Teoría de Lenguajes
Teoría de la Programación I
(Soluciones)

Ejercicio 1 **[Teoría de Lenguajes]**
Indique si las siguientes afirmaciones son verdaderas o falsas, justificando en cada caso.

a) Sean $L_1 \bigcap L_2 \neq \varnothing$; $L_1 \bigcap L_3$ finito ; $L_2 \bigcap L_3$ Libre de Contexto no Regular.
Entonces L_1 es Regular.

b) Sean $L_4 \bigcap L_5$ regular ; $L_4 \bigcap L_6$ Libre de Contexto no Regular ; $L_4 \neq L_5$.
Entonces $L_4 \bigcap L_5 \bigcap L_6$ es Regular

c) Si $(L_7 \cap L_8)^c = \varnothing$, entonces L_7 es Libre de Contexto

Solución:

a) **Falso**. Contrajemplo con Σ={a,b}

Con $L_1=\{a^nb^n, n\geq 0\}$ y $L_2=L_3=\{b^na^n, n\geq 0\}$ se tiene que
$L_1 \bigcap L_2 = \{\varepsilon\} \neq \varnothing$
$L_1 \bigcap L_3 = \{\varepsilon\}$ es finito
$L_2 \bigcap L_3 = \{b^na^n, n\geq 0\}$ es libre de contexto no regular
y $L_1=\{a^nb^n, n\geq 0\}$ no es regular

b) **Falso**. Contrajemplo con Σ={a,b}

Con $L_4= \Sigma^*$, $L_5=\Sigma^*-\{\varepsilon\}$ y $L_6=\{a^nb^n, n>0\}$ se tiene que
$L_4 \bigcap L_5 =\Sigma^*-\{\varepsilon\}$ es regular
$L_4 \bigcap L_6 =\{a^nb^n, n>0\}$ es libre de contexto no regular, $L_4 \neq L_5$
y $L_4 \bigcap L_5 \bigcap L_6 =\{a^nb^n, n>0\}$ no es regular

c) **Verdadero**.

Si $(L_7 \cap L_8)^c = \varnothing$ entonces $L_7= L_8= \Sigma^*$ (y Σ^* es LC)

Ejercicio 2
Sea $L_2 = \{ w_1\#w_2\#...\#w_n\# , w_i \in \{0,1\}^* \text{ y } \exists\, w_i, w_{i-1} \ / \ |w_i|_0 = |w_{i-1}|_1 \}$

a) Clasifique L_2 según la Jerarquía de Chomsky.
b) Construya una gramática G_2 / $L_2 = L(G_2)$. ¿Está simplificada? Justifique.
c) Construya un autómata M_2 / $L_2 = L(M_2)$ ¿Es determinista? Justifique.

Solución

a) L_2 es Libre de Contexto, lo cual se demuestra con la G.L.C. de la parte b) o con el A.P.D. de la parte c). Demostraremos que no es Regular, aplicando el Contra reciproco del Pumping Lemma 1, para lenguajes regulares.

Sea N la constante del Pumping Lemma, elegimos $z = 1^N\#0^N\# \in L_2$, $|z|=2N+2$.
Consideramos todas las descomposiciones de z = wxy, tal que: $|wx|\leq N$ y $|x|>0$

Caso	1^N	#	0^N	#
1	wx			

Caso 1
$w = 1^p$ $\quad p+q\leq N$
$x = 1^q$ $\quad q>0$
$y = 1^{N-p-q}\#0^N\#$

Nota: Las gramáticas y los autómatas deben corresponderse con el tipo del lenguaje considerado en cada caso, según la Jerarquía de Chomsky. Se valora positivamente la simplicidad de las soluciones propuestas así como una breve explicación de éstas. Todas las respuestas deben estar debidamente justificadas.

$z_i = 1^p (1^q)^i 1^{N-p-q}\#0^N\#$
Si elegimos i = 2, $z_2 = 1^p 1^{2q} 1^{N-p-q}\#0^N\# = 1^{N+q}\#0^N\# \notin L_2$, dado que no se cumple: $|w_2|_0 = |w_1|_1$ y no existen otro par de w_i y w_{i-1}.

Como estas son todas las descomposiciones posibles de z en wxy que cumplen: $|wx| \leq N$ y $|x|>0$, por el CR del PL 1 concluimos que L_2 no es regular.

b) G_2:

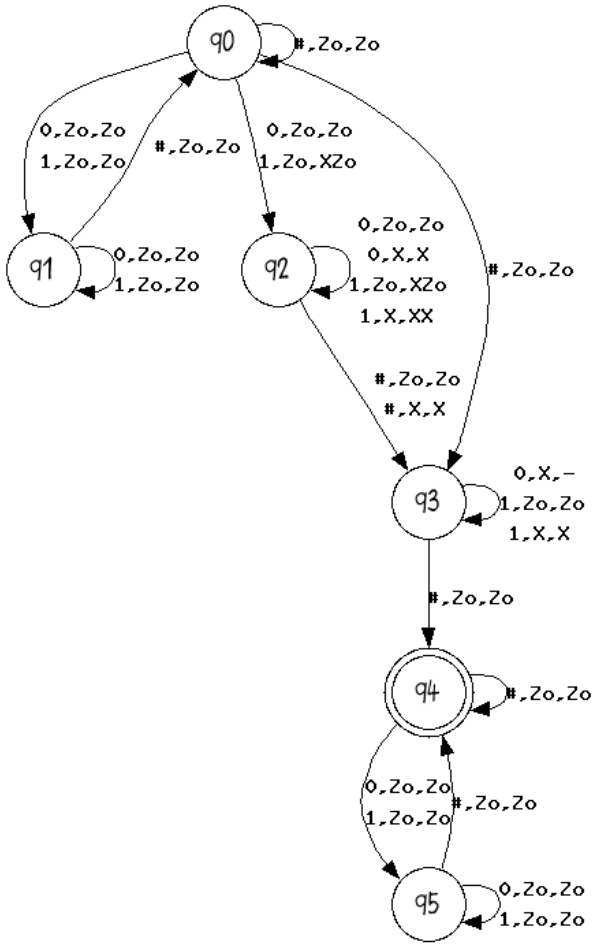
S → R T # R

T → C 1 C T U 0 U | # // genero: w_{i-1} # w_i
C → 0 C | ε
U → 1 U | ε

R → ε | W # R // genero: $w_1\#w_2\#...w_{i-2}\#$ o $w_{i+1}\#w_{i+2}\#...w_n\#$
W → 0 W | 1 W | ε

G_2 es un gramática que no esta simplificada, porque tiene transiciones épsilon.

c) M_2:



M_2 es un APD no determinista, dado que de q0 salen dos transiciones con el mismo símbolo y el mismo tope en el stack hacia estados distintos.

Nota: Las gramáticas y los autómatas deben corresponderse con el tipo del lenguaje considerado en cada caso, según la Jerarquía de Chomsky. Se valora positivamente la simplicidad de las soluciones propuestas así como una breve explicación de éstas. Todas las respuestas deben estar debidamente justificadas.

Ejercicio 3

Sea $L_3 = \{ x_1x_2...x_n\#y_1y_2...y_n\#z_1z_2...z_n$ con $x_i, y_i, z_i \in \{0,1\}; n \geq 1 ; z_i = x_i \oplus y_i \}$

Son ejemplos de tiras válidas:

1100#0101#1001
001#101#100

a) Clasifique L_3 según la Jerarquía de Chomsky.
b) Construya una gramática G_3 / $L_3 = L(G_3)$.
c) Construya un autómata M_3 / $L_3 = L(M_3)$.

Solución

a) El lenguaje L_3 propuesto es recursivamente enumerable, lo que se demuestra en la parte b) dando una gramática irrestricta que lo genera o en la parte c) mediante una máquina de Turing que lo reconoce. No es un lenguaje libre de contexto, lo cual se prueba aplicando el contra-recíproco del Pumping Lema para lenguajes libres de contexto.

Dado N, elegimos para el pumping la tira $Z= 0^N\#0^N\#0^N$. Tenemos que $|Z| = 3N+2 > N$, y además se cumple que $Z \in L_3$ ya que las tres secuencias de ceros tienen igual largo y la tercer secuencia es el XOR de las dos primeras.

Analizamos las descomposiciones de z = uvwxy y consideramos aquellas familias que cumplen las condiciones $|vx| \geq 1$ y $|vwx| \leq N$.

familia	**0^N**	**#**	**0^N**	**#**	**0^N**
1	v x				
2			v x		
3					v x
4	v x	x	x		
5	v		x		
6	v	v	v x		
7			v x	x	x
8			v		x
9			v	v	v x

Familia 1:

$u = 0^p$
$v = 0^q$ $q+s >= 1$
$w = 0^r$ $q+r+s <= N$
$x = 0^s$
$y = 0^{N-p-q-r-s}\#0^N\#0^N$

$z_i = 0^{N+(q+s)(i-1)}\#0^N\#0^N$
$z_0 = 0^{N-(q+s)(}\#0^N\#0^N$

Como q+s >= 1 , el largo de la primera secuencia de 0's es menor estricto que el de las otras 2, y por lo tanto $z_0 \notin L_3$.

Familias 2 y 3:
Análogas a la familia 1, en ambos casos $z_0 \notin L_3$. La segunda y tercera secuencias de 0's quedan mas cortas que las otras dos respectivamente.

Familia 4:
$u = 0^{N-p-q-r}$
$v = 0^{p}$
$w = 0^{q}$
$x = 0^{r}\#0^{s}$ $p+q+r+s+1 <= N$ $p+r+s+1 >= 1$
$y = 0^{N-s}\#0^{N}$

$z_i = 0^{N+p(i-1)-r} (0^{r}\#0^{s})^{i} 0^{N-s}\#0^{N}$
$z_0 = 0^{2N-p-s-r}\#0^{N}$
luego z_0 tiene un único "#", y por lo tanto $z_0 \notin L_3$.

Familias 6,7 y 9:
Análogas a la familia 4, eligiendo i=0 en todos los casos $z_0 \notin L_3$ por tener sólo un "#".

Familia 5:
$u = 0^{N-p-q}$
$v = 0^{p}$ $p+s >= 1$
$w = 0^{q}\#0^{r}$ $p+q+1+r+s <= N$
$x = 0^{s}$
$y = 0^{N-r-s}\#0^{N}$

$zi = 0^{N+p(i-1)}\# 0^{N+s(i-1)}\#0^{N}$
$z_2 = 0^{N+p}\#0^{N+s}\#0^{N}$

Como p+s >= 1 , debe ser p>=1 o s>=1

Si p >= 1 :
N + p >= N + 1, haciendo que el largo de la primera secuencia de 0's sea estrictamente mayor que el largo de la tercera secuencia de 0's, y por consiguiente $z_2 \notin L_3$.

Si s >= 1 :
N + s >= N + 1, haciendo que el largo de la segunda secuencia de 0's sea estrictamente mayor que el largo de la tercera secuencia de 0's, y por consiguiente $z_2 \notin L_3$.

Familia 8:
Es análoga a la familia 5 anterior, tomando i=2 ya sea la segunda o la tercera secuencia de 0's quedan de mayor largo que la primera y por lo tanto $z_2 \notin L_3$.

Como estas son todas las descomposiciones que cumplen las condiciones $|vx| \geq 1$ y $|vwx| \leq N$ y para cada una encontramos un i / $z_i \notin L_3$, entonces podemos afirmar que **L_3 no es Libre de Contexto.**

b) Se construirá una gramática irrestricta G_3 / $L_3 = L(G_3)$.
w = xi,x2,...,xn#y1,y2,...,yn#z1,z2...,zn

Idea de la gramática: Se generan primero las secuencias arbitrarias de x_i y de y_i de igual largo. Por cada y_i se genera una variable adicional C ó U según sea y_i =0 o y_i =1 respectivamente. Se agrega una marca M al comienzo que se mueve a la derecha hasta el primer "#", creando variables C_x cuando encuentra un x_i =0 y variables U_x cuando encuentra un x_i =1. Las variables C y U se mueven a la derecha del segundo "#", renombrandose cuando pasan el mismo a C_y y U_y. Las variables C_x y U_x se mueven a la derecha hasta encontrar una variable C_y o U_y, momento en que el par de variables se transforma en el literal resultado del XOR correspondiente.

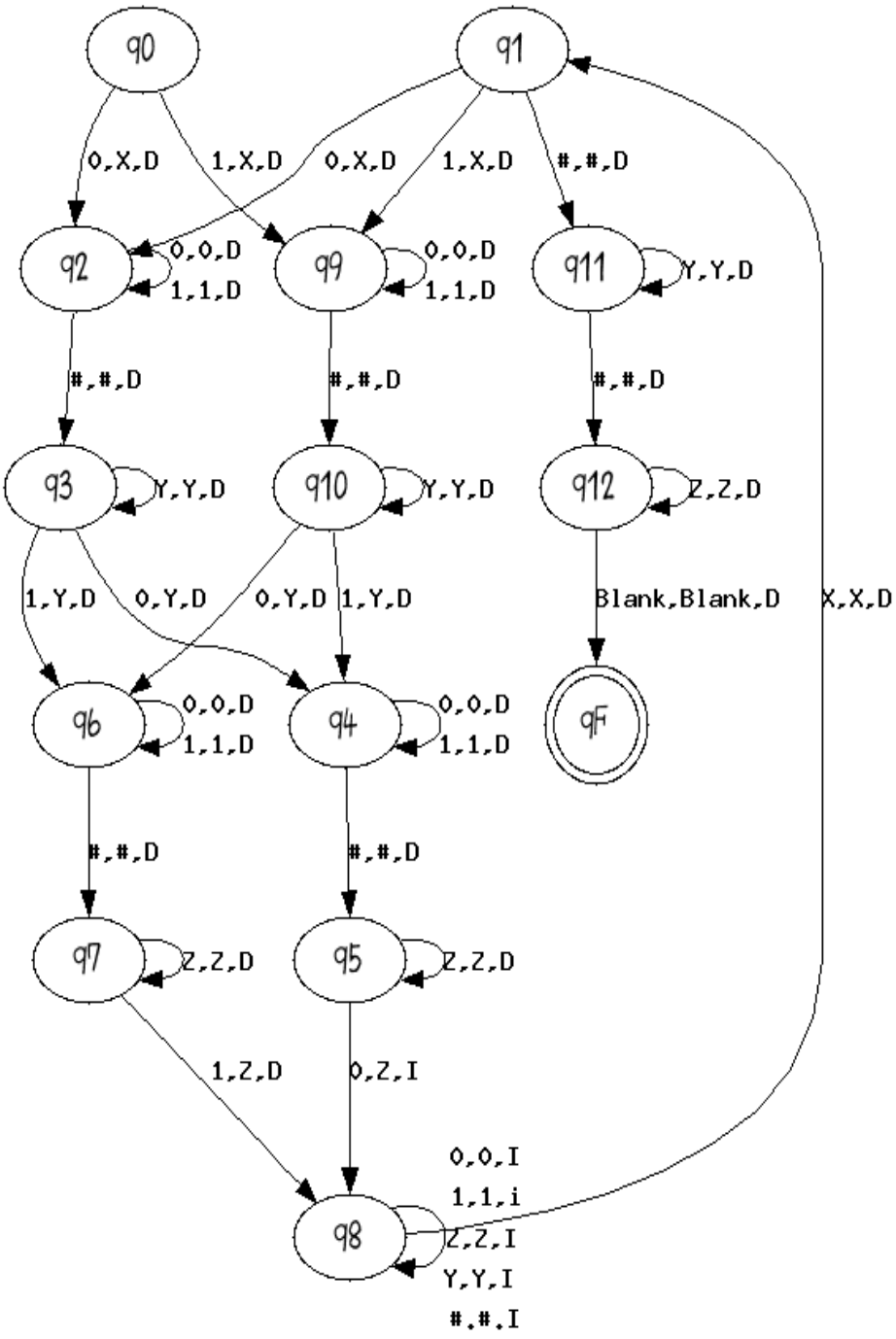
S -> MS'#
S'-> 0S'0C | 0S'1U | 1S'0C | 1S'1U | 0#0C | 0#1U | 1#0C | 1#1U
// se pasan las variables C y U a la derecha del segundo "#", renombrándose
C0 -> 0C
C1 -> 1C
C# -> #C_y
U0 -> 0U
U1 -> 1U
U# -> #U_y
// se mueve M a la derecha generando C_x y U_x
M0 -> 0MC_x //Es necesario que la M vaya a la izquierda de las C_x y de las U_x
M1 -> 1MU_x
M# -> #
// se mueven las C_x y U_x a la derecha, hasta encontrar una C_y o una U_y
C_x0 -> 0C_x
C_x1 -> 1C_x
C_x# -> #C_x
U_x0 -> 0U_x
U_x1 -> 1U_x
U_x# -> #U_x
// se aplica la tabla del XOR para generar los zi
C_xC_y -> 0
C_xU_y -> 1
U_xC_y -> 1
U_xU_y -> 0

Otra solución, donde se generan las zi al comienzo y luego solo se reordenan símbolos:
//A = C = 0
//B = U = 1
// A y B corresponden a la segunda tira, C y U a la tercera tira

S -> PXMF
X -> YX | Y
Y = 0AC | 0BU | 1AU | 1BC
//terminales de la segunda tira
A0 -> 0A
A1 -> 1A
AM ->M0
B0 -> 0B
B1 -> 1B
BM ->M1
//terminales de la tercera tira
C0 -> 0C
C1 -> 1C
CM ->MC
CF -> F0

U0 -> 0U
U1 -> 1U
UM ->MU
UF -> F1
//inserción de los #
P0 -> 0P
P1 -> 1P
PM -> #P
PF -> #

c) Se construye la siguiente MT



Ejercicio 4

a) Pasaje AFND-ε → AFND

ε-clausura(q0) = {q0,q1,q2,q3}
ε-clausura(q1) = {q0,q1,q2,q3}
ε-clausura(q2) = {q0,q1,q2,q3}
ε-clausura(q3) = {q3}
ε-clausura(q4) = {q4}
ε-clausura(q5) = {q5,q7}
ε-clausura(q6) = {q3,q6}
ε-clausura(q7) = {q7}
ε-clausura(q8) = {q8}

Aplicamos la fórmula: δ'(q,a) = ε-clausura(δ(ε-clausura(q),a))

δ'	a	b
q0	{q0,q1,q2,q3}	{q4}
q1	{q0,q1,q2,q3}	{q4}
q2	{q0,q1,q2,q3}	{q4}
q3	{}	{q4}
q4	{q5,q7}	{q8}
q5	{q3,q6}	{}
q6	{}	{q4}
q7	{q3,q6}	{}
q8	{}	{}

Pasaje AFND → AFD

	δ''	a	b
p0	[q0]	[q0q1q2q3]	[q4]
p1	[q0q1q2q3]	[q0q1q2q3]	[q4]
p2	[q4]	[q5q7]	[q8]
p3	[q5q7]	[q3q6]	-
p4	[q3q6]	-	[q4]
p5	[q8]	-	-

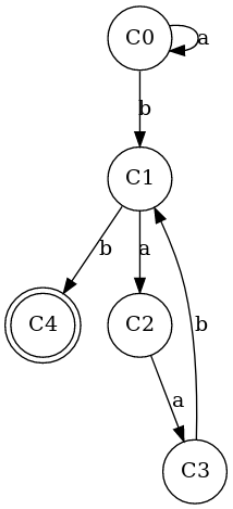
Quedando como estados finales: {p5}

Pasaje de AFD -> AFDM

$\prod_0$: [p0 p1 p2 p3 p4] [p5]
$\prod_1$: [p0 p1] [p2] [p3] [p4] [p5]
$\prod_2$: [p0 p1] [p2] [p3] [p4] [p5]
$\prod_2 = \prod_1$

Luego tomamos:
C0 = [p0p1]
C1 = [p2]
C2 = [p3]
C3 = [p4]
C4 = [p5]

Haciendo el diagrama de estados quedaría:



ii) Se calculan las expresiones regulares de las clases de equivalencia de $R_{M'4}$ para el autómata mínimo M'_4 obtenido en la parte anterior.
Como el AFD es mínimo, por Myhill-Nerode las clases de R_{M4} coinciden con las de R_{L4} y por lo tanto las expresiones regulares que se obtienen son las solicitadas.

Para cada estado C_i se tiene una expresión regular X_i correspondiente a las tiras de Σ^* que partiendo del estado inicial C_0 terminan en dicho estado C_i.

$X_0 = X_0a \mid \varepsilon$
$X_1 = X_0b \mid X_3b$
$X_2 = X_1a$
$X_3 = X_2a$
$X_4 = X_1b$

Además agregamos la ecuación para el estado pozo:

$X_p = X_2b \mid X_3a \mid X_4(a|b) \mid X_p(a|b)$

Sustituyendo y aplicando Arden resulta:

$X_0 = a^*$

$X_1 = a^*b \mid X_2ab$
$X_1 = a^*b \mid X_1aab$
$X_1 = a^*b \mid a^*b(aab)^*$

X_1 = a*b(aab)*

X_2 = a*b(aab)*a

X_3 = a*b(aab)*aa

X_4 = a*b(aab)*b

X_p = a*b(aab)*ab | a*b(aab)*aaa | a*b(aab)*b(a|b) | (a|b)* =
= a*b(aab)*(ab | aaa | ba | bb) | (a|b)*

iii) Finalmente se construye una Gramática Lineal Izquierda con las siguientes producciones:

S → Rb
R → Raab | Xb | b
X → Xa | a

Está simplificada porque:
- No tiene producciones épsilon
- No tiene producciones unitarias
- Todas las variables son útiles

Ejercicio 1 [Teoría de la Programación I]

a) Computable: Este es un programa en P que la computa:

```
PROGRAM(<i,k,m,p>)
      X0:=1;                          -- resultado
      X1:=k;
      X2:=suc(m);                     -- X1 varia entre k y m
      WHILE (X1<X2) & (X0<>0) DO
        X3:=EVAL_PROG_STEP(i,X1,p); -- p pasos de i, entrada X1
        X4:= FST(X3);
        IF (X4=0) THEN X0:=0 FI;      -- si no terminó i, devuelvo 0
         X1:=SUC(X1)                  -- aumento la entrada
      END WHILE;

      RESULT(X0)
```

El programa anterior simula p pasos del programa i probando, una a una, todas las entradas entre k y m. Si esa cantidad de pasos no le alcanza al programa i para converger en algún valor, X0 se lleva a cero, y este es el valor devuelto por la función. En caso contrario, va a terminar el ciclo exterior sin cambiar el valor de X0, y la función devuelve 1. Entonces, este programa computa a la función **f**.

b) i) Un problema es NP-completo, si pertenece a NP (hay una algoritmo polinomial no determinístico que lo resuelve) y todo otro problema en NP se reduce en tiempo polinomial a él.

ii)El problema SAT es el problema de saber si, dada una expresión booleana con variables y sin cuantificadores, hay alguna asignación de valores para sus variables que hace que la expresión sea verdadera.

iii) El Teorema de Cook demuestra que el problema de la satisfactibilidad booleana es NP-completo