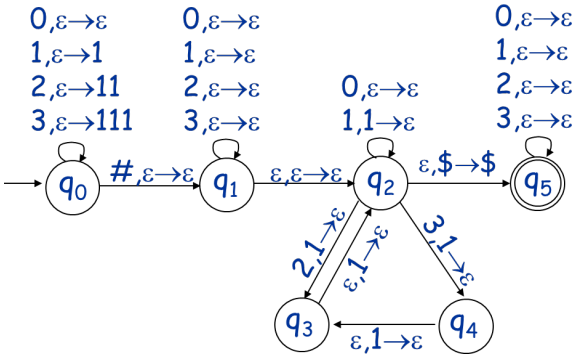
Tarea #3 - Resolución

1. **[13 pt]** Consideremos el alfabeto $\Sigma = \{0, 1, 2, 3, \#\}$, y el lenguaje formado por todas las palabras de la forma $u\#v$, donde $u, v \in \{0, 1, 2, 3\}^+$ son dos palabras tales que v contiene alguna subpalabra que suma lo mismo que lo que suma u. Por ejemplo, 23#021101121301 está en el lenguaje, pues la segunda parte contiene 21101, que suma 5, y esa es la suma de la primera parte. 23#31331 en cambio no estaría, pues ninguna subpalabra de la segunda parte suma 5.

   Construya un autómata de pila que reconozca este lenguaje.

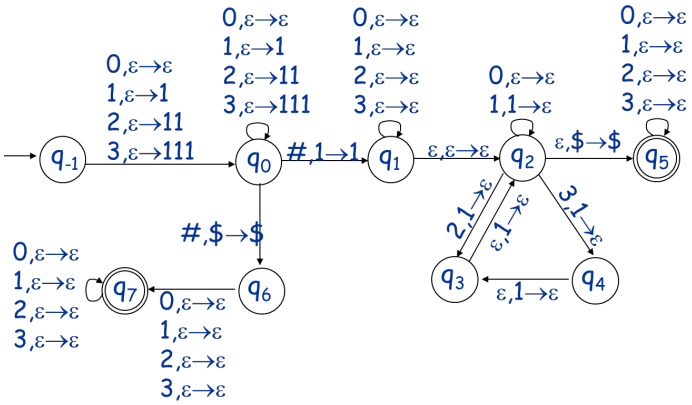
   R:

   Acá va uno:



   Primero en q_0 apilamos la suma de lo que viene antes del #. Después, en q_1, saltamos los dígitos que queramos, hasta que de manera no determinista decidimos que hemos llegado a la zona que queremos que sume lo deseado, y nos vamos a q_2. Allí desapilamos; los 0 se pueden ignorar, por cada 1 saco un 1 de la pila, y para sacar 2 y 3 de la pila (al leer un 2 ó un 3, respectivamente) usamos los estados extra q_3 y q_4. Si logramos vaciar la pila, significa que los dígitos lograron sumar lo que queríamos, con lo que la condición se cumplió y podemos irnos a q_5, de aceptación, donde además podemos seguir leyendo cualquier dígito extra que venga.

   Update: Me hicieron notar que el caso de suma cero podía dar problemas... Pero ojo, si la suma es cero, entonces sí o sí estaremos en el lenguaje, porque la subpalabra "ε" (que inevitablemente es subpalabra de lo que venga después del #) cumpliría. Y es subpalabra legítima, según las definimos en su momento en el ppt. Lo que sí noté a raíz de eso, es que el PDA no está chequeando que el u y el v sea no nulos (que era parte del enunciado). La forma más simple que se me ocurrió para parchar eso es:



El q_{-1} es para asegurarnos de que u no sea nulo. Por otro lado, la salida hacia q_6 cubre el caso de suma 0, asegurándonos de aceptar pero chequeando antes que v no sea nulo. La salida hacia q_1 es ahora para todos los casos de suma mayor que 0 (y ahí sí o sí tendremos que leer algún símbolo para llegar a q_5, así que v no podrá ser nulo).

2. **[30 = 6×5 pt]** Considere la gramática, de variable inicial S:

$$\begin{aligned} S &\to ZSb \mid ZZTab \mid Xb \mid T \\ T &\to Xa \mid \varepsilon \\ U &\to YUY \\ V &\to XabTc \\ W &\to b \mid aY \mid YZ \\ X &\to ZT \mid ZUT \\ Y &\to Uc \mid cU \mid \varepsilon \\ Z &\to a \mid Yb \mid WY \end{aligned}$$

   (a) Elimine producciones nulas.

   (b) Elimine producciones unitarias.

   (c) Elimine variables redundantes.

   (d) Elimine variables inútiles.

   (e) Pase la gramática a forma normal de Chomsky.

   (f) Describa el lenguaje generado en e.q.s.a.e.

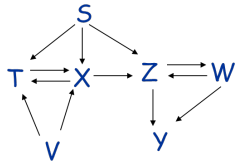
   R:

   Primero que nada, haré la revisión de variables inútiles, para no trabajar de sobra. Eso sí: de todos modos habrá que hacerla de nuevo después.

   ¿Hay variables que no se vayan? Hacemos la bolsita de cosas útiles, Uti= $\{a, b, c\}$, y revisamos. En primera vuelta, se agregan, en ese orden, T (por su producción ε, que le permite fabricar palabras), W (por su producción b), Y (por su producción ε) y Z (por su producción a, o por la Yb). Con eso, ahora Uti= $\{a, b, c, T, W, Y, Z\}$. En segunda ronda se agregan además S (por su producción $ZZTab$, o por la T) y la X (por su producción ZT). Recién en la tercera ronda agregamos V, y en la cuarta ya no se agrega nada. La que quedó fuera es U, y la elimino. [Nota: sí, es legítimo también si a ojo dicen "elimino U porque no se va", sin tanto detalle.] Nos queda

$$\begin{aligned} S &\to ZSb \mid ZZTab \mid Xb \mid T \\ T &\to Xa \mid \varepsilon \\ V &\to XabTc \\ W &\to b \mid aY \mid YZ \\ X &\to ZT \\ Y &\to \varepsilon \\ Z &\to a \mid Yb \mid WY \end{aligned}$$

   Ahora, variables a las que nadie llama. Es más o menos fácil ver que la inútil en ese sentido es la V. De todos modos, acá está el grafo de llamadas entre variables:



donde es claro que la única inaccesible desde S es la V. La quitamos entonces:

$$\begin{aligned} S &\to ZSb \mid ZZTab \mid Xb \mid T \\ T &\to Xa \mid \varepsilon \\ W &\to b \mid aY \mid YZ \\ X &\to ZT \\ Y &\to \varepsilon \\ Z &\to a \mid Yb \mid WY \end{aligned}$$

Nótese, por si acaso, que de las tres letras que parecía tener el alfabeto de la gramática, quedan sólo a y b: no se generan palabras que incluyan la c.

(a) Directamente anulables son T e Y. Indirectamente, la S (por su producción T) y nadie más. Como S es inútil, agregamos el S_0 de rigor. Además eliminamos las producciones ε, y agregamos las versiones anuladas de las producciones que incluyen S, T ó Y:

$$\begin{aligned} S_0 &\to S \mid \varepsilon \\ S &\to ZSb \mid Zb \mid ZZTab \mid ZZab \mid Xb \mid T \\ T &\to Xa \\ W &\to b \mid aY \mid a \mid YZ \mid Z \\ X &\to ZT \mid Z \\ Y &\to \\ Z &\to a \mid Yb \mid b \mid WY \mid W \end{aligned}$$

Epa. ¿Qué pasó aquí? ¡La Y se quedó sin producciones! Si queremos seguir mecánicamente el algoritmo, podríamos dejarla donde está y se eliminaría después cuando nos demos cuenta de que ahora es inútil (pues es de las que no se van, al no poder producir palabras). Pero es demasiado obvio así que quitémosla altiro:

$$\begin{aligned} S_0 &\to S \mid \varepsilon \\ S &\to ZSb \mid Zb \mid ZZTab \mid ZZab \mid Xb \mid T \\ T &\to Xa \\ W &\to b \mid a \mid Z \\ X &\to ZT \mid Z \\ Z &\to a \mid b \mid W \end{aligned}$$

(b) Las producciones unitarias son $S \to T$, $W \to Z$, $X \to Z$ y $Z \to W$. Indirectamente eso además nos da la derivación unitaria $X \to^* W$. Eliminemos las producciones y copiemos todo lo pertinente:

$$\begin{aligned} S_0 &\to S \mid \varepsilon \\ S &\to ZSb \mid Zb \mid ZZTab \mid ZZab \mid Xb \mid Xa \\ T &\to Xa \\ W &\to b \mid a \\ X &\to ZT \mid a \mid b \\ Z &\to a \mid b \end{aligned}$$

(c) Claramente W y Z hacen lo mismo; quitaré la Z por redundante:

$$S_0 \to S \mid \varepsilon$$

$$
\begin{aligned}
S &\to WSb \mid Wb \mid WWTab \mid WWab \mid Xb \mid Xa \\
T &\to Xa \\
W &\to b \mid a \\
X &\to WT \mid a \mid b
\end{aligned}
$$

(d) Si hubiésemos andado trayendo aún la U, la V o la Y, sería el momento de eliminarlas. Como no están, la verdad es que no hay nada que hacer: un rápido nuevo chequeo muestra que todas las que quedan son útiles.

(e) Primero agreguemos A y B para resolver símbolos terminales problemáticos:

$$
\begin{aligned}
S_0 &\to S \mid \varepsilon \\
S &\to WSB \mid WB \mid WWTAB \mid WWAB \mid XB \mid XA \\
T &\to XA \\
W &\to b \mid a \\
X &\to WT \mid a \mid b \\
A &\to a \\
B &\to b
\end{aligned}
$$

Ahora, para las cadenas largas, necesitamos variables auxiliares. Una opción sería

$$
\begin{aligned}
S_0 &\to S \mid \varepsilon \\
S &\to WV_1 \mid WB \mid V_2V_4 \mid V_2V_3 \mid XB \mid XA \\
T &\to XA \\
W &\to b \mid a \\
X &\to WT \mid a \mid b \\
A &\to a \\
B &\to b \\
V_1 &\to SB \\
V_2 &\to WW \\
V_3 &\to AB \\
V_4 &\to TV_3
\end{aligned}
$$

(f) El eqsae está un poco más complejo de lo que pensé inicialmente, pero de poderse se puede. Primero que nada, retomaré la gramática desde la forma en que la teníamos antes de eliminar las producciones nulas:

$$
\begin{aligned}
S &\to ZSb \mid ZZTab \mid Xb \mid T \\
T &\to Xa \mid \varepsilon \\
W &\to b \mid aY \mid YZ \\
X &\to ZT \\
Y &\to \varepsilon \\
Z &\to a \mid Yb \mid WY
\end{aligned}
$$

Como X e Y tienen una sola producción cada uno, los sustituiré para tener algo más compacto:

$$
\begin{aligned}
S &\to ZSb \mid ZZTab \mid ZTb \mid T \\
T &\to ZTa \mid \varepsilon \\
W &\to b \mid a \mid Z \\
Z &\to a \mid b \mid W
\end{aligned}
$$

y como además vimos (y vemos) que W y Z son la misma cosa,

$$\begin{aligned} S &\to WSb \mid WWTab \mid WTb \mid T \\ T &\to WTa \mid \varepsilon \\ W &\to b \mid a \end{aligned}$$

Claramente W genera $(a+b)$. Por lo tanto, la recursión de T lo que genera es algo de la forma wa^n, donde $|w| = n$. O sea: una palabra donde la segunda mitad son a. Abusando un poco de la notación, lo anotaré como $W^n a^n$ (entendiendo que la W se convierte en a o en b). Veamos ahora qué pasa con la S. Tiene una llamada recursiva (su primera producción), de la tendrá que salir en algún momento saliendo usando alguna de las otras tres producciones. Mientras está haciendo la llamada recursiva, va generando algo de la forma $W^m S b^m$, para algún m. Supongamos que sale de ahí usando $S \to T$; en tal caso, por lo que vimos antes sobre la T, la palabra generada será de la forma $W^m W^n a^n b^m = W^{m+n} a^n b^m$, para algún $n, m \geq 0$. ¿Qué pasa si en lugar de eso, sale de la recursión usando $S \to WTb$? Pues entonces generará algo de la forma $W^m W W^n a^n b b^m = W^{m+n+1} a^n b^{m+1}$, que estaba cubierto en el caso anterior. Y si sale usando $S \to WWTab$? Generará $W^m WWW^n a^n abb^m = W^{m+n+2} a^{n+1} b^{m+1}$... ¡también cubierto en el caso anterior!

Por lo tanto (y gracias, abuela, por la paciencia), el lenguaje consiste en palabras de largo par formadas por a y b, en las cuales la segunda mitad consiste en un bloque de a's seguido por un bloque de b's.

(Si alguien le achuntó sin mucho análisis, vale. El análisis acá es principalmente para convencerlos.)

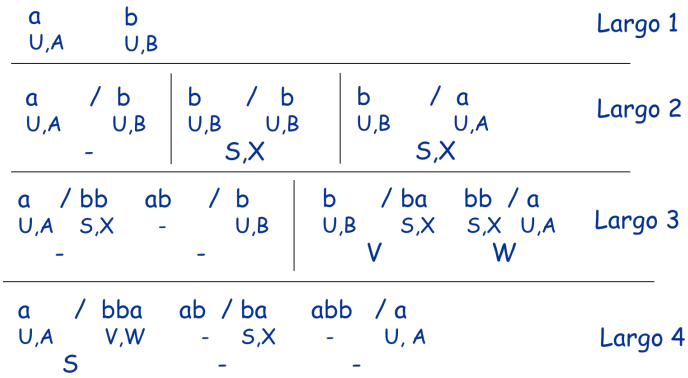
3. **[14 pt]** Considere la gramática de variable inicial S definida por

$$\begin{aligned} S &\to AW \mid VU \mid BU \\ X &\to VU \mid BU \\ U &\to a \mid b \\ V &\to BX \\ W &\to SU \\ A &\to a \\ B &\to b \end{aligned}$$

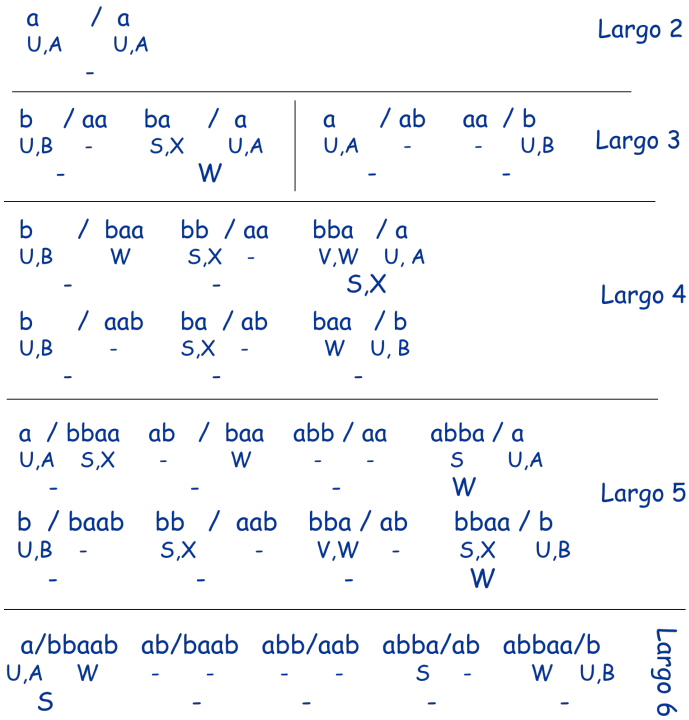
Use el algoritmo CYK para determinar acaso las palabras *abba* y *abbaab* pertenecen al lenguaje generado.

R:

La verdad es que esta gramática genera el mismo lenguaje de la pregunta anterior, sólo que transpuesto. Pero olvidé que lo había traspuesto, y puse dos palabras que están en el lenguaje (usualmente en los ejercicios de CYK pongo una que sí y una que no). En fin, acá van las tablitas. La primera, para *abba*, es



y por lo tanto *abba* está en el lenguaje. Para *abbaab* (obviando las que ya están en la tabla anterior) tenemos la que sigue, y que muestra que también está en el lenguaje.



4. **[11 = 4+3+4 pt]** Considere la gramática G de variable inicial S definida por

$$\begin{aligned} S &\to XS \mid SX \mid \varepsilon \\ X &\to aX \mid bS \end{aligned}$$

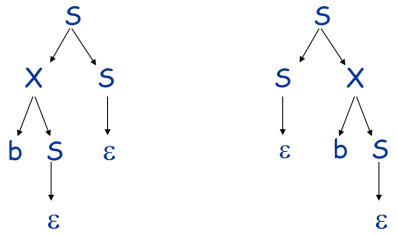
(a) Muestre que G es ambigua.

(b) Describa $L(G)$ en e.q.s.a.e.

(c) Escriba una gramática *no ambigua* que genere $L(G)$.

R:

(a) Acá van dos árboles distintos para la palabra b:



(b) La variable S básicamente genera una cadena de tantos X como queramos. Cada X, por su parte, genera algo de la forma $a^n bS\ldots$ y esa S a su vez generará más X, o muere. Por lo tanto, las palabras serán concatenaciones de cosas como esa. El lenguaje es regular, con expresión $(a^*b)^*$. Abuelita: son las palabras que no terminan con a.

(c) Serviría, por ejemplo,

$$\begin{aligned} S &\to XS \mid \varepsilon \\ X &\to aX \mid b \end{aligned}$$

Aquí forzosamente cada b está generada por un X distinto, así que para una palabra que contenga n letras b, el árbol consistirá en n reemplazos de S por XS, un reemplazo final de esa S por ε, y bajo cada X, una llamada a aX por cada a que haya que poner entre esa b y la anterior.

5. **[32 = 4×8 pt]** ¿Contexto libre o no? Determine la respuesta en cada caso y demuéstrelo.

   (a) $L_1 = \{w \in \{a,b\}^* : w = a^m b^n a^p$, para $n, m, p \geq 0$ tales que $nmp \geq n^2\}$.

   (b) $L_2 = \{w \in \{a,b\}^* : w = a^m b^n a^p$, para $n, m, p \geq 0$ tales que nmp es múltiplo de $3\}$.

   (c) $L_3 = \{w \in \{a,b,c\}^* : |w|_a \leq 2|w|_b \leq 3|w|_c\}$

   (d) $L_4 = \{w \in \{a,b\}^* : w = ab^n a^m$ para $0 \leq m < \frac{n}{2}\}$.

   R:

   (a) NO ES DE CONTEXTO LIBRE. Primero que nada, notemos que la condición es equivalente a pedir que $n = 0$ (y en ese caso no hay restricción sobre m y p) o bien $mp \geq n$. Como el primer caso está contenido en el segundo, en realidad es esa la condición. Sea ahora N la constante de bombeo, y tomemos $w = a^N b^{N^2} a^N$, que está en el lenguaje pues tiene $m = p = N$, $n = N^2$, y la desigualdad se cumple. Si el lenguaje fuera de contexto libre el lema de bombeo garantiza una descomposición $w = uvxyz$, con v, y bombeables, $|vy| > 0$, y $|vxy| < N$. Esa última condición nos garantiza que habrá 5 casos para la ubicación del vxy: ① dentro de la primera zona (las a del comienzo), ②dentro de la segunda zona (las b), ③ dentro de la tercera zona (las a del final), ④ entre la primera y la segunda zona, y ⑤ entre la segunda y la tercera. Para lo que sigue, llamaré $|vy|_a = p$, $|vy|_b = q$.

   - Caso ①: Aquí $p > 0$, $q = 0$. Bombeamos con $k = 0$, obteniendo $a^{N-p} b^{N^2} a^N$. El producto de los largos de los bloques de a será $(N-p)N < N^2$, así que salimos del lenguaje.
   - Caso ②: Aquí $p = 0$, $q > 0$. Bombeamos con $k = 2$, obteniendo $a^N b^{N^2-q} a^N$. El producto de los largos de los bloques de a será N^2, pero el bloque de b es de largo $N^2 - q < N^2$. Así que salimos también de L_1.
   - Caso ③: Simétrico al ①, también bombeamos con $k = 0$.
   - Caso ④: Aquí tanto p como q son mayores que cero. Bombeamos con $k = 0$ y obtenemos $a^{N-p} b^{N^2-q} a^N$. Para estar en el lenguaje, necesitaríamos que $(N-p)N \geq N^2 - q$. Pero entonces (despejando) $pN \leq q$, lo cual es imposible porque $p \geq 1$, por lo que tendríamos $N \leq pN \leq q$, y q no puede ser mayor que N (pues $p + q = |vy| < N$). Así que no puede ser; ergo, salimos del lenguaje.
   - Caso ⑤: Simétrico al anterior.

   Como en todos los casos podemos bombear y salir del L_1, no es de contexto libre.

   (b) SÍ ES DE CONTEXTO LIBRE. Basta notar que para que se cumpla la condición, basta y sobra con que cualquiera de las tres cantidades sea múltiplo de 3. De modo que el lenguaje es regular (y por lo tanto de contexto libre), con expresión regular

   $$(aaa)^* b^* a^* + a^* (bbb)^* a^* + a^* b^* (aaa)^*$$

   (c) NO ES DE CONTEXTO LIBRE. Sea N la constante de bombeo y tomemos $w = a^{6N} b^{3N} c^{2N}$, que está en L_3.

   Versión rápida: para los casos en que vxy está en la primera o segunda zona, o montado entre la primera y la segunda, bombeamos hacia arriba y así violamos la condición $|w|_a \leq 3|w|_c$ o bien $2|w|_b \leq 3|w|_c$. Si está entre la segunda y tercera zona, o dentro de la tercera, bombeamos hacia abajo ($k = 0$) y con eso violamos $|w|_a \leq 3|w|_c$.

   Versión más detallada: Numeremos los casos igual que en la parte (a), aprovechando que hay tres zonas también.

- Caso ①: Aquí $p > 0$, $q = 0$. Bombeamos con $k = 2$, obteniendo $a^{6N+p}b^{3N}c^{2N}$. Con eso $|w|_a = 6N + p > 6N = 3|w|_c$, así que salimos del lenguaje.
- Caso ②: Aquí $p = 0$, $q > 0$. Bombeamos con $k = 2$, obteniendo $a^{6N}b^{3N+q}c^{2N}$. Con eso $2|w|_b = 6N + 2q > 6N = 3|w|_c$, así que salimos del lenguaje.
- Caso ③: Acá volvemos a tener $p > 0$, $q = 0$. Bombeamos hacia abajo ($k = 0$), obteniendo $a^{6N}b^{3N}c^{2N-p}$. Con eso $2|w|_b = 6N > 6N - 3p = 3|w|_c$, así que salimos del lenguaje.
- Caso ④: Tenemos $p > 0$, $q > 0$. Bombeamos hacia arriba, con $k = 2$, y obtenemos $a^{6N+p}b^{3N+q}c^{2N}$. Con eso $|w|_a = 6N + p > 6N = 3|w|_c$, así que salimos del lenguaje.
- Caso ⑤: Tenemos $p > 0$, $q > 0$. Bombeamos hacia abajo, con $k = 0$, y obtenemos $a^{6N}b^{3N-q}c^{2N-p}$. Con eso $|w|_a = 6N > 6N-3p = 3|w|_c$, así que salimos del lenguaje.

(d) SÍ ES DE CONTEXTO LIBRE. Sirve, por ejemplo, la gramática

$$\begin{aligned} S &\to aYZ \\ Y &\to bY \mid \varepsilon \\ Z &\to bbZa \mid \varepsilon \end{aligned}$$

o, para PDA-lovers,