Prácticas Modelos de Computación

Javier Gómez Luzón A3

Índice

Práctica 1
Lenguajes y gramáticas

1. Describir el lenguaje generado por las siguientes gramáticas en {0, 1}*.

a) **$S \rightarrow 0S_1 1$** **$S_1 \rightarrow 0S_1 \mid 1S_1 \mid \varepsilon$**

El lenguaje comienza por un 0 y termina por un 1. L={0 u 1; u E {0,1}*}.

b) **$S \rightarrow S_1 101 S_1$** **$S_1 \rightarrow 0S_1 \mid 1S_1 \mid \varepsilon$**

Este lenguaje contiene la subcadena "101". L={u 101 v: u, v E {0,1}*}.

c) **$S \rightarrow 0S1 \mid S_1$** **$S_1 \rightarrow 1S_1 0 \mid 1S_2 0$** **$S_2 \rightarrow 0S_2 1 \mid \varepsilon$**

Este lenguaje se forma por 0 y 1. L={$0^i\ 1^j\ 0^x\ 1^x\ 1^j\ 0^i$ | i, j E {1,2,3,...} xE N }.

2. Encontrar gramáticas de tipo 2 para los siguientes lenguajes sobre el alfabeto {0, 1}. En cada caso determinar si los lenguajes generados son de tipo 3, estudiando si existe una gramática de tipo 3 que los genera.
a) Palabras que comienzan con la subcadena "10" y acaban en "001".

La gramática de tipo 2:
$S \rightarrow 10X01$ $X \rightarrow 1Y$ $X \rightarrow \varepsilon$ $X \rightarrow 0X$ $X \rightarrow 0$
$Y \rightarrow 0X$ $Y \rightarrow 1Y$ $Y \rightarrow 0$
La gramática de tipo 3:
$S \rightarrow ACX$ $A \rightarrow 10$ $X \rightarrow 1X \mid 0X \mid B \mid \varepsilon$ $B \rightarrow 001$ $C \rightarrow 1D \mid \varepsilon$

b) Palabras que tienen 2 o 3 "0".

La gramática de tipo 2:
$S \rightarrow A0A0ABA$ $A \rightarrow 1A \mid \varepsilon$ $B \rightarrow 0 \mid \varepsilon$
La gramática de tipo 3:
$S \rightarrow A$ $A \rightarrow 1A \mid 0B$ $B \rightarrow 1B \mid 0C$ $C \rightarrow 1C \mid 0D \mid \varepsilon$ $D \rightarrow 1D \mid \varepsilon$

c) Palabras que no contienen la subcadena "011".

La gramática de tipo 2:
$S \rightarrow 1C0$ $A \rightarrow 0B \mid 0$ $B \rightarrow 1A \mid \varepsilon$ $C \rightarrow 1C0 \mid A \mid \varepsilon$ $D \rightarrow 0D \mid B \mid \varepsilon$
La gramática de tipo 3:
$S \rightarrow C$ $A \rightarrow 0B \mid D$ $B \rightarrow 1A \mid \varepsilon$ $C \rightarrow 1C \mid A \mid \varepsilon$ $D \rightarrow 0D \mid B \mid \varepsilon$

3. Como empleado de la empresa de desarrollo de videojuegos "MoreThanDungeons", se le ha pedido diseñar una gramática que represente los niveles de un juego de exploración de mazmorras y las salas de éstas, con una serie de restricciones.
En cada nivel:

- **Existen salas grandes (g) y pequeñas (p) que deberán ser limpiadas de monstruos para avanzar. (Los niveles más sencillos tienen al menos una sala grande)**
- **Hay al menos una sala de tendero (t), donde recuperar fuerzas y comprar objetos.**
- **Habrá una sola sala secreta (x), siempre le precede una sala grande. Es decir, siempre habrá una "g" delante de "x".**
- **Cada nivel de la mazmorra debe acabar con una sala final de jefe (j).**

Por ejemplo, la cadena terminal "ppgxtj" representa el nivel en el que el jugador debe de pasar por dos habitaciones pequeñas "pp", seguidas de una grande "g". En esta, podrá encontrar la sala secreta "x". A continuación, podrá recuperar fuerzas en la tienda "t". Para finalmente, enfrentarse al jefe final "j" del nivel.
Elabore una gramática que genere estos niveles con sus restricciones. Cada palabra del lenguaje es UN SOLO NIVEL. ¿A qué tipo de la jerarquía de Chomsky pertenece la gramática que ha diseñado?

S→ gGX | pPX
G→ gG | P | t | ε
P→ pP | G | t | ε
X→ gxGPj

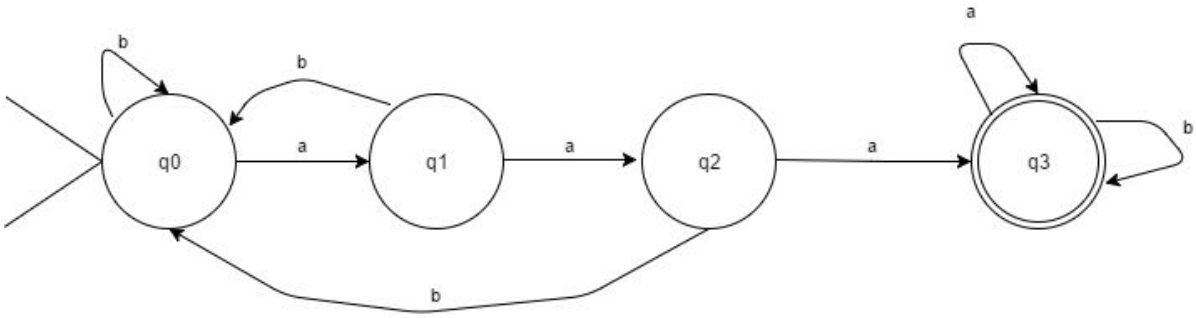
¿Podría diseñar una gramática de tipo 3 para dicho problema?

La gramática que he diseñado inicialmente ya era de tipo 3.

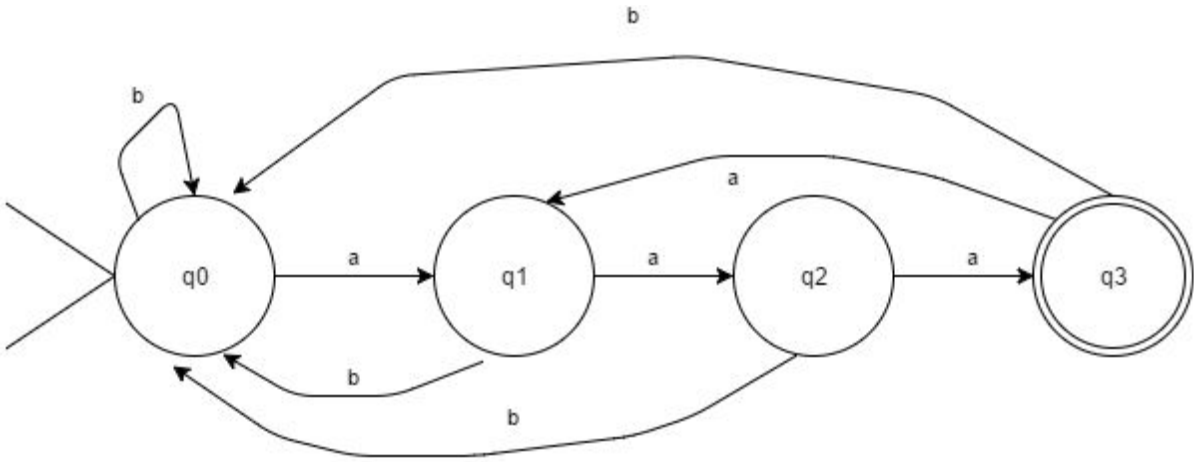
Práctica 2
AFD/AFND

1. Construir un AFD que acepte cada uno de los siguientes lenguajes con alfabeto {a,b}:

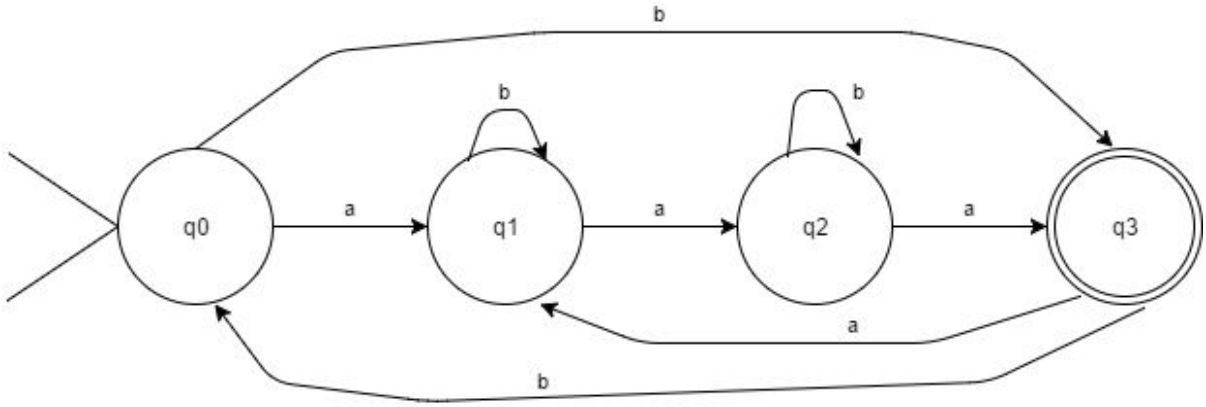
a. El lenguaje de las palabras que contienen la subcadena aaa.



b. El lenguaje de las palabras que empiezan o terminan (o ambas cosas) en aaa.

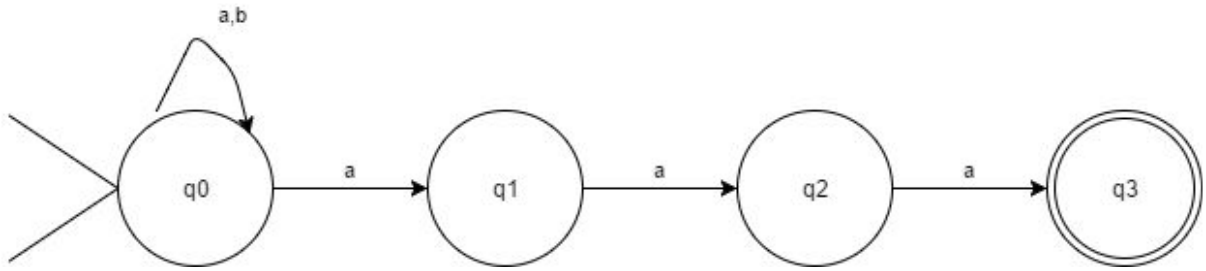


c. El lenguaje formado por las cadenas donde el número de a's es divisible por 3.

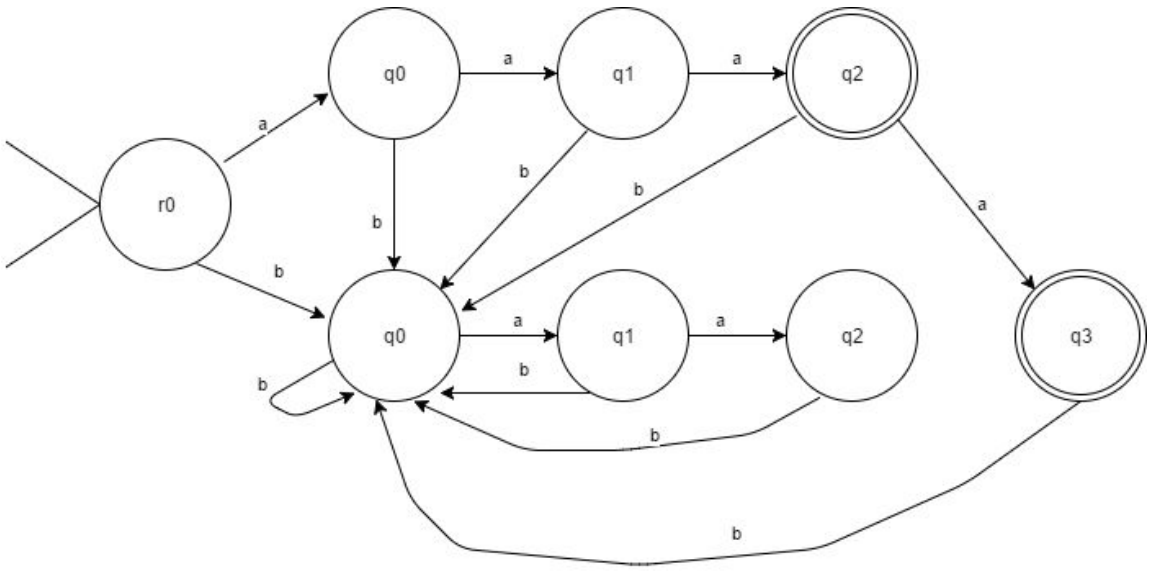


2. Construir un AFND que acepte cada uno de los siguientes lenguajes con alfabeto {a,b}:

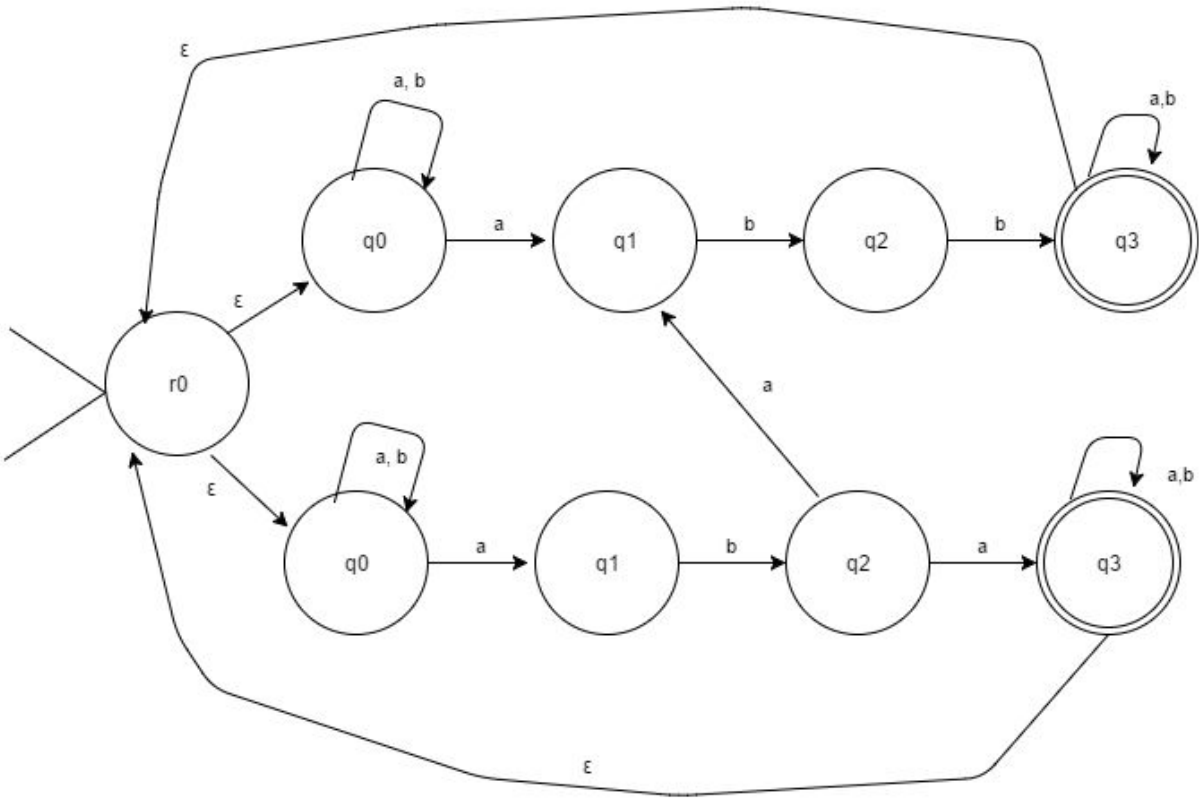
a. El lenguaje de las palabras que terminan en aaa.



b. El lenguaje de las palabras que empiezan o terminan (o ambas cosas) en aaa.



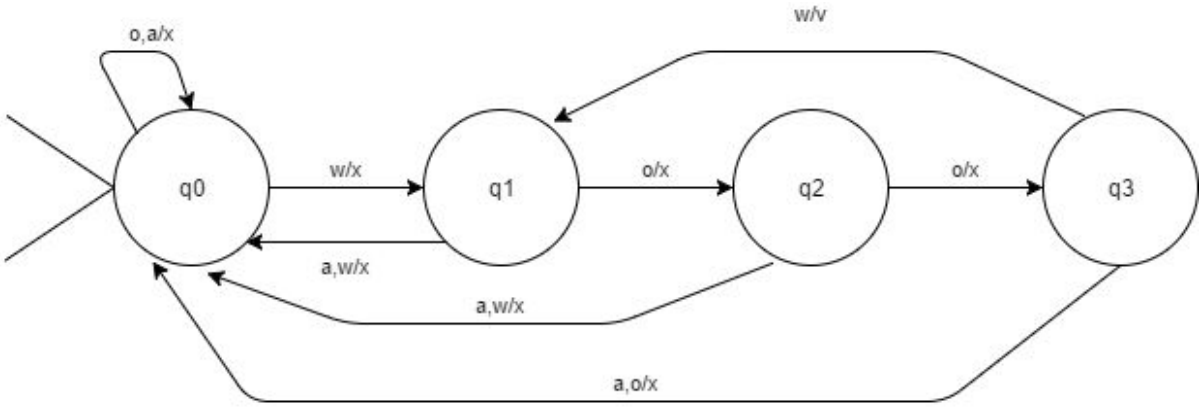
c. El lenguaje de las palabras que contengan, simultáneamente, las subcadenas aba y abb. Este AFND también acepta cadenas en la que estas subcadenas están solapadas (por ejemplo, las palabras "ababb" y "aaabbbaba" serían aceptadas).



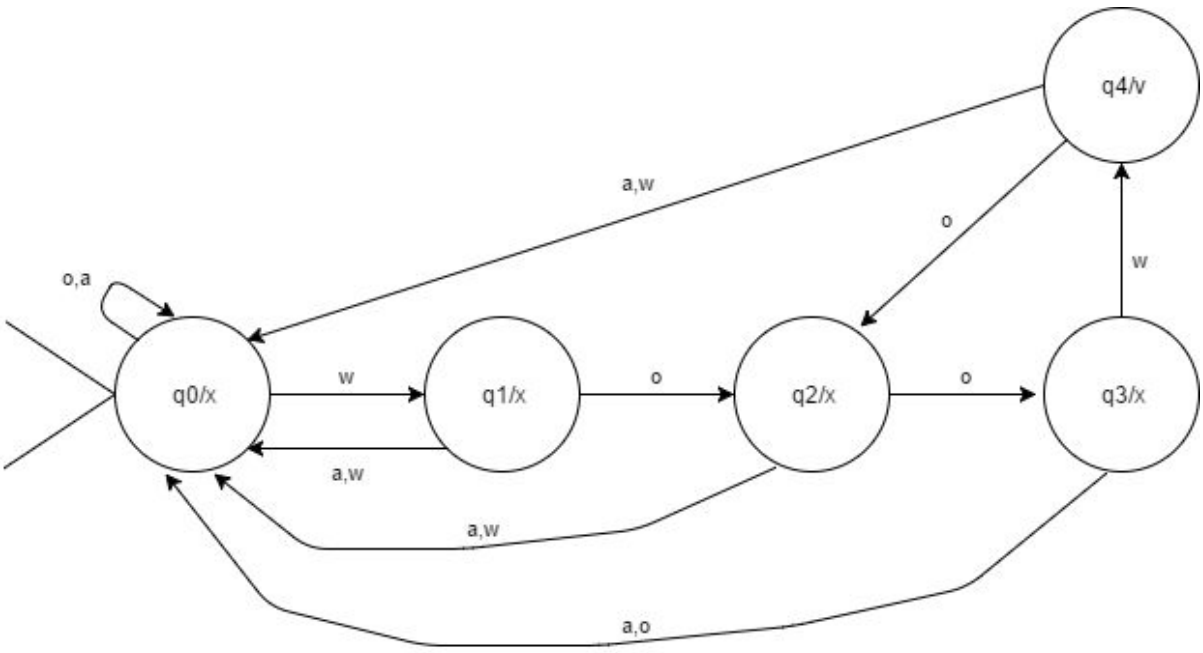
3. Diseñar una Máquina de Mealy o de Moore que, dada una cadena usando el alfabeto A={'a', 'w', 'o'}, encienda un led verde (salida 'V') cada vez que se detecte la cadena "woow" en la entrada, apagándolo cuando lea cualquier otro símbolo después de esta cadena (representamos el led apagado con la salida"X"). El autómata tiene que encender el led verde (salida 'V') , tantas veces como aparezca en la secuencia "woow" en la entrada, y esta secuencia puede estar solapada. Por ejemplo, ante la siguiente entrada, la Máquina de Mealy/Moore emitirá la salida:

entrada	**aaawoawoowoowwoowa**
salida	**XXXXXXXXXVXXVXXXVX**

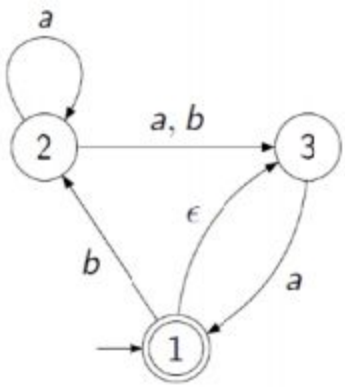
Mealy


o,a/x
w/v
q0
w/x
q1
o/x
q2
o/x
q3
a,w/x
a,w/x
a,o/x

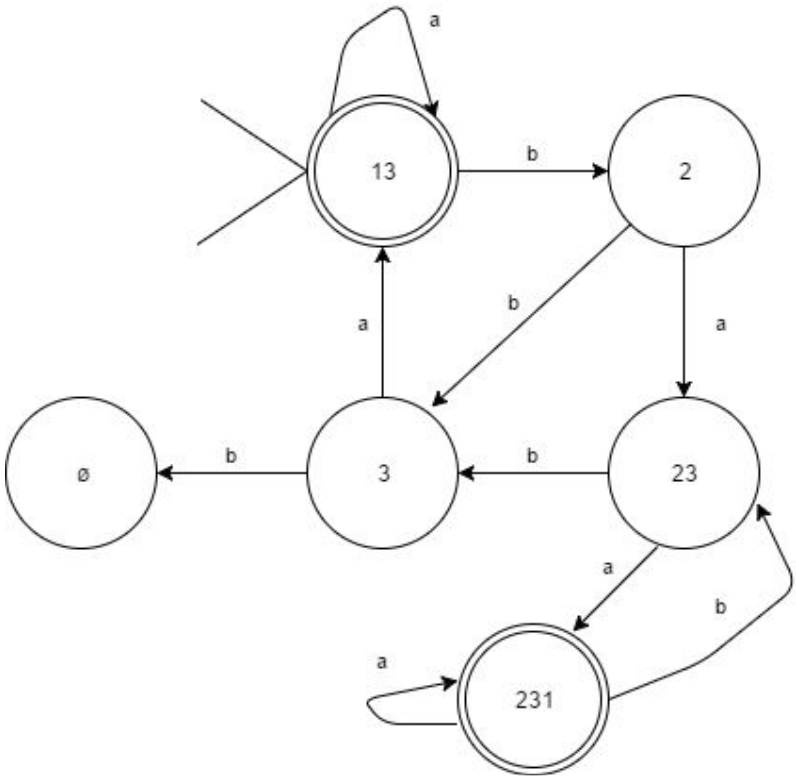
Moore


q4/v
a,w
o
w
o,a
q0/x
w
q1/x
o
q2/x
o
q3/x
a,w
a,w
a,o

4. Obtener un AFD equivalente al AFND siguiente:



	a	b
13	13	2
2	23	3
23	231	3
231	231 (estado final)	23
3	13 (estado final)	ø



Práctica 3
LEX

Explicación de la regla:

([_A-Za-z\.0-9-]+)*

Cadenas con letras mayúsculas, minúsculas, números, puntos, barras bajas, guiones. Esto lo consigue el ([]+)*

Ya que aqui iria el nombre del usuario en si, por ejemplo javigomez, javi_gomez, javigomez_95, javi-go.mez, etc.

Luego va el arroba seguido, lo que obliga que ahora después del nombre de usuario haya el arroba.

([\.a-z0-9-]+)*

Combinaciones de puntos, letras minusculas, numeros. He puesto el punto al principio porque si lo ponia al final de la regla (es decir, lo puesto antes de la "a", en vez de después del guión) no me aceptaba por ejemplo "correo.ugr". Poniendo al principio si los aceptaba.

"."([a-z]{2,4})

Un punto y combinaciones de 2 a 4 letras minúsculas para el .es, .com, .en…

He puesto toda la cadena esté entre comillas ya que esto busca cadenas de un código html de la página de la bandeja de entrada del gmail. Donde me fije que las direcciones de correo están entre comillas dobles. Y he hecho las pruebas con ese .html.

Código:

```
%{
#include <stdio.h>
%}

%%

.	{}	//Esta linea hace que los caracteres que no estan en ninguna otra regla (salvo los saltos de linea) no los muestre
\n	{}	//Esta linea hace que tampoco se muestren los campos de linea, que era lo que faltaba en la linea anterior
"\""([_A-Za-z\.0-9-]+)*@([\.a-z0-9-]+)*"."([a-z]{2,4})"\""	{
	//La regla anterior lo que hace es que coge las cadenas que estan entre parentesis por lo de "\"" ("loquesea")
	//Las cadenas que empiezan por combinaciones con las letras de "a" a "z" (mayusculas incluidas), con numeros de 0 a 9, barras bajas, puntos y guiones ("l0q9SE.ae")
```

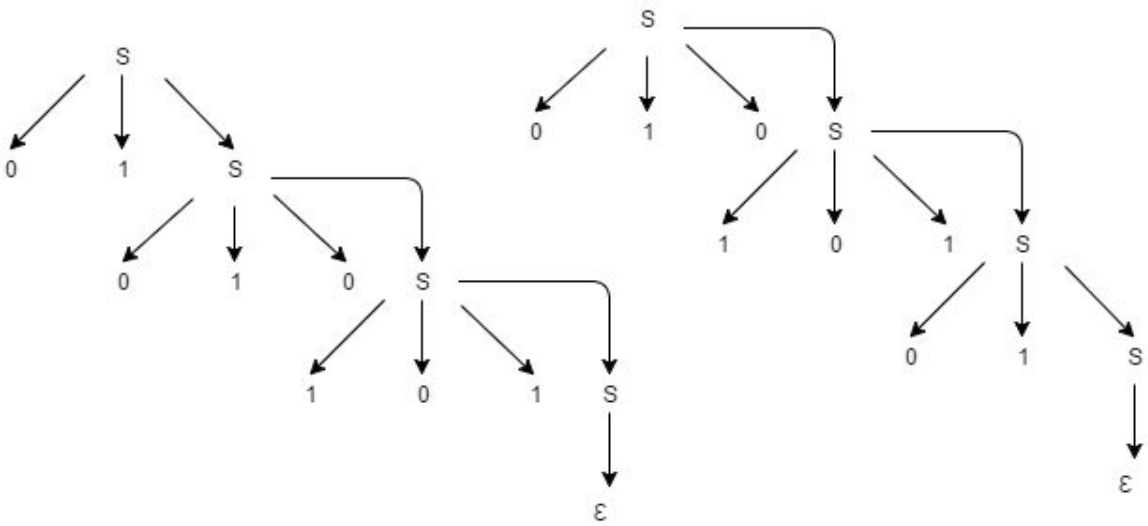
```
	//a esas cadenas les seguirÃ¡ un arroba y otra combinacion de letras, numeros, guiones y puntos ("l0q9SE.ae@correo.ugr")
	//al final de las cadenas (y antes del los "\"") ira un punto y una cadena de 2 a 4 caracteres de letras para el .es, .com ("l0q9SE.ae@correo.ugr.es")
	int i;
	for(i=1;i<yyleng-1;i++){
	//como en yytext en yytext[0] y yytext[n-1] hay unas comillas no mostramos esos caracteres
		printf("%c",yytext[i]);
	};printf("\n");}

%%

int main(int argc, char * argv[]){
	if(argc==2){
		yyin=fopen(argv[1],"r");
		printf("Fichero abierto\n");
		if(yyin==NULL){
			printf("El fichero %s no se puede abrir\n", argv[1]);
			exit(-1);
		}
	}
	else yyin=stdin;
	yylex();
}
```

Enlaces:
Código de la práctica: https://drive.google.com/open?id=0BwhDbQ4Uf6nfUUVxYWhqdWpOams
Código html de mi bandeja de entrada de gmail para que pruebe el código aquí:
https://drive.google.com/open?id=0BwhDbQ4Uf6nfeU5rTTB4SU1aVkk

PRÁCTICA 4
LENGUAJES LIBRES DE CONTEXTO

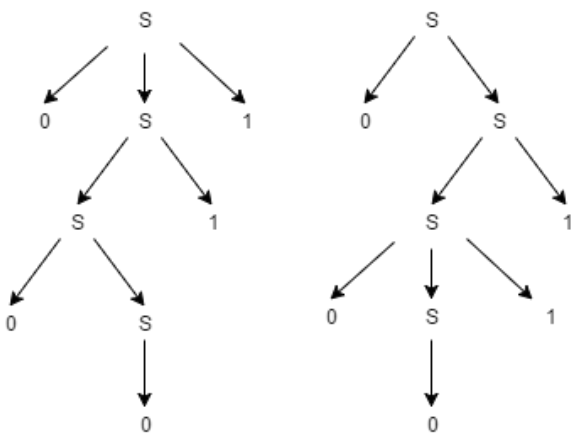
1. Determinar cuáles de las siguientes gramáticas son ambiguas y, en su caso, comprobar si los lenguajes generados son inherentemente ambiguos. Justificar la respuesta.

a) S → 01S | 010S | 101S | ε



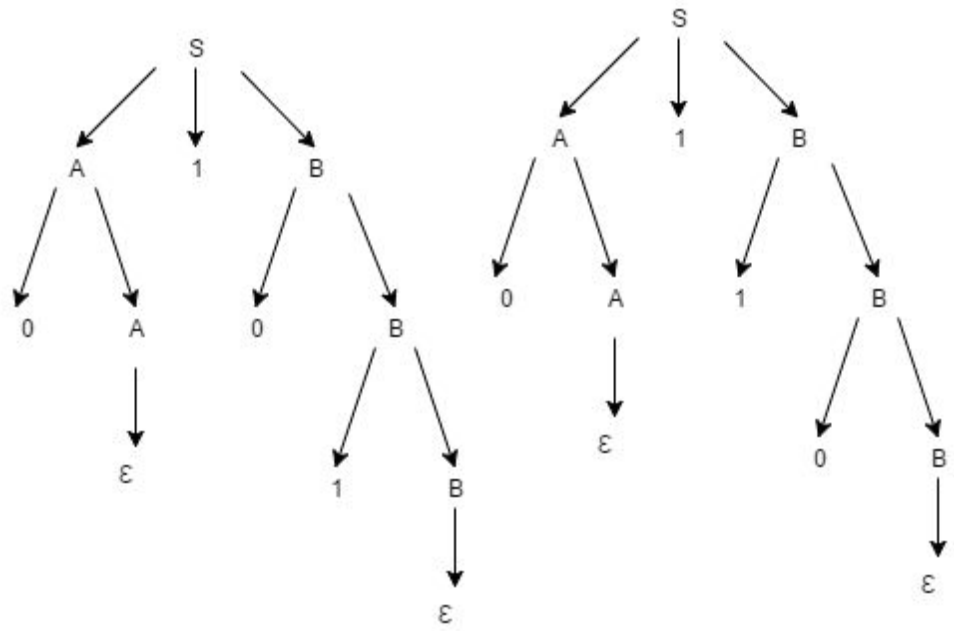
En ambos árboles obtenemos la misma secuencia "01010101" pero por árboles de derivación distintos distintos. Por lo que es una gramática ambigua. Y esta gramática no es inherentemente ambigua ya su lenguaje es de tipo 3, por lo tanto es regular y tiene un AFD asociado y por tanto una gramática no ambigua asociada.

b) S → 0S1 | S1 | 0S | 0



En ambos árboles obtenemos la misma secuencia “00011” pero por árboles de derivación distintos distintos. Por lo que es una gramática ambigua.

c) S→ A1B A→0A | ε B→0B | 1B | ε



Esta gramática no es ambigua. Ya que no existe más de un árbol por el que se pueda acceder a una misma combinación. Es decir no hay ninguna gramática que sea ambigua.

2. Eliminar símbolos y producciones inútiles. Realizar el procedimiento paso por paso, indicando las variables descartadas y el motivo.

S→moA; S→cI; A→dEs; A→jBI;
B→bb; B→D; E→elO; E→Perl;
D→de; C→c; j→kC; I→fI;
O→o; P→oIa;

Eliminamos las variables desde las que no se puede llegar a una palabra y las producciones en las que aparezcan.

V={B,O,D,C}
V’={B,O,D,C,A,E,J}
V’’={B,O,D,C,A,E,J,S}

Eliminamos la I y P. Eliminamos I porque al estar I misma en su producción nunca acabaría de producir y P la eliminamos porque P produce a I, por lo que tampoco acabaría de producir nunca. S y A no la eliminamos porque podemos evitar el camino en el que producimos a I. Quedaría.

S→moA; A→dEs; B→bb; B→D;
E→elO; D→de; C→c; J→kC;
O→o;

Eliminamos ahora los símbolos inalcanzables desde el estado inicial S y las producciones en las que aparezcan V={S} T={m,o}

V'={S,A} T={m,o,d,s}
V''={S,A,E} T={m,o,d,s,e,r,l}
V'''={S,A,E,O} T={m,o,d,s,e,r,l}

Eliminamos las variables y terminales inútiles y nos quedamos con;

S→moA
E→elO
A→dEs
O→o

3. Eliminar producciones nulas y unitarias, en el orden correcto. Realizar los procedimientos paso por paso, indicando las producciones descartadas en cada momento.

S→XYZ S→XYz X→xxX X→ε
y→yyY Y→ε Z→yxZ Z→X

Identificamos las anulables:

H={S,Y,X,Z}

Y e X llevan ambas al vacío. El símbolo S lleva solo a símbolos en una de sus producciones y Z también.

Vamos producción por producción revisandolas buscando un **ε** y eliminamos en las que lo haya

X→ε Y→ε (quedan eliminadas)

Que quedaría:

S→X S→Y S→Z S→XY S→XZ S→YZ S→XYZ
S→Xz S→Yz X→xx Y→yy Z→yx Z→X S→XYz

Ahora procedemos a eliminar las producciones unitarias. Vemos que caminos hay formando parejas:

H={(S,X), (S,Y),(S,Z), (Z,X)}

S lleva a X S→X
S lleva a Y S→Y
S lleva a Z S→Z
Z lleva a X Z→X

Eliminamos producciones unitarias y resulta:

S→X la cambiamos por S→xx (ya que X→xx)
S→Y la cambiamos por S→yy (ya que Y→yy)
S→Z la cambiamos por S→yx (ya que Z→yx)
Z→X la cambiamos por Z→xx (ya que X→xx)

S→xx S→yy S→yx Z→xx

Las añadimos y quedaría:

S→xx S→yy S→yx S→XY S→XZ S→YZ S→XYZ
S→Xz S→Yz Z→xx S→XYz

4. Pasar la siguiente gramática a forma normal de Greibach:

S→ a | CD | cs A→ a | b | SS
C→ a D→ AS

1. Renombramos :

   $S=A_1$
   $C=A_2$
   $D=A_3$
   $A=A_4$

2. $A_i \rightarrow a\ Y$ ó $A_i \rightarrow A_j\ Y$, siendo i<j

   $A_1 \rightarrow a \mid A_2\ A_3 \mid A_2\ A_1$
   $A_2 \rightarrow a$
   $A_3 \rightarrow A_4\ A_1$
   $A_4 \rightarrow a \mid b \mid A_1\ A_1$

   Ahora sustituimos para que se cumpla que i<j

   $A_4 \rightarrow a \mid b \mid \mathbf{A_1}\ A_1$

   Lo único que habría que cambiar es el símbolo en rojo.

$A_1\ A_1$ por $a \mid A_2\ A_3 \mid A_2\ A_1$

Sustituyo e identifico con color verde la parte de A_4 y con color azul la parte de A_1 quedaria asi:

$a\ A_1 \mid A_2\ A_3\ A_1 \mid A_2\ A_1\ A_1$

cambiamos otra vez los símbolos en rojo

$a\ A_1 \mid A_2\ A_3\ A_1 \mid A_2\ A_1\ A_1$

Sustituyo e identifico con color verde la parte de A_4 y con color azul la parte de A_2 quedaria asi:

$a\ A_1 \mid a\ A_3\ A_1 \mid a\ A_1\ A_2$

Esto es lo que llevamos hasta ahora:

$A_1 \rightarrow a \mid A_2\ A_3 \mid A_2\ A_1$
$A_2 \rightarrow a$
$A_3 \rightarrow A_4\ A_1$
$A_4 \rightarrow a \mid b \mid a\ A_1 \mid a\ A_3\ A_1 \mid a\ A_1\ A_2$

Vemos que A_2 y A_4 ya están en la forma normal de Greibach.

3. Sustituimos desde n-1 a 1:

Sustituyo e identifico con color verde la parte de A_1 y con color azul la parte de A_2 quedaria asi:

$A_1 \rightarrow a \mid A_2\ A_3 \mid A_2\ A_1 \rightarrow a \mid a\ A_3 \mid a\ A_1$
$A_2 \rightarrow$ No son necesarios cambios.

Sustituyo e identifico con color verde la parte de A_3 y con color azul la parte de A_4 quedaria asi:

$A_3 \rightarrow A_4\ A_1 \rightarrow a\ A_1 \mid b\ A_1 \mid a\ A_1\ A_1 \mid a\ A_3\ A_1\ A_1 \mid a\ A_1\ A_2\ A_1$
$A_4 \rightarrow$ No son necesarios cambios.

La solución es por tanto:

$A_1 \rightarrow a \mid a\ A_3 \mid a\ A_1$
$A_2 \rightarrow a$
$A_3 \rightarrow a\ A_1 \mid b\ A_1 \mid a\ A_1\ A_1 \mid a\ A_3\ A_1\ A_1 \mid a\ A_1\ A_2\ A_1$
$A_4 \rightarrow a \mid b \mid a\ A_1 \mid a\ A_3\ A_1 \mid a\ A_1\ A_2$