Teoría de Lenguajes
Teoría de la Programación I
Soluciones

Ejercicio 1

a)

El lenguaje es recursivamente enumerable como vermos en la parte b) al construir una G.I. que lo genera o en c) al construir una M.T. que lo reconoce. Demostraremos que NO es libre de contexto, utilizando el CR del PL (para Libres de Contexto).
Para eso elegimos la tira $z = 0^N01^N\#0^N11^N = 0^{N+1}1^N\#0^N1^{N+1}$, siendo N la constante del Pumping Lemma.
Consideramos todas las descomposiciones posibles de la tira z en uvwxy que cumplen $|vx|>0$ y $|vwx| \leq N$, y probamos que existe un i tal que uv^iwx^iy no pertenece a L_1

Familia	0^{N+1}	1^N	#	0^N	1^{N+1}
1	v x				
2		v x			
3				v x	
4					v x
5	v	x			
6				v	x
7	v x	x			
8	v	v x			
9				v x	x
10				v	v x
11		v x	x	x	
12		v	v	v x	
13		v		x	

Familia 1
$u = 0^j$
$v = 0^k$
$w = 0^l$
$x = 0^m$
$y = 0^{N+1-j-k-l-m}1^N\#0^N1^{N+1}$ con $k+m>0$
$z_0 = 0^{N+1-k-m}1^N\#0^N1^{N+1}$ no pertenece al lenguaje pues al ser $k+m>0$, la cantidad de simbolos a la izquierda del # es distinta de la cantidad de simbolos a la derecha del # a's y b's

Familias 2, 3, 4, 5, 6, 7, 8, 9 y 10
En forma analoga a la familia 1, se puede demostrar que eligiendo i=0, z_0 no pertenece al lenguaje, dado que la cantidad de simbolos a ambos lados del # no son iguales para i=0.

Familias 11
$u = 0^{N+1}1^j$
$v = 1^k$
$w = 1^{N-j-k-l}$
$x = 1^l\#0^m$
$y = 0^{N-m}1^{N+1}$
$z_0 = 0^{N-k}1^{N-l}0^{N-m}1^{N+1}$ no pertenece al lenguaje pues z_0 no tiene el #.

Familias 12
En forma analoga a la familia 11, se puede demostrar que eligiendo i=0, z_0 no pertenece al lenguaje, dado que z_0 no tiene el #.

Familia 13
$u = 0^{N+1}1^j$
$v = 1^k$
$w = 1^{N-j-kl}\#0^l$
$x = 0^m$
$y = 0^{N-l-m}1^{N+1}$ con k+m > 0
Si k≠m entonces al elegir i=0, z_0 no pertenece al lenguaje, porque queda con distinta cantidad de simbolos a ambos lados del #.
Si k=m entonces elegimos i=4, $z_3 = 0^{N+1}1^{N+3k}\#0^{N+3k}1^{N+1}$ no pertenece al lenguaje pues como k=m>0 en el peor caso vale 1, y la cantidad 0s a la izquierda y derecha el # difieren al menos en 2 (o más si k>1), por lo tanto la cantidad de simbolos distintos entre w y w' es mayor a 1.

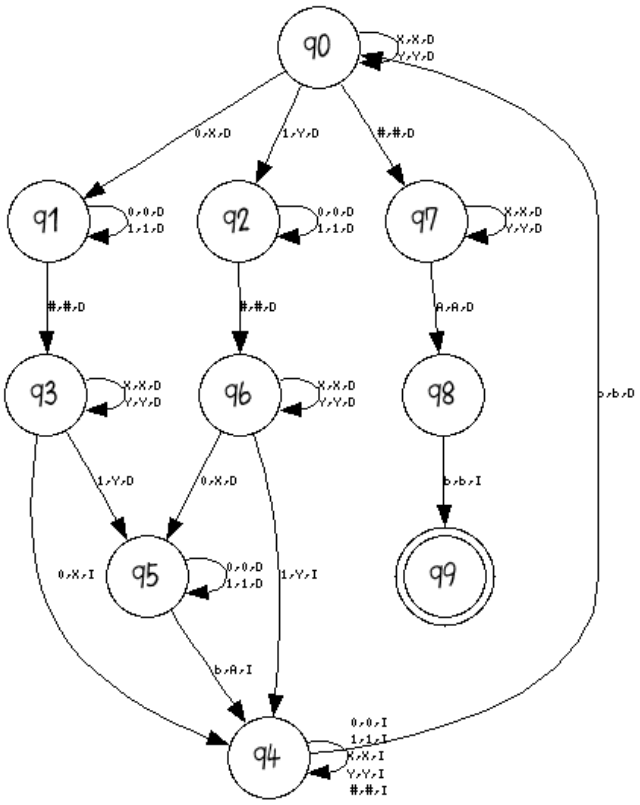
Como estas son todas las descomposiciones posibles de z en uvwxy que cumplen |vx|>0 y |vwx| ≤ N, por el CR del PL el lenguaje no es libre de contexto.

b)

La idea es generar por cada 1 una U, por cada 0 una C, excepto en un caso, donde es al revés. Luego mover a la derecha las U y las C y transformarlas en 1 y 0

```
S  -> 0CS | 1US | S'
S' -> 0UA | 1CA
A  -> 0CA | 1UA | #
C0 -> 0C
C1 -> 1C
U0 -> 0U
U1 -> 1U
U# -> #1
C# -> #0
```

c)



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.

La idea básica es: por cada simbolo del lado izquierdo del # se busca su correspondiente del lado derecho del # en la misma posición relativa desde el comienzo de la tira y desde el # respectivamente. Los 0s se marcan con Xs y los 1s con Ys, salvo cuando no hay una correspondencia (los simbolos en la misma posición no coinciden) en tonces se agrega una marca A al fina, de la tira. Como solo se permite un simbolo distinto, solo puede haber una marca A, si se intenta agregar otra marca A la MT se tranca. Cuando termina de recorrer las tiras w y w' deber verificar que al final se haya agregado una marca A al final.

Ejercicio 2

a) FALSO

Sea $L_1 = \{a^n b^n / n \in N\}$ y sea $L_2 = L_1^C$. Luego, $L_1 \cup L_2 = \Sigma^*$ que es regular. Sin embargo, L_2 no es regular (si lo fuera, L_1 también sería regular por clausura de los L.R. respecto al complemento). La afirmación es falsa.

b) VERDADERA

Supongamos que L_2 es finito. Sea $L_3 = L_1 \cup L_2$. Trabajando sobre conjuntos se tiene que
$L_1 = (L_3 \cap L_2^C) \cup (L_1 \cap L_2)$
Si L_3 (por la letra) y L_2 (porque supongo finito) son regulares, $L_3 \cap L_2^C$ también es regular (clausura de los lenguajes regulares respecto a la intersección y al complemento).
Asimismo, como L_2 es finito, $L_1 \cap L_2$ también lo es, y por lo tanto $L_1 \cap L_2$ es regular.
Como L_1 es la unión de dos Lenguajes regulares.
Finalmente llegamos a que L_1 es regular, lo cual contradice la hipótesis "L_1 es no regular". Se concluye entonces que el supuesto no puede darse y por lo tanto L_2 es no finito.
La afirmación es verdadera.

c) FALSO

L_2 es finito. Luego $L_1 \cap L_2$ es finito y por lo tanto regular. La afirmación es falsa.

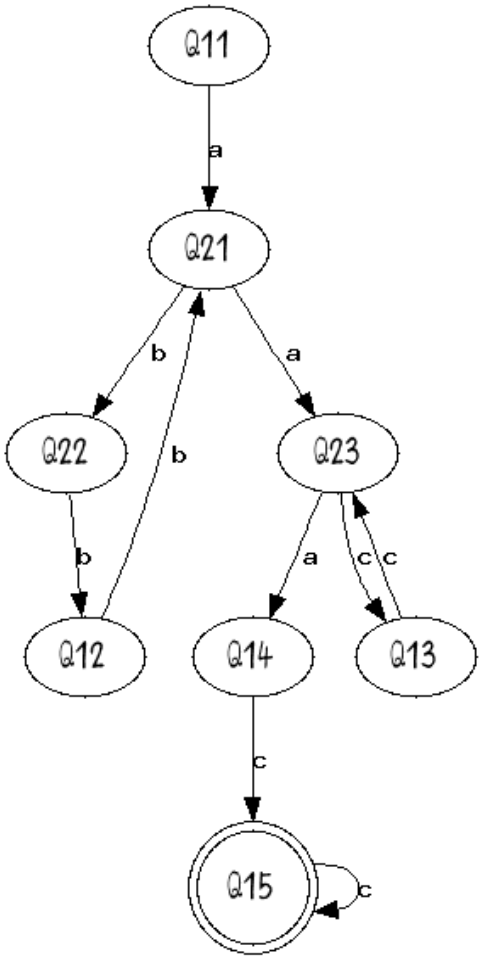
d) FALSO

Sea $L_1 = \{x/x$ es de la forma $a^n b^n c^n, n \geq 0\}$. L_1 es R.E. y no L.C.
Sea $L_2 = L(a^*b^*c^*) - \{\varepsilon\}$. L_2 es regular.

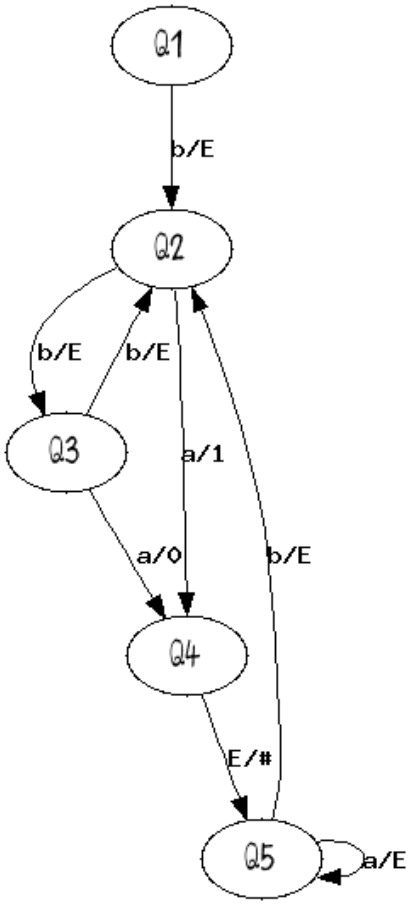
La tira ε pertenece a L_1 y no pertenece a L_2, por lo que podemos afirmar que L_1 no está incluido en L_2. $L_1 \cup L_2 = L(a^*b^*c^*)$ que es regular. Entonces la afirmación es falsa.

Ejercicio 3

a) AFD de 2 cintas



b) Máquina de Mealy



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 4 [Teoría de Lenguajes]

a)

L_4 es Libre de Contexto pero no regular. Se demuestra por el contrarrecíproco del Pumping Lemma para Regulares y luego en b) se construye una Gramática Libre de Contexto.

Sea N la constante del PL, y sea $z= a^N100b^N$; notar que cumple $|z|>N$

Consideramos todas las descomposiciones posibles para z = uvw que cumplan:
$|uv| \leq N$ y $|v|\geq 1$

Se puede observar que como los N primeros símbolos de la z elegida contienen solo a's, la única familia de descomposiciones posible es:

i) $u=a^p$ $p+j \leq N$
$v=a^j$ $j \geq 1$
$w= a^{N-p-j}100b^N$

$z_i= a^{N+(i-1)j}100b^N$ En este caso, para i=2 z_2 quedaría $a^{N+j}100b^N$ pero como $j \geq 1$, la cantidad de a's antes de la subtira 10 es mayor o igual a la cantidad de b's. Por lo tanto z_2 no pertenece al lenguaje.

Como se dijo antes, es la única familia a analizar bajo los supuestos $|uv| \leq N$ y $|v|\geq 1$
Con lo cual, como esas son todas las descomposiciones posibles L_4 no es un lenguaje regular.

b)

Una posible gramática es la que tiene las siguientes producciones:
{ $S \rightarrow aSb \mid aAb$
$A \rightarrow 1A0 \mid 1A00 \mid 100$ }

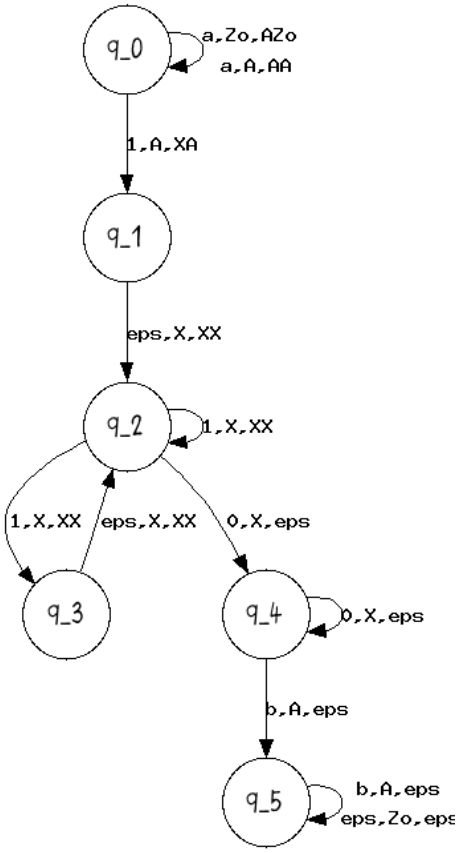
No contiene producciones Epsilon, ni tiene producciones unitarias. Todas las reglas contienen símbolos útiles (se puede ver aplicando los algoritmos POS y ALC vistos en el curso)

Luego, es una gramática simplificada.

c)

Se construirá un APD.

La idea es que sea NO Determinista. Se agrega una A por cada a leída. Cuando vengan los 1's agrego o una X o dos X. Uso un estado con transición epsilon para el caso de agregar 2 X. El primer uno agrega una X para garantizar luego que k<m.
Cuando comiencen a venir 0´s, se cambia a otro estado y se van sacando las "marcas" X (tantas como 0´s vengan; a lo sumo vendrán 2k-1 0´s).
Finalmente se vuelve a cambiar de estado ante la primer b (teniendo en cuenta que en el tope del stack solo puede haber A. Se sacan validando la misma cantidad de a's que b's.



d) Como L_4 libre de contexto no regular, tiene infinitas clases de equivalencia.
Esto se justifica por el teorema de Myhill-Nerode (ver teórico)

Ejercicio 4 [Teoría de la Programación 1]

i. Falso. Supongamos que A es un conjunto decidible; luego, su función característica total es P-computable. Sea MA una macro que la computa.

Consideremos una macro MQ que dado el índice de un programa i, calcula el índice del siguiente programa:

```
PROGRAM(X0)
   X1:= EVAL_PROG(i,i);
RESULT(X0)
```

Este programa computa, o bien la función identidad (cuando el programa de índice i con entrada i converge), o bien la función vacía (cuando diverge).

Ahora, construimos el siguiente programa en P de índice p:

```
PROGRAM(X0)
   X1:= MQ(X0);
   X2:= MA(X1);
RESULT(X2)
```

Este programa siempre para, pues MQ y MA siempre lo hacen. Además se cumple que:

$$<I_x(p), i>\Rightarrow 1 \Leftrightarrow^1 MA(MQ(i))=1 \Leftrightarrow^2 MQ(i) \in A \Leftrightarrow^3 \phi_{MQ(i)}(MQ(i))=MQ(i) \Leftrightarrow^4$$
$$\Leftrightarrow^4 <I_x(i), i>\downarrow \Leftrightarrow^5 \theta(i)=1$$

1. Por construcción de p.
2. MA función característica total de A.
3. Por definición de A.
4. Por construcción, MQ(i) computa la identidad sii $I_x(i)$ converge con entrada i.
5. Definición de θ.

Luego, el programa de índice p computa a la función θ, lo que es un absurdo que se desprende de suponer que A es decidible $\Rightarrow$ A no es decidible.

ii. False. Supongamos que B es un conjunto r.e., luego, existe su función característica parcial. Sea MB una macro que la computa.

Sea MQ la macro de la parte anterior. Construimos el siguiente programa en P de índice p:

```
PROGRAM(X0)
   X1:= MQ(X0);
   X2:= MB(X1);
RESULT(X2)
```

Este programa cumple que:

$$<I_x(p), i>\downarrow \Leftrightarrow^1 MB(MQ(i))\downarrow \Leftrightarrow^2 MQ(i) \in B \Leftrightarrow^3 \phi_{MQ(i)} \neq Id_N \Leftrightarrow^4$$
$$\Leftrightarrow^4 <I_x(i), i>\uparrow \Leftrightarrow^5 i \in C(K)$$

1. Por construcción de p.
2. MB función característica parcial de B.
3. Por definición de B.
4. Por construcción, MQ(i) no computa la identidad sii $I_x(i)$ diverge con entrada i.
5. Definición de C(K).

Dado que el programa de índice p cuando converge devuelve 1, éste computa a la función característica parcial de C(K), con lo que el conjunto es r.e. Esto es un absurdo (teoremas 19 y 20), que se desprende de suponer que B es r.e $\Rightarrow$ B no es r.e.

iii. False. Como se cumple que $B=C^c$, si C fuese decidible, también lo sería B (teorema 17), lo que implicaría que B sería además r.e. (teorema 18), algo que demostramos en la parte anterior que no se cumple. $\Rightarrow$ C no es decidible.

Ejercicio 5 [Teoría de la Programación 1]

Ver Garey & Johnson.