Universidad de Valladolid
Departamento de Informática

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

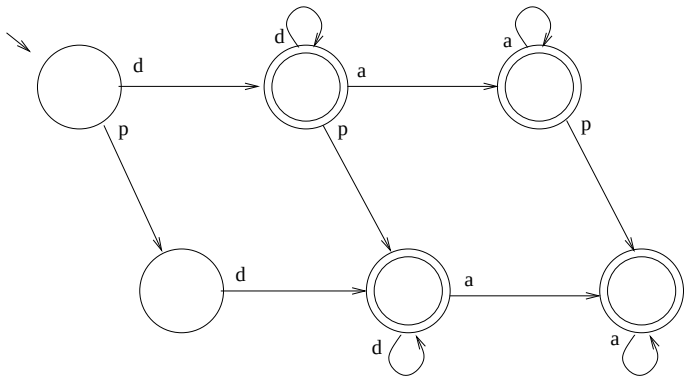
Examen de primera convocatoria. 15 de junio de 2010 **PARTE 1**

Notas para la resolución del examen.

(REVISAR)

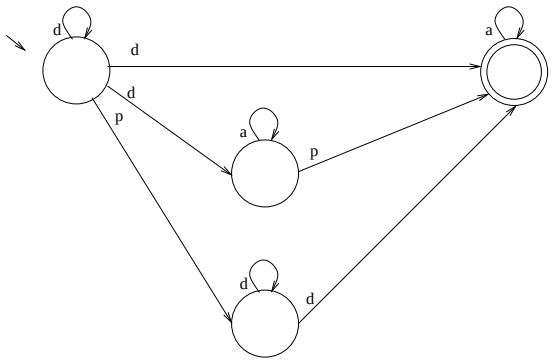
---

1 **(1'5 p.)** El reconocedor pedido es el siguiente (en la forma de diagrama no se ha reflejado el estado "de fallo")



	d	a	p
$\rightarrow 1$	2	7	4
(2)	2	3	5
(3)	7	3	6
4	5	7	7
(5)	7	6	7
(6)	7	6	7
7	7	7	7

El lenguaje está bien descrito mediante la expresión $d^*(d|da^*p|pdd^*)a^*$, lo que se justifica, por ejemplo, construyendo a partir de esa expresión un RFN como el siguiente:



	d	a	p
$\rightarrow 1$	$1,2,4$		3
2		2	4
3	$3,4$		
(4)		4	

El algoritmo de determinación obtendría

	d	a	p	
$\rightarrow A$	B	C	D	1
(B)	B	E	F	$1,2,4$
C	C	C	C	
D	F	C	C	3
(E)	C	E	G	$2,4$
(F)	F	G	C	$3,4$
(G)	C	G	C	4

que es (isomorfo a) el primero (la equivalencia de nombres es: A=1 B=2 C=7 D=4 E=3 F=5 G=6)

2 **(1 p.)** ▪ Simulación con las cadenas de entrada *aaabba* y *baa*

Como el autómata es indeterminista, hay varias posibilidades para cada cadena:
para la primera de ellas:

$(q, aaabba, Z) \vdash (q, aaabba, \epsilon)$

$(q, aaabba, Z) \vdash (q, aabba, PZ) \vdash (q, abba, bZ) \vdash (q, bba, PbZ) \vdash (q, ba, abZ) \vdash (q, a, aaabZ) \vdash (q, \epsilon, aabZ)$

$(q, aaabba, Z) \vdash (q, aabba, PZ) \vdash (q, abba, bZ) \vdash (q, bba, bPZ) \vdash (q, ba, PZ) \vdash (q, a, aZ) \vdash (q, \epsilon, Z) \vdash (q, \epsilon, \epsilon)$

Se ve por lo tanto que el autómata acepta (por vaciado de pila) a la cadena *aaabba*

Para *baa* hay 2 computaciones:

$(q, baa, Z) \vdash (q, baa, \epsilon)$

$(q, baa, Z) \vdash (q, aa, aaZ) \vdash (q, a, aZ) \vdash (q, \epsilon, Z) \vdash (q, \epsilon, \epsilon)$

Y por lo tanto también se acepta esta cadena

- El algoritmo correspondiente da la siguiente gramática:

$$\begin{array}{rl} S & \to [q\,Z\,q] \\ [q\,Z\,q] & \to a[q\,P\,q]\,[q\,Z\,q] \\ [q\,Z\,q] & \to b[q\,a\,q]\,[q\,a\,q]\,[q\,Z\,q] \\ [q\,Z\,q] & \to \epsilon \\ [q\,a\,q] & \to a \\ [q\,a\,q] & \to b[q\,a\,q]\,[q\,a\,q]\,[q\,a\,q] \\ [q\,P\,q] & \to a[q\,b\,q] \\ [q\,P\,q] & \to b[q\,a\,q] \\ [q\,b\,q] & \to a[q\,P\,q]\,[q\,b\,q] \\ [q\,b\,q] & \to a[q\,b\,q]\,[q\,P\,q] \\ [q\,b\,q] & \to b \end{array}$$

Aunque no se pedía más, se pueden hacer las siguientes transformaciones que muestran más claramente el mecanismo de generación de la gramática: Llamando Z al símbolo $[q\,Z\,q]$, que ahora será el inicial, P a $[q\,P\,q]$, A a $[q\,a\,q]$ y B a $[q\,b\,q]$, la gramática es

$$\begin{array}{rl} Z & \to aPZ \mid bAAZ \mid \epsilon \\ A & \to a \mid bAAA \\ P & \to aB \mid bA \\ B & \to b \mid aPB \mid aBP \end{array}$$

que es equivalente a

$$\begin{array}{rl} Z & \to aaBZ \mid abAZ \mid bAAZ \mid \epsilon \\ A & \to a \mid bAAA \\ B & \to b \mid aaBB \mid abAB \mid aBaB \mid aBbA \end{array}$$

3 **(1'5 p.)** Considere los lenguajes siguientes:

- $L_0 = \{w \in (a|b)^* \ / \ |w| \text{ es múltiplo de } 3 \ \}$
- $L_1 = \{w \in (a|b)^* \ / \ |w|_a = 2|w|_b\}$
- $L_2 = \{a^n w \ / \ n \geq 0\,,\ w \in (a|b)^*\,,\ |w|_a = |w|_b = n\}$
- $L_3 = \{aab, aba\}$

(a) Demuestre que $L_3 \subset L_2 \subset L_1 \subset L_0$

La cadena aab es $a^1 w$, siendo $w = ab$ y claramente $|w|_a = |w|_b = 1$, luego está en $L2$. De forma análoga, $aba = a^1 v$, siendo $v = ba$ así que también cumple la condición de las cadenas de L_2 y por lo tanto $L_3 \subset L_2$

Si una cadena está en L_2 es de la forma $x = a^n w$ cumpliendo que $|w|_a = |w|_b = n$. Por lo tanto, se tiene que $|x|_a = n + |w|_a = 2n$ y $|x|_b = |w|_b = n$ y por lo tanto $|x|_a = 2|x|_b$, así que $x \in L_3$. Es decir, $L_2 \subset L_1$

Si $w \in L_1$, se tendrá que $|w| = |w|_a + |w|_b = 2|w|_b + |w|_b = 3|w|_b$, y por lo tanto múltiplo de 3, de modo que $L_1 \subset L_0$

(b) Demuestre que L_0 y L_3 son regulares

L_0 es regular porque es finito (su expresión regular es $aba \,|\, aab$)

L_3 es regular porque se puede expresar mediante la expresión $(\,(a|b)\,(a|b)\,(a|b)\,)^*$

(c) Demuestre que L_1 es (estrictamente) independiente de contexto

L_1 es independiente de contexto porque se puede generar, por ejemplo, mediante la gramática

$$S \to aSaSbS \mid aSbSaS \mid bSaSaS \mid \epsilon$$

lo que puede razonarse viendo que cada aplicación de una de las 3 primeras reglas introduce exactamente dos aes y una b, además de otras posibles subcadenas de la misma forma en cualquier lugar intermedio, y

que cualquier cadena con el doble de aes que de bes deberá empezar por compensar cada b con dos aes en alguna parte.

Y es estrictamente independiente de contexto, porque su intersección con $a^+b^+a^+$ es el lenguaje

$\{a^nb^na^n \ / \ n > 0\}$

que se sabe que no es regular porque no cumple el lema de bombeo de los regulares. (La cadena $a^Nb^Na^N$ no admite ninguna subcadena bombeable entre los N primeros caracteres dentro del lenguaje, por muy grande que sea N)

(d) Demuestre que L_2 no es independiente de contexto

El problema de L_2 es que debe conservar de algún modo el número incial de aes para comprobar que coincide con el número de bes, y ademaás conservarlo de nuevo para comprobar que vuelve a haber el mismo número de aes después de las primeras. Si se almacenan las aes iniciales (o parte de ellas) para comparar con el número de bes, al sacarlas de la pila, ya no se pueden conservar para colver a "contar" el resto de las aes.

Para demostrar que efectivamente L_2 no es independiente de contexto, considérese por ejemplo

$L_4 = L_2 \cap a^+baba^+b^+ = \{a^na^rbaba^{n-r}b^{n-2} \ / \ n > r \geq 0\,,\ n \geq 2\}$

Es la intersección de L_2 con un cierto regular. Este lenguaje no cumple el lema de bombeo de los independientes de contexto, luego no es independiente de contexto, y por lo tanto L_2 tampoco.

Para ver que L_4 no cumple el lema de bombeo de los independientes de contexto, considérese, para una hipotética constante N, por ejemplo la cadena de L_4 siguiente

$z = a^{N+3}baba^Nb^N$

Las dos cadenas que deberían existir :

- tendrían que tener, entre ambas, el doble de aes que de bes (para permanecer en L_2)
- tendrían que evitar "tocar" la zona "central" bab (para no salirse de $a^+baba^+b^+$ al repetirlas)
- tendrían que estar a distancia menor o igual que N

Así que deben estar en los trozos finales (a^Nb^N) y ser $y = a^{2s}, v = b^s$. Pero entonces, al repetirlas, se tendrían cadenas de la forma $a^{N+3}baba^{N+2is}b^{N+is}$ que ya no cumplen que hay (al menos) tantas aes al principio como bes en total.

4 **(1 p.)** Justifique las siguientes afirmaciones:

(a) Todo lenguaje finito es recursivo

Todo lenguaje finito es regular, y por lo tanto recursivo. (Un reconocedor finito que lo cat¡racterice ya es una máquina de Turing que lo caracteriza).

(b) Todo lenguaje cuyo complementario sea finito es recursivo

Si el complementario de L es finito, es regular, por lo tanto su complementario (o sea, L) también es regular y por lo tanto recursivo.

Sea L es un lenguaje recursivamente numerable no recursivo y M una máquina reconocedora de L.

(c) L es infinito y su complementario son infinitos

Sólo hay que aplicar los apartados anteriores:

Si L no es recursivo, no puede ser finito, porque por el apartado a) sería recursivo.

Si $\overline{L}$ no es recursivo, no puede ser finito, porque por el apartado b) sería recursivo.

(d) El lenguaje de parada de M es infinito

Si el lenguaje de parada fuera finito, podría caracterizarse con una máquina de Turing (un Reconocedor finito, de hecho), y su complementario también. Así que, dada una cadena, se podría (mediante un máquina de Turing) determinar si M se va a parar o no ante ella.

Se podría entonces construir un algoritmo (o máquina de Turing) de la siguiente manera: Se lee la cadena de entrada. Si M no va a parar con x, se escribe "NO", y en caso contrario, se aplica M para determinar si dicha cadena perteneco o no a L. Este mecanismo garantarizaría que L es recursivo, y se sabe que no lo es. Por lo tanto, el lenguaje de parada no puede ser finito.

(e) $\{x \in \Sigma^* \ / \ M$ no se para ante $x\}$ es infinito

Si este lenguaje fuera finito, su complementario sería recursivo, es decir, el lenguaje de parada sería recursivo, e igualmente podríamos construir el algoritmo del apartado anterior que probaría que L es recursivo, lo que es absurdo.

5 **(1'5 p.)** Elimine la recursión a la izquierda, factorice y calcule la TASP del resultado, especificando los conjuntos Primeros y Siguientes necesarios para este cálculo. ¿Puede hacerse el análisis deseado?.

$$G: \begin{cases} L \rightarrow L\,c\,V \mid V \mid M\,D \\ V \rightarrow i \mid i\,a\,R\,b\,M \\ M \rightarrow a\,R\,b\,M \mid \epsilon \end{cases} \qquad \begin{cases} R \rightarrow n\,P \\ P \rightarrow c\,R \mid \epsilon \\ D \rightarrow c\,D\,c\,M \mid \epsilon \end{cases}$$

Justifique si es o no posible que esta gramática genere alguna cadena

- que contenga la subcadena ci
- que contenga la subcadena cc
- que contenga la subcadena nb
- que contenga la subcadena ab

Gramática:

$L \rightarrow VL' \mid MDL'$
$L' \rightarrow cVL' \mid \epsilon$
$V \rightarrow iM$
$M \rightarrow aRbM \mid \epsilon$
$R \rightarrow nP$
$P \rightarrow cR \mid \epsilon$
$D \rightarrow cDcM \mid \epsilon$

Primeros	L	L'	V	M	R	P	D	
	i	c	i	a	n	c	c	
	a	ϵ		ϵ		ϵ	ϵ	
	c							
	ϵ							

Siguientes	L	L'	V	M	R	P	D	
	\$	\$	c	c	b	b	c	
			\$	\$			\$	

TASP	a	b	c	i	n	\$
L	$L \rightarrow MDL'$		$L \rightarrow MDL'$	$L \rightarrow VL'$		$L \rightarrow MDL'$
L'			$L' \rightarrow cVL'$			$L' \rightarrow \epsilon$
V				$V \rightarrow iV$		
M	$M \rightarrow aRbM$		$M \rightarrow \epsilon$			$M \rightarrow \epsilon$
R					$R \rightarrow nP$	
P		$P \rightarrow \epsilon$	$P \rightarrow cR$			
D			$D \rightarrow cDcM$ / $D \rightarrow \epsilon$			$D \rightarrow \epsilon$

(NOTA: es posible también hacer esta tabla haciendo la factorización por la izquierda del siguiente modo: $V \rightarrow iV'$ siendo $V' \rightarrow \epsilon \mid aRbM$, en cuyo caso hay un símbolo más (V') idéntico a M en cuanto a primeros, siguientes y tabla).

Los tres primeros casos son perfectamente posibles. Se justifica teniendo en cuenta que todas las reglas son útiles y que

- aparece en la gramática original el consecuente LcV, y claramente i está en primeros de v
- aparece en la gramática original el consecuente $cDcM$, y claramente D es anulable, y a c le siguen los primeros de DcM, y en particular c
- el consecuente $aRbM$ garantiza que b sigue a R. Como aparece la regla $R \rightarrow nP$, y P es anulable, lo que sigue a R sigue también a n, en particular b

o, lo que es más fácil, construyendo derivaciones como las siguientes:

$L \Rightarrow LcV \Rightarrow VcV \Rightarrow icV \Rightarrow ici$

$L \Rightarrow MD \Rightarrow D \Rightarrow cDcM \Rightarrow ccM \Rightarrow cc$

$L \Rightarrow V \Rightarrow iaRbM \Rightarrow ianPbM \Rightarrow ianbM \Rightarrow ianb$

(Da lo mismo trabajar con la gramática original o la construída, porque son equivalentes)

SIn embargo el tercer caso no es posible, porque de la misma manera que se calculan los siguientes de los auxiliares, se pueden calcular los siguientes de los terminales, y en este caso, $siguientes(a) = primeros(RbM) - \epsilon = \{n\}$

6 **(0'5 p.)** Para la gramática

$$S \rightarrow \mathbf{i}\, S \mid \mathbf{i}\, S\, \mathbf{e}\, S \mid \mathbf{n}$$

Pivotes de **i i n e n**:

Las únicas derivaciones más a la derecha posibles para $iinen$ son

$S \Rightarrow iS \Rightarrow iiSeS \Rightarrow iiSen \Rightarrow iinen$

y

$S \Rightarrow iSes \Rightarrow iSen \Rightarrow iiSen \Rightarrow iinen$

En ambos casos el pivote es la primera n, con la regla $S \rightarrow n$

Pivotes de **i i** S **e n**:

Las dos únicas posibles derivaciones más a la derecha para $iiSen$ son (las mismas del caso anterior hasta el anteúltimo paso):

$S \Rightarrow iS \Rightarrow iiSeS \Rightarrow iiSen$

y

$S \Rightarrow iSes \Rightarrow iSen \Rightarrow iiSen$

lo que justifica dos pivotes para esta cadena:

- n con la regla $S \rightarrow n$
- iS con la regla $S \rightarrow iS$

¿Escribiría algún mensaje Yacc al tratar esta gramática? Si es así cuál y a qué se debería:

Detectaría un conflicto desplazamiento-reducción (al menos) y escribiría un mensaje informando de ello. Se debe a que la gramática no es $LALR(1)$, porque es ambigua, como se ve por las dos derivaciones más a la derecha que acaban de mostrarse.

7 **(1 p.)**

1. ¿Qué característica debe cumplir una máquina de Mealy para ser considerada de Moore?
   Las salidas asociadas a todos los arcos con el mismo destino coinciden.
2. ¿Qué relación hay entre lenguajes reconocibles por vaciado de pila por algún autómata con pila determinista y lenguajes independientes de contexto?
   Los primeros forman un subconjunto propio de los segundos (todo lenguaje reconocible por vaciado de pila por algún A.P. Determinista es independiente de contexto, pero no todo lenguaje independiente de contexto es reocnocible por vaciado de pila por un AP Determinista)
3. "Para cualquier lenguaje existirá una gramática no restringida, más o menos difícil de encontrar, que lo genere". Razone si la afirmación es cierta o falsa.
   Falso. Existen lenguajes no recursivamente numerables, es decir, que no son de tipo 0, o sea, que no pueden describirse con ninguna gramática.
4. ¿En qué fases se descompone habitualmente el proceso de compilación?
   Análisis léxico, análisis sintáctico, análisis semántico (fases de análisis), generación de código intermedio, optimización de código y generación de código (fases de síntesis).

8 **(1 p.)** ¿Qué es un autómata con pila "extendido" (Extended Push-Down Automaton)?

Los elementos son los mismos que los autómatas con pila estudiados en clase $(\Sigma, \Gamma, Q, q_1, Z_0, f, F)$, salvo porque la función de transición f está definida de $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma^*$ en $\mathcal{P}(Q \times \Gamma)$

Es decir: en cada movimiento, en lugar de salir un elemento de la pila a cambio de entrar una cadena, sale de la pila una cadena de símbolos y entra uno sólo.