Modelos de Computación(2015-2016)
Grado en Ingeniería Informática
Universidad de Granada

Prácticas Modelos de Computación

Antonio de la Vega Jiménez

8 de octubre de 2016

Índice

Índice de figuras

1. Práctica 1

Explica con tus propias palabras que lenguaje genera la siguiente gramática: G=(V, T, P, S) donde V = S, A, B y T = a, b y las reglas de producción son:

$$\begin{array}{llll} S \to aB & S \to bA & A \to a & A \to aS \\ A \to bAA & B \to b & B \to bS & B \to aBB \end{array}$$

Esta gramática genera el lenguaje que esta formado por las cadenas de "a" y "b" que cumplen la condición de que el número de apariciones "a" es igual al número de apariciones "b". Como se pude ver en las producciones, al principio se puede añadir una "a" o una "b" , pero no se añaden solos, sino que se añaden junto a una variable, si se añade una "a" se añade junto a la variable "B" y si se añade una "b" se añade junto a la variable "A". La variable "A" y la variable "B" siempre acaban siendo sustituidas por una "a" y una "b" respectivamente, es decir se podría interpretar como que la variable "A" significa añadir una "a" y la variable "B" añadir un "b" , por ello al principio la "a" se añade junto a una "B" (y la "b" junto a una "A") de forma que finalmente el número de ambos símbolos terminales será el mismo. Por ejemplo si empezamos con $S \to aB$ añadimos el símbolo terminal "a" a la cadena y la variable "B", y como para quitar la "B" se acaba poniendo un "b", el número de símbolos terminales "a" y "b" esta equilibrado. En caso de querer añadir otra "a", hacemos uso de la producción $B \to aBB$, esta producción añade una"b" de más , por lo que al usar las dos producciones anteriores quedaría la cadena $aaBB$ y como "B" siempre se acaba sustituyendo por "b" la cadena sigue cumpliendo la propiedad de que el número de apariciones de "a" es igual al número de apariciones de "b". Los símbolos "a" y "b" pueden ir apareciendo en cualquier orden ya que las producciones $B \to bS$ *y* $A \to aS$ permiten cambiar entre un símbolo y otro manteniendo la propiedad de la cadena. En caso de que se comience por $S \to bA$ ocurre exactamente lo mismo.

2. Práctica 2

Determinar si G=(S, A, B, a, b ,c, d, P, S) donde P es:

$$\begin{array}{lll} S \to AB & A \to Ab & A \to a \\ B \to cB & B \to d & \end{array}$$

genera un lenguaje de tipo 3 (Regular).

En primer lugar hay que mirar que lenguaje genera la gramática. Mirando las producciones se puede observar que la cadena tiene que empezar por AB una vez tenemos eso a partir de "A" se pueden generar dos cosas , una "a" o una cadena de "b" que comienza con una "a" ya que la "A" solo se puede quitar añadiendo una "a", por lo tanto a partir de "A" se obtiene $\{ab^i\} : i \in \mathbb{N}$. A partir de la "B" se puede generar una

"d" o una cadena de "c" con una una "d" al final. Por lo tanto el lenguaje generado es $\{ab^i c^j d\} : i, j \in \mathbb{N}$. Para ver si es un lenguaje de tipo 3 tenemos que encontrar una gramática de tipo 3 que genere lo mismo que la anterior, es decir una gramatical que genere cadenas que comienzan por una "a" luego tengan todas las "b" que se quieran, después todas las "c" que se quiera y finalmente una "d" y que cumpla las condiciones "$A \to uB$ *o* $A \to u$ *donde* $u \in T*$ *y* $A, B \in V$", por ejemplo nos vale la gramática $A \to aB,\ B \to bB,\ B \to C,\ C \to cC,\ C \to c$. Como en la jerarquía de Chomsky se dice que una gramática que cumple las condiciones anteriores genera un lenguaje de tipo 3 y la gramática que hemos creado cumple esas condiciones y genera el lenguaje $\{ab^i c^j d\} : i, j \in \mathbb{N}$, entonces el lenguaje es de tipo 3.

3. Práctica 3

Crear el autómata finito determinista que reconoce cadenas que comienzan con cualquier número de 1 y 0 (pudiendo no haber ninguno), luego contiene la subcadena ***0110****, después tienen cualquier número de 1 y 0 (pudiendo no haber ninguno) , tras esto contiene la subcadena* ***1000*** *y finalmente tiene cualquier número de 0 y 1 (pudiendo no haber ninguno).*

En primer lugar creamos el autómata finito no determinista mostrado en la figura 3.1 que acepta las cadenas con la estructura que deseamos (cadenas que empiezan con cualquier cosa, luego tiene la subcadena 0110 luego cualquier cosa , después la subcadena 1000 y finalmente cualquier cosa).

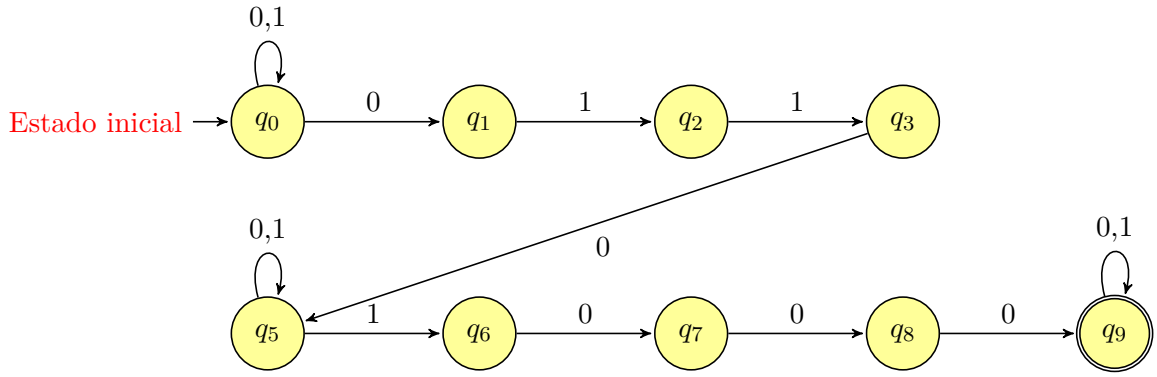

Figura 3.1: Autómata finito no determinista.

Una vez que tenemos el autómata finito no determinista (figura 3.1) obtenemos el autómata finito determinista mostrado en la figura 3.2 y ya tenemos un autómata finito determinista que reconoce cadenas con la estructura deseada.

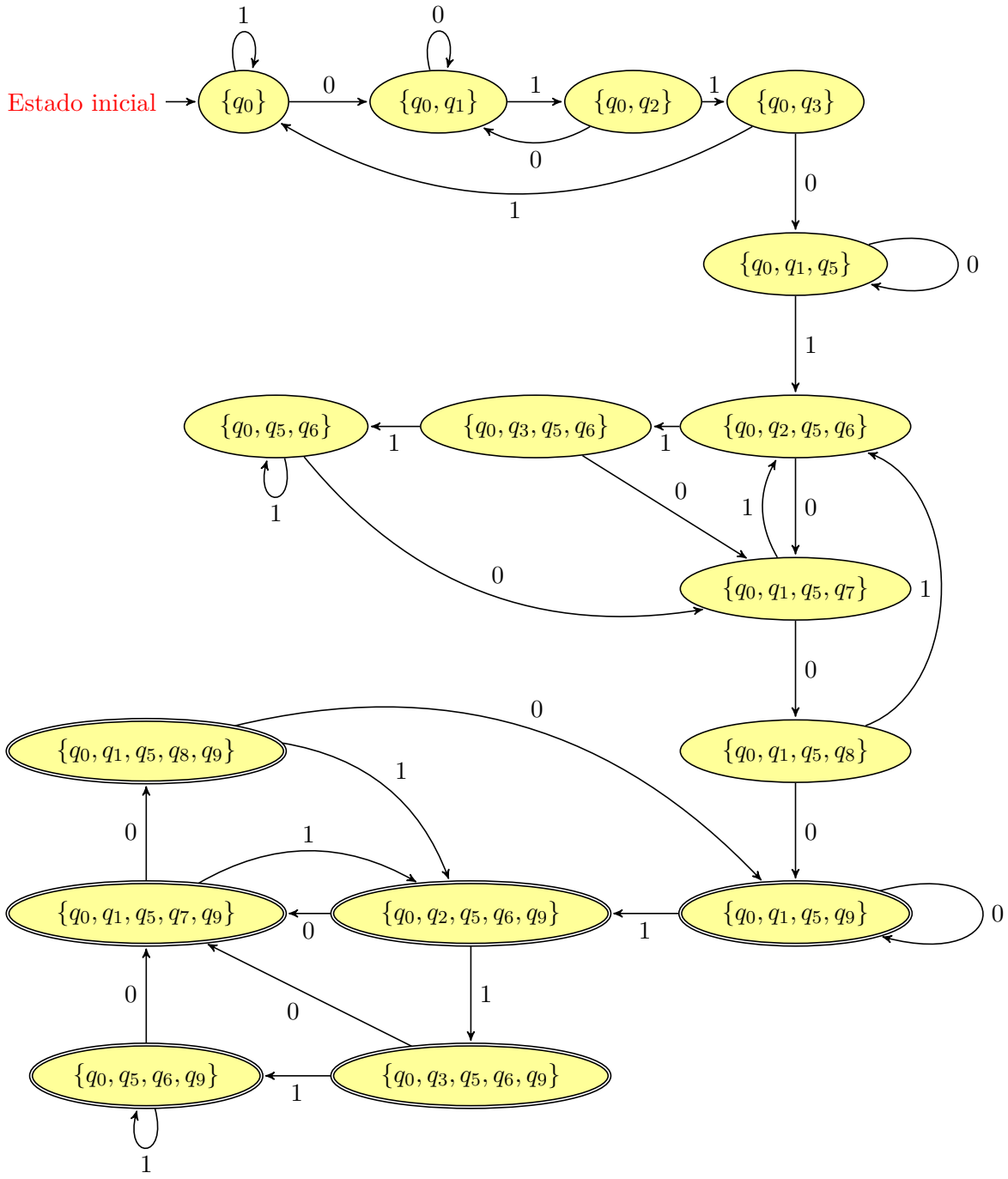

Figura 3.2: Autómata finito determinista.

4. Práctica 4

Crear un fichero Lex con el código mostrado en el tema 2 y comprobar que funciona correctamente.

Para esta practica he usado el fichero Lex del tema 2 con una pequeña modificación para que ademas de reconocer números que usan un "." para los decimales , también reconozca a los que usan un ",". El fichero Lex es el mostrado en la figura 4.1. Siguiendo el proceso mostrado en la figura 4.2 he comprobado que compila correctamente y además haciendo uso del archivo mostrado en la figura 4.3 he comprobado que funciona adecuadamente.

```
car         [a-zA-Z]
digito      [0-9]
signo       (\-|\+)
suc         ({digito}+)
enter       ({signo}?{suc})
real1       ({enter}\.{digito}*)
real2       ({signo}?\.{suc})
real3       ({enter}\,{digito}*)
real4       ({signo}?\,{suc})

    int ent=0, real=0, ident=0, sumaent=0;
%%
    int i;
{enter}                         {ent++; sscanf(yytext," %d",&i); sumaent += i;
                                printf("Numero entero %s\n",yytext);}
({real1}|{real2}|{real3}|{real4})   {real++; printf("Num. real %s\n",yytext);}
{car}({car}|{digito})*          {ident++; printf("Var. ident. %s\n",yytext);}
.|\n {;}

%%

yywrap()
  {printf("Numero de Enteros %d, reales %d, ident %d,\n
          Suma de Enteros %d \n",ent,real,ident,sumaent); return 1;}
```

Figura 4.1: Fichero lex.

```
[antonio@antonio-pc Desktop]$ flex practica4
[antonio@antonio-pc Desktop]$ gcc lex.yy.c -o prac4 -lfl
[antonio@antonio-pc Desktop]$ ./prac4 < test > resultado
[antonio@antonio-pc Desktop]$ cat resultado
Num. real 123.2343
Num. real +.2323
Var. ident. A1234
Numero entero 33
Num. real 12,3
Numero entero 21
Num. real 7,7
Var. ident. A234
Var. ident. Abcde1
Num. real +1.125
Numero de Enteros 2, reales 5, ident 3,
 Suma de Enteros 54
[antonio@antonio-pc Desktop]$
```

Figura 4.2: Proceso de compilación y ejecución del fichero lex.

```
1  123.2343 +.2323 A1234
2  33 12,3 21 7,7 A234
3  Abcde1 +1.125
```

Figura 4.3: Fichero para realizar la prueba.

5. Práctica 5

Dado el lenguaje $L = \{0u1 \ / \ u \in \{0,1\}^*\}$ *obtener la expresión regular, el autómata finito determinista y las gramáticas lineales por la izquierda y por la derecha.*

En primer lugar obtenemos la expresión regular que se corresponde con ese lenguaje, que seria la siguiente:

$$0(0+1)^*1$$

Haciendo uso de la expresión regular podemos obtener el autómata finito no determinista con transiciones nulas mostrado en la figura 5.1 que reconoce cadenas con esa estructura.

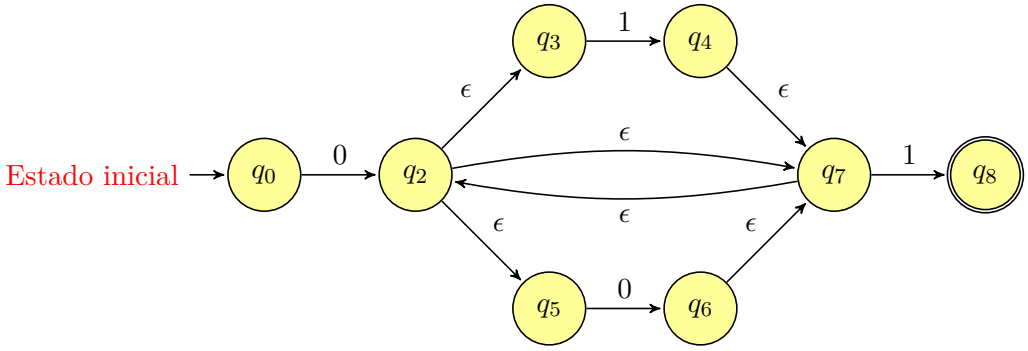

Figura 5.1: Autómata finito no determinista con transiciones nulas.

A partir del autómata finito no determinista con transiciones nulas (figura 5.1) obtenemos el autómata finito determinista de la figura 5.2.

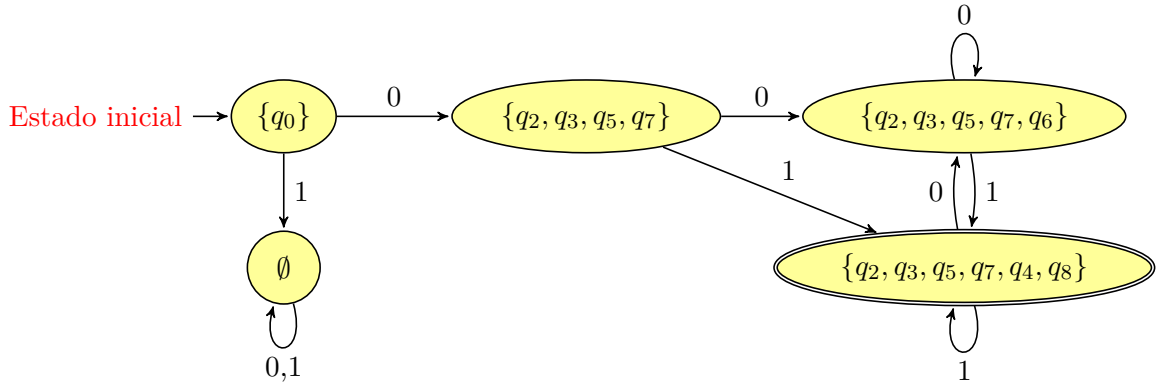

Figura 5.2: Autómata finito determinista

Para obtener la gramática lineal por la derecha hacemos uso del autómata anterior (figura 5.2) pero cambiamos los nombres de los nodos para que las producciones queden con nombres mas simples. El autómata resultante es el mostrado en la figura 5.3.

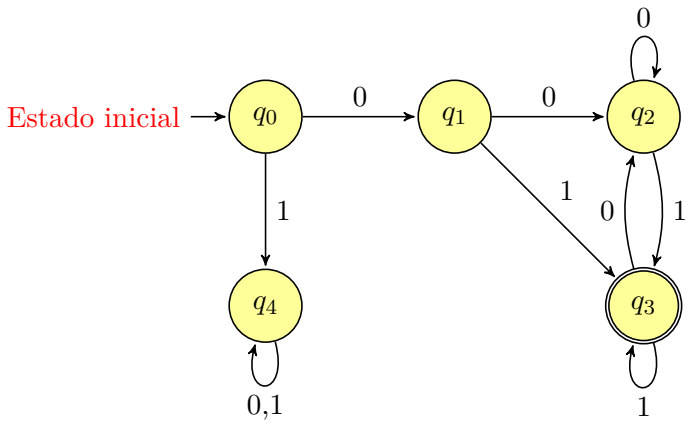

Figura 5.3: Autómata 5.2 simplificado.

Haciendo uso del autómata 5.3 pasamos a obtener la gramática, que queda como se muestra a continuación (el estado inicial es q_0):

$$
\begin{array}{llll}
q_0 \to 0q_1 & q_1 \to 0q_2 & q_1 \to 1q_3 & q_2 \to 0q_2 \\
q_2 \to 1q_3 & q_3 \to 0q_2 & q_3 \to 1q_3 & q_3 \to \varepsilon \\
q_0 \to 1q_4 & q_4 \to 1q_4 & q_4 \to 0q_4 &
\end{array}
$$

Para obtener la gramática lineal por la izquierda, en primer lugar invertimos el autómata 5.3 quedando como se muestra en la figura 5.4.

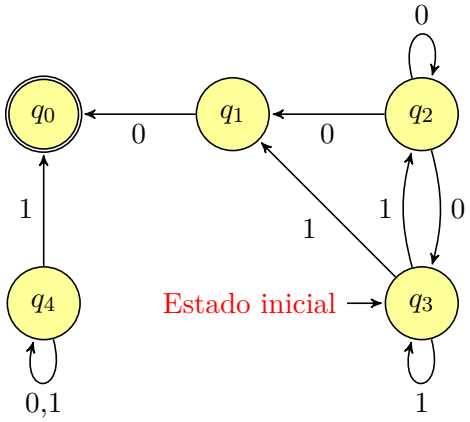

Figura 5.4: Autómata 5.3 invertido.

Como al invertir el autómata ha dejado de ser determinista hay que volverlo a convertir en determinista como se muestra a continuación:

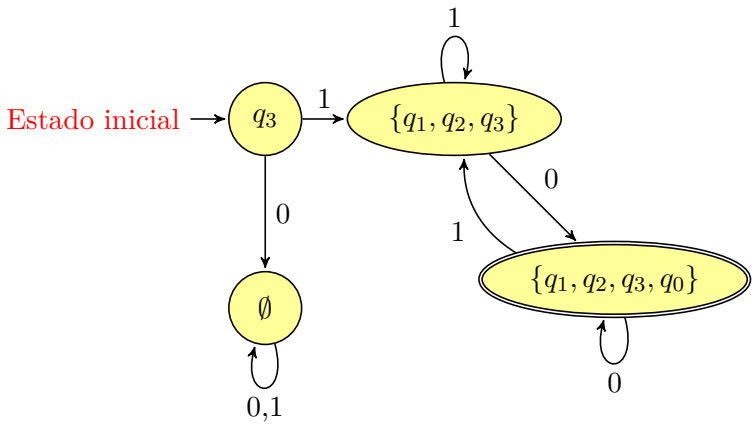

Figura 5.5: Autómata determinista obtenido del autómata 5.4

Ahora simplificamos los nombres de los estados para que los nombres de las producciones sean mas sencillas:

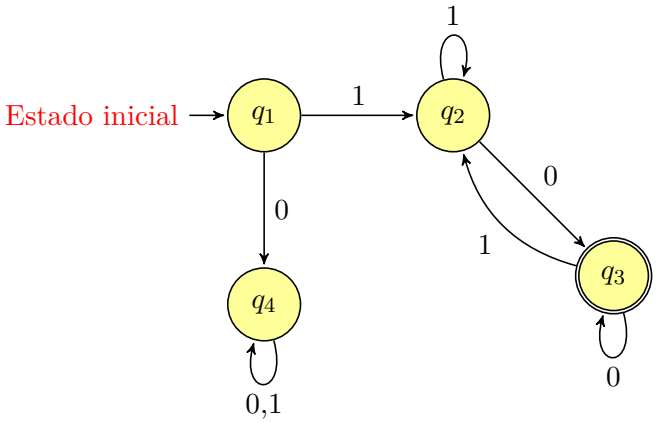

Figura 5.6: Autómata 5.5 simplificado.

A partir del autómata 5.6 obtenemos la siguiente gramática (el estado inicial es q_1):

$$q_1 \to 0q_4 \qquad q_1 \to 1q_2 \qquad q_2 \to 1q_2 \qquad q_2 \to 0q_3$$
$$q_3 \to 0q_3 \qquad q_3 \to 1q_2 \qquad q_3 \to \varepsilon$$

Por último invertimos la parte derecha de las producciones y obtenemos la gramática lineal por la izquierda mostrada a continuación (el estado inicial es q_1):

$$q_1 \to q_40 \qquad q_1 \to q_21 \qquad q_2 \to q_21 \qquad q_2 \to q_30$$

$q_3 \to q_30 \qquad q_3 \to q_21 \qquad q_3 \to \varepsilon$

6. Práctica 6

Proponer una gramática libre del contexto que contenga alguna producción nula y alguna producción unitaria. Eliminar las producciones nulas y unitarias y pasar a la forma normal de Chomsky.
Para esta practica voy a usar la siguiente gramática:

$$\begin{aligned}
&S \to A \mid BCa \mid aDcd \\
&A \to aAb \mid c \\
&B \to CD \mid Ad \mid \varepsilon \\
&C \to Ca \mid Bb \mid c \\
&D \to aDd \mid Dd \mid \varepsilon
\end{aligned}$$

En primer lugar buscamos y eliminamos las producciones nulas, tras esto la gramática queda de la siguiente forma:

$$\begin{aligned}
&S \to A \mid BCa \mid aDcd \mid Ca \mid acd \\
&A \to aAb \mid c \\
&B \to CD \mid Ad \mid \cancel{\varepsilon} \mid C \\
&C \to Ca \mid Bb \mid c \mid b \\
&D \to aDd \mid Dd \mid \cancel{\varepsilon} \mid ad \mid d
\end{aligned}$$

$$H = \{B, D, S\}$$

El siguiente paso es eliminar las producciones unitarias, tras realizar esto, la gramática queda como se muestra a continuación:

$$\begin{aligned}
&S \to \cancel{A} \mid BCa \mid aDcd \mid Ca \mid acd \mid aAb \mid c \\
&A \to aAb \mid c \\
&B \to CD \mid Ad \mid \cancel{C} \mid Ca \mid Bb \mid c \mid b \\
&C \to Ca \mid Bb \mid c \mid b \\
&D \to aDd \mid Dd \mid ad \mid d
\end{aligned}$$

$$H = \{B, D, S\}$$

Una vez que ya tenemos la gramática sin producciones nulas ni unitarias, la pasamos a la forma normal de Chomsky en la que todas las producciones tienen la forma $A \to BC$ *o* $A \to a$.El primer paso para pasar a la forma normal de Chomsky es eliminar los símbolos terminales en las producciones que no sean $A \to a$, tras hacer esto la gramática queda de la siguiente forma:

$$
\begin{array}{ll}
X_a \to a & S \to \ BCX_a \mid X_aDX_cX_d \mid CX_a \mid X_aX_cX_d \mid X_aAX_b \mid X_c \\
X_b \to b & A \to X_aAX_b \mid X_c \\
X_c \to c & B \to CD \mid AX_d \ \mid CX_a \mid BX_b \mid X_c \mid X_b \\
X_d \to d & C \to CX_a \mid BX_b \mid X_c \mid X_b \\
 & D \to X_aDX_d \mid DX_d \mid X_aX_d \mid X_d
\end{array}
$$

Por último tenemos que quitar las producciones que tienen mas de dos variables a la derecha, tras hacer esto, nos queda:

$$
\begin{array}{ll}
X_a \to a & S \to \cancel{BCX_a} \mid \cancel{X_aDX_cX_d} \mid CX_a \mid \cancel{X_aX_cX_d} \mid \cancel{X_aAX_b} \mid X_c \\
X_b \to b & A \to \cancel{X_aAX_b} \mid X_c \\
X_c \to c & B \to CD \mid AX_d \ \mid CX_a \mid BX_b \mid X_c \mid X_b \\
X_d \to d & C \to CX_a \mid BX_b \mid X_c \mid X_b \\
 & D \to \cancel{X_aDX_d} \mid DX_d \mid X_aX_d \mid X_d \\
 & S \to \ BZ_1 \mid X_aZ_2 \mid X_aZ_4 \mid X_aZ_5 \\
 & A \to X_aZ_6 \qquad D \to X_aZ_7 \qquad Z_1 \to CX_a \\
 & Z_2 \to DZ_3 \qquad Z_3 \to X_cX_d \qquad Z_4 \to X_cX_d \\
 & Z_5 \to AX_b \qquad Z_6 \to AX_b \qquad Z_7 \to DX_d
\end{array}
$$

Finalmente ordenando lo anterior, la gramática en forma normal de Chomsky es:

$$
\begin{array}{ll}
X_a \to a & S \to \ CX_a \mid X_c \mid BZ_1 \mid X_aZ_2 \mid X_aZ_4 \mid X_aZ_5 \\
X_b \to b & A \to \ X_c \mid X_aZ_6 \\
X_c \to c & B \to CD \mid AX_d \ \mid CX_a \mid BX_b \mid X_c \mid X_b \\
X_d \to d & C \to CX_a \mid BX_b \mid X_c \mid X_b \\
 & D \to \ DX_d \mid X_aX_d \mid X_d \mid X_aZ_7 \\
 & Z_1 \to CX_a \\
 & Z_2 \to DZ_3 \qquad Z_3 \to X_cX_d \qquad Z_4 \to X_cX_d \\
 & Z_5 \to AX_b \qquad Z_6 \to AX_b \qquad Z_7 \to DX_d
\end{array}
$$

7. Práctica 7

Obtener una gramática libre del contexto que genere las palabras del lenguaje que reconoce un autómata con pila.

Para realizar esta practica he elegido un autómata con pila que reconoce las palabras que contienen la misma cantidad de unos que de ceros:

$$M = (\{q_1\}, \{0,1\}, \{R, X, Y\}, \delta, q_1, R, \emptyset)$$

$$
\begin{array}{ll}
\delta(q_1, 0, R) = \{(q_1, XR)\} & \delta(q_1, 0, X) = \{(q_1, XX)\} \\
\delta(q_1, 1, R) = \{(q_1, YR)\} & \delta(q_1, 1, Y) = \{(q_1, YY)\} \\
\delta(q_1, 1, X) = \{(q_1, \varepsilon)\} & \delta(q_1, 0, Y) = \{(q_1, \varepsilon)\} \\
\delta(q_1, \varepsilon, R) = \{(q_1, \varepsilon)\} &
\end{array}
$$

A partir del autómata anterior obtenemos la siguiente gramática que genera las palabras que tiene el mismo número de unos que de ceros:

$$
\begin{array}{ll}
[q_1, R, q_1] \to 0[q_1, X, q_1][q_1, R, q_1] & [q_1, X, q_1] \to 0[q_1, X, q_1][q_1, X, q_1] \\
[q_1, R, q_1] \to 1[q_1, Y, q_1][q_1, R, q_1] & [q_1, Y, q_1] \to 1[q_1, Y, q_1][q_1, Y, q_1] \\
[q_1, X, q_1] \to 1 & [q_1, Y, q_1] \to 0 \\
[q_1, R, q_1] \to \varepsilon &
\end{array}
$$