Teoría de Lenguajes

2do. Parcial – Curso 2022

Soluciones

Ejercicio 1 [Evaluación individual del obligatorio]

a) Escriba usando la sintaxis de NLTK una gramática regular para la expresión a*b*a
b) Indique que consideraciones tuvo para segmentar en tokens la entrada de los distintos programas.
c) ¿En que programas (1, 2 o 3) tuvo que recorrer explícitamente el árbol completo para generar la salida? Justifique.

a)

```
"""
S -> 'a' S | 'b' B | 'a'
B -> 'b' B | 'a'
"""
```

b) En el programa 1 es conveniente segmentar por caracteres tal como se mostraba en el programa de ejemplo (programa 0).
En los programas 2 y 3, si bien se puede mantener la misma segmentación, resulta conveniente segmentar utilizando los espacios de la entrada. Así los nombres de los operadores (ej. and, or) y las palabras (ej. Juan) son considerados como un único token y no como la concatenación de sus caracteres.

c) El único programa que es necesario recorrer el árbol completo es el 2 para escribir en notación prefija cada operador.
En el programa 3 basta con procesar el primer nivel (o primeros dependiendo de la gramática construida) para reescribir la oración en voz pasiva; y en el programa 1 no había que recorrer el árbol en absoluto, solamente verificar que existe.

Ejercicio 2

Dado el siguiente lenguaje:

$$L_2=\{ (ab)^q\#^r a^s \ / \ s=q+r+t,\ q \geq 1,\ t \geq 1,\ r \geq 0 \}$$

a) Sabiendo que L_2 no es regular, clasifíquelo en la Jerarquía de Chomsky.
b) Construya - si fuera posible - una gramática simplificada G_2 / $L(G_2) = L_2$. Justifique.
c) Construya un autómata M_2 / $L(M_2) = L_2$. ¿Es determinista? Justifique.

a) El lenguaje es no regular y libre de contexto. Qué no es regular es dato de la letra. Para ver que es libre de contexto, se prueba con la parte b) o la parte c).

b) El lenguaje L_2 puede ser escrito como $L_2=\{(ab)^q\#^r a^{r+q+t}: q \geq 1, t \geq 1, r \geq 0\}$ donde se visualiza de manera más sencilla tanto cómo se tiene que usar el stack del autómata push-down de la parte c) así como la construcción de la GLC

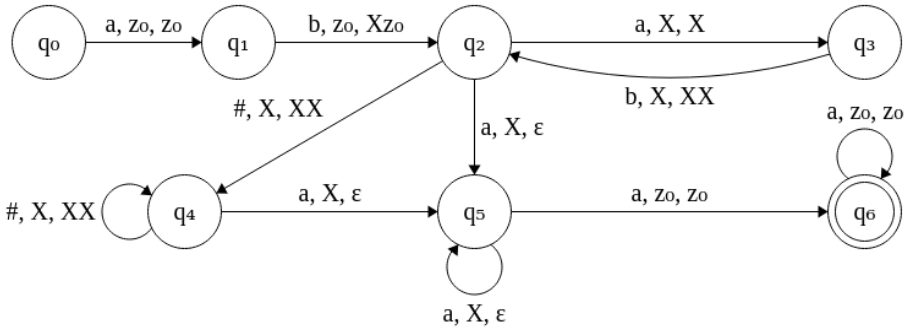
Se construye una gramática que tiene las siguientes reglas de producción:

S –> Sa | Aa
A –> abAa | abBa | aba
B –> #Ba | #a

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.

Esta gramática ya está simplificada porque:
- no contiene producciones épsilon
- no contiene producciones unitarias
- todos las variables son útiles (son alcanzables desde S y son todas positivas)

c)



El autómata construido **no es determinista** puesto que para el estado q_2 y tope de stack Q con la entrada a existen dos transiciones posibles: una a q_3 dejando igual el stack y otra a q_6 realizando un pop.

Ejercicio 3

Dados los siguientes lenguajes:

$L_{31}=\{(ab)^n xy \;/\; n \geq 1 \;,\; x,y \in \{a,b\}^*,\; |x| \leq 1,\; |y| = |x|+1\}$

$L_{32}=\{(aa)^n b^m a^p \;/\; m > n \geq 0 \;,\; p \geq n\}$

a) Clasifique ambos lenguajes en la Jerarquía de Chomsky. Justifique.
b) Construya gramáticas G_{31} / $L(G_{31}) = L_{31}$ y G_{32} / $L(G_{32}) = L_{32}$ de acuerdo a las clases en las que pertenecen cada uno según la parte a).

a)

L_{31}: Es un lenguaje regular, podríamos escribir una ER, construir un AF o una Gramática Lineal como haremos en la parte b).

L_{32}: Es un lenguaje recursivamente enumerable NO libre de contexto. Esto último se prueba con el contrarecíproco del Pumping Lema II. En b) se construye una Gramática Irrestricta.

Dado N constante cualquiera, elegimos $z = (aa)^N b^{N+1} a^N = a^{2N} b^{N+1} a^N$ / $z \in L_{32}$ y $|z|=4N+1$
Estudiamos todas las descomposiciones de z=uvwxy que cumplen $|vx|>0$ y $|vwx| \leq N$

Es importante notar que las condiciones que se deben cumplir para que las tiras $\in L_{32}$ son:

(1) 2*cant_b(z) > cant_a(z) – de las primeras **a**'s

(2) 2*cant_a(z) ≥ cant_a(z) – comparando las últimas **a**'s contra las primeras **a**'s

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

Familia 1

$u=a^{2N-p-q-r-s}$
$v=a^p$ $p+r > 0$
$w=a^q$ $p+q+r \leq N$
$x=a^r$
$y=a^s b^{N+1} a^N$

$z_i = a^{2N-p-q-r-s} (a^p)^i a^q (a^r)^i a^s b^{N+1} a^N = a^{2N-p+pi-r+ri} b^{N+1} a^N = a^{2N-(p+q)(i-1)} b^{N+1} a^N$
Eligiendo i=2 queda $z_2 = a^{2N+p+q} b^{N+1} a^N$
donde falla la propiedad (1), es decir, la cant_a(z_2) del comienzo de la tira es ≥ 2*cant_b(z_2) porque p+r>0 entonces $z_2 \notin L_{32}$

Familia 2, razonamiento similar.

$u=a^{2N} b^s$
$v=b^p$ $p+r > 0$
$w=b^q$ $p+q+r \leq N$
$x=b^r$
$y=b^{N+1-s-p-q-r} a^N$

$z_i = a^{2N} b^s (b^p)^i b^q (b^r)^i b^{N+1-s-p-q-r} a^N = a^{2N} b^{N+1+(i-1)(p+q)} a^N$
Eligiendo i=0 queda $z_0 = a^{2N} b^{N+1-p-q} a^N$
donde también falla la (1), la cant_a(z_0) del comienzo de la tira es ≥ 2*cant_b(z_2) porque p+r>0 entonces $2N \geq 2(N+1-p-q) = 2N+2-2(p+q)$ y $z_0 \notin L_{32}$

Familia 3, razonamiento similar al 1, hay que tomar i=0, con lo cual ahí la cantidad de **a**'s es la que disminuye con respecto a las primeras **a**'s (falla la (2)).

Familia 4

$u=a^{2N-p-q-r}$
$v=a^p$ $p+r+s > 0$ (s >0 sino estamos en Familia 1)
$w=a^q$ $p+q+r+s \leq N$ (r >0 sino estamos en Familia 8)
$x=a^r b^s$
$y=b^{N+1-s} a^N$

$z_i = a^{2N-p-q-r} (a^p)^i a^q (a^r b^s)^i b^{N+1-s} a^N$
Eligiendo i=2 queda $z_2 = a^{2N+p} b^s a^r b^{N+1} a^N$
donde $z_2 \notin L_{32}$ porque hay más de dos segmentos de **b**'s separados por **a**'s.

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.

Familia 5,6 7 razonamiento análogo y argumento análogo.
Siempre consideramos los casos NO contemplados en las otras familias (cuando hay más de un símbolo en v o x)

Familia 8

$u=a^{2N-p-q}$
$v=a^{p}$ $p+r > 0$
$w=a^{q}b^{s}$ $p+q+r+s \leq N$
$x=b^{r}$
$y=b^{N+1-s-r}a^{N}$

$z_i = a^{2N-p-q}(a^{p})^{i}a^{q}b^{s}(b^{r})^{i}b^{N+1-s-r}a^{N} = a^{2N-p}(a^{p})^{i}(b^{r})^{i}b^{N+1-r}a^{N}$

Como p+r>0, al menos uno de los dos es mayor a 0. Esto nos deja tres casos posibles:

- si p>0 y r>0:

Eligiendo i=0, tenemos $Z_0= a^{2N-p}b^{N+1-r}a^{N}$ donde falla la propiedad (2) ya que no se cumple (por ser p>0) que 2N ≥ 2N-p. Por lo tanto $z_0 \notin L_{32}$

- si p=0 (r>0):

Eligiendo i=0, tenemos $Z_0= a^{2N}b^{N+1-r}a^{N}$ donde falla la propiedad (1) ya que no se cumple (por ser r>0) que 2(N+1-r) > 2N. Por lo tanto $z_0 \notin L_{32}$

- si r=0 (p>0):

Eligiendo i=2, tenemos que $Z_2 = a^{2N+p}b^{N+1+r}a^{N}$ donde falla la propiedad (2) ya que no se cumple (por ser p>0) que 2N ≥ 2N+p. Por lo tanto $z_2 \notin L_{32}$

Familia 9 es similar,

$u=a^{2N}b^{N+1-p-s}$
$v=b^{p}$ $p+r > 0$
$w=b^{s}a^{q}$ $p+q+r+s \leq N$
$x=a^{r}$
$y=a^{N-q-r}$

$z_i = a^{2N}b^{N+1-p-s}(b^{p})^{i}b^{s}a^{q}(a^{r})^{i}a^{N-q-r} = a^{2N}b^{N+1-p}(b^{p})^{i}(a^{r})^{i}a^{N-r} = a^{2N}b^{N+1+(i-1)p}a^{N+(i-1)r}$
Eligiendo i=0 queda $z_0= a^{2N}b^{N+1-p}a^{N-r}$

Como p+r>0, al menos uno de los 2 es >0:
- si p=0 (r>0) falla la propiedad (2)
- si r=0 (p>0) falla la propiedad (1)
- si ambas (p>0 y r>0) por lo pronto falla seguro la propiedad (1)

Por lo tanto $z_0 \notin L_{32}$

Como estas son todas las descomposiciones posibles que cumplen |vx|>0 y |vwx|≤ N, se ha logrado demostrar por el contra-recíproco del Pumping Lema para lenguajes libres de contexto que L_{32} NO es un lenguaje libre de contexto.

b)
Para L_{31} se construye una gramática lineal derecha con las siguientes reglas de producción:

S → ab S | abX | abXXX
X → a | b

Para L_{32} se construye una gramática irrestricta con las siguientes reglas de producción:

S –> aaXBaA | YA

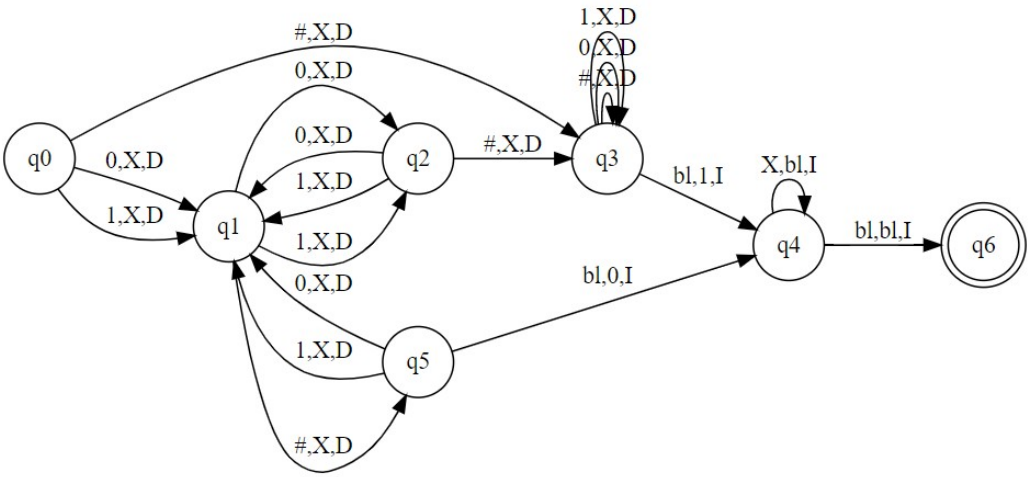
A –> aA | ε
Y –> bY | b
X –> aaxBa | bF
aB –> Ba
FB –> FBB | bF
Fa –> a

Ejercicio 4

Construya una Máquina de Turing que compute una función que recibe una tira formada por secuencias (eventualmente vacías) de 0's y 1's terminadas en # y devuelve **1** si al menos una de las secuencias tiene cantidad par de símbolos y **0** en caso contrario.
Ejemplos:

01#1110#	1
000##00#1110#	1
101#101##	1
1#111#01001#0#	0
0#1#0#	0

Nota: al terminar, en la cinta solo debe quedar la respuesta (0 o 1).



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.