Teoría de Lenguajes

Consideraciones generales	
i)	Escriba nombre y C.I. en todas las hojas.
ii)	Numere todas las hojas.
iii)	En la primera hoja indique el total de hojas.
iv)	Comience cada ejercicio en una hoja nueva.
v)	Utilice las hojas de un solo lado.
vi)	Entregue los ejercicios en orden.

Ejercicio 1

a) Sea L_1 = { tiras de la forma x# / $x \in \{a,b\}^* \wedge |x|_a \bmod 2 = |x|_b \bmod 3$ }

i) Defina la relación R_L para un lenguaje L cualquiera.

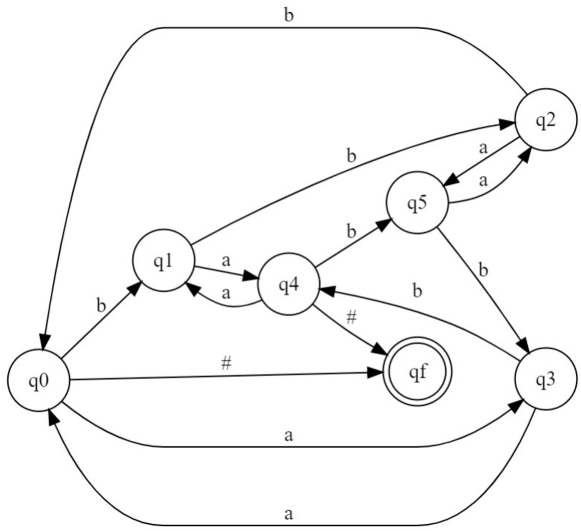
Sea $L \subseteq \Sigma^*$, $x,y \in \Sigma^*$
Se dice que $x\ R_L\ y$ <==> $\forall z \in \Sigma^*$ se cumple que $xz \in L \wedge yz \in L$ o $xz \notin L \wedge yz \notin L$

ii) Construya un autómata determinista M_1 / $L_1 = L(M_1)$.

Se ve que las tiras para pertenecer al lenguaje, deben de cumplir simultáneamente las dos condiciones respecto a las cantidades de a's y b's.
Podemos armar la siguiente tablita con las distintas combinaciones posibles para las cantidades y asociarle a cada una un "estado" del autómata.

	$\|x\|_a$ mod 2	$\|x\|_b$ mod 3
q_0	0	0
q_1	0	1
q_2	0	2
q_3	1	0
q_4	1	1
q_5	1	2

De donde el diagrama de estados del autómata podría ser el siguiente:



iii) ¿Cuántas clases de equivalencia se definen para el lenguaje L_1 según la relación R_L definida en i) ? Justifique.

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.

Para responder debiera hacer el autómata mínimo.
Si aplicamos el algoritmo de minimización podemos verificar que nos queda el mismo, con lo cual, los estados mi indican la cantidad de clases de equivalencia del lenguaje (#clases(R_M) = #clases(R_L) si M es el mínimo.

En nuestro ejemplo serían 8; las dibujadas más el pozo (recordar que lo que se hace es una partición de las clases sobre Σ^*

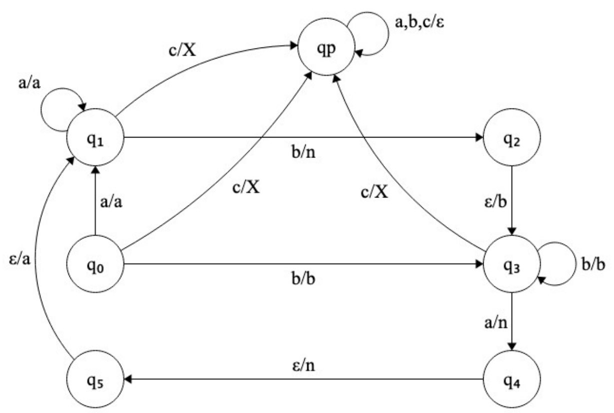
b) Se quiere construir un autómata que, dado una entrada sobre el alfabeto {a,b,c}, la traduzca generando una salida sobre {a,b,c,n,X} con las siguientes reglas:

- *n* representa el salto de línea
- se mantendrán las a's consecutivas hasta que venga una **b**. En ese caso se salta a la siguiente línea generando una *n*
- análogamente, se tratará de modo a la secuencia de b's con respecto a una **a**, pero se deberá hacer un salto de línea extra; es decir, se generan dos ***n***'s
- en caso de venir una c, la ejecución *se aborta* y debe generarse un símbolo "X" y luego consumir la entrada sin generar ningún símbolo.

Ejemplos:

Entrada	Salida
aaabbbaaa	aaanbbbnnaaa
bababba**c**	bnnanbnnanbbnnaX
b**c**aaaabbba	bX
ba**c**	bnnaX
cba	X

Construya un autómata con salida que resuelva la traducción.



Ejercicio 2

Sea $L_2 = \{ x \mathbin{/} x = d_1d_2d_3\ldots d_na^p \mathbin{/} d_i \in\{1,2,3\} \wedge \sum_{i=1}^{i=n} di = 3*p\ ,\ p>0\}$

a) Clasifique L_2 según la Jerarquía de Chomsky. Justifique.

Se intentará probar que L2 NO es un lenguaje Regular utilizando el contrarecíproco del Pumping Lema. Sea N cte del PL
Se considera $z = 1^{3N}a^N \in L_2$ Con $|z|=4N$

La única familia a estudiar $z_i = uv^iw$ tal que $|uv| \leq N$ y $|v| > 0$ es:

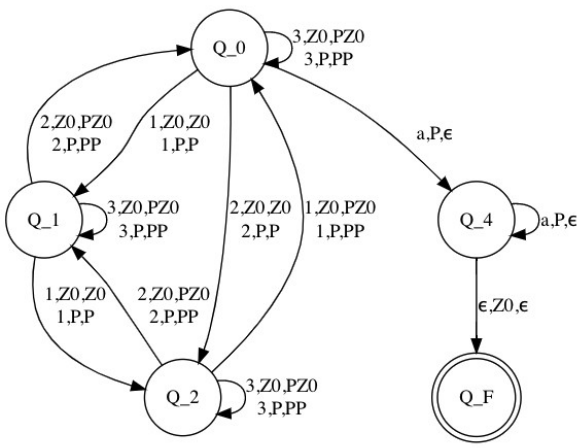
$u = 1^p$
$v = 1^q$
$w = 1^{3N-(p+q)}a^N$

Siendo $q>0$, $p \geq 0$, $p+q \leq N$

Tomando i=0, $z_0 = 1^{3N-q}a^N$, por lo que 3*cant 1's < cant a's (q>0) y entonces $z_0 \notin L_2$

Por lo tanto L_2 No es Regular.
Luego, por parte b) o c) se prueba que es Libre de Contexto ya que existe un APD (b) y una gramática libre de contexto (c).

b) Construya un autómata $M_2 \mathbin{/} L_2 = L(M_2)$. ¿Es determinístico? Justifique.



El autómata construido es determinístico.

c) Construya una gramática simplificada $G_2 \mathbin{/} L_2 = L(G_2)$.

Puede ser una gramática con las siguientes reglas:

$S \rightarrow 1A \mid 2B \mid 3Ca \mid 3a$
$A \rightarrow 1B \mid 2C \mid 3Aa \mid 2a$
$B \rightarrow 1Ca \mid 2Aa \mid 3Ba \mid 1a$
$C \rightarrow 1Aa \mid 2Ba \mid 3Ca \mid 3a$

La gramática está simplificada porque no contienen producciones epsilon, ni unitarias y todos sus símbolos son útiles.

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 = { $x_0 \ldots x_n\#ay_1 \ldots y_n\#w_0 \ldots w_n$ / $x_i \in \{0,1\}$, $y_i \in \{a,b\}$, $w_i \in \{0,1\}$ y se cumple que:

- si $y_i = a \rightarrow w_i = x_i$
- si $y_i = b \rightarrow w_i = x_{i-1}$

a) Sabiendo que L_3 NO es un lenguaje Libre de Contexto, construya una gramática $G_3/L_3=L(G_3)$.

Como se trata de un lenguaje que no es Libre de Contexto, una posible solución sería una gramática irrestricta que tenga las siguientes reglas:

S → IXaACF | IYaBUF

A → XaAC | YaBU | XbAC | YbBC | GH
B → XaAC | YaBU | XbAU | YbBU | GH

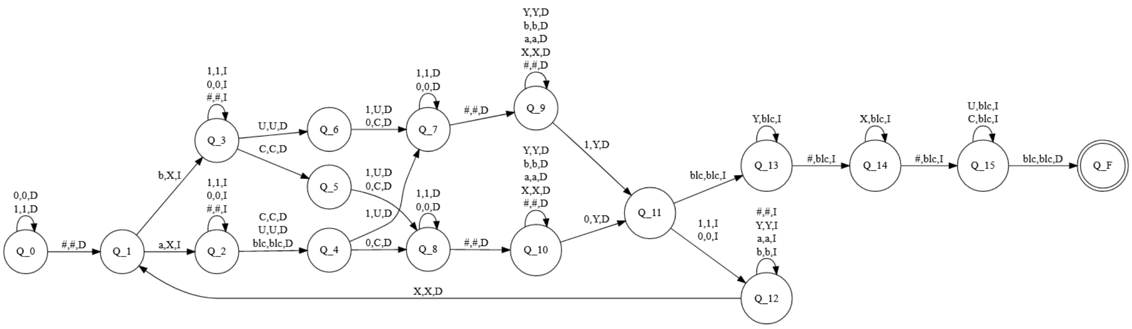
IX → 0I
IY → 1I
aG → Ga
bG → Gb
IG → #

UF → 1F
CF → 0F

U1 → 1U
U0 → 0U
C1 → 1C
C0 → 0C

H0 → 0H
H1 → 1H
HF → #

b) Construya un autómata M_3 / L_3 = $L(M_3)$.¿Es determinístico? Justifique.



Ejercicio 4 [Teoría de la Programación I]

a) El teorema de Cook es la piedra fundamental de la teoría de la complejidad computacional:

i Enuncie este teorema

ii ¿En qué radica su importancia?

i. El problema de satisfacer una fórmula booleana [SAT] es un problema NP-Completo.

ii.Cook elabora la primera prueba de que un problema es NP-Completo. Luego, para probar que un problema B es NP-Completo, no es necesario reducir TODO problema de NP a B, sino que basta con reducir únicamente a SAT.

b) Sea el conjunto A = {<i,m> / ∃n, Ix(i,n)↓ en a los sumo m pasos}
Es correcto afirmar que A no es r.e. Justifique.

Falso

Es r.e. porque el siguiente programa computa la función característica parcial de A:

```
PROGRAM(X0)                          -- X0=<i,m>
    X1:=FST(X0);                      -- índice del programa
    X2:=SND(X0);                      -- cantidad de pasos a probar
    X3:=0;                            -- entrada a probar
    X4:=1;                            -- flag de fin
    WHILE (X4≠0) DO
        X5:=EVAL_PROG_STEP(X1, X3, X2);
        IF FST(X5) ≠0 THEN
          X4:=0
        ELSE
             X3:=SUC(X3)
    FI
    END;
    X4=SUC(0);
RESULT(X4)
```