Modelos de computación (2016-2017)
Grado en Ingeniería Informática
Universidad de Granada

Memoria de practicas

Antonio Rodríguez Alaminos

22 de febrero de 2017

Índice

Índice de figuras

Índice de tablas

1. Práctica 1

1.1. Ejercicio 1

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

1.1.1. Apartado A

$S \to 0S_1 1$ $\qquad\qquad$ $S_1 \to 0S_1|1S_1|\epsilon$

Cuando tenemos que $S_1 \to \epsilon$:
$S \to 0S_1 1 \Rightarrow 0\epsilon 1 \Rightarrow \mathbf{01}$

Ahora comprobamos cuando $S_1 \to 0S_1$:
$S \to 0S_1 1 \to 00S_1 1 \to 000S_1 1 \to \mathbf{00000...1}$

Continuamos cuando $S_1 \to 1S_1$:
$S \to 0S_1 1 \Rightarrow 01S_1 1 \Rightarrow 01S_1 11 \Rightarrow \mathbf{0...11111}$

Y por ultimo cuando tenemos que $S_1 \to 1S_1$ o $S_1 \to 0S_1$:
$S \to 0S_1 1 \Rightarrow 01S_1 1 \Rightarrow 01S_1 11 \Rightarrow 0S_1 111 \Rightarrow 000S_1 111 \Rightarrow \mathbf{0...0101...1010...1}$

Por lo tanto tenemos que $L_n = \{0u1 \ tq \ u \in \{0,1\}^*\}$

1.1.2. Apartado B

$S \to S_1 101S_1$ $\qquad\qquad$ $S_1 \to 0S_1|1S_1|\epsilon$

Aplicamos los mismos pasos del ejercicio anterior, empezamos con $S_1 \to \epsilon$:
$S \to S_1 101S_1 \Rightarrow \epsilon 101\epsilon \Rightarrow \mathbf{101}$

Ahora comprobamos cuando $S_1 \to 0S_1$:
$S \to S_1 101S_1 \Rightarrow 0S_1 1010S_1 \Rightarrow 0S_1 010100S_1 \Rightarrow \mathbf{00000...101...00000}$

Continuamos cuando $S_1 \to 1S_1$:
$S \to S_1 101S_1 \Rightarrow 1S_1 1011S_1 \Rightarrow 1S_1 110111S_1 \Rightarrow \mathbf{11111...101...11111}$

Y por ultimo cuando tenemos que $S_1 \to 1S_1$ o $S_1 \to 0S_1$:
$S \to S_1 101S_1 \Rightarrow 1S_1 1011S_1 \Rightarrow 1S_1 110111S_1 \Rightarrow 0S_1 11101110S_1 \Rightarrow 0S_1 0111011100S_1 \Rightarrow$
$\mathbf{0101....101...1010}$

Por lo tanto tenemos que $L_n = \{u101v \ tq \ u, v \in \{0,1\}^*\}$

1.1.3. Apartado C

$S \to 0S_1|S_1 \quad || \quad S_1 \to 1S_10|1S_20 \quad || \quad S_2 \to 0S_21|\epsilon$

Por lo tanto tenemos que $L_n = \{0^i\ 1^j\ 0^k\ 1^k\ 0^j\ tq\ i \in [0,1],\ j \geq 1,\ k \geq 0\}$

1.2. Ejercicio 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.

1.2.1. Apartado A

Palabras que comienzan con la subcadena "10" y acaban en "001".

- **Tipo 2**
  $G = \{S, \{A\}, \{01\}, P, S\}$
  $P = \{S \to 10A001 \quad A \to 1A \quad A \to 0A \quad A \to \epsilon\}$

- **Tipo 3**
  $G = \{S, \{S_1, S_2\}, \{01\}, P, S\}$
  $P = \{S \to 10S_1|S_2 \quad S_1 \to 1S_1|0S_1|S_2 \quad S_2 \to 001\}$

1.2.2. Apartado B

Palabras que tienen 2 o 3 "0".

- **Tipo 2**
  $G = \{S, \{A, B, C\}, \{01\}, P, S\}$
  $P = \{S \to 1S|0S_1|0S_30 \quad S_1 \to 1S_1|0S_2|0S_3 \quad S_2 \to 1S_2|0S_3 \quad S_3 \to \epsilon\}$

- **Tipo 3**
  $G = \{S, \{A, B, C\}, \{01\}, P, S\}$
  $P = \{S \to 1A \quad S \to 0B \quad A \to 1A \quad A \to 0B \quad B \to 1B$
  $B \to 0C \quad C \to 1C \quad C \to 0D \quad C \to \epsilon \quad D \to 1D \quad D \to \epsilon\}$

1.2.3. Apartado C

Palabras que no contienen la subcadena "011".

- **Tipo 3**
  $G = \{S, \{A, B\}, \{01\}, P, S\}$
  $P = \{S \to 1S \quad S \to 0A \quad S \to \epsilon \quad A \to 0A \quad A \to 1B$
  $A \to \epsilon \quad B \to 0S \quad B \to \epsilon\}$

1.3. Ejercicio 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 estas, 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?

$G = \{S, A, \{g, x, t, p\}, P, S\}$
$P = \{S \to AtAgxAj \qquad A \to gA \qquad A \to pA \qquad A \to tA \qquad A \to \epsilon\}$

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

$G = \{S, A, \{g, x, t, p\}, P, S\}$
$P = \{S \to gS|pS|tS_1|gxS_2 \qquad S_1 \to gS_1|pS_1|tS_1|gxS_3 \qquad S_2 \to gS_2|pS_2|tS_3$
$S_3 \to gS_3|pS_3|tS_3|j\}$

2. Practica 2

2.1. Ejercicio 1

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

2.1.1. Apartado A

El lenguaje de las palabras que contienen la subcadena aaa.

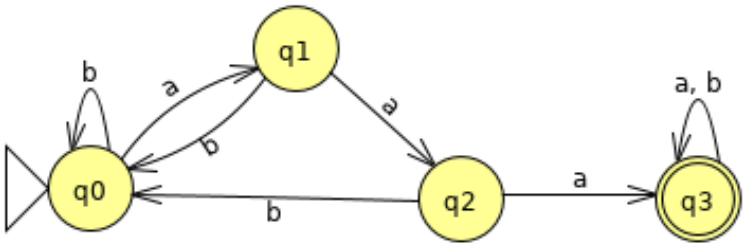


Figura 2.1: Grafo del ejercicio 1A

2.1.2. Apartado B

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

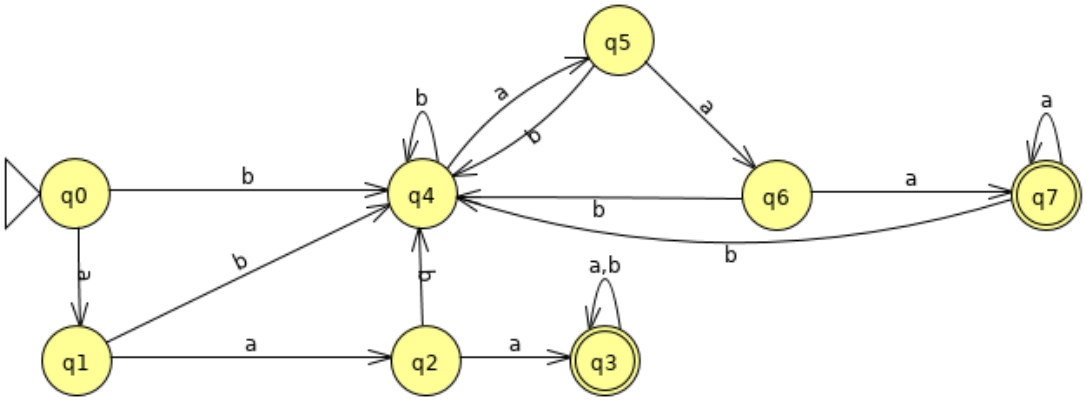


Figura 2.2: Grafo del ejercicio 1B

2.1.3. Apartado C

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

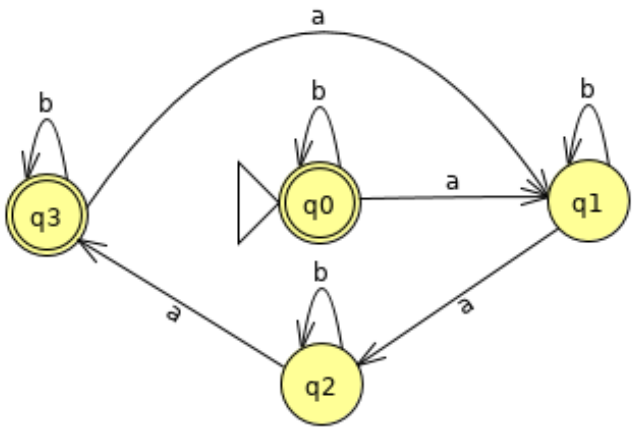


Figura 2.3: Grafo del ejercicio 1C

2.2. Ejercicio 2

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

2.2.1. Apartado A

El lenguaje de las palabras que terminan en aaa.

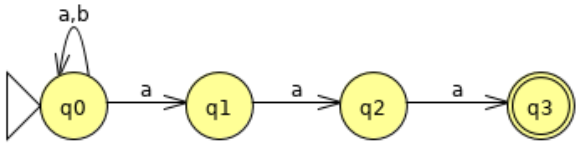


Figura 2.4: Grafo del ejercicio 2A

2.2.2. Apartado B

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

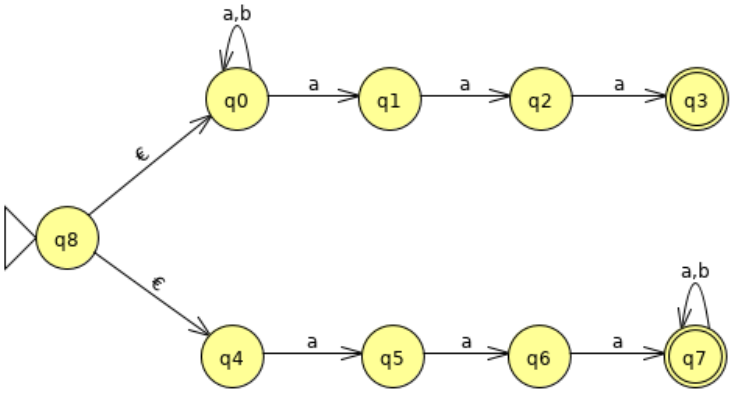


Figura 2.5: Grafo del ejercicio 2B

2.2.3. Apartado 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).

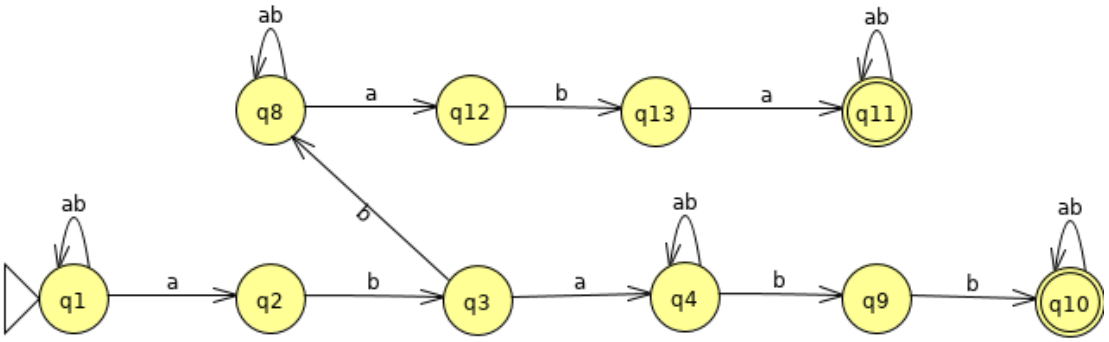


Figura 2.6: Grafo del ejercicio 2C

2.3. Ejercicio 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

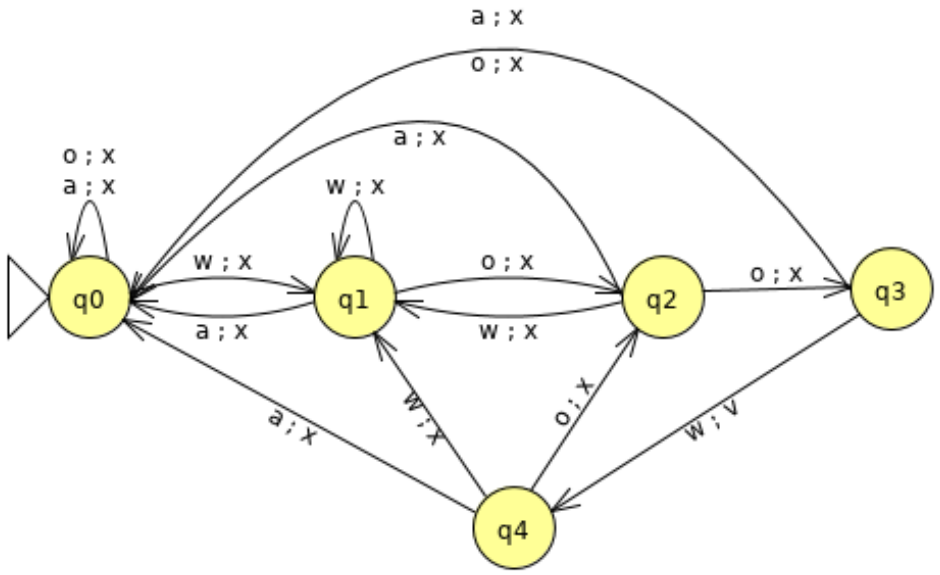
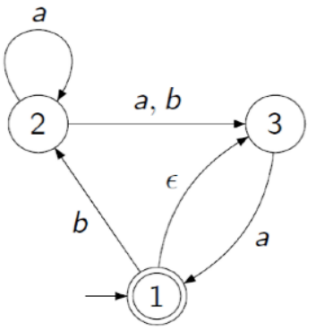


Figura 2.7: Grafo del ejercicio 3

2.4. Ejercicio 4

Obtener un AFD equivalente al AFND siguiente:



Estado/Atributo	a	b
q_1q_3	q_1q_3	q_2
q_2	q_2q_3	q_3
q_2q_3	$q_2q_3q_1$	q_3
$q_2q_3q_1$	$q_2q_3q_1$	q_3q_2
q_3	q_1q_3	$\emptyset$

Tabla 2.1: Caminos, ejercicio 4

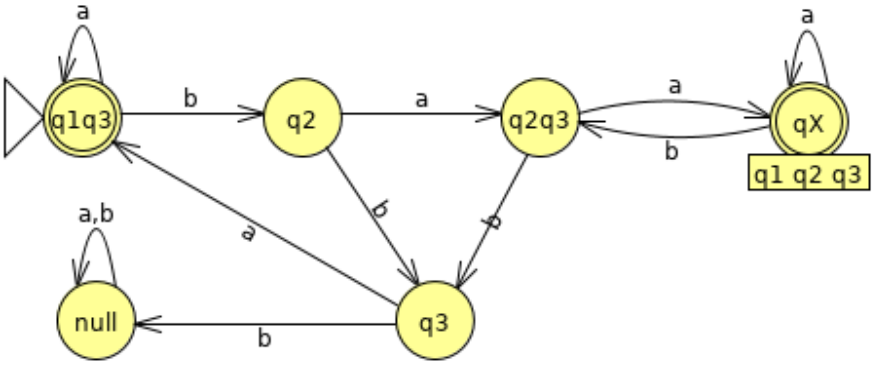


Figura 2.8: Grafo del ejercicio 4

3. Practica 3

3.1. Ejercicio 1

Pensar un problema original de procesamiento de textos. Para la resolución de este problema debe ser apropiado el uso de Lex, o sea, se debe resolver mediante el emparejamiento de cadenas con expresiones regulares y la asociación de acciones a cada emparejamiento. Se presentará una descripción por escrito del problema. Consultar al profesor de prácticas acerca de la complejidad del problema propuesto.

El problema real que voy a resolver, trata en obtener: las cadenas que tienen la información relevante sobre los log obtenidos tras una investigación de clasificación sobre datos no balanceados.

3.2. Ejercicio 2

Resolver el problema propuesto usando Lex.

Código del autómata en lex.

```
        /*------ Seccion de Declaraciones ------*/

%{

        #include <stdio.h>
        #include <string.h>
        #include <stdbool.h>

        int tiempoIntervalo = 0;
        int tiempo = 0;
        int contador = 1;
        char charTimeInit[11];
        char charTimeEnd[11];
        void escribirNombre(char* dato1, int tam);
        void escribirInicio(char* dato1, int tam);
        void escribirFin(char* dato1, int tam);

%}

initTime            ("***␣BEGIN␣OF␣EXPERIMENT␣")
fechaDia            {initTime}([a-zA-Z]{3})
fechaMes            {fechaDia}.([a-zA-Z]{3})
fechaDiaNum         {fechaMes}.([0-9]{2})
hora                {fechaDiaNum}.([0-9]{2}\:[0-9]{2}\:[0-9]{2})
cet                 {hora}.([a-zA-Z]{3})
initProceso         {cet}.([0-9]{4})

initNombre           ("The␣name␣is:␣")
nombreProceso       {initNombre}([a-zA-Z]+).([a-zA-Z]+)*
```

```
endTime             ("*** END OF EXPERIMENT ")
endfechaDia         {endTime}([a-zA-Z]{3})
endfechaMes         {endfechaDia}.([a-zA-Z]{3})
endfechaDiaNum      {endfechaMes}.([0-9]{2})
endhora             {endfechaDiaNum}.([0-9]{2}\:[0-9]{2}\:[0-9]{2})
endcet              {endhora}.([a-zA-Z]{3})
endProceso          {endcet}.([0-9]{4})

%%

        /*----- Seccion de Reglas -----*/

{initProceso}               { escribirInicio(yytext, yyleng); }
{nombreProceso}             { escribirNombre(yytext, yyleng); }
{endProceso}                { escribirFin(yytext, yyleng); }
.                           {}
\n                          {}

%%

        /*----- Seccion de Procedimientos -----*/

int main (int argc, char *argv[]){
        if (argc == 2)  {
                yyin = fopen (argv[1], "rt");

                if (yyin == NULL)     {
                        printf ("El fichero %s no se puede abrir\n",
                            argv[1]);
                        exit (-1);
                }
        }
        else yyin = stdin;

        yylex ();
        printf ("\n");
        printf ("\n -->  Tiempo Final: %is", tiempo);
        printf ("\n");

        return 0;
}

void escribirInicio(char* dato1, int tam){
        char* tipo = "Inicio: ";
        char* aux;
        char letra;
        int tamanio = 32;

        printf ("\n## Inicio Proceso: %i ## \n", contador);
        printf ("\n%s",tipo);

        aux = dato1;
```

```
        for(int i=24; i < strlen(dato1); i++){
                letra = dato1[i];
                printf ("%c",letra);
        }

        for(int i=0; i < 11; i++){
                letra = aux[tamanio];
                charTimeInit[i] = letra;
                tamanio++;
        }
}

void escribirFin(char* dato1, int tam){
        char* tipo = "Fin: ";
        char letra;
        char* aux;
        int tamanio = 30;

        if (tieneNombre != true){
                printf ("\nNombre: No tiene nombre. ");
        }

        printf ("\n%s",tipo);

        aux = dato1;

        for(int i=22; i < strlen(dato1); i++){
                letra = dato1[i];
                printf ("%c",letra);
        }

        for(int i=0; i < 11; i++){
                letra = aux[tamanio];
                charTimeEnd[i] = letra;
                tamanio++;
        }

        tiempoTotal();

        printf ("\n \n## Fin Proceso: %i ## \n \n
           ####################################\n", contador);
        \\

        tieneNombre = false;

        contador++;
}

void escribirNombre(char* dato1, int tam){
        char* tipo = "Nombre: ";
        char letra;

        printf ("\n%s",tipo);
```

```
        for(int i=13; i < strlen(dato1); i++){
                letra = dato1[i];
                printf ("%c",letra);
        }

        tieneNombre = true;
}

void tiempoTotal(){
        int tamanio = 0, tiempoTotalInit = 0, tiempoTotalEnd = 0;
        char CdiaInit[2], CdiaEnd[2], ChoraInit[2], ChoraEnd[2],
            CminInit[2], CminEnd[2], CsegInit[2], CsegEnd[2];

        for(int i = 0; i < 2; i++){
                CdiaInit[i] = charTimeInit[tamanio];
                CdiaEnd[i] = charTimeEnd[tamanio];
                tamanio++;
        }

        tamanio++;

        for(int i = 0; i < 2; i++){
                ChoraInit[i] = charTimeInit[tamanio];
                ChoraEnd[i] = charTimeEnd[tamanio];
                tamanio++;
        }

        tamanio++;

        for(int i = 0; i < 2; i++){
                CminInit[i] = charTimeInit[tamanio];
                CminEnd[i] = charTimeEnd[tamanio];
                tamanio++;
        }

        tamanio++;

        for(int i = 0; i < 2; i++){
                CsegInit[i] = charTimeInit[tamanio];
                CsegEnd[i] = charTimeEnd[tamanio];
                tamanio++;
        }

        tiempoTotalInit = atoi(CdiaInit)*24*60*60;
        tiempoTotalInit += atoi(ChoraInit)*60*60;
        tiempoTotalInit += atoi(CminInit)*60;
        tiempoTotalInit += atoi(CsegInit);

        tiempoTotalEnd = atoi(CdiaEnd)*24*60*60;
        tiempoTotalEnd += atoi(ChoraEnd)*60*60;
        tiempoTotalEnd += atoi(CminEnd)*60;
        tiempoTotalEnd += atoi(CsegEnd);

        tiempoIntervalo = tiempoTotalEnd - tiempoTotalInit;
```

```
            tiempo += tiempoIntervalo;

            printf ("\n\t\t\t--> Tiempo Proceso: %is", tiempoIntervalo);
    }
```

Enlaces de descarga:

- Automata lex.
- Prueba 1
- Prueba 2
- Resultado

3.3. Ejercicio 3

Realizar un documento presentando el problema y la solución con Lex.

El principal problema que tiene el extraer esta información sin ayuda de los autómatas es la cantidad de tiempo y lo compleja que encontrarla entre mas de 45000 de líneas de código.

Solución obtenida con el autómatas en diversos casos:

- **Prueba 20**

  ## Inicio Proceso: 1 ##

  Inicio: Fri Jan 06 00:51:34 CET 2017
  Nombre: IterativePartitioningFilter
  Fin: Fri Jan 06 01:14:07 CET 2017
  –>Tiempo Proceso: 1353s

  ## Fin Proceso: 1 ##

  ##

  ## Inicio Proceso: 2 ##

Inicio: Fri Jan 06 01:14:07 CET 2017
Nombre: IterativePartitioningFilter
Fin: Fri Jan 06 01:36:40 CET 2017
–>Tiempo Proceso: 1353s

Fin Proceso: 2

##

.....

##

Inicio Proceso: 31

Inicio: Fri Jan 06 03:29:08 CET 2017
Nombre: No tiene nombre.
Fin: Fri Jan 06 03:29:08 CET 2017
–>Tiempo Proceso: 0s

Fin Proceso: 31

##

–>Tiempo Final: 9454s

4. Practica 4

4.1. Ejercicio 1

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

4.1.1. Apartado A

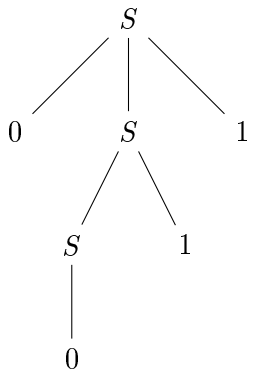
$S \to 01S \mid 010S \mid 101S \mid \epsilon$

Como por norma las gramáticas de tipo 3 no son ambiguas podemos decir que esta al serlo no puede ser ambigua.

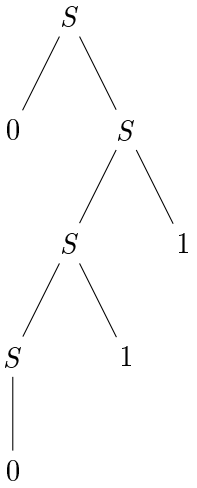
4.1.2. Apartado B

$S \to 0S1 \mid S1 \mid 0S \mid 0$

Palabra 0011



Palabra 0011



Como tenemos dos grafos que dan la misma palabra podemos decir que la gramática es ambigua. Y como podemos encontrar la gramática $S \to S1|S0|0$, diremos también que es inherentemente ambigua.

4.1.3. Apartado C

$S \to A1B$ $\quad$ $A \to 0A \mid \epsilon$ $\quad$ $B \to 0B \mid 1B \mid \epsilon$

Como podemos ver a simple vista, las S generan A por la izquierda y B por la derecha por lo que nunca podremos encontrar dos arboles iguales. Esto nos dice que la gramática no es ambigua y por lo tanto el lenguaje no puede ser inherentemente ambiguo.

4.2. Ejercicio 2

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

$S \to moA$ $\quad$ $S \to cI$ $\quad$ $A \to dEs$ $\quad$ $A \to jBI$

$B \to bb$ $\quad$ $B \to D$ $\quad$ $E \to elO$ $\quad$ $E \to Perl$

$D \to de$ $\quad$ $C \to c$ $\quad$ $J \to kC$ $\quad$ $I \to fl$

$O \to o$ $\quad$ $P \to ola$

En primer lugar, eliminaremos las variables que no llegan a las palabras T^* y las producciones donde estas estén:

$V_t = \{C\}$

$S \to moA$ $\quad$ $S \to cI$ $\quad$ $A \to dEs$ $\quad$ $A \to jBI$

$B \to bb$ $\quad$ $B \to D$ $\quad$ $E \to elO$ $\quad$ $E \to Perl$

$D \to de$ $\quad$ $C \to c$ $\quad$ $J \to kC$ $\quad$ $I \to fl$

$O \to o$ $\quad$ $P \to ola$

$V_t = \{C, O\}$

$S \to moA$ $S \to cI$ $A \to dEs$ $A \to jBI$

$B \to bb$ $B \to D$ $E \to elO$ $E \to Perl$

$D \to de$ $C \to c$ $J \to kC$ $I \to fl$

$O \to o$ $P \to ola$

$V_t = \{C, O, B\}$

$S \to moA$ $S \to cI$ $A \to dEs$ $A \to jBI$

$B \to bb$ $B \to D$ $E \to elO$ $E \to Perl$

$D \to de$ $C \to c$ $J \to kC$ $I \to fl$

$O \to o$ $P \to ola$

$V_t = \{C, O, B, D\}$

$S \to moA$ $S \to cI$ $A \to dEs$ $A \to jBI$

$B \to bb$ $B \to D$ $E \to elO$ $E \to Perl$

$D \to de$ $C \to c$ $J \to kC$ $I \to fl$

$O \to o$ $P \to ola$

$V_t = \{C, O, B, D, J\}$

$S \to moA$ $S \to cI$ $A \to dEs$ $A \to jBI$

$B \to bb$ $B \to D$ $E \to elO$ $E \to Perl$

$D \to de$ $C \to c$ $J \to kC$ $I \to fl$

$O \to o$ $P \to ola$

$V_t = \{C, O, B, D, J, E\}$

$S \to moA$	$S \to cI$	$A \to dEs$	$A \to jBI$
$B \to bb$	$B \to D$	$E \to elO$	$E \to Perl$
$D \to de$	$C \to c$	$J \to kC$	$I \to fl$
$O \to o$	$P \to ola$		

$V_t = \{C, O, B, D, J, E, A\}$

$S \to moA$	$S \to cI$	$A \to dEs$	$A \to jBI$
$B \to bb$	$B \to D$	$E \to elO$	$E \to Perl$
$D \to de$	$C \to c$	$J \to kC$	$I \to fl$
$O \to o$	$P \to ola$		

$V_t = \{C, O, B, D, J, E, A, S\}$

$S \to moA$	$S \to cI$	$A \to dEs$	$A \to jBI$
$B \to bb$	$B \to D$	$E \to elO$	$E \to Perl$
$D \to de$	$C \to c$	$J \to kC$	$I \to fl$
$O \to o$	$P \to ola$		

$V_{resultado} = V - V_t = \{P, I\}$

Ahora continuamos eliminando los estados que no se pueden alcanzar desde el estado inicial S, y las producciones que estos tienen:

{ v_S: variables obtenidas T_S: símbolos terminales J: variables por analizar{S} }

$S \to moA$ $S \to cI$ $A \to dEs$ $A \to jBI$

$B \to bb$ $B \to D$ $E \to elO$ $E \to Perl$

$D \to de$ $C \to c$ $J \to kC$ $O \to o$

{ v_S: { S } T_S: { m,o } J: { A } }

$S \to moA$ $S \to cI$ $A \to dEs$ $A \to jBI$

$B \to bb$ $B \to D$ $E \to elO$ $E \to Perl$

$D \to de$ $C \to c$ $J \to kC$ $O \to o$

{ v_S: { S,A } T_S: { m,o,d,s } J: { E } }

$S \to moA$ $S \to cI$ $A \to dEs$ $A \to jBI$

$B \to bb$ $B \to D$ $E \to elO$ $E \to Perl$

$D \to de$ $C \to c$ $J \to kC$ $O \to o$

{ v_S: { S,A,E } T_S: { m,o,d,s,e,l } J: { O } }

$S \to moA$ $S \to cI$ $A \to dEs$ $A \to jBI$

$B \to bb$ $B \to D$ $E \to elO$ $E \to Perl$

$D \to de$ $C \to c$ $J \to kC$ $O \to o$

{ v_S: { S,A,E,O } T_S: { m,o,d,s,e,l,o } J: { } }

Por lo tanto las produciones que nos quedan son:

$S \to moA$ $A \to dEs$ $E \to elO$ $O \to o$

Estas produciones proporcionan la siguiente palabra:

$S \to moA \Rightarrow modEs \Rightarrow modelOs \Rightarrow modelos$

4.3. Ejercicio 3

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

$S \to XYZ$ $S \to XYz$ $X \to xxX$ $X \to \epsilon$

$Y \to yyY$ $Y \to \epsilon$ $Z \to yxZ$ $Z \to X$

En el primer paso **quitamos las producciones nulas**.

$S \to XYZ$ $S \to XYz$ $X \to xxX$ $X \to \epsilon$

$Y \to yyY$ $Y \to \epsilon$ $Z \to yxZ$ $Z \to X$

Vemos que variables son anulables para extraerlas, y obtenemos que:

$X \to \epsilon$
$Z \to X \to \epsilon$
$Y \to \epsilon$
$S \to XYZ \to \epsilon$

Teniendo en cuenta que S es anulable la palabra vacía podría crearse con esta gramática.
$H = \{X, Y, Z, S\}$

Parte dos del proceso, **añadir producciones**.

En la tabla señalamos en **rojo** la parte que se elimina o no se edita y quedaran fuera en el futuro:

	S → XYZ	**X → xxX**	**Y → yyY**	**S → XYz**	**Z → yxZ**	**Z → X**
delet X	**S → YZX**	**X → xxX**	**Y → yyY**	**S → XYz**	**Z → yxZ**	**Z → X**
delet Y	**S → XYZ**	**X → xxX**	**Y → yyY**	**S → XYz**	**Z → yxZ**	**Z → X**
delet Z	**S → XYZ**	**X → xxX**	**Y → yyY**	**S → XYz**	**Z → yxZ**	**Z → X**
delet XY	**S → XYZ**	**X → xxX**	**Y → yyY**	**S → XYz**	**Z → yxZ**	**Z → X**
delet YZ	**S → XYZ**	**X → xxX**	**Y → yyY**	**S → XYz**	**Z → yxZ**	**Z → X**
delet XZ	**S → XYZ**	**X → xxX**	**Y → yyY**	**S → XYz**	**Z → yxZ**	**Z → X**

Por lo tanto limpiando los eliminados tendríamos los siguientes valores:

	S → XYZ	**X → xxX**	**Y → yyY**	**S → XYz**	**Z → yxZ**	**Z → X**
delet X	**S → YZ**	**X → xx**		**S → Yz**		
delet Y	**S → XZ**		**Y → yy**	**S → Xz**		
delet Z	**S → XY**				**Z → yx**	
delet XY	**S → Z**	**X → xx**	**Y → yy**	**S → z**		
delet YZ	**S → X**		**Y → yy**	**S → Xz**	**Z → yx**	
delet XZ	**S → Y**	**X → xx**		**S → Yz**	**Z → yx**	

Eliminamos los repetidos que son los de color azul:

	S → XYZ	**X → xxX**	**Y → yyY**	**S → XYz**	**Z → yxZ**	**Z → X**
delet X	**S → YZ**	**X → xx**		**S → Yz**		
delet Y	**S → XZ**		**Y → yy**	**S → Xz**		
delet Z	**S → XY**				**Z → yx**	
delet XY	**S → Z**	**X → xx**	**Y → yy**	**S → z**		
delet YZ	**S → X**		**Y → yy**	**S → Xz**	**Z → yx**	
delet XZ	**S → Y**	**X → xx**		**S → Yz**	**Z → yx**	

Limpiamos la tabla y obtenemos los siguientes valores:

	S → XYZ	**X → xxX**	**Y → yyY**	**S → XYz**	**Z → yxZ**	**Z → X**
delet X	**S → YZ**	**X → xx**		**S → Yz**		
delet Y	**S → XZ**		**Y → yy**	**S → Xz**		
delet Z	**S → XY**				**Z → yx**	
delet XY	**S → Z**			**S → z**		
delet YZ	**S → X**					
delet XZ	**S → Y**					

El siguiente paso el eliminar las unidades unitarias. Como tenemos que:

$$S \to Z \qquad S \to Y \qquad S \to X \qquad Z \to X$$

Como podemos ver, Z esta a la derecha y a la izquierda por lo tanto tendríamos que añadir (S, X) pero como ya esta no es necesario, ni necesitamos meterla. Ya que si luego tendríamos que eliminarla cuando limpiemos.

Por lo que al final nos queda que: { (S,X) (S,Y) (S,Z) (Z,X)}

S → XYZ	**X → xxX**	**Y → yyY**	**S → XYz**	**Z → yxZ**
S → YZ	**X → xx**		**S → Yz**	
S → XZ		**Y → yy**	**S → Xz**	
S → XY				**Z → yx**
	$S \to xxX$	$S \to yyY$		$S \to yxZ$
	$S \to xx$	$S \to yy$		$S \to yx$
	$Z \to xxX$			
	$Z \to xxX$			

4.4. Ejercicio 4

Pasar la siguiente gramática a forma normal de Greibach:

$S \to a \mid CD \mid CS$ $A \to a \mid b \mid SS$ $C \to a$ $D \to AS$

Primero renombramos las variables quedando lo siguiente:

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

$A_1 \to a \mid A_2A_3 \mid A_2A_1$ $A_2 \to a$ $A_3 \to A_4A_1$
$A_4 \to a \mid b \mid A_1A_1$

Ahora continuamos con la transformación pre-Greibach, para ello nos fijamos que todas las producciones han de estar de la forma $i < j$.

$A_4 \to A_1A_1 \Rightarrow A_4 \to aA_1$
$A_4 \to A_2A_3A_1 \Rightarrow A_4 \to aA_3A_1$
$A_4 \to A_2A_1A_1 \Rightarrow A_4 \to aA_1A_1$

$A_1 \to a \mid A_2A_3 \mid A_2A_1$ $A_2 \to a$ $A_3 \to A_4A_1$
$A_4 \to a \mid b \mid aA_1 \mid aA_3A_1 \mid aA_1A_1$

Ahora pasamos todas las formas formales de Greibach.

$A_3 \to A_4A_1$
$\Rightarrow A_3 \to aA_1 \Rightarrow A_3 \to bA_1 \Rightarrow A_3 \to aA_1A_1 \Rightarrow A_3 \to aA_3A_1A_1 \Rightarrow A_3 \to aA_1A_1A_1$

$A_3 \to A_2A_3$
$\Rightarrow A_1 \to aA_3$

$A_1 \to A_2 A_1$
$\Rightarrow A_1 \to aA_1$

Entonces nuestra gramática quedaría de la siguiente forma:

$A_1 \to a \mid aA_3 \mid aA_1$
$A_2 \to a$
$A_3 \to aA_1 \mid bA_1 \mid aA_1A_1 \mid aA_3A_1A_1 \mid aA_1A_1A_1$
$A_4 \to a \mid b \mid aA_1 \mid aA_3A_1 \mid aA_1A_1$