Teoría de Lenguajes – Teoría de la Programación I
Soluciones

Ejercicio 1
Sean

L_a = { w ∈ {a,b,c}*, w es de la forma $a^n b^m c^p$, con p=n MOD 3 y m=p*n ; n>0}
L_b = { w ∈ {a,b,c}*, w es de la forma $a^n b^m c^p$, con p=n MOD m ; m>0}

a) Clasifique L_a y L_b según la Jerarquía de Chomsky.

Demostraremos que L_a es L.C. no regular.
Es L.C. por la gramática L.C. de la parte b), o por el APD de la parte c).
No es Regular por el contra-recíproco del PL para lenguajes regulares, como mostraremos a continuación:

Sea N constante del P.L., elegimos $z = a^{3N+1} b^{3N+1} c \in L_a$ y $|z| = 6N+3 \geq N$
Obs.: tomamos 3N+1 como la cantidad de a_s para garantizar que el modulo 3 sea igual a 1 (la cantidad de c_s).

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

Caso 1:
$u = a^p$ $p+q \leq N$
$v = a^q$ q c 1
$w = a^{3N+1-p-q} b^{3N+1} c$

$z_i = a^p (a^q)^i a^{3N+1-p-q} b^{3N+1} c = a^{3N+1+(i-1)q} b^{3N+1} c$, eligiendo i= 0, entonces $z_0 = a^{3N+1-q} b^{3N+1} c$ que no pertenece a L_a dado que p=1, pero m no es igual a p*n, $3N+1 \neq 1*(3N+1-q)$ con $q \geq 1$

Este es el único caso a analizar bajo los supuestos $|uv| \leq N$ y $|v| \geq 1$. Por lo tanto por el C.R. del P.L. para lenguajes regulares, L_a no es un lenguaje regular.

Demostraremos que L_b es R. E. no L. C.
Es R.E. por la gramática irrestricta de la parte b), o por la M.T. de la parte c).
No es L.C. por el contra-recíproco del P.L. para lenguajes L.C., como mostraremos a continuación:

Sea N constante del P.L., elegimos $z = a^N b^{N+1} c^N \in L_b$ y $|z| = 3N+1 \geq N$
Consideramos todas las descomposiciones posibles para z = uvwxy que cumplen:
$|vx|>0$ y $|vwx| \leq N$

Caso	a^N	b^{N+1}	c^N
1	v x		
2	v x	x	
3	v	x	
4	v	v x	
5		v x	
6		v x	x
7		v	x
8		v	v x
9			v x

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.

Caso 1:
La descomposición de este caso sería:

$u = a^p$
$v = a^q$ $|vx| = q+s \geq 1$
$w = a^r$ $|vwx| = q+r+s \leq N$
$x = a^s$
$y = a^{N-p-q-r-s}\, b^{N+1}\, c^N$

$z_i = a^p\, (a^q)^i\, a^r\, (a^s)^i\, a^{-p-q-r-s} b^{N+1} c^N = a^{N+(i-1)(q+s)} b^{N+1} c^N$

eligiendo i=0, $z_0 = a^{N-(q+s)} b^{N+1} c^N$ no pertenece a L_b dado que (N-(q+s)) mod (N+1) $\neq$ N porque q+s >0.

Caso 2:
La descomposición de este caso sería:

$u = a^{N-p-q-r}$
$v = a^p$ $|vx| = p+r+s \geq 1$
$w = a^q$ $|vwx| = p+q+r+s \leq N$
$x = a^r b^s$ $r>0$ y $s>0$
$y = b^{N+1-s} c^N$

$z_i = a^{N-p-q-r}\, (a^p)^i\, a^q\, (a^r b^s)^i\, b^{N+1-s} c^N$

eligiendo i=2, se mezclan las a_s con las b_s, por lo tanto z_2 no pertenece a L_b.

Caso 4, 6 y 8:
son análogos al caso 2, eligiendo i=2 se mezclan en el caso 4 las a_s con las b_s y en los casos 6 y 8 las b_s y c_s.

Caso 3:
La descomposición de este caso sería:

$u = a^{N-p-q}$
$v = a^p$ $|vx| = p+s \geq 1$
$w = a^q\, b^r$ $|vwx| = p+q+r+s \leq N$
$x = b^s$
$y = b^{N+1-r-s}\, c^N$

$z_i = a^{N-p-q}\, (a^p)^i\, a^q\, b^r\, (b^s)^i\, b^{N+1-r-s}\, c^N = a^{N+(i-1)p}\, b^{N+1+(i-1)s}\, c^N$

Si p>0, eligiendo i=0 la cantidad de a_s es menor que la cantidad de c_s, entonces z_0 no pertenece a L_b dado que el resto (la cantidad de c_s) no puede ser mayor que el dividendo (la cantidad de a_s).
Si p=0, entonces s>0 porque p+s ≥ 1. Elegimos i=0, quedando en z_0 el divisor (la cantidad de b_s) menor o igual que el resto (la cantidad de c_s), por lo tanto z_2 no pertenece a L_b.

Caso 5:
La descomposición de este caso sería:

$u = a^N\, b^{N+1-p-q-r-s}$
$v = b^p$ $|vx| = p+r \geq 1$
$w = b^q$ $|vwx| = p+q+r \leq N$
$x = b^r$
$y = b^s c^N$

$z_i = a^N\, b^{N+1-p-q-r-s}\, (b^p)^i\, b^q\, (b^r)^i\, b^s\, c^N = a^N\, b^{N+1+(i-1)(p+r)}\, c^N$

eligiendo i=0, $z_0 = a^N\, b^{N+1-(p+r)}\, c^N$ no pertenece a L_b. dado que el divisor (la cantidad de b_s) no puede ser menor igual al resto (la cantidad de c_s).

Caso 7:
La descomposición de este caso sería:

$u = a^N b^{N+1-p-q}$
$v = b^p$ $\quad |vx| = p+s \geq 1$
$w = b^q c^r$ $\quad |vwx| = p+q+r+s \leq N$
$x = c^s$
$y = c^{N-r-s}$

$z_i = a^N b^{N+1-p-q} (b^p)^i b^q c^r (c^s)^i c^{N-r-s} = a^N b^{N+1+(i-1)p} c^{N+(i-1)s}$

Si s>0, eligiendo i=2 la cantidad de c_s es mayor que la cantidad de a_s, entonces z_2 no pertenece a L_b dado que el resto (la cantidad de c_s) no puede ser mayor que el dividendo (la cantidad de a_s).
Si s=0, entonces p>0 porque p+s ≥ 1. Elegiendo i=0, quedaría en z_0 el divisor (la cantidad de b_s) menor o igual que el resto (la cantidad de c_s), por lo tanto z_2 no pertenece a $L_{b.}$

Caso 9:
La descomposición de este caso sería:

$u = a^N b^{N+1} c^{N-p-q-r-s}$
$v = c^p$ $\quad |vx| = p+r \geq 1$
$w = c^q$ $\quad |vwx| = p+q+r \leq N$
$x = c^r$
$y = c^s$

$z_i = a^N b^{N+1} c^{N-p-q-r-s} (c^p)^i c^q (c^r)^i c^s = a^N b^{N+1} c^{N+(i-1)(p+r)}$

eligiendo i=2, $z_2 = a^N b^{N+1} c^{N+(p+r)}$ no pertenece a L_b. dado que el resto (la cantidad de c_s) no puede ser mayor que el dividendo (la cantidad de a_s).

Estos son todos los casos a analizar bajo los supuestos |vwx| ≤ N y |vx|≥1. Por lo tanto por el C.R. del P.L. para lenguajes L.C., L_b no es un lenguaje L.C.

b) Construya gramáticas G_a y G_b / $L_a = L(G_a)$ y $L_b = L(G_b)$.

```
/** Gramática Ga **/

// Para n MOD 3 tomo 3 casos: {0, 1, 2}
// X: n MOD 3 = 0
// Y: n MOD 3 = 1
// Z: n MOD 3 = 2

S -> X | Yc | Zcc

X -> aaa | Xaaa                // n > 0
Y -> ab | aaaYbbb
Z -> aabbbb | aaaZbbbbbb



/** Gramática Gb **/

// Para p = n MOD m tomo 3 casos
// X: n = m => p = 0
// Y: n < m => p = n
// Z: n > m => p = n MOD m

S -> PS'F      // P y F son marcas de principio y fin
```

```
S' -> X | Y | Z

X -> ab | aXb

Y -> b | Yb | Y'b        // Genero las b "extra" para que m > n
Y' -> abC | aY'bC        // Genero igual cantidad de a, b y c

Cb -> bC        // Muevo C hacia la derecha
CF -> Fc        // Muevo F hacia la izquierda cambiando las C por c

Pa -> aP        // Muevo P hacia la derecha salteando las a
Pb -> bP        // Muevo P hacia la derecha salteando las b

PF -> eps       // Borro variables de marca

// Para Z tomo 2 casos:
// Z': m = 1 => p = 0
// Z'': m > 1

Z -> Z'b | IZ''  // I se utiliza como factor multiplicador de la cantidad de a

Z' -> aa | aZ'  // No se toma el caso de cantidad de a = 1 porque está considerado
en X

Z'' -> aAb | aAZ''b     // Genero igual cantidad de a y b

bI -> Ib
aI -> Ia
AI -> IAa       // I se mueve a la izquierda "multiplicando" la cantidad de a

Db -> bD
Da -> aD
DA -> AaD       // D se mueve a la derecha "multiplicando" la cantidad de a

PI -> PD | PE   // Cambio "multiplicadores" de izquierda y derecha, o cambio por
                   variable de borrado
DF -> IF | EF // E se encarga de desbalancear por lo menos en 1 la cantidad de a y b

bE -> Eb
Eb -> bE
aE -> Ea
Ea -> aE        // Busco la primer A, sea que venga de la izquierda o de la derecha

AE -> M
EA -> M         // Desbalanceo a y b, y cambio por M, quien mantiene por lo menos
                   una a de diferencia

bM -> Mb
Mb -> bM
aM -> Ma
Ma -> aM        // Busco la primer A, sea que venga de la izquierda o de la derecha

AM -> TaC       // Mantengo por lo menos una a de diferencia
MA -> aCT       // T se encarga de terminar de generar las C y a restantes

bT -> Tb
Tb -> bT
aT -> Ta
Ta -> aT        // Busco las A, sea que venga de la izquierda o de la derecha
```

AT -> TaC | T // Puedo elegir borrar la A o generar una a y una C
TA -> aCT | T

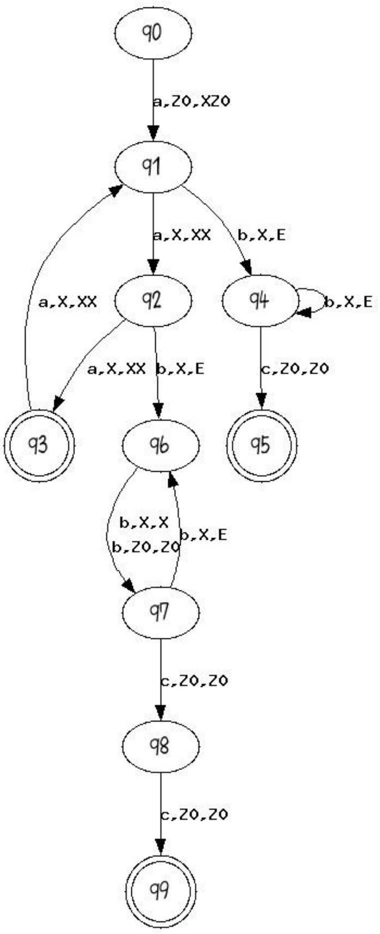
TF -> F
PT -> P // Llegué al principio o al final y borro la marca T

Ca -> aC // Muevo las C hacia la derecha

// Recordar que más arriba tengo definidas reglas de movimiento de P, F y C

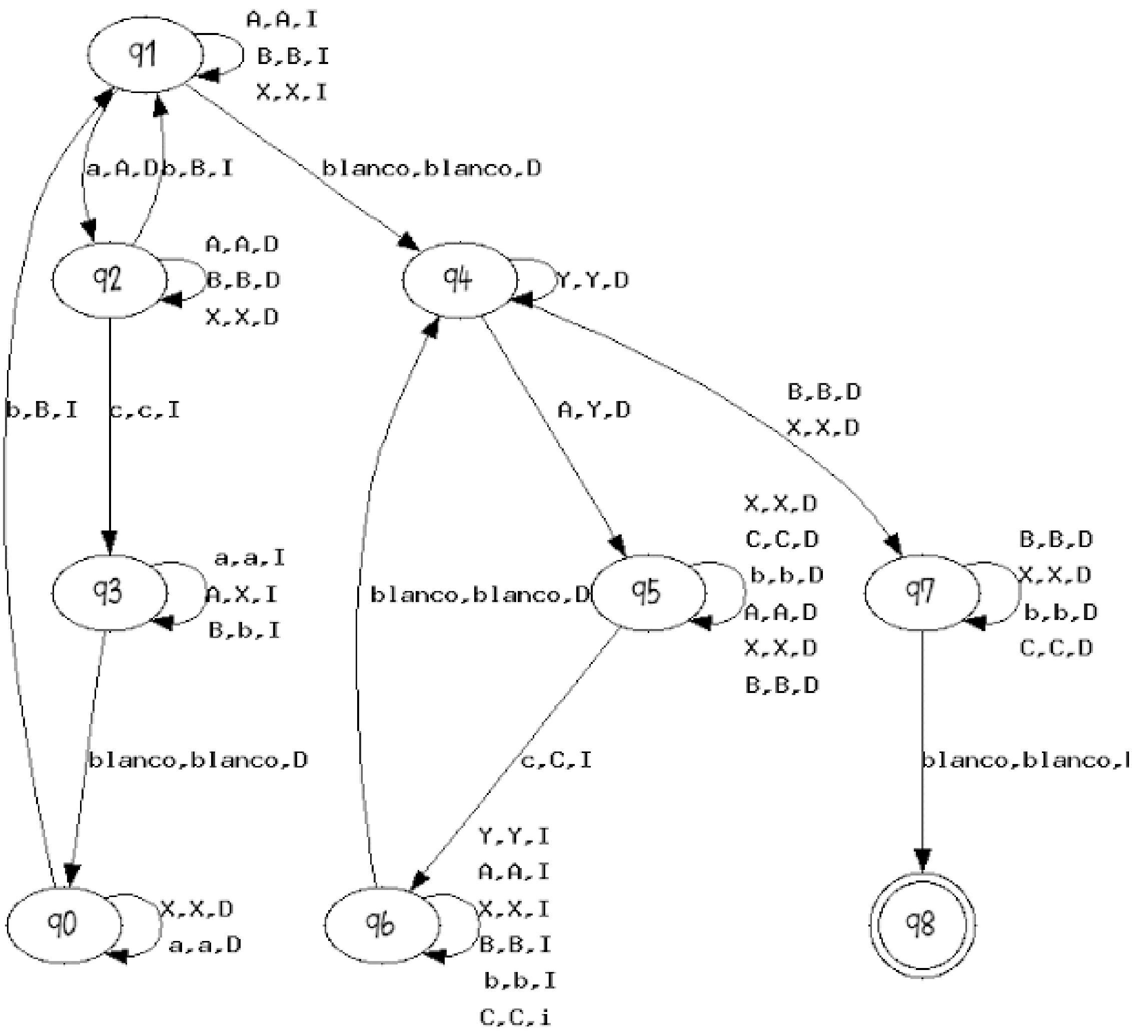
c) Construya autómatas M_a y M_b / $L_a = L(M_a)$ y $L_b = L(M_b)$.
d) ¿Son los autómatas M_a y M_b construidos deterministas? Justifique.

Ma



Es un APD Determinista

Mb



Se trata de una MT Determinista

Ejercicio 2

Indique si las siguientes afirmaciones son verdaderas o falsas, justificando en cada caso.

a) Sean L_1 y L_2 lenguajes Libres de Contexto pero NO Regulares entonces $L_1 \cup L_2$ es también un lenguaje Libre de Contexto NO Regular
Falso: Sean por ejemplo $L_1=\{0^k1^j, k>j\geq0\}$ y $L_2=\{0^k1^j, j\geq k\geq0\}$ de donde $L_1 \cup L_2 = \{o^*1^*\}$ que es regular

b) Sean L_3 un lenguaje Regular y dos tiras x, w $\in L_3$, entonces $xR_{L3}w$
Falso: Sea por ejemplo L dado por la ER 0*| 1* y tomemos las tiras x=0 y w=1 y puede verse que 1 R_{L3} 0 NO se cumple puesto que por ejemplo, si le concatenamos una tira de 0's, (000) 1000 $\notin L_3$ y 0000 $\in L_3$

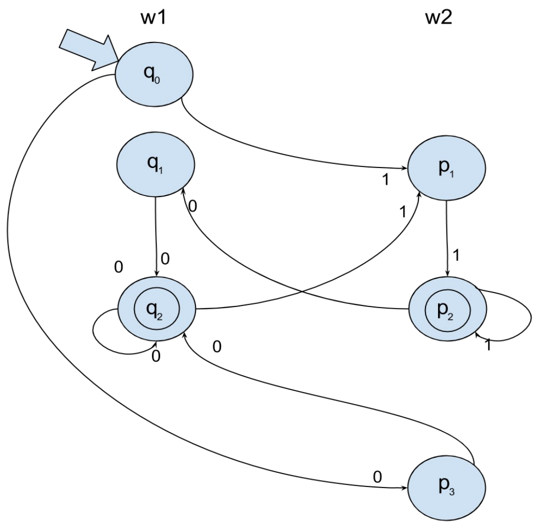
c) Si L_4 no es Libre de Contexto y L_5 es finito, entonces $L_4 \cap L_5$ no es Regular
Falso: $L_4 \cap L_5$ contiene a lo sumo, todas las tiras de L_5 que es finito; con lo cual, la intersección es finita y por lo tanto es un lenguaje regular.

d) Al aplicar el algoritmo de conversión de AFND-ε a AFND el conjunto de estados finales se mantiene

Falso: Sea un AFND-ε M=(Q,Σ,δ,qo,F) y al aplicar el algoritmo obtenemos un AFND M'=(Q,Σ,δ',qo,F'), F=F' a menos que la ε-clausura(qo) contenga un estado final de F y qo $\notin$ F ; en cuyo caso hay quye agregarlo al nuevo conjunto de estados finales del AFND,
es decir F' = F $\cup$ {qo}

Ejercicio 3

a)



b) Sea el siguiente AFND-**ε** M = ({q_o, q_1, q_2, q_3}, {a, b}, δ , q_o, {q_3}) , donde δ

δ	a	b	ε
q_o	{q_1}	{q_2}	{q_1}
q_1	{q_0}	{ }	{q_3}
q_2	{q_3}	{ }	{q_o}
q_3	{ }	{q_3}	{}

i) Dé el AFDM (Autómata Finito Determinista Mínimo) M' / L(M) = L(M').

1. AFND-**ε** → AFND

Se calcula la **ε**-clausura para cada estado:

- **ε**-clausura (q_o) = {q_o, q_1, q_3}
- **ε**-clausura (q_1) = {q_1, q_3}
- **ε**-clausura (q_2) = {q_o, q_1, q_2, q_3}
- **ε**-clausura (q_3) = {q_3}

La nueva función de transición es:

δ_{AFND}	**a**	**b**
q_o	$\{q_o, q_1, q_3\}$	$\{q_o, q_1, q_2, q_3\}$
q_1	$\{q_o, q_1, q_3\}$	$\{q_3\}$
q_2	$\{q_o, q_1, q_3\}$	$\{q_o, q_1, q_2, q_3\}$
q_3	$\{\}$	$\{q_3\}$

Dado que:

- **ε**-clausura $(q_o) \cap \{q_3\} \neq \emptyset$ entonces q_o es un estado final del AFND

El AFND resultante es $M_{AFND} = (\{q_o, q_1, q_2, q_3\}, \{a, b\}, \delta_{AFND}, q_o, \{q_o, q_3\})$

2. AFND → AFD

La nueva función de transición es:

δ_{AFD}	**a**	**b**
$[q_o]$	$[q_o, q_1, q_3]$	$[q_o, q_1, q_2, q_3]$
$[q_o, q_1, q_3]$	$[q_o, q_1, q_3]$	$[q_o, q_1, q_2, q_3]$
$[q_o, q_1, q_2, q_3]$	$[q_o, q_1, q_3]$	$[q_o, q_1, q_2, q_3]$

Renombrando los estados de la siguiente forma:

- $p_o = [q_o]$
- $p_1 = [q_o, q_1, q_3]$
- $p_2 = [q_o, q_1, q_2, q_3]$

La nueva función de transición queda:

δ_{AFD}	**a**	**b**
p_o	p_1	p_2
p_1	p_1	p_2
p_2	p_1	p_2

Dado que todos los estados p_i del nuevo AFD contienen al estado final q_o del AFND original, entonces todos los estados del nuevo AFD son finales.

El AFD resultante es $M_{AFD} = (\{p_o, p_1, p_2\}, \{a, b\}, \delta_{AFD}, p_o, \{p_o, p_1, p_2\})$

3. AFD → $AFD_{Mínimo}$

$\prod_0$: $\emptyset$ $\{p_o, p_1, p_2\}$ → Particionamiento inicial: finales - no finales
$\prod_1$: $\emptyset$ $\{p_o, p_1, p_2\}$ → Dado que no varia el particionamiento se termina el algoritmo

Sea c_0 la clase que representa a los estados $\{p_o, p_1, p_2\}$, la nueva función de transición queda:

δ_{AFDM}	**a**	**b**
c_o	c_0	c_0

El AFD Mínimo resultante es $M_{AFDM} = (\{c_o\}, \{a, b\}, \delta_{AFDM}, c_o, \{c_o\})$

ii) Defina la relación R_M para un AFD cualquiera.

Dado un Autómata Finito Determinista M = (Q, Σ, δ, q_o, F), sean x, y $\in \Sigma^*$ se dice que:

$$x \; R_M \; y \Leftrightarrow \hat{\delta}(q_o, x) = \hat{\delta}(q_o, y)$$

iii) ¿Cuántas clase de equivalencia se definen para $R_{M'}$? Justifique. Escriba expresiones regulares para cada una de las clases.

Dado que existe un único estado, se tiene una única clase de equivalencia para $R_{M'}$. Para calcular formalmente su expresión se debe resolver el siguiente sistema de ecuaciones:

$$Y_0 = \boldsymbol{\varepsilon} \mid Y_0(0|1)$$

Dado que $\boldsymbol{\varepsilon} \notin L(0|1)$, es posible aplicar el Lema de Arden:

$$Y_0 = \boldsymbol{\varepsilon}(0|1)^* = (0|1)^*$$

Ejercicio 3 **[Teoría de la Programación I]**

a) Indique si la siguiente función es o no computable. Justifique.

$$f(i,j) = \begin{cases} 1 & \text{si } \exists\, k, p \in N \text{ tales que } \langle I_x(i),k\rangle \Rightarrow p \text{ y } \langle I_x(j),p\rangle \Rightarrow k \\ 0 & \text{en caso contrario} \end{cases}$$

- *f* no es computable

Supongamos que f es computable; sea entonces MF una macro que la computa.
Sea la macro MJ, la cual, dado un natural **i**, calcula el índice **j** del siguiente programa constante:

```
PROGRAM(X0)
  X1:=EVAL_PROG(i,i);
  X1:=0
RESULT(X1)
```

El programa de índice **j** o bien converge a cero con toda entrada (si Ix(i,i)↓ lo hace), o bien diverge con todas ellas (si Ix(i,i)).
Se construye el siguiente programa en P de índice p:

```
PROGRAM(X0)
  X1:=MJ(X0);
  X2:=MF(X1,X1);
RESULT(X2)
```

Se cumple que:

- La convergencia del programa **p** está garantizada: tanto MJ como MF siempre paran.
- Además:

$$Ix(p,i) \Rightarrow 1 \Leftrightarrow^1 MF(MJ(i), MJ(i)) \Rightarrow 1 \Leftrightarrow^2 \exists\, k, p \in N \text{ tales que } \langle I_x(j),k\rangle \Rightarrow p \,\wedge\, \langle I_x(j),p\rangle \Rightarrow k \Leftrightarrow^3 I_x(i,i)\downarrow$$

1. Por construcción del programa **p**.
2. MF computa la función *f*.
3. Por construcción del programa **j**, en caso que éste termine, siempre se cumple la condición para k=p=0

Luego, el programa de índice **p** calcula la función θ. **Absurdo** que se genera al suponer *f* computable $\Rightarrow$ tesis.

b) Explique brevemente:

i) ¿Qué significa que un problema de decisión se transforma polinomialmente en otro?.
ii) ¿Qué significa que un problema de decisión es NP?
iii) ¿Qué significa que un problema de decisión es NP-Completo?

Ver bibliografía