# Tarea 2

Estudiante: Sebastián Cortés
CC3102-1 Teoría de la Computación
6 de Diciembre de 2019

# P1

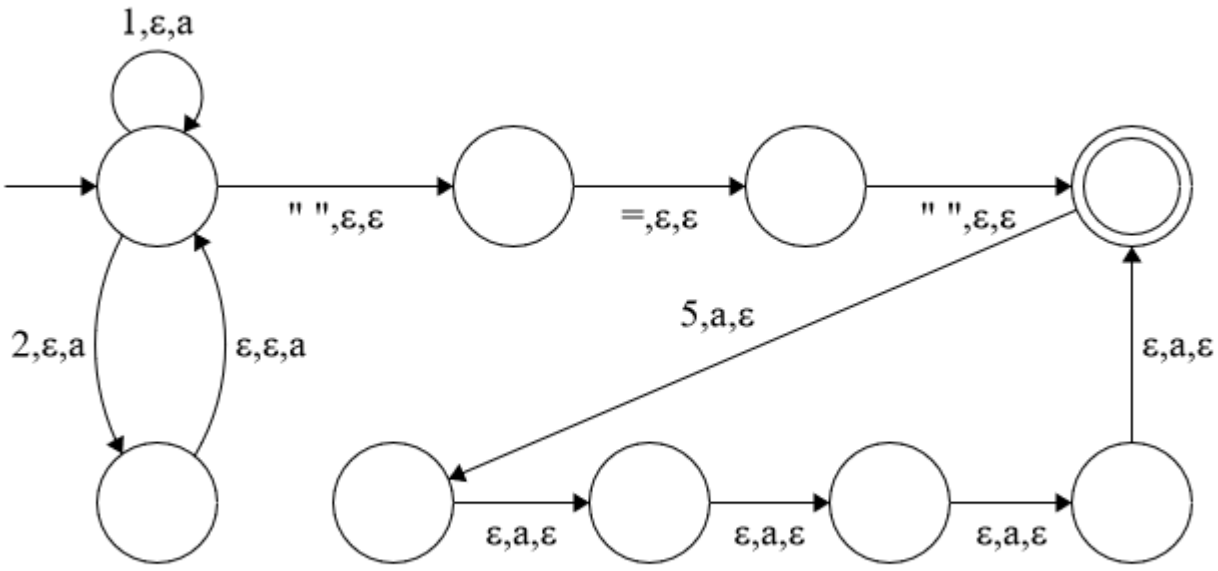
El lenguaje pedido se puede generar con la siguiente gramática de libre contexto

$$
\begin{gathered}
S \rightarrow X \\
X \rightarrow Y|AYB|BYA|CY|YCY|X111X|X_2 11X_2|\epsilon \\
X_2 \rightarrow Y|AYB|BYA \\
Y \rightarrow YY|1YA|\epsilon \\
A \rightarrow 0|\epsilon \\
B \rightarrow 00|\epsilon \\
C \rightarrow 000|\epsilon
\end{gathered}
$$

Lo que hace esta gramática a grandes rasgos es crear unos y ceros en la misma cantidad o solo 1's (en la regla del "$Y$" donde crea "$1YA$"), tambien puede replicar esta acción en simultáneo cuantas veces quiera ("$Y$" crea "$YY$"), ademas las reglas del "$X$" pueden generan el "$Y$" acompañado de los "$A$", $B$"o"$C$" de tal manera que el número de ceros pueda exceder el numero de unos hasta un máximo de 3, ademas puede replicarse habiendo cumplido los 3,2 o 1 de manera restringida (regla del "$X111X$", "$X_2 11X_2$).

# P2

Para la creación del autómata de pila usaremos la notación (a,b,c) en las transiciones, tal que leo "$a$" del input, saco "$b$" de la pila e ingreso "$c$" a la pila, cualquier estado en el cual no haya camino para el input a seguir o no hay dicho "$b$" en el tope de la pila llevará a un estado sumidero y posterior rechazo.



En la parte izquierda los dos estados analizan la parte a la izquierda del " = " del input donde por cada "1" se agrega un "$a$" a la pila y por cada "2" se agregan 2 "$a$" a la pila. Luego se procede a la derecha del " = " a través de las tres transiciones a la derecha del dibujo para llegar al estado de aceptación, cabe mencionar que este estado no va a aceptar hasta que se vacié la pila, lo cual el autómata lo hace sacando "$a"'s$ de 5 en 5 para cumplir con el objetivo, si se queda sin "$a"'s$ en medio de la vuelta para volver al estado final el autómata rechazará como se explicó al principio.
Finalmente el autómata va a aceptar el lenguaje pedido, pues la cantidad de "$a"'s$ que entran deben ser igual a las que salen (para que cumpla la igualdad) y la cantidad de "$a"'s$ que entran y salen son las descritas en el lenguaje dado que por cada "1" se agrega un "$a$", por cada "2" se agregan 2 "$a"'s$ y por cada "5" despues del " = " se sacan exactamente 5 "$a"'s$.

# P3

**1.** $\{0^n1^n0^n1^n | n \geq 0\}$

Para demostrar que este lenguaje no es de libre contexto usaremos el lema del bombeo, probando que este no lo cumple y que por lo tanto no es un LLC.
Primero, tomaremos un "$l$" arbitrario perteneciente al lenguaje tal que $l = uvxyz$, $|vy| \geq 1$, $|l| > N$, $|vxy| \leq N$, con "$N$" el largo del bombeo y mayor o igual a 1.
Ahora, por demostrar que existe un bombeo "$i$" tal que $uv^ixy^iz$ se sale del lenguaje, no importando el "l.º la partición de este, nos encontraremos con dos casos.
El primero caso es cuando "$v$" o "$y$" posee al menos un 1 y un 0. En este caso vemos que si bombeamos a mas de 1 ($i > 1$) el patron con el cual el lenguaje debe cumplir se rompe, pues el lenguaje pide que se repita esta secuencia, $1's, 0s, 1s, 0s$, lo cual si bombeamos y repetimos el "$v$" o "$y$" (que al menos uno de ellos posee 1 y 0), rompemos la condición pues estariamos agregando un $1's, 0's$ por cada bombeo que aumentamos.
El otro caso es cuando ambos, "$v$" y "$y$", poseen solo un carácter cada uno. En este caso también basta con usar un bombeo distinto de 1 ($i \neq 1$) puesto que cuando $i = 1$ la palabra cumple que su estructura de $1's, 0s, 1s, 0s$ y cada corrida, de las 4, del mismo numero tiene que poseer el mismo largo que las otras, si usamos otro bombeo rompe con esta estructura porque provocaría que una corrida sea mas larga o corta que otra y dejaría de pertenecer al lenguaje.
Como se mostró que este lenguaje, no importando el "l"ni como particionemos este mismo, no cumple el lema del bombeo para LLC decimos que este lenguaje no es de libre contexto.

**2.** $\{t_1t_2...t_k | k \geq 2$, cada $t_i \in \{a, b\}^*$, y $t_i = t_j$ para algún $i \neq j\}$

En la demostración de que este lenguaje no es de libre contexto haremos lo mismo que en el caso anterior, probando que este lenguaje no cumple el lema del bombeo y que por lo tanto no es un LLC.
Tomaremos un "$l$" arbitrario perteneciente al lenguaje tal que $l = uvxyz$, $|vy| \geq 1$, $|l| > N$, $|vxy| \leq N$, con "$N$" el largo del bombeo y mayor o igual a 1.
Ahora, al igual que en el caso anterior notamos que exiten dos grandes casos de como puede ser nuestra partición de "$l$" independiente del "l"mientras pertenezca al lenguaje.
El primero, es cuando $vxy$ contiene el símbolo , con lo cual, si el  está en "$v$" o "$y$" basta con bombear a 0 para que "$l$" se salga del lenguaje porque los $t_i$ deben estar separados por , por otro lado si  está en "$x$" sabemos que hay un bombeo distinto de 1 tal que rompe la condición de "$t_i = t_j$ para algún $i \neq j$".

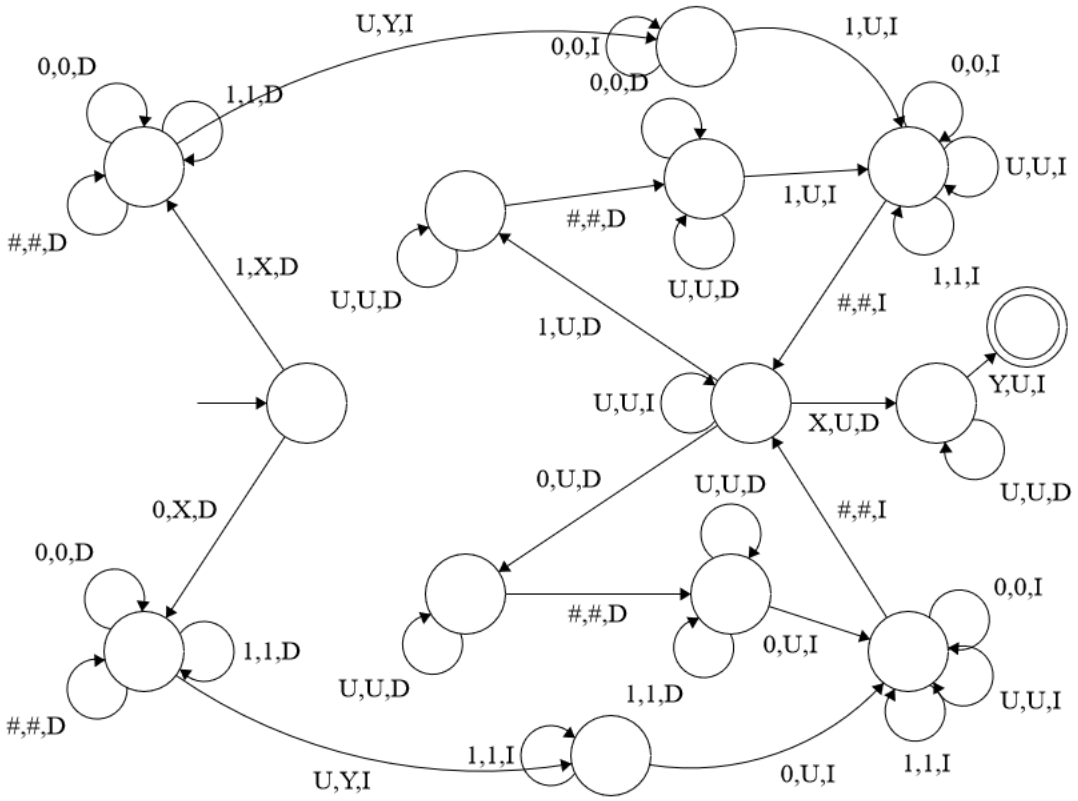
Para el otro caso tenemos que "$vxy$" está dentro de un $t_i$, en este caso también basta con bombear distinto de 1 ($i \neq 1$) puesto a que solo modificamos un $t_i$ y tambien rompemos la condición de "$t_i = t_j$ para algún $i \neq j$"del lenguaje.
Por lo anterior sabemos que no hay particiones para un "$l$" arbitrario del lenguaje con el que se cumpla el lema del bombeo para LLC, sigue que este lenguaje no es de libre contexto.

## P4

Por demostrar que los lenguajes de libre contexto son cerrados bajo el operador SUFFIX, diremos que para SUFFIX(A) existe un autómata de pila que reconoce SUFFIX(A) con A un lenguaje de libre contexto.
Para conseguir dicho autómata crearemos uno a partir del automata X que reconoce el lenguaje A (sabemos que existe pues A es un LLC) y otro Y exactamente igual. Luego unimos estos dos autómatas de pila en uno P que comienzo donde comienza el autómata Y (con el fin de analizar lo anterior al sufijo primero), ahora hacemos que las transiciones en Y no avanzen en el input (puesto que aun tenemos que simular que avanzamos en la parte anterior del sufijo antes de analizar el sufijo) cambiando en la transición la parte del input por un $\epsilon$. Para analizar el "$v$" (el sufijo que damos por input al autómata) haremos que cada transición del sub-autómata Y vaya sin avanzar en el input, ni agregando ni sacando nada de la pila, a su correspondiente en el sub-autómata X (con transiciones con solo $\epsilon$), así el sub-autómata X procedera a avanzar en el input "$v$" de tal manera que el sufijo solo aceptará si está en A.
Como se explicitó la existencia de un autómata de pila que acepta los sufijos de A, siendo A un LLC cualquiera, se muestra que los lenguajes de libre contexto son cerrados bajo el operador SUFFIX.

## P5

Para la maquina de Turing se utilizará la siguiente convención, cada transición tendrá al menos una 3-tupla (a,b,c), donde "$a$" será lo que lee el cabezal en el ese momento del input, "$b$" será lo que escribe el cabezal y "$c$" será la dirección a donde seguirá el cabezal, siendo "$D$" derecha y "$I$", si el input actual no está presente en las transiciones del estado actual o si la maquina entrá en un loop o si se va hacia el ïnfinito"derecho de la cinta la maquina lo llevará al estado sumidero para posterior rechazó.
Esta es la maquina de Turing que reconoce el lenguaje pedido (disculpas si quizás no es tan eficiente o si está desordenado), otra cosa que mencionar es que en la maquina los "$U$" son $\sqcup$ (no pude escribir dicho símbolo en la página que utilicé para hacer el dibujo). La idea tras la máquina es que marque un carácter a la izquierda y a su vez el mismo en la derecha y que verifique si poseen la misma cantidad de cada carácter.




La maquina comienza con dos posibles caminos (uno para cada posible carácter), ambos semejantes, asi que sin perdida de generalidad la máquina lee uno de los carácters y marca con una "$X$" con el fin de marcar el inicio del string y utilizarlo par cuando se acepte una palabra, ahora la maquina con un carácter marcado irá hasta la derecha hasta encontrarse con un $\sqcup$ que le indica que se acabó el string y ya está en la cinta sin tocar, en esta situación la maquina marca con un "Y"para indicar que hasta ahí llega el string. Ahora con el string acotado a ambos lados, la máquina procede a ir marcando un caracter en ambos lados del , primero marca el carácter con el que empezó, luego de eso se devuelve hasta el otro lado del  para ver si quedan 1's o 0's, si encuentra uno lo marca en los dos lados del  y luego vuelve a la izquierda. Si la maquina cuando vuelva a la izquierda solo encuentra "$U$"$'s$ (es decir$\sqcup$) llegará al "$X$" inicial y así indicará que no quedan carácteres por marcar en la parte izquierda. Ahora la maquina empieza a devolverse hasta la derecha leyendo $\sqcup$ y marcando con $\sqcup$ hasta llegar al "Y"del final del string, notar que si se encuentra con un 1 o 0 la máquina no tiene transición para eso por lo que, como mencioné al principio, la máquina rechazará el input y solo aceptará cuando luego de marcar a ambos lados de igual manera solo queden "$U$"$'s$" ($\sqcup's$)