

Modelos de Computacion - Practicas

Eloy Bedia Garcia

27-11-2017

1 Práctica 1

1. Dada la gramática $G = (\{S, A\}, \{a, b\}, P, S)$ donde $P = \{S \rightarrow abAS, abA \rightarrow baab, S \rightarrow a, A \rightarrow b\}$. Determinar el lenguaje que genera:

$$L = \{ua \mid u \in \{baab, abb\}^+\}$$

2. Sea la gramática $G = (V, T, P, S)$ donde:

$$V = \{< numero >, < digito >\}$$

$$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$S = < numero >$$

Las reglas de producción P son las siguientes:

$$< numero > \rightarrow < digito > < numero >$$

$$< numero > \rightarrow < digito >$$

$$< digito > \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Determinar el lenguaje que genera:

$$L = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^+$$

3. Encontrar si es posible una gramática libre de contexto que genera el lenguaje L , siendo $L \subseteq \{a, b, c\}^*$. La palabra $u \in L \iff$ o contiene el mismo número de símbolos b que de símbolos c .

Sea la gramática $G = (V, T, P, S)$ donde:

$$V = \{S, A\}$$

$$T = \{a, b, c\}$$

$$S = S$$

Las reglas de producción P son las siguientes:

$$< numero > \rightarrow bSc \mid A$$

$$< numero > \rightarrow Aa \mid \epsilon$$

4. Encontrar gramáticas de tipo 2 para $L \subseteq \{0, 1\}^*$. En cada caso, indicar si los lenguajes generados son regulares indicando, si existe una gramática regular que los genera. Para los siguientes casos:

- a. Palabras que comienzan con 000 y terminan en 111.

$G = (\{S, A\}, \{0, 1\}, P, S)$ donde P son las producciones siguientes:

$$S \rightarrow 000A111$$

$$A \rightarrow 0A \mid 1A \mid \epsilon$$

Gramática regular $G = (\{S, A, B\}, \{0, 1\}, P, S)$ donde P son las producciones siguientes:

$$S \rightarrow 000A$$

$$A \rightarrow 0A \mid B \mid 111$$

$$A \rightarrow 1B \mid A \mid 11$$

- b. Palabras que no contienen dos 0 seguidos (00).

$G = (\{S, A\}, \{0, 1\}, P, S)$ donde P son las producciones siguientes:

$$S \rightarrow 0A \mid 1S \mid \epsilon$$

$$A \rightarrow 1S \mid \epsilon$$

Este lenguaje es regular, es decir, puede ser generado por una gramática regular (tipo 3)

c. Palabras que si contienen tres ceros (000) le sigue al menos un 1.

$G = (\{S, A, B, C\}, \{0, 1\}, P, S)$ donde P son las producciones siguientes:

$$\begin{aligned} S &\rightarrow 0A \mid C \mid \epsilon \\ A &\rightarrow 0B \mid C \mid \epsilon \\ B &\rightarrow 0C \mid C \mid \epsilon \\ C &\rightarrow 1S \end{aligned}$$

Este lenguaje es regular, es decir, puede ser generado por una gramática regular (tipo 3)

5. En una empresa de videojuegos “Dungeons of the Hell” están planteando diseñar una gramática capaz de generar niveles de un juego de exploración de mazmorras, y sus salas correspondientes, siguiendo el siguiente conjunto de restricciones:

- Hay 2 tipos de salas, grandes (g) y pequeñas (p).
- Hay 2 tipos de monstruos, fuertes (f) y débiles (d).
- Las salas grandes, contienen al menos 1 monstruo fuerte y 2 débiles. Y las salas pequeñas, contienen a lo sumo 1 monstruo fuerte.
- Existen unas salas especiales donde se pueden reponer fuerzas y comprar armas. Estas salas son las llamadas salas de tendero (t).
- Siempre, tras la sala grande, se va a encontrar una sala secreta (s). Al final de cada nivel habrá una sala final donde hay que rescatar a una princesa (x).

Elaborar una gramática que genere estos niveles con sus restricciones. Cada palabra del lenguaje es un solo nivel. ¿A que tipo de gramática dentro de la jerarquía de Chomsky pertenece la gramática diseñada? ¿Sería posible diseñar una gramática de tipo 3 para dicho problema?

$G = (\{S, G, P, M\}, \{g, p, f, d, t, x\}, P, S)$ donde P son las producciones siguientes:

$$\begin{aligned} S &\rightarrow GS \mid PS \mid tS \mid x \\ M &\rightarrow fM \mid dM \mid \epsilon \\ G &\rightarrow gMfMdMdM \mid gMdMfMdM \mid gMdMdMf \\ P &\rightarrow pMdM \end{aligned}$$

Esta gramática, según la Jerarquía de Chomsky, es de Tipo 2 (Independiente del contexto).

No existe una gramática regular que satisfaga estas restricciones, ya que hay que comprobar que exista un mínimo de monstruos determinados, y para ello es necesaria una gramática de tipo 2

2 Práctica 2

Dada la URL de un video de Youtube, el software te facilita los siguientes datos:

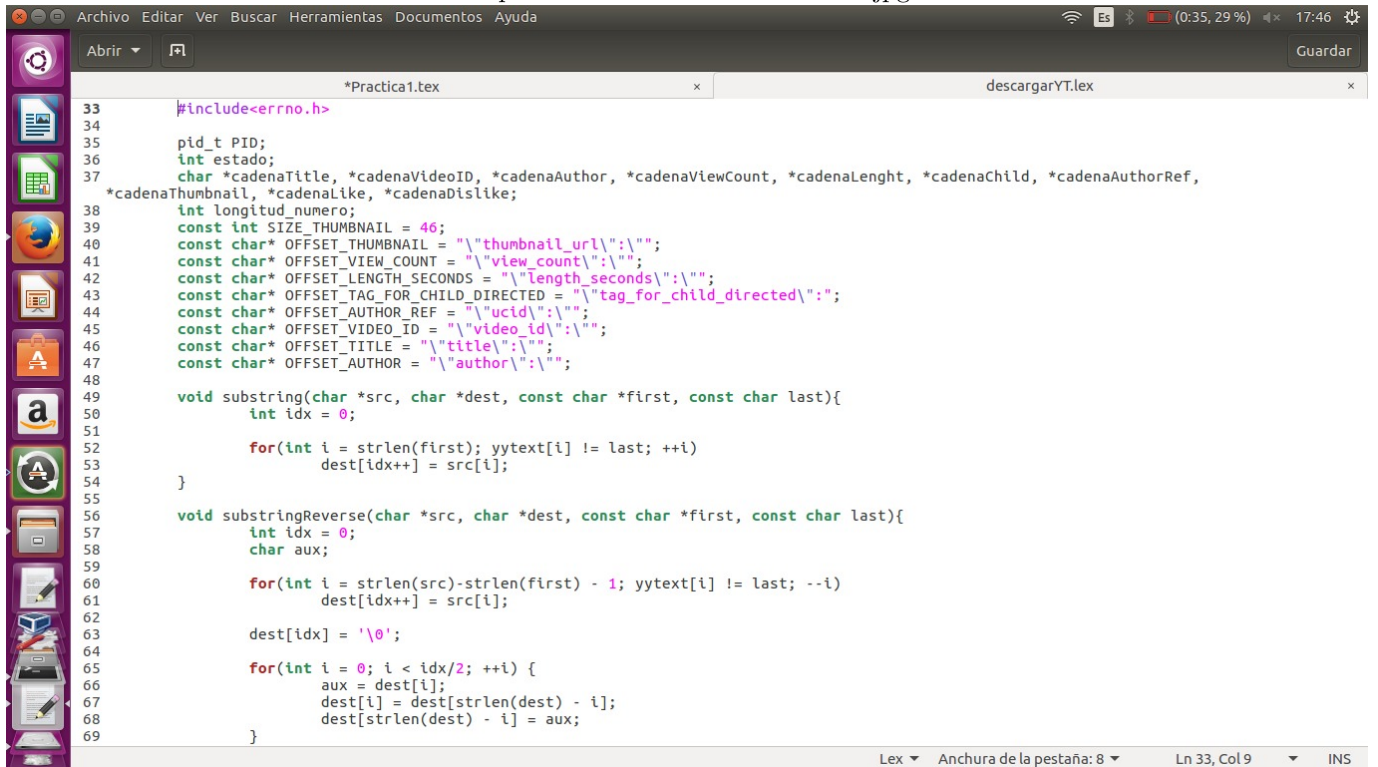
-Título

- Autor
- ID
- Miniatura
- Nº Likes
- Nº Dislikes
- Nº Visitas
- Duración (s)
- ¿Apt para niños?

de pantalla de 2018-01-07 17-46-40.jpg

```
1 ID [0-9a-zA-Z]{11}
2 VIDEO_ID \\"video_id\\":\\"{ID}\\\"
3 TITLE \\"title\\":\\"{^}\\\"
4 AUTHOR \\"author\\":\\"{0-9a-zA-Zá-ú\\\"
5 URL_ENCODED_FMT_STREAM_MAP \\"url_encoded_fmt_stream_map\\":\\"{^url=\\\"}url=https[^\\\"}+\\\"
6 VIEW_COUNT \\"view_count\\":\\"{0-9}\\\"
7 LENGTH_SECONDS \\"length_seconds\\":\\"{0-9}\\\"
8 TAG_FOR_CHILD_DIRECTED \\"tag_for_child_directed\\":\\"{true|false}\\\"
9
10 AUTHOR_REF \\"ucid\\":\\"{0-9a-zA-Z\\\"{24}\\\"
11
12 SUFIX \\"\\default\\.jpg\\\"
13 PREFIX \\"thumbnail_url\\":\\"https:\\/\\/i\\.ytimg\\.com\\/vi\\\"
14 THUMBNAIL_URL {PREFIX}{ID}{SUFIX}
15
16 LIKE \\"<button class=\\\"yt-ux-button yt-ux-button-size-default yt-ux-button-opacity yt-ux-button-has-icon no-
icon-markup like-button-renderer-like-button like-button-renderer-like-button-clicked yt-ux-button-toggled hid yt-ux-tooltip\\\" type=
\\\"button\\\" onclick=\\\";return false;\\\" title=\\\"{^}\\\" aria-label=\\\"{1-9}\\\" data-position=\\\"bottomright\\\"
data-orientation=\\\"vertical\\\" data-force-position=\\\"true\\\">\\\"{NUMBERPOINT}\\\"</span>\\\"
17
18 DISLIKE \\"<button class=\\\"yt-ux-button yt-ux-button-size-default yt-ux-button-opacity yt-ux-button-has-icon no-
icon-markup like-button-renderer-dislike-button like-button-renderer-dislike-button-clicked yt-ux-button-toggled hid yt-ux-tooltip\\\"
type=\\\"button\\\" onclick=\\\";return false;\\\" title=\\\"{^}\\\" aria-label=\\\"{1-9}\\\" data-position=\\\"bottomright\\\"
\\\" data-orientation=\\\"vertical\\\" data-force-position=\\\"true\\\">\\\"{NUMBERPOINT}\\\"</span>\\\"
19
20
21 SUBS .longSubscriberCountText.+
22
23 NUMBERPOINT [1-9]{1,3}\\\"{0-9}\\\"{3}\\\"*
24
25 NONE .|\\n
26
27 %{
28 #include<unistd.h>
29 #include<stdio.h>
30 #include<sys/types.h>
31 #include <sys/wait.h>
32 #include<stdlib.h>
```

de pantalla de 2018-01-07 17-46-55.jpg

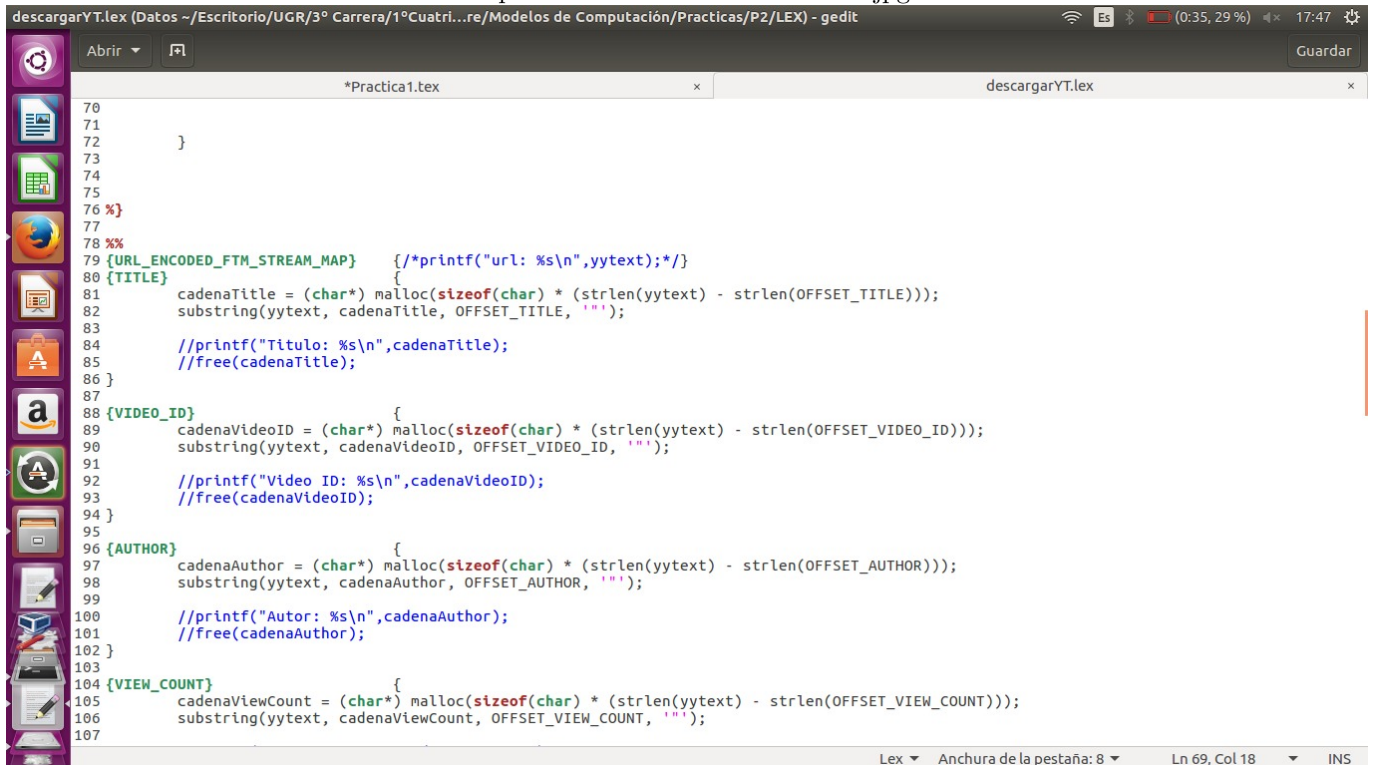


```

33 #include<errno.h>
34
35 pid_t PID;
36 int estado;
37 char *cadenaTitle, *cadenaVideoID, *cadenaAuthor, *cadenaViewCount, *cadenaLenght, *cadenaChild, *cadenaAuthorRef,
*cadenaThumbnail, *cadenaLike, *cadenaDislike;
38 int longitud_numero;
39 const int SIZE_THUMBNAIL = 46;
40 const char* OFFSET_THUMBNAIL = "\"thumbnail_url\":\":";
41 const char* OFFSET_VIEW_COUNT = "\"view_count\":\":";
42 const char* OFFSET_LENGTH_SECONDS = "\"length_seconds\":\":";
43 const char* OFFSET_TAG_FOR_CHILD_DIRECTED = "\"tag_for_child_directed\":\":";
44 const char* OFFSET_AUTHOR_REF = "\"ucid\":\":";
45 const char* OFFSET_VIDEO_ID = "\"video_id\":\":";
46 const char* OFFSET_TITLE = "\"title\":\":";
47 const char* OFFSET_AUTHOR = "\"author\":\":";
48
49 void substring(char *src, char *dest, const char *first, const char last){
50     int idx = 0;
51
52     for(int i = strlen(first); yytext[i] != last; ++i)
53         dest[idx++] = src[i];
54 }
55
56 void substringReverse(char *src, char *dest, const char *first, const char last){
57     int idx = 0;
58     char aux;
59
60     for(int i = strlen(src)-strlen(first) - 1; yytext[i] != last; --i)
61         dest[idx++] = src[i];
62
63     dest[idx] = '\0';
64
65     for(int i = 0; i < idx/2; ++i) {
66         aux = dest[i];
67         dest[i] = dest[strlen(dest) - i];
68         dest[strlen(dest) - i] = aux;
69     }

```

de pantalla de 2018-01-07 17-47-06.jpg

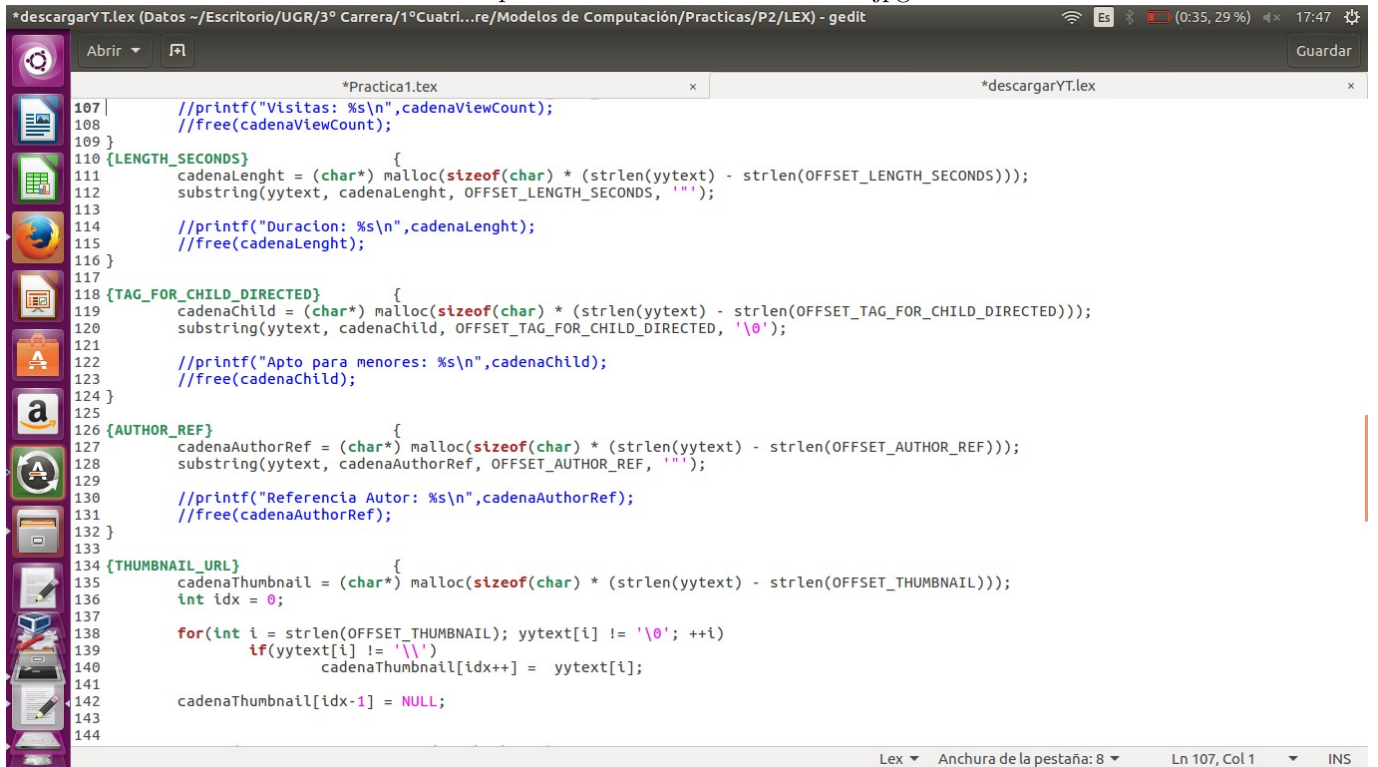


```

70
71 }
72
73
74
75
76 %}
77
78 %%
79 {URL_ENCODED_FTM_STREAM_MAP} { /*printf("url: %s\n",yytext);*/}
80 {TITLE} {
81     cadenaTitle = (char*) malloc(sizeof(char) * (strlen(yytext) - strlen(OFFSET_TITLE)));
82     substring(yytext, cadenaTitle, OFFSET_TITLE, '"');
83
84     //printf("Titulo: %s\n",cadenaTitle);
85     //free(cadenaTitle);
86 }
87
88 {VIDEO_ID} {
89     cadenaVideoID = (char*) malloc(sizeof(char) * (strlen(yytext) - strlen(OFFSET_VIDEO_ID)));
90     substring(yytext, cadenaVideoID, OFFSET_VIDEO_ID, '"');
91
92     //printf("Video ID: %s\n",cadenaVideoID);
93     //free(cadenaVideoID);
94 }
95
96 {AUTHOR} {
97     cadenaAuthor = (char*) malloc(sizeof(char) * (strlen(yytext) - strlen(OFFSET_AUTHOR)));
98     substring(yytext, cadenaAuthor, OFFSET_AUTHOR, '"');
99
100     //printf("Autor: %s\n",cadenaAuthor);
101     //free(cadenaAuthor);
102 }
103
104 {VIEW_COUNT} {
105     cadenaViewCount = (char*) malloc(sizeof(char) * (strlen(yytext) - strlen(OFFSET_VIEW_COUNT)));
106     substring(yytext, cadenaViewCount, OFFSET_VIEW_COUNT, '"');
107 }

```

de pantalla de 2018-01-07 17-47-23.jpg

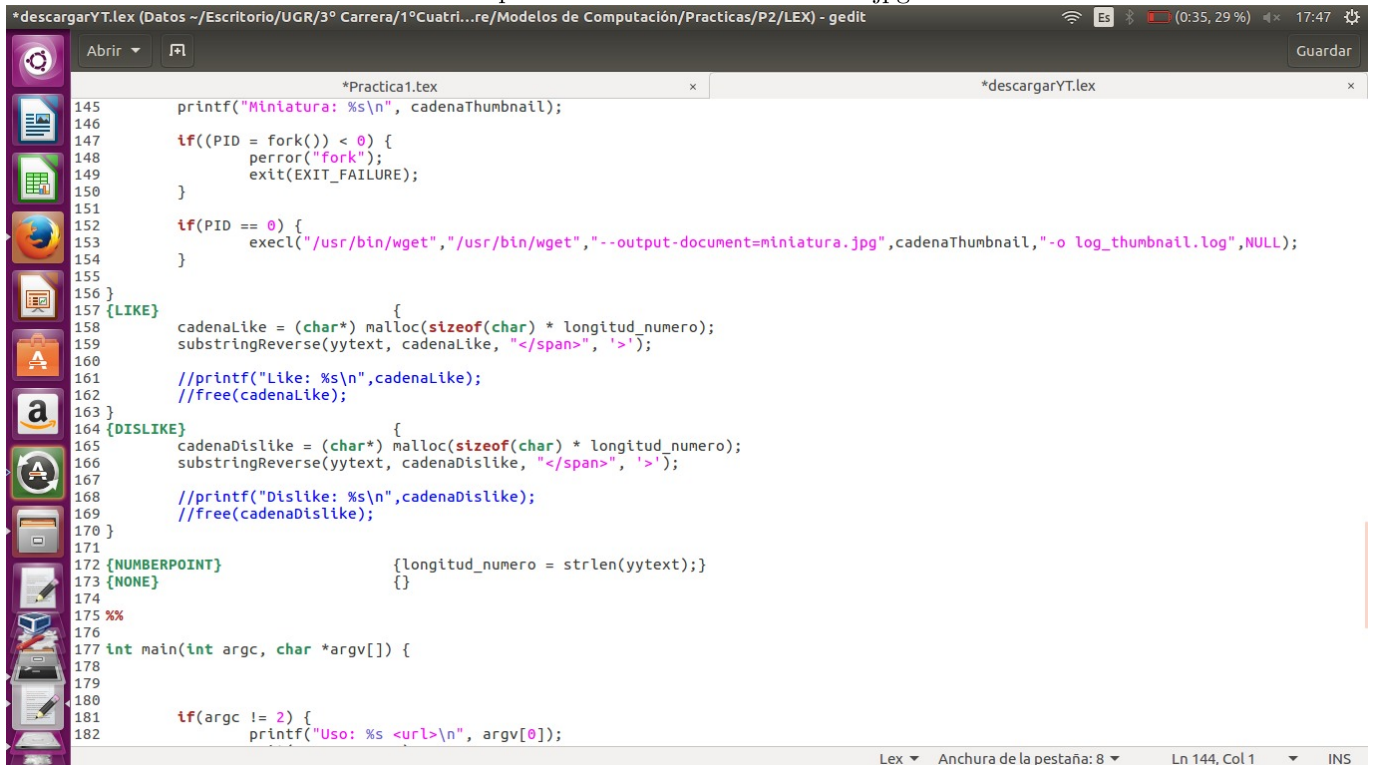


```
*descargarYT.lex (Datos ~/Escritorio/UGR/3º Carrera/1º Cuatri...re/Modelos de Computación/Practicas/P2/LEX) - gedit
Abrir Guardar

*Practica1.tex
107 //printf("Visitas: %s\n",cadenaViewCount);
108 //free(cadenaViewCount);
109 }
110 {LENGTH_SECONDS} {
111     cadenaLenght = (char*) malloc(sizeof(char) * (strlen(yytext) - strlen(OFFSET_LENGTH_SECONDS)));
112     substring(yytext, cadenaLenght, OFFSET_LENGTH_SECONDS, '\0');
113 }
114 //printf("Duracion: %s\n",cadenaLenght);
115 //free(cadenaLenght);
116 }
117
118 {TAG_FOR_CHILD_DIRECTED} {
119     cadenaChild = (char*) malloc(sizeof(char) * (strlen(yytext) - strlen(OFFSET_TAG_FOR_CHILD_DIRECTED)));
120     substring(yytext, cadenaChild, OFFSET_TAG_FOR_CHILD_DIRECTED, '\0');
121 }
122 //printf("Apto para menores: %s\n",cadenaChild);
123 //free(cadenaChild);
124 }
125
126 {AUTHOR_REF} {
127     cadenaAuthorRef = (char*) malloc(sizeof(char) * (strlen(yytext) - strlen(OFFSET_AUTHOR_REF)));
128     substring(yytext, cadenaAuthorRef, OFFSET_AUTHOR_REF, '\0');
129 }
130 //printf("Referencia Autor: %s\n",cadenaAuthorRef);
131 //free(cadenaAuthorRef);
132 }
133
134 {THUMBNAIL_URL} {
135     cadenaThumbnail = (char*) malloc(sizeof(char) * (strlen(yytext) - strlen(OFFSET_THUMBNAIL)));
136     int idx = 0;
137
138     for(int i = strlen(OFFSET_THUMBNAIL); yytext[i] != '\0'; ++i)
139         if(yytext[i] != '\\')
140             cadenaThumbnail[idx++] = yytext[i];
141
142     cadenaThumbnail[idx-1] = NULL;
143 }
144
```

Lex Anchura de la pestaña: 8 Ln 107, Col 1 INS

de pantalla de 2018-01-07 17-47-31.jpg

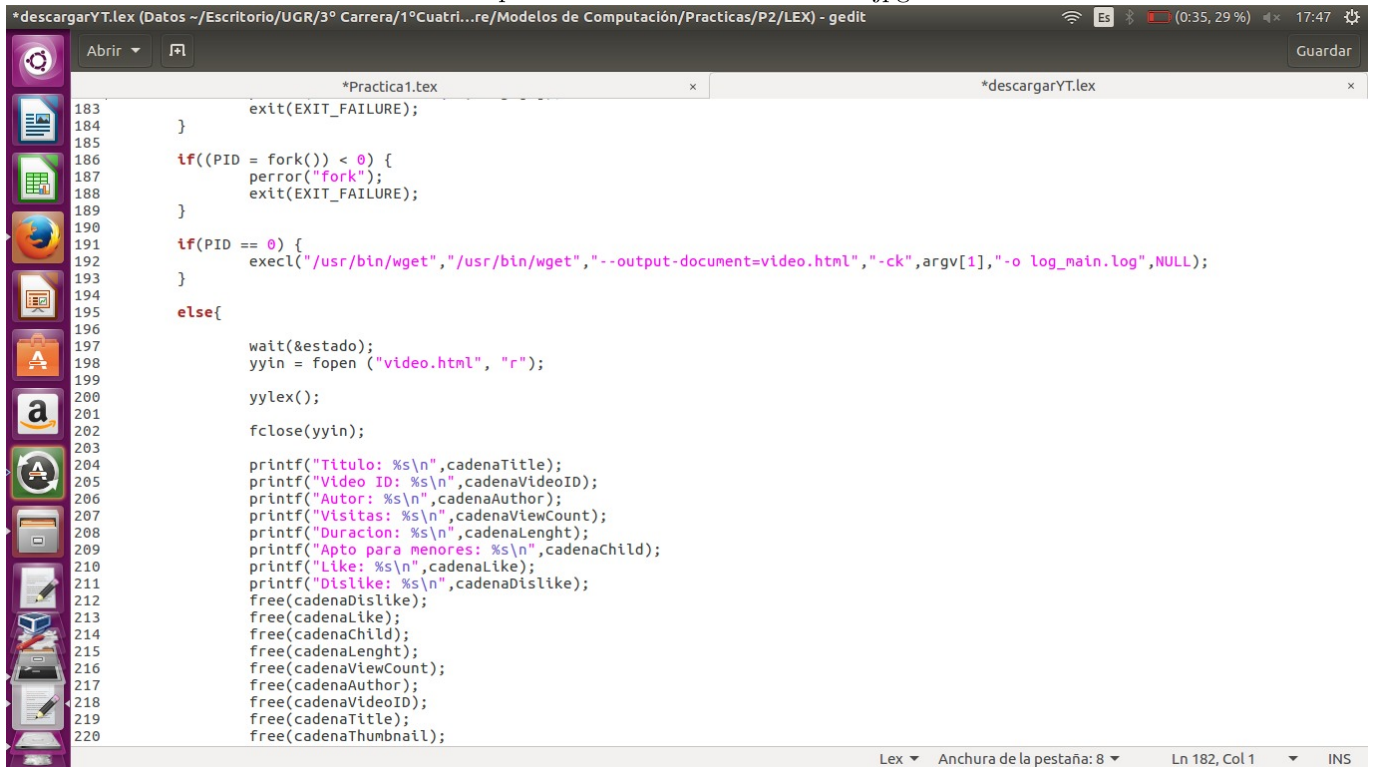


```
*descargarYT.lex (Datos ~/Escritorio/UGR/3º Carrera/1º Cuatri...re/Modelos de Computación/Practicas/P2/LEX) - gedit
Abrir Guardar

*Practica1.tex
145 printf("Miniatura: %s\n", cadenaThumbnail);
146
147 if((PID = fork()) < 0) {
148     perror("fork");
149     exit(EXIT_FAILURE);
150 }
151
152 if(PID == 0) {
153     execl("/usr/bin/wget", "/usr/bin/wget", "--output-document=miniatura.jpg", cadenaThumbnail, "-o log_thumbnail.log", NULL);
154 }
155
156 }
157 {LIKE} {
158     cadenaLike = (char*) malloc(sizeof(char) * longitud_numero);
159     substringReverse(yytext, cadenaLike, "</span>", '>');
160 }
161 //printf("Like: %s\n",cadenaLike);
162 //free(cadenaLike);
163 }
164 {DISLIKE} {
165     cadenaDislike = (char*) malloc(sizeof(char) * longitud_numero);
166     substringReverse(yytext, cadenaDislike, "</span>", '>');
167 }
168 //printf("Dislike: %s\n",cadenaDislike);
169 //free(cadenaDislike);
170 }
171
172 {NUMBERPOINT} {longitud_numero = strlen(yytext);}
173 {NONE} {}
174
175 %%
176
177 int main(int argc, char *argv[]) {
178
179
180
181     if(argc != 2) {
182         printf("Uso: %s <url>\n", argv[0]);
183     }
184 }
```

Lex Anchura de la pestaña: 8 Ln 144, Col 1 INS

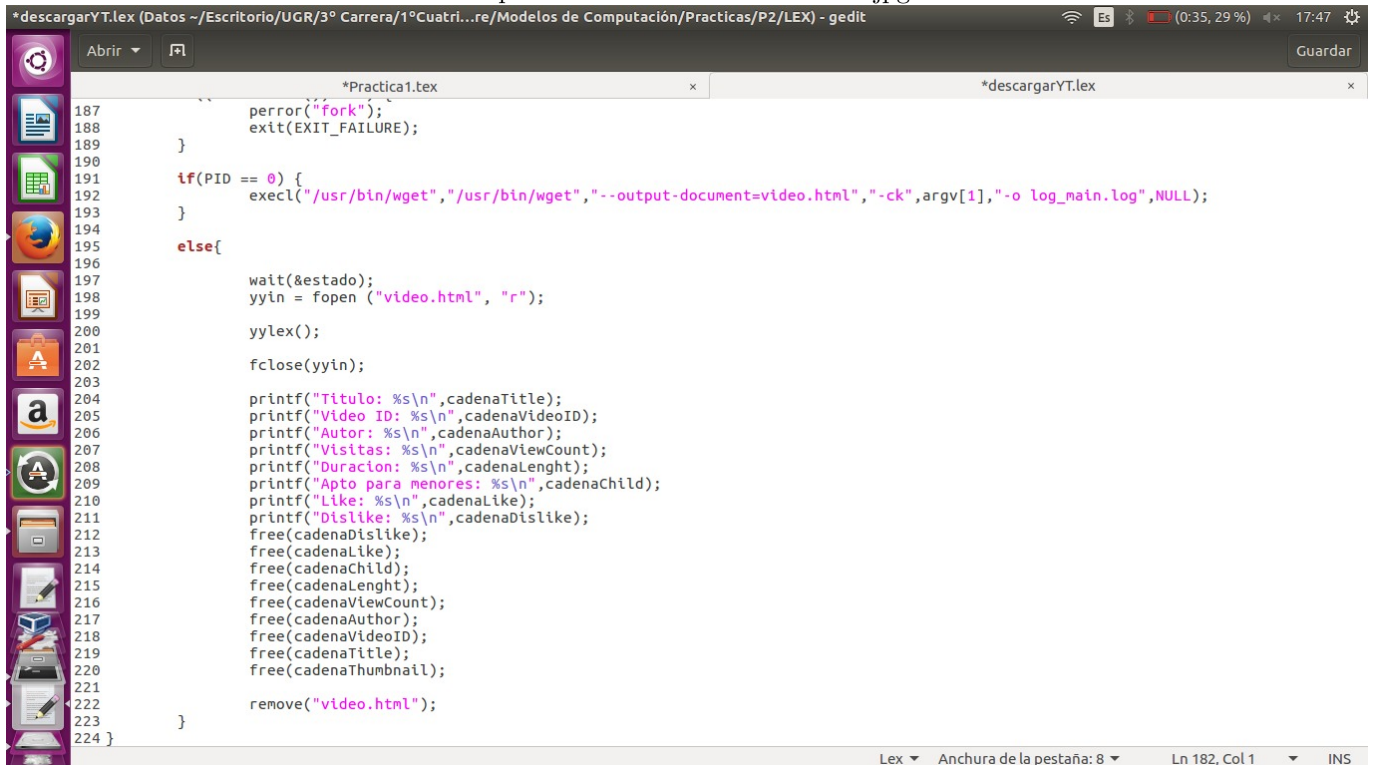
de pantalla de 2018-01-07 17-47-43.jpg



```
*descargarYT.lex (Datos ~/Escritorio/UGR/3º Carrera/1º Cuatri...re/Modelos de Computación/Practicas/P2/LEX) - gedit
Abrir Guardar

*Practica1.tex
183     exit(EXIT_FAILURE);
184 }
185
186 if((PID = fork()) < 0) {
187     perror("fork");
188     exit(EXIT_FAILURE);
189 }
190
191 if(PID == 0) {
192     execl("/usr/bin/wget", "/usr/bin/wget", "--output-document=video.html", "-ck", argv[1], "-o log_main.log", NULL);
193 }
194
195 else{
196     wait(&estado);
197     yyin = fopen ("video.html", "r");
198     yylex();
199     fclose(yyin);
200
201     printf("Titulo: %s\n", cadenaTitle);
202     printf("Video ID: %s\n", cadenaVideoID);
203     printf("Autor: %s\n", cadenaAuthor);
204     printf("Visitas: %s\n", cadenaViewCount);
205     printf("Duracion: %s\n", cadenaLenght);
206     printf("Apto para menores: %s\n", cadenaChild);
207     printf("Like: %s\n", cadenaLike);
208     printf("Dislike: %s\n", cadenaDislike);
209     free(cadenaDislike);
210     free(cadenaLike);
211     free(cadenaChild);
212     free(cadenaLenght);
213     free(cadenaViewCount);
214     free(cadenaAuthor);
215     free(cadenaVideoID);
216     free(cadenaTitle);
217     free(cadenaThumbnail);
218 }
219
220
Lex Anchura de la pestaña: 8 Ln 182, Col 1 INS
```

de pantalla de 2018-01-07 17-47-58.jpg



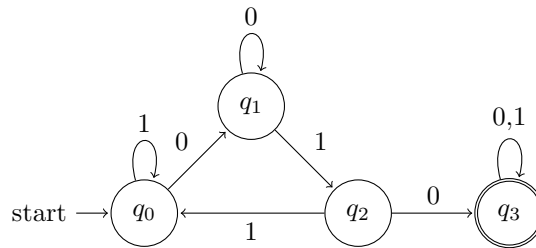
```
*descargarYT.lex (Datos ~/Escritorio/UGR/3º Carrera/1º Cuatri...re/Modelos de Computación/Practicas/P2/LEX) - gedit
Abrir Guardar

*Practica1.tex
187     perror("fork");
188     exit(EXIT_FAILURE);
189 }
190
191 if(PID == 0) {
192     execl("/usr/bin/wget", "/usr/bin/wget", "--output-document=video.html", "-ck", argv[1], "-o log_main.log", NULL);
193 }
194
195 else{
196     wait(&estado);
197     yyin = fopen ("video.html", "r");
198     yylex();
199     fclose(yyin);
200
201     printf("Titulo: %s\n", cadenaTitle);
202     printf("Video ID: %s\n", cadenaVideoID);
203     printf("Autor: %s\n", cadenaAuthor);
204     printf("Visitas: %s\n", cadenaViewCount);
205     printf("Duracion: %s\n", cadenaLenght);
206     printf("Apto para menores: %s\n", cadenaChild);
207     printf("Like: %s\n", cadenaLike);
208     printf("Dislike: %s\n", cadenaDislike);
209     free(cadenaDislike);
210     free(cadenaLike);
211     free(cadenaChild);
212     free(cadenaLenght);
213     free(cadenaViewCount);
214     free(cadenaAuthor);
215     free(cadenaVideoID);
216     free(cadenaTitle);
217     free(cadenaThumbnail);
218     remove("video.html");
219 }
220
221
222
223
224 }
Lex Anchura de la pestaña: 8 Ln 182, Col 1 INS
```

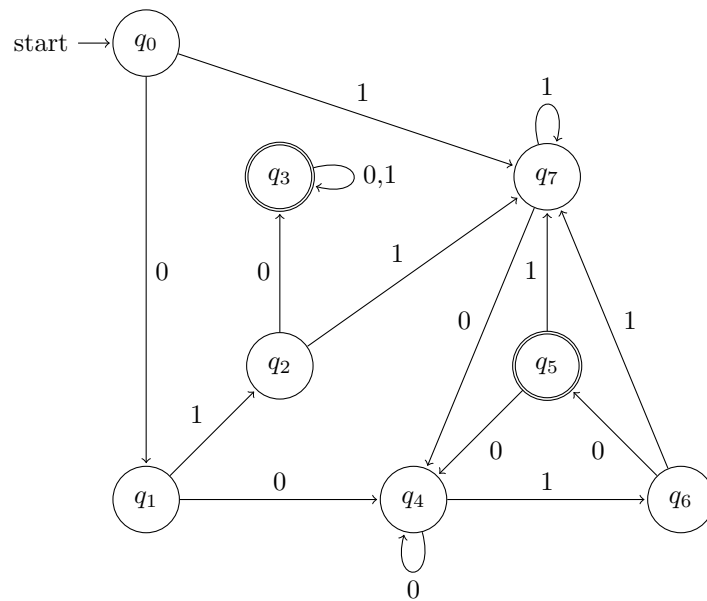
3 Práctica 3

1. Construir un AFD que acepte cada uno de los siguientes lenguajes con alfabeto 0,1:

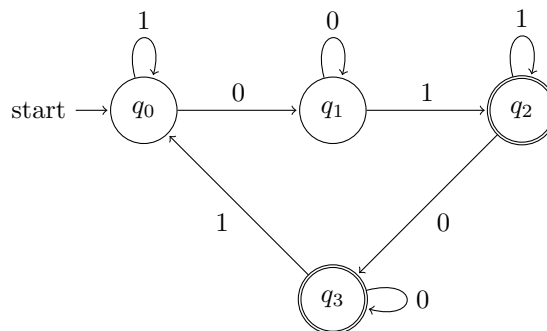
a. El lenguaje de las palabras que contienen la subcadena 010.



b. El lenguaje de las palabras que empiezan o terminan (o ambas cosas) en 010.

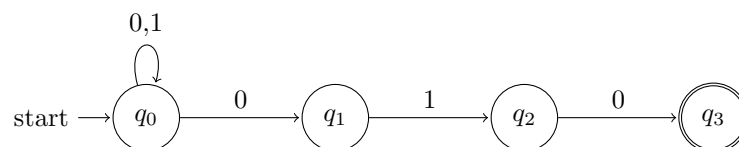


c. El lenguaje $L \subset \{0,1\}^*$ de todas las palabras con un número impar de ocurrencias de la subcadena 01.

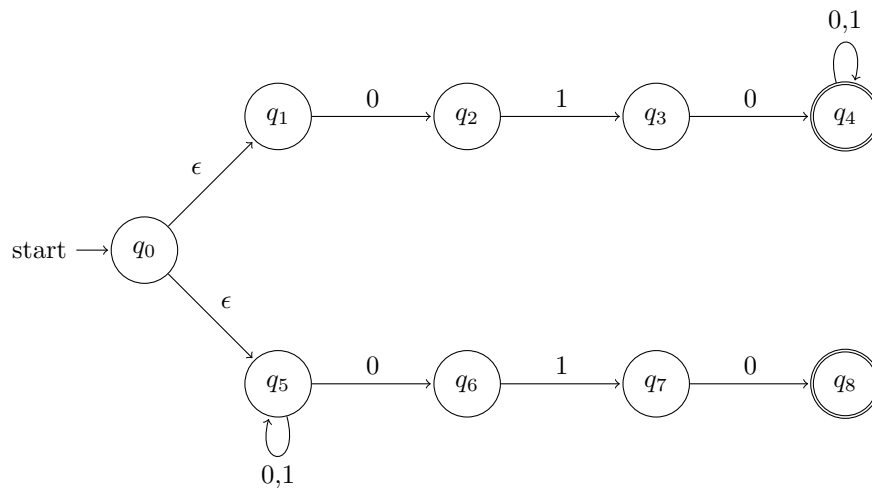


2. Construir un AFND que acepte cada uno de los siguientes lenguajes con alfabeto 0,1:

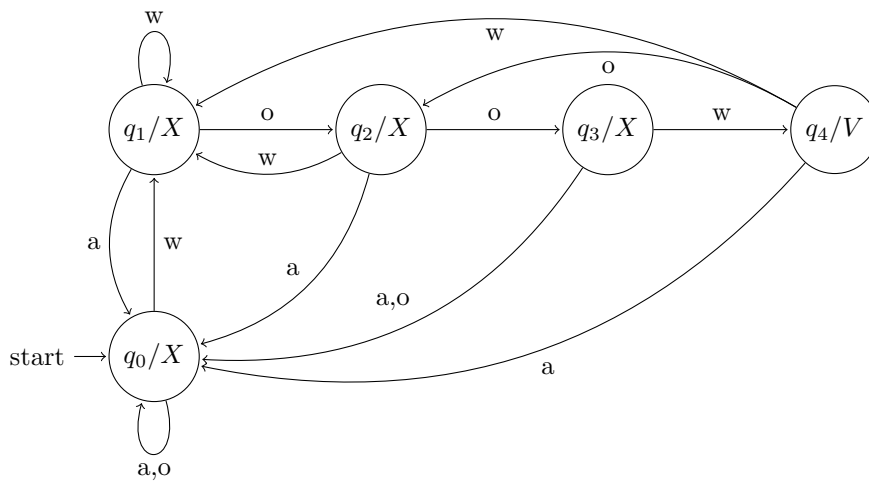
a. El lenguaje de las palabras que terminan en 010.



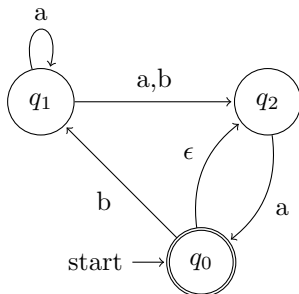
- b. El lenguaje de las palabras que empiezan o terminan (o ambas cosas) en 010.

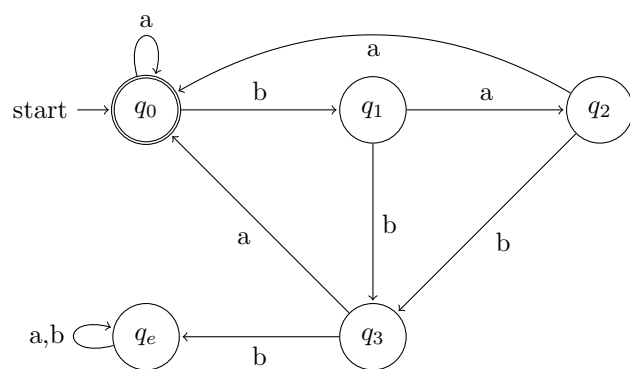


3. Diseñar una Máquina de Mealy o de Moore que, dada una cadena usando el alfabeto $A = \{a, w, o\}$ encienda un led verde (salida 'V') cada vez que se detecte la cadena "woow" en la entrada apagándolo cuando lea cualquier otro símbolo después de esta cadena (representamos el led apagado con la salida "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:



4. Obtener un AFD equivalente al AFND siguiente:





4 Práctica 4

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. $S \rightarrow AbB \mid A \rightarrow aA \mid \epsilon \mid B \rightarrow aB \mid bB \mid \epsilon$

No es ambigua, ya que produce a's al principio; y a's y b's al final.

b. $S \rightarrow abaS \mid baS \mid babS \mid \epsilon$

Es ambigua, ya haciendo 3 veces la producción 2 se puede llegar al mismo resultado que haciendo una vez la 3 y la 1.

El lenguaje no es inherentemente ambiguo, ya que la siguiente gramática no ambigua genera el mismo lenguaje:

$$S \rightarrow aA \mid bC \mid \epsilon$$

$$A \rightarrow bB$$

$$B \rightarrow aS$$

$$C \rightarrow aD$$

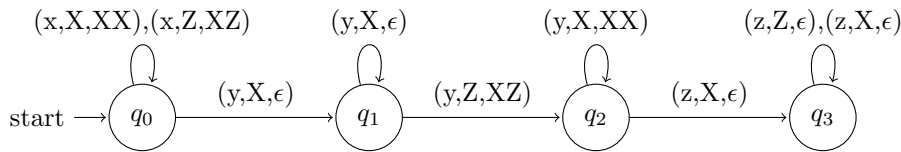
$$C \rightarrow bS \mid aA \mid \epsilon$$

c. $S \rightarrow aSA \mid \epsilon \mid A \rightarrow bA \mid \epsilon$

Es ambigua, ya que produciendo N a's puedo producir hasta N b's, con lo que llevando a vacío cualquier $A \leq N$ b's puedo obtener una palabra con N a's y N - A b's.

2. Encontrar el autómata con pila que acepte el siguiente lenguaje L.

$$L = x^i y^j z^k \mid i + k = j; i, j, k \in N$$



Gramática:

$$S \rightarrow aSb \mid bSa$$

3. Pasar este autómata con pila a gramática independiente del contexto. Una vez conseguida la gramática, proceder a eliminar variables y producciones inútiles. ¿Qué lenguaje genera dicha gramática?

$$S \rightarrow aAS \mid \epsilon$$

$$A \rightarrow aAA \mid b$$

El lenguaje generado por esta gramática pertenece a $\{a, b\}^*$ tal que para toda a perteneciente a $\{a, b\}^*$ si la divides por la mitad tiene el mismo número de a's que de b's

4. Dar gramáticas libres de contexto o regulares no ambiguas (cuando sea posible) para los siguientes lenguajes:

a. $L_1 = (ab)^i (bc)^j \mid i, j \geq 0$

$$S \rightarrow abS \mid B$$

$$B \rightarrow bcB \mid \epsilon$$

b. $L_2 = a^i b^j c^{i+j} \mid i, j \geq 0$

$$S \rightarrow aSc \mid A$$

$$A \rightarrow bAc \mid \epsilon$$

c. $L_3 = a^i b^j c^j d^i \mid i, j \geq 0$

$$S \rightarrow aSd \mid A$$

$$A \rightarrow bAc \mid \epsilon$$

d. L_4 definido como el conjunto de palabras de alfabeto a, b, c que empiezan por aab y acaban por bbc y tales que estas dos subcadenas no aparecen nunca en el interior de la palabra (sólo al comienzo y al final).

$$S \rightarrow aabAbbc$$

$$A \rightarrow aA_1 \mid bA_3 \mid cA$$

$$A_1 \rightarrow aA_2 \mid bA_3 \mid cA$$

$$A_2 \rightarrow aA_2 \mid cA$$

$$A_3 \rightarrow bA_4 \mid aA_1 \mid cA$$

$$A_4 \rightarrow aA_1 \mid bA_4$$