Universidad de Valladolid
Departamento de Informática

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

Solución al examen extraordinario de 5 de septiembre de 2005.

1

	k=0	k=1	k=2	k=3
α_{11}	$a \mid \epsilon$			
α_{12}	b	a^*b		
α_{13}	$\emptyset$	$\emptyset$	a^*bb^*a	$(a^*bb^*a)^+$
α_{21}	$\emptyset$	$\emptyset$	$\emptyset$	
α_{22}	$b \mid \epsilon$	$b \mid \epsilon$		
α_{23}	a	a		
α_{31}	a			
α_{32}	b	a^*b		
α_{33}	ϵ	ϵ	$\epsilon \mid a^*bb^*a$	

El lenguaje reconocido viene dado por la expresión α_{13}^3

2 RFD equivalente:

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

Minimización:

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

(el autómata es mínimo).

3 Máquina de Moore equivalente:

	a	b
$1/1$	2^0	3^0
$2^0/0$	2^1	1
$2^1/1$	2^1	1
$3^0/0$	1	3^1
$3^1/1$	1	3^1

4 **El lenguaje L_s no es regular**: si lo fuese, también lo sería $L_s \cap 01^*01^*0 = \{01^n01^n0 \,/\, n \geq 0\}$, que no lo es porque no cumple el lema de bombeo de los regulares. Efectivamente, si N fuera la constante del lema de bombeo, la cadena $z = 01^N01^N0$ (que está en el lenguaje y tiene longitud $> N$) debería admitir una subcadena bombeable no nula, v, entre los primeros N caracteres. v no puede contener un 0, porque al eliminarla el resultado contendría sólo 2 ceros. Entonces $v = 1^p$ con $p > 0$, pero en este caso, al eliminarla quedaría $01^{N-p}01^N0$ que tampoco está en el lenguaje.

L_s **es independiente de contexto:** la gramática siguiente lo genera:

$$G = \begin{cases} S & \rightarrow & P0 \\ P & \rightarrow & 1P1 \mid 0Q \\ Q & \rightarrow & 1Q1 \mid 0 \end{cases}$$

porque claramente

- Q genera todas las cadenas de la forma 1^m01^m con $m \geq 0$, y sólo éstas,
- P genera todas las formas 1^n0Q1^n con $n \geq 0$ y sólo éstas son generables sin la reaparición de P y sin la desaparición de Q;
- por lo tanto P genera exactamente las cadenas de terminales $1^n01^m01^m1^n$, $m, n \geq 0$
- y de ahí que $L(G)$ sea el lenguaje pedido.

Por lo tanto (como es generable por una gramática independiente de contexto) el lenguaje es independiente de contexto.

Puede construirse un **autómata con pila** a partir de una forma normal de Greibach de G mediante el algoritmo conocido. Resulta en este caso que el autómata es **determinista** pero reconoce por vaciado de pila; puede modificarse para que reconozca por estado final:

Forma normal de Greibach para G:

$$G' = \begin{cases} S & \rightarrow & 1PUZ|0QZ \\ P & \rightarrow & 1PU \mid 0Q \\ Q & \rightarrow & 1QU \mid 0 \\ Z & \rightarrow & 0 \\ U & \rightarrow & 1 \end{cases}$$

Autómata con pila que reconoce por estado final:

$\Gamma = \{S, P, Q, Z, U, A\}$, símbolo inicial de la pila: A; $Q = \{p_0, q, r\}$, estado inicial: p_0, $F = \{r\}$, función de transición:

p_0	0	1	ϵ
A			(q, SA)

q	0	1	ϵ
S	(q, QZ)	(q, PUZ)	
P	(q, Q)	(q, PU)	
Q	(q, ϵ)	(q, QU)	
Z	(q, ϵ)		
U		(q, ϵ)	
A			(r, A)

De forma alternativa, se puede construir un **autómata a pila determinista** observando la forma de las cadenas de L_s: se apilan los unos de los dos primeros fragmentos, y se desapilan con el tercero:

$\Gamma = \{A, 1\}$, símbolo inicial de la pila: A; $Q = \{p, q, r, s\}$, estado inicial: p, $F = \{s\}$, función de transición:

p	0	1
A	(q, A)	$(p, 1A)$
1	$(q, 1)$	$(p, 11)$

apilando unos
cambiar al primer 0

q	0	1
A	(r, A)	$(q, 1A)$
1	$(r, 1)$	$(q, 11)$

apilando unos
cambiar al segundo 0

r	0	1
A	(s, A)	
1		(r, ϵ)

desapilando unos
terminar al tercer 0

Nota: $000 \in L_s$

Como hay un autómata a pila cuyo lenguaje reconocido es L_s, el lenguaje **es independiente de contexto** (de esta manera no es necesario plantear la gramática).

El lenguaje complementario de uno independiente de contexto no tiene por qué serlo, pero en este caso **lo es**, porque la existencia de este autómata a pila *determinista* permite construir otro para el complementario, sencillamente "negándolo". En el último autómata propuesto bastaría añadir otro estado, t y rellenar los dos huecos de las tablas con (t, A), para tener un autómata a pila determinista completo. Las cadenas de ceros y unos llevarán al autómata a pila de p a algún estado. Si éste es s, la cadena está en L_s; si no, la cadena está en su complementario. Por lo tanto, si hacemos que el conjunto de estados finales sea $\{p, q, r, t\}$, el lenguaje reconocido será el complementario de L_s.

L_p **no es independiente de contexto** porque no cumple el lema de bombeo de los independientes de contexto (versión fuerte):

Supongamos que N fuera una constante para dicho lema de bombeo. Tomemos $z = 1^N 01^N 01^{N^2} 0$. La necesaria descomposición en $xyuvw$ para el lema de bombeo obliga a que tanto y como v tengan que estar compuestas por unos, porque el número de ceros es siempre 3 y no puede cambiarse. Por lo tanto, $y = 1^p$ y $v = 1^q$, con $p + q = |yv| > 0$. Además $|yuv|$ debe ser *menor o igual* que N.

Como $|yuv| \leq N$, la elección de y y v tiene las siguientes posibilidades:

- estar ambas en el mismo grupo de unos.
  - en el primer grupo: no puede ser, porque entonces $xy^0uv^0w = 1^{N-(p+q)}01^N01^{N^2}0$ y $(N - (p+q))N = N^2 - (p+q)N \neq N^2$
  - en el segundo grupo: no puede ser, porque entonces $xy^0uv^0w = 1^N01^{N-(p+q)}01^{N^2}0$ y también $N(N - (p+q)) = N^2 - (p+q)N \neq N^2$
  - en el tercer grupo: tampoco puede ser, porque entonces $xy^0uv^0w = 1^N01^N01^{N^2-(p+q)}0$ y $NN \neq N^2 - (p+q)$
- estar y en el primer grupo y v en el segundo. Entonces $xy^2uv^2w = 1^{N+p}01^{N+q}01^{N^2}0$ y $(N+p)(N+q) = N^2 + (p+q)N + pq > N^2$

- estar y en el segundo y v en el tercero. Entonces
  $xy^0uv^0w = 1^N01^{N-p}01^{N^2-q}0$
  debería estar en L_p y por lo tanto debería darse que
  $N(N-p) = N^2 - q$ y por lo tanto $q = pN$, en cuyo caso $p + q = p + pN = p(N+1)$. Pero entonces $|yuv| > p + q = p(N+1) > N$ en contra de lo supuesto.

NOTA: L_p cumple la versión débil: para $1^n01^m01^{nm}0$ se podría bombear, por ejemplo, el último 1 del segundo grupo y los n primeros del último dando $1^n01^{n-1}1^i01^{ni}1^{nm-n}0$, que están en L_p.

5 Sea L un lenguaje recursivamente numerable y a un símbolo del alfabeto.

1. Si L es recursivamente numerable, aL **es** recursivamente numerable: basta para afirmarlo considerar la máquina de Turing que existe y genera L y, a la salida, escribir una a delante de cada cadena que imprima, para tener una máquina de Turing que genera aL. También puede razonarse construyendo una máquina reconocedora de aL a base de

   ```
   leer y
   si y comienza por a, (y=ax), entonces
        aplicar la máquina reconocedora de L a x
             si ésta responde "si" entonces responder "si"
             si no, si responde "no" entonces responder "no"
             fin si
   si no
        responder "no"
   fin si
   ```

   Es posible que este algoritmo (máquina) no se pare (porque no lo haga la de L), pero lo hará con toda seguridad respondiendo afirmativamente con las cadenas de aL.
2. Se considera el conjunto de cadenas de L que comienzan por a, y se les quita a todas ellas esa primera a. Por ejemplo,

   $$\begin{array}{rcl} L & = & \{\ a,\ ab,\ baa,\ aaa,\ ababb,\ aabb,\ \ldots\} \\ L' & = & \{\ \epsilon,\ b,\ \ aa,\ \ babb,\ \ abb,\ \ldots\} \end{array}$$

   Es decir: $y \in L' \Leftrightarrow ay \in L$. Si L es recursivamente numerable, el conjunto de cadenas resultante **es** recursivamente numerable: se puede construir una máquina de Turing que tome la generadora de L y, cada vez que ésta escriba una cadena comprobar el primer símbolo; si no es una a, no se escribe; si es una a, se omite esta a y se escribe el resto de la cadena. Otra forma de verlo:

   ```
   leer y
   aplicar la máquina reconocedora de L a ay
        si ésta responde "si" entonces responder "si"
        si no, si responde "no" entonces responder "no"
        fin si
   fin si
   ```

   Nuevamente es posible que este algoritmo (máquina) no se pare, pero lo hará con toda seguridad respondiendo afirmativamente con las cadenas del lenguaje considerado.
3. Si se hace la misma operación anterior con un lenguaje recursivo, el resultado **es** un lenguaje recursivo: el algoritmo anterior siempre se parará, puesto que lo hace la máquina reconocedora de L.

6

1. El método CYK se suele clasificar entre los que no son ninguna de las dos cosas, aunque su forma de detectar si la cadena de entrada está o no en el lenguaje en cuestión es ascendente.
2. El que YACC encuentre un conflicto al analizar una gramática **no** significa que sea ambigua. Hay gramáticas no ambiguas que provocan conflictos en YACC. Lo que significa un conflicto en YACC es que la gramática no es LALR(1), es decir, que el conocimiento del prefijo de la cadena ya leída y un símbolo más no permiten que el algoritmo que Yacc aplica pueda decidir en ese momento, en todos los casos, cuál es el pivote.
3. Un pivote de una forma sentencial derecha (α) es
   - una subcadena de la forma sentencial (como tal subcadena, en una posición concreta de dicha forma sentencial) (β tal que $\alpha = \alpha_1\beta\alpha_2$)
   - y una regla de la gramática cuyo consecuente es esa subcadena ($r = A \to \beta$)

   tales que, si se sustituye en la forma sentencial (α) la subcadena (β) por el antecedente de la regla (A), el resultado ($\alpha_1 A \alpha_2$) es la forma sentencial anterior de una derivación más a la derecha con resultado la forma sentencial de partida ($S \Rightarrow^*_{md} \alpha_1 A \alpha_2 \Rightarrow^*_{md} \alpha_1\beta\alpha_2 = \alpha$).
   Una forma sentencial derecha puede tener varios pivotes. Si la gramática no es ambigua, sólo tendrá 1. Si la gramática es ambigua, puede tener varios.

4. Dado un lenguaje independiente de contexto L:
   a) No siempre existe un autómata a pila determinista cuyo lenguaje reconocido es L. Lo que siempre existe es un autómata a pila cuyo lenguaje reconocido es L, pero no necesariamente determinista.
   b) Es posible que exista un reconocedor finito cuyo lenguaje reconocido sea L: exactamente en el caso particular de que además sea regular.
5. Dada una máquina de Turing y uno de sus estados q, el conjunto de cadenas de entrada que provocan que la máquina se pare en q, se puede garantizar que es al menos recursivamente numerable, porque podemos considerar a este estado como el único final (de aceptación), y entonces estamos justamente en la definición de lenguaje recursivamente numerable. (Si además la máquina se para ante cualquier entrada, el lenguaje también será recursivo).

7

Primeros	S	P	F	Q	A	B	D
	e	c	f	e	a	b	c
	c			a	ϵ	ϵ	b
				b			
				ϵ			

Siguientes	S	P	F	Q	A	B	D
	\$	e	\$	f	b	f	e
	f	a	f	d	f	d	a
		b		\$	d	\$	b
		\$			\$		f
		f					

TASP	c	e	d	f	a	b	\$
S	$S \to PQ$	$S \to eDQF$					
P	$P \to cSf$						
F				$F \to f$			
Q		$Q \to eQd$	$Q \to AB$	$Q \to AB$	$Q \to AB$	$Q \to AB$	$Q \to AB$
A			$A \to \epsilon$	$A \to \epsilon$	$A \to a$	$A \to \epsilon$	$A \to \epsilon$
B			$B \to \epsilon$	$B \to \epsilon$		$B \to b$	$B \to \epsilon$
D	$D \to cD$					$D \to b$	

8 **Fuente para Lex**:

```
%{
#include "y.tab.h"
%}
 int cont=0;
%%
[ \t]+  ;
[0-9]+       { yylval.numero = atoi(yytext); return NUM; }
FIN            return FIN;
[a-zA-Z]+    { yylval.caract=yytext[cont]; return CARACTER; }
"\n"         { cont++;                     return '\n' ; }
.            { printf ("Error léxico\n"); }
```

Fuente para Yacc (Nota: se ha considerado que siempre hay al menos una línea):

```
%{
#include<stdio.h>
yyerror (char*s){ fprintf (stderr, "%s\n", s); }
%}

%union{
      int numero;
      char caract;
      struct {
            char cad [100];
            int hora;
            int min;
      } registro;
}

%token <numero>NUM
%token <caract>CARACTER
%token FIN
%type <registro> linea, E,S
%type <numero>NUM
%%
S      :  E FIN '\n'       { if ($1.hora<0 || $1.hora>23 || $1.min<0 ||$1.min>59)
                                 printf ("Error semántico\n");
                             else
                               printf ("%s %d:%d\n", $1.cad, $1.hora, $1.min);
                           }
      ;

E      : E linea           { $$.hora = $1.hora+$2.hora;
                             $$.min = $1.min+$2.min;
                             strcpy ($$.cad, strcat ($1.cad, $2.cad));
                           }
      | linea              { $$.hora = $1.hora;
                             $$.min = $1.min;
                             strcpy ($$.cad, $1.cad);
                           }
      ;
linea : NUM CARACTER NUM '\n'
                           { $$.hora = $1;
                             $$.min = $3;
                             $$.cad[0]=$2; $$.cad[1]='\0';
                           }
      ;
%%
main(){
      /* yylval.registro.cad[0]='\0';  */
      yyparse();
}
```