyo penúltimo carácter es un 0

Solución:

El lenguaje formado por las cadenas binarias cuyo penúltimo carácter es 0 es evidentemente el descrito por la expresión regular $(0|1)^*0(0|1)$. El método de Thompson obtiene el siguiente reconocedor finito a partir de ella:



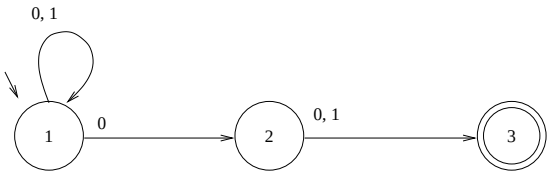
	0	1	ϵ
$\to 1$			$2, 8$
2			$3, 5$
3	4		
4			7
5		6	
6			7
7			$2, 8$
8	9		
9			$10, 12$
10	11		
11			14
12		13	
13			14
(14)			

El algoritmo de determinación obtiene:

	0	1	
$\to A$	B	C	$1, 2, 3, 5, 8$
B	D	E	$2, 3, 4, 5, 7, 8, 9, 10, 12$
C	B	C	$2, 3, 5, 6, 7, 8$
(D)	D	E	$2, 4, 7, 8, 9, 10, 11, 12, 14$
(E)	B	C	$2, 3, 5, 6, 7, 13, 14$

que minimizado (A es equivalente a C), es el del apartado 1; por lo tanto el lenguaje reconocido es efectivamente el propuesto.

De forma alternativa, se podría partir del reconocedor



	0	1
$\to 1$	$1,2$	1
2	3	3
(3)		

del que se obtiene el determinista

$\to A$	B	A	1
B	C	D	$1,2$
(C)	C	D	$1,2,3$
(D)	B	A	$1,3$

que coincide con el del apartado 1.

También podría resolverse el ejercicio justificando que el lenguaje reconocido es el que se propone, analizando cualquiera de los autómatas (el no determinista propuesto en el apartado 1 o el determinista obtenido a partir de él), para probar que

- si el reconocedor acepta una cadena, su penúltimo carácter es un 0
- si el penúltimo carácter de la cadena es un 0, el reconocedor la acepta (o alternativamente, si el reconocedor no la acpta, lo anterior no ocurre)

Partiendo del no determinista propuesto:

- si se acepta una cadena, se habrá llegado

  al estado s: únicamente por venir de q con un solo símbolo, 0, y para llegar a q, necesariamente se ha debido leer un 0, que será el penúltimo carácter

  al estado t: se procede de q ó de r, mediante un 1, y a q y r se accede necesariamente mediante un 0

  Por lo tanto, si se acepta una cadena, el penúltimo carácter es un 0
- recíprocamente, si el penúltimo carácter de una cadena es 0, será de la forma $x0a$ con $a \in \{0,1\}$ y $x \in (0|1)^*$. El reconocedor podrá con esa cadena hacer las transiciones:
  $(p, x0a) \vdash (p, 0a) \vdash (q, a)$
  y entonces pasar a (s, ϵ) si $a = 0$, o a (t, ϵ) si $a = 1$, y en ambos casos aceptar

Partiendo del determinista obtenido:

- si se acepta una cadena, se ha llegado a C ó a D.
  La llegada a C supone haber leído dos símbolos al menos, y, quitando el último símbolo, se ha venido de C, o de B, o de D, en cualquier caso con un 0.
  La llegada a D supone llegar con un 1, de B ó C, al que se ha llegado con un 0
- como el reconocedor es completo, cualquier cadena que no se acepta, lleva al autómata a A ó a B. Por lo tanto, o bien la cadena no tiene penúltimo carácter (porque es de longitud 0 ó 1), o, sí lo tiene; en este caso, la llegada a B procede de A ó D, a los que se llega necesariamente con un 1 (penúltimo carácter), mientras que la llegada a A procede del propio A o de D, a los que igualmente se llega con un 1 Por lo tanto, si la cedena no se acepta, o no tiene penúltimo carácter, o lo tiene y es un 1

Finalmente, también se puede resolver este ejercicio obteniendo algorítmicamente de cualquiera de los reconocedores el lenguaje reconocido, y probando que coincide con $(0|1)^*0(0|1)$

3 **(1'5 p.)** Considere el conjunto de cadenas no vacías de paréntesis y corchetes equilibrados, entendiendo que los paréntesis se equilibran con paréntesis y los corchetes con corchetes. (Por ejemplo, ([[ ][ ]])() está equilibrada, pero ( ][ ) no lo está). Recodifique los paréntesis (y) como a y b respectivamente.

1. Diseñe una gramática que genere este lenguaje

   Solución:

   La gramática

   $$S \to [\,S\,]\,S \mid (S)S \mid \epsilon$$

   genera las cadenas equilibradas de paréntesis y corchetes. Por lo tanto, la gramática

   $$\begin{array}{rcl} S & \to & [\,S\,]\,S \mid [\,S\,] \mid [\,]\,S \mid [\,] \\ & & \mid (S)S \mid (S) \mid ()S \mid () \end{array}$$

   genera el lenguaje pedido.

2. Diseñe un autómata con pila determinista que lo reconozca

   Solución

   Considere el autómata con pila en el que

$\Sigma = \{a, [, b,]\}$, $\Gamma = \{a, [, Z\}$, $Q = \{q_1, q_2, q_2\}$, $F = \{q_3\}$, el estado inicial es q_1, el símbolo inicial de la pila es Z, y la función de transición viene dada por las siguientes tablas:

$\rightarrow q_1$	a	$[$
Z	q_2, aZ	$q_2, [\, Z$

(q_3)	a	$[$
Z	q_2, aZ	$q_2, [\, Z$

q_2	a	$[$	b	$]$	ϵ
a	q_2, aa	$q_2, [\, a$	q_2, ϵ		
$[$	$q_2, a\,[$	$q_2, [\, [$		q_2, ϵ	
Z					q_3, Z

Desde el estado inicial, la lectura de un elemento de apertura (`a` ó `[`) lleva al estado q_2 apilando tal elemento. En el estado q_2, hay uno o varios elementos "abiertos", que estarán en la pila. En ese estado se continúa, apilando los elementos de apertura, y desapilándolos si aparece el correspondiente de cierre. Se detectará que la cadena está equilibrada cuando se encuentre el símbolo de inicio de pila en ella, en cuyo caso se pasa al estado q_3, de aceptación. Si en este estado la cadena ha terminado, se ha aceptado; pero la aparición de otro elemento de apertura llevará de nuevo al estado q_2, para permanecer en él hasta que se vuelva a equilibrar la cadena, y repetir el proceso tantas veces como sea necesario.

3. Demuestre que no es regular.

   Solución:

   Si se interseca el lenguaje (L) con a^*b^* se obtiene el lenguaje $\{a^n b^n \ / \ n > 0\}$, que no es regular. Por lo tanto L no puede serlo, ya que la intersección de dos lenguajes regulares es regular.

4 **(1 p.)** Probar las siguientes afirmaciones

1. La intersección de dos lenguajes recursivos es recursiva.

   Solución:

   Sean L_1 y L_2 dos lenguajes recursivos. Se sabe que la unión de dos lenguajes recursivos lo es, y que el complementario de un recursivo también es recursivo.

   Por lo tanto, $\overline{L_1}$ y $\overline{L_1}$ son recursivos, y también $\overline{L_1} \cup \overline{L_2}$ lo es y también $\overline{\overline{L_1} \cup \overline{L_2}}$, que es, según las leyes de De Morgan, $L_1 \cap L_2$

   Como solución alternativa, se puede construir, a partir de las máquinas reconocedoras de L_1 y L_2 una máquina reconocedora de $L_1 \cap L_2$. La nueva máquina lee la cadena de entrada y se la da a la máquina para L_1. Esta máquina parará aceptando o no. Si no acepta, se detiene la construcción sin aceptarla, y si acepta, se le da una copia de la cadena a la máquina para L_2 que también parará aceptando o no. Si ésta no acepta, se para la construcción sin aceptar, y si acepta, se para aceptando.

```
leer x
aplicar MR1 a x
si la respuesta es "NO", entonces escribir "NO" y parar (x no está en L1)
si no
    aplicar MR2 a x
    si la respuesta es "NO" entonces escribir "NO" y parar (x no está en L2)
    si no escribir "SI" (x está en L1 y L2)
    fin si
fin si
```

2. La intersección de dos lenguajes recursivamente numerables es recursivamente numerable.

   Solución:

   Si L_1 y L_2 son recursivamente numerables, existe para cada uno de ellos una máquina reconocedora que se parará aceptando las cadenas del lenguaje (aunque posiblemente no se pare para alguna de las cadenas que no estén en él). Se puede construir una máquina reconocedora de la intersección mediante el siguiente "algoritmo":

```
leer x
aplicar MR1 a x (puede que no se pare)
si la respuesta es "SI" entonces (llegar aquí supone que MR1 se ha parado)
    aplicar MR2 a x (puede que no se pare)
    si la respuesta es "SI" entonces (también MR2 se ha parado)
         escribir "SI" y parar
    si no escribir "NO" y parar
    fin si
si no, escribir "NO" y parar
fin si
```

Puede que la máquina basada en el algoritmo anterior no se pare, pero

- se para ante cualquier cadena de $L_1 \cap L_2$, aceptando
- si se para aceptando, la cadena estará en $L_1 \cap L_2$

por lo tanto reconoce la intersección (aunque pueda no pararse ante alguna cadena de su complementario), lo que hace a la intersección recursivamente numerable.

3. Si L_r es recursivo y L_n es recursivamente numerable, entonces $L_n - L_r$ es recursivamente numerable.

   Solucion:

   Si L_r es recursivo, también lo es $\overline{L_r}$, y por lo tanto también es recursivamente numerable. Por el resultado del apartado anterior, $L_n \cap \overline{L_r}$ será recursivamente numerable, y es precisamente $L_n - L_r$

4. Si L_r es recursivo y L_n es recursivamente numerable, entonces $L_r - L_n$ no tiene por qué ser recursivamente numerable, es decir, puede serlo o no serlo.

   Solución:

   Se trata de encontrar un ejemplo en el que $L_r - L_n$ sea recursivamente numerable y otro en el que no lo sea.

   - Sea L_n un lenguaje recursivamente numerable que también sea recursivo. Entonces su complementario es recursivo, de forma que $L_r \cap \overline{L_n} = L_r - L_n$ es recursivo, luego recursivamente numerable.
     Otro ejemplo: sea L_r un lenguaje recursivo sobre el alfabeto $\{a, b\}$ y L_n un lenguaje recursivamente numerable (recursivo o no, como se quiera), sobre el alfabeto $\{c, d\}$. Evidentemente ambos son lenguajes sobre el alfabeto $\{a, b, c, d\}$, y su diferencia es el lenguaje vacío, que es recursivamente numerable.
   - Sea L_n un lenguaje recursivamente numerable no recursivo sobre el alfabeto Σ, y sea $L_r = \Sigma^*$ (que es recursivo; de hecho es hasta regular). $L_r - L_n$ es el complementario de L_n, que no puede ser recursivamente numerable.

5 **(0.5 p.)** Dé un ejemplo de tres lenguajes no vacíos L_1, L_2 y L_3 tales que $L_2 \neq L_3$ pero $L_1L_2 = L_1L_3$

Ejemplos posibles:

L_1	L_2	L_3	$L_1L_2 = L_1L_3$
a^*	a^*	ϵ	a^*
a^*	a	aa^*	aa^*
$(a\|b)^*$	a^*	b^*	$(a\|b)^*$
a^*b^*	b^*	ϵ	a^*b^*
$(a\|b)^*$	$\{a^nb^n \ / \ n \geq 0\}$	a^*b^*	$(a\|b)^*$

6 **(1 p.)** Construya la Tabla de Análisis Sintáctico Predictivo para la gramática siguiente, una vez eliminados los símbolos y reglas inútiles, y especificando los primeros y siguientes de cada auxiliar.

$S \rightarrow DA \mid PA \mid PQR$ $\quad$ $B \rightarrow BA \mid A$ $\quad$ $P \rightarrow R \mid cP$ $\quad$ $R \rightarrow bR \mid \epsilon$

$A \rightarrow BP$ $\quad$ $D \rightarrow aD \mid \epsilon$ $\quad$ $Q \rightarrow aPS \mid c$

Gramática sin reglas inútiles:

$S \rightarrow P\,Q\,R$
$P \rightarrow R \mid c\,P$
$Q \rightarrow a\,P\,S \mid c$
$R \rightarrow b\,R \mid \epsilon$

A y B son no terminables. Al eliminarlos se obtiene una gramática en la que D es inaccesible.

Primeros	S	P	Q	R			
	b	b	a	b			
	c	c	c	ϵ			
	a	ϵ					

Siguientes	S	P	Q	R			
	\$	**a**	**b**	a			
	b	**c**	\$	c			
		b		b			
				\$			

(En los siguientes aparecen en negrita los símbolos que surgen antes de hacer las transferencias de siguientes de S a Q y R, de P a R, y de Q a S)

TASP	a	b	c	\$
S	S → P Q R	S → PQR	S → PQR	
P	P → R	P → R	P → R ; P → cP	
Q	Q → aPS		Q → c	
R	R → ϵ	R → bR ; R → ϵ	R → ϵ	R → ϵ

7 **(0.5 p.)** Determine los pivotes de las cadenas siguientes respecto a la gramática $S \rightarrow aSbS \mid bSaS \mid \epsilon$ (Nota: como la gramática es ambigua, es posible que alguna cadena tenga más de un pivote).

- $aabb$: pivotes: ϵ en la posición 3 ($aa\underline{\epsilon}bb$), con la regla $S \rightarrow \epsilon$
- $abab$: pivotes: ϵ en la posición 3 ($ab\underline{\epsilon}ab$) con la regla $S \rightarrow \epsilon$; ϵ en la posición 2 ($a\underline{\epsilon}bab$) con la regla $S \rightarrow \epsilon$

8 **(0.5 p.)** La gramática

$$R \rightarrow R + R \mid R \bullet R \mid R* \mid (R) \mid \mathbf{0} \mid \mathbf{1} \mid \mathbf{a} \mid \mathbf{b} \mid \mathbf{c}$$

genera las expresiones regulares sobre el alfabeto { **a**, **b**, **c** } pero de forma ambigua (el terminal **1** representa a la expresión regular ϵ y el terminal **0** a la expresión regular $\emptyset$. La concatenación se expresa siempre con $\bullet$, el asterisco no se pone como superíndice y la unión se representa con el símbolo $+$, de forma que lo que típicamente se escribe $(a|b)^*abb|\epsilon$ debe escribirse ahora como $(a+b)*\bullet a \bullet b \bullet b + 1$).

Obténgase una gramática no ambigua para este lenguaje que refleje las precedencias y asociatividades habituales.

$R \rightarrow R + T \mid T$ $\quad$ $F \rightarrow F* \mid U$

$T \rightarrow T \bullet F \mid F$ $\quad$ $U \rightarrow (R) \mid \mathbf{0} \mid \mathbf{1} \mid \mathbf{a} \mid \mathbf{b} \mid \mathbf{c} \mid$

Universidad de Valladolid
Departamento de Informática

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

Examen de primera convocatoria. 18 de junio de 2009

Apellidos, Nombre.....

Notas para la resolución del examen: Entregar estos ejercicios separados de los anteriores.

---

9 **(1'25 p.)** Construya un programa, utilizando las herramientas Lex y Yacc, que obtenga, para una expresión regular dada sobre el alfabeto { **a**, **b**, **c** } el número de estados y el número de arcos ϵ que generaría el método de Thompson al construir el autómata finito (no determinista) a partir de dicha expresión. (Ver ejercicio 8).

Por ejemplo, para la expresión regular $(a + b) * \bullet a \bullet b \bullet b$ debería obtenerse que tiene 11 estados y 8 arcos ϵ.

10 **(0.75 p.)** Se desea construir un analizador sintáctico para las expresiones de sumas de vectores de números reales de hasta 10 elementos. Los elementos de cada vector estarán separados por espacios en blanco y encerrados entre corchetes. El analizador léxico para resolver el problema se construiría utilizando la herramienta Lex. El programa debería devolver el resultado de realizar las operaciones algebraicas indicadas en la expresión de la entrada sólo en el caso de que ésta sea correcta. Por ejemplo:

Entrada : `([1.0  0  -1  1 ] + [1  0.1  -1  0]) + [1  4.3  5  7.2]`

Salida esperada : `[3.0  4.4  3  8.2]`

Se pide **solamente** determinar una gramática y tipos semánticos necesarios para poder resolver este problema y especificar en cuál de los códigos fuente (Lex o Yacc) deberían estar definidos.