Universidad de Valladolid

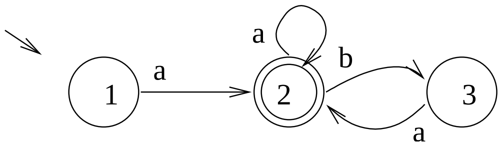
Departamento de Informática

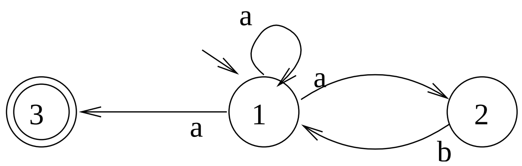
Teoría de autómatas y lenguajes formales . 2º I.T.Informática. Gestión.

Examen extraordinario. 5 de septiembre de 2003.

Soluciones.

$\boxed{1}$ **(2'5 p.)** 1. Para la construcción del RFD para las expresiones consideradas se puede partir de los RF no deterministas siguientes (respectivamente):





de tablas

	a	b
1	2	
(2)	2	3
3	2	

y

	a	b
1	1, 2, 3	
2		1

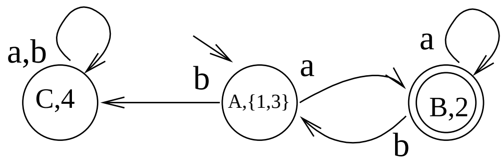
Dan lugar a los RFD de tablas

	a	b
1	2	4
(2)	2	3
3	2	4
4	4	4

y

	a	b	
A	B	C	1
(B)	B	A	1, 2, 3
C	C	C	

El algoritmo de minimización del primero da lugar al segundo:



y por lo tanto las expresiones son equivalentes.

2. *a*) Cierta, dado que

$$a(a|ba)^+ = a(a|ba)^*(a|ba) \quad = \quad \text{(apartado 1)}$$
$$(a|ba)^*a(a|ba) = (a|ba)^*(aa|aba) = (a|ba)^*(a|ab)a = (a|ba)^+a$$

b) Falsa . Por ejemplo baa está representada en la segunda expresión y no en la primera y aba en la primera y no en la segunda.

c) Falsa. Por ejemplo $abba$ está representada en la segunda expresión y no en la primera y a en la primer y no en la segunda.

d) Falsa. Por ejemplo $abaa$ está representada en la primera expresión y no en la segunda

e) Falsa. Por ejemplo $abaa$ está representada en la primera expresión y no en la segunda

3. La tabla siguiente resume el proceso de determinación, suponiendo que los estados de la figura se numeran $2, 1, 3, 4$

	a	b	
A	B	C	1
B	B	B	
C	D	C	1, 2, 3
(D)	B	C	1, 4

y el algoritmo de minimización construye

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

y por lo tanto el autómata obtenido es mínimo.

Se pueden obtener las expresiones regulares siguientes (equivalentes)

- $(b|ba)^*ba$ obtenida "leyendo" el RFN de partida.
- $b(b|ab)^*a$ o $b^+(ab^+)^*a$, obtenidas "leyendo" el RFD obtenido.
- $bb^*a(bb^*a)^*$ o sea $(b^+a)^+$ aplicando el teorema de análisis al RFD obtenido: renombrando $A \to 1$, $B \to 4$, $C \to 2$ y $D \to 3$ se tiene:

$$\begin{aligned}
\alpha^4_{13} &= \alpha^3_{13}|\alpha^3_{14}(\alpha^3_{44})^*\alpha^3_{43} = \alpha^3_{13}\\
\alpha^3_{13} &= \alpha^2_{13}|\alpha^2_{13}(\alpha^2_{33})^*\alpha^2_{33} = \alpha^2_{13}(\alpha^2_{33})^*\\
\alpha^2_{13} &= \alpha^1_{13}|\alpha^1_{12}(\alpha^1_{22})^*\alpha^1_{23}\\
\alpha^1_{13} &= \alpha^0_{13}|\alpha^0_{11}(\alpha^0_{11})^*\alpha^0_{13} = \emptyset|\lambda(\lambda)^*\emptyset = \emptyset\\
\alpha^1_{12} &= \ldots = b\\
\alpha^1_{22} &= \ldots = \lambda|b\\
\alpha^1_{23} &= \ldots = a\\
\alpha^2_{13} &= bb^*a\\
\alpha^2_{33} &= \alpha^1_{33}|\alpha^1_{32}(\alpha^1_{22})^*\alpha^1_{23}\\
\alpha^1_{33} &= \lambda\\
\alpha^1_{32} &= b\\
\alpha^2_{33} &= = \lambda|bb^*a\\
\alpha^4_{13} &= bb^*a(\lambda|bb^*a)^* = bb^*a(bb^*a)^*
\end{aligned}$$

$\boxed{2}$ **(2'5 p.)** 1. La gramática con la que se trabaja es

$$S \to aSbS \mid \lambda$$

(reglas (1) y (2) de la tabla).

El lenguaje generado es el de los paréntesis equilibrados, si consideramos que a es (y b es). Es decir: cadenas que tienen el mismo número de aes que de bes, de forma que además cualquiera de sus prefijos tiene al menos tantas aes como bes.

La gramática no es ambigua puesto que es $LALR(1)$ (dado que Yacc do detecta conflictos).

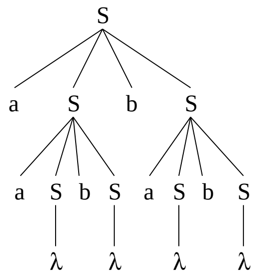
2.

$PILA$		$Entrada$	$Acción$	$Regla$	$Ir-a$
$\$0$	0	$aabbab\$$	$s1$		
$\$0a1$	0 1	$abbab\$$	$s1$		
$\$0a1a1$	0 1 1	$bbab\$$	$r2$	$S \to \lambda$	ir-a[1,S]=3
$\$0a1a1S3$	0 1 1 3	$bbab\$$	$s4$		
$\$0a1a1S3b4$	0 1 1 3 4	$bab\$$	$r2$	$S \to \lambda$	ir-a[4,S]=5
$\$0a1a1S3b4S5$	0 1 $\underline{1\,3\,4\,5}$	$bab\$$	$r1$	$S \to aSbS$	ir-a[1,S]=3
$\$0a1S3$	0 1 3	$bab\$$	$s4$		
$\$0a1S3b4$	0 1 3 4	$ab\$$	$s1$		
$\$0a1S3b4a1$	0 1 3 4 1	$b\$$	$r2$	$S \to \lambda$	ir-a[1,S]=3
$\$0a1S3b4a1S3$	0 1 3 4 1 3	$b\$$	$s4$		
$\$0a1S3b4a1S3b4$	0 1 3 4 1 3 4	$\$$	$r2$	$S \to \lambda$	ir-a[4,S]=5
$\$0a1S3b4a1S3b4S5$	0 1 3 4 $\underline{1\,3\,4\,5}$	$\$$	$r1$	$S \to aSbS$	ir-a[4,S]=5
$\$0a1S3b4S5$	0 $\underline{1\,3\,4\,5}$	$\$$	$r1$	$S \to aSbS$	ir-a[0,S]=2
$\$0S2$	0 2	$\$$	$accept$		

La derivación se obtiene leyendo las acciones en orden inverso, y es la derivación más a la derecha:

$$S \Rightarrow aSbS \Rightarrow aSbaSbS \Rightarrow aSbaSb \Rightarrow aSbab \Rightarrow aaSbSbab \Rightarrow aaSbbab \Rightarrow aabbab$$

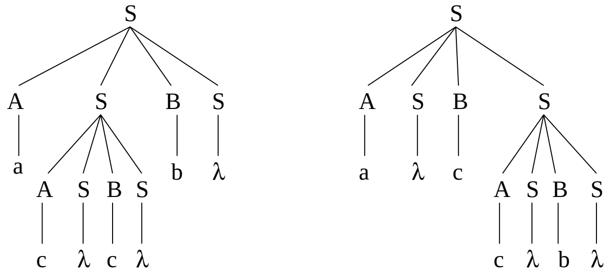
que describe el árbol de derivación



3. Una gramática posible:

$$\begin{array}{lcl} S & \rightarrow & ASBS \mid \lambda \\ A & \rightarrow & a \mid c \\ B & \rightarrow & b \mid c \end{array}$$

Es es ambigua. Se comprueba, por ejemplo, considerando los árboles de derivación para la cadena *accb* siguientes



(correspondientes a las interpretaciones *aabb* y *abab* respectivamente).

$\boxed{3}$ **(1'5 p.)** 1. Si $L \subseteq a^*$ es independiente de contexto, verificará el lema de bombeo correspondiente:

$$\exists N/(z \in L \wedge |z| \geq N) \Rightarrow (\exists x, y, u, v, w \in a^*/|uv| > 0 \wedge \forall i \geq 0, xy^iuv^iw \in L)$$

Por fuerza, tanto u como v estarán constituídos por aes, o sea que $y = a^p$ y $v = a^q$ para ciertos p y q con $0 < p + q \leq N$, y $z = a^m$ con $m \geq N$ de forma que $xy^iuv^iw = a^{m+(p+q)*(i-1)}$. Por lo tanto, para ese mismo N, todo $z \in L$ de longitud mayor o igual que N admite una subcadena a^{p+q} bombeable (elegida en cualquier posición posible de z, por ejemplo en la primera posición). Lo verdaderamente relevante en esta demostración es que hay un único carácter en la constitución de las cadenas.

2. Evidentemente, si $z = a^mb^m$ con $m >= 7$, se pueden bombear simultáneamente una a y una b, porque se mantendrá la igualdad entre el número de aes y bes, estando las aes delante de las bes, y, incluso cuando se eliminen ($i = 0$), se obtendrán como poco 6 aes y 6 bes, permaneciendo en L. Por lo tanto una constante posible para N es 14.
Sin embargo no es la mínima, ya que para a^6b^6 y a^4b^4 se pueden bombear a^2 y b^2 (el número de aes y bes será siempre par, luego nunca 5); para a^3b^3 se pueden bombear a^3 y b^3 (el número de aes y bes será siempre múltiplo de 3, luego nunca 5); y para a^2b^2 se pueden bombear también a^2 y b^2.
Ahora bien, ab no admite ningún par de cadenas bombeables (bombear a y b pasaría por a^5b^5).
Por lo tanto, la constante mínima es 3 (toda cadena de L de longitud mayor o igual que 3 es a^2b^2 o a^3b^3 o a^4b^4 o a^6b^6 o $a^{7+r}b^{7+r}$ para algún $r \geq 0$ y en todos los casos se encuentra un par de subcadenas bombeables[1] en la forma que exige el lema de bombeo).

[1] No tienen por qué ser siempre las mismas.

$\boxed{4}$ **(1'5 p.)** 1. b) es recursivamente numerable (el programa P es precisamente un reconocedor de L)

e) puede que sea recursivo y puede que no (puede existir *otro* programa que lo reconozca parándose siempre, con un "sí" o un "no", o puede que no exista)

2. a) puede que sea recursivamente numerable y puede que no (puesto que es el complementario de L, que es recursivamente numerable)

e) puede que sea recursivo y puede que no (si L, además de recursivamente numerable, fuese recursivo, su complementario también lo sería. Si, por el contrario, L no fuera recursivo, su complementario no sería ni siquiera recursivamente numerable, luego tampoco recursivo).

3. tipo 1 = generables por gramáticas no contractivas $\subset$ recursivos $\subset$ tipo 0

Y las contenciones son estrictas.

$\boxed{5}$ **(0'75 p.)** 1. `grep -n "^$"` *fichero*

2. `sed '/^$/d'` *fichero*

3. `awk '!/^$/{}'` *fichero*

$\boxed{6}$ **(1'25 p.)** Fuente Lex:

```
%{
#include "y.tab.h"
%}
%%
[0-9]+                  { yylval=atof(yytext);
                          return (nume);       }
[0-9]+"."[0-9]+         { yylval=atof(yytext);
                          return (numr); }
[tT][oO][tT][aA][lL]    { return (total); }
[a-zA-Z]+               { return (producto); }
"\n"                    { return (salto); }
.                       { printf ("ERROR LEXICO"); }
```

Fuente Yacc:

```
%{
#define YYSTYPE real
%}
%token nume numr total producto salto
%%
A    : S total  { printf ("Total : %d, IVA %d ", $1, $1*0.07); }
     ;
S    : producto nume numr salto S        {$$=$2*$3+$5;}
     | producto nume numr salto          {$$=$2+$3;}
     | producto nume nume salto S        {$$=$2*$3+$5;}
     | producto nume nume salto          {$$=$2+$3;}
     ;
%%
main () {
    yyparse();
}
```