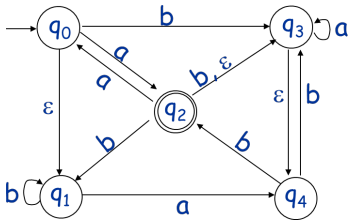
Tarea #2 - Resolución

1. **[33 = 3×11 pt]** A partir del siguiente AFND+ε, de alfabeto $\{a, b\}$,



   (a) Obtenga una expresión regular equivalente, u.e.a.q.v.e.c.[1]. Muestre sus pasos intermedios.

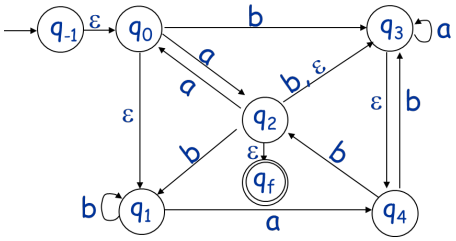
   (b) Obtenga un AFD, u.e.a.q.v.e.c.. Precise a qué subconjunto de estados del AFND+ε corresponde cada estado del AFD.

   (c) Minimice el AFD resultante, u.e.a.q.v.e.c. Muestre los pasos intermedios y la tabla final de palabras que distinguen los estados.

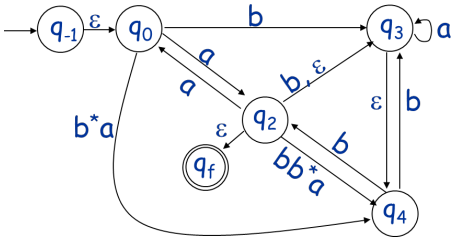
   (d) **[Bonus, 6pt]** Describa el lenguaje en e.q.s.a.e.[2]

   R:

   (a) Primero que nada, normalicemos:

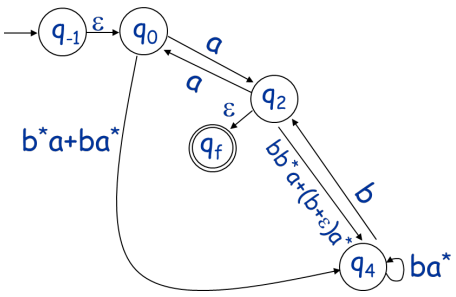


   Ahora partimos eliminando q_1, donde se forman sólo dos flechas (en todos los otros serían más):

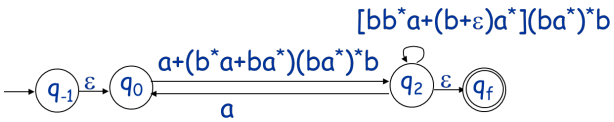


   Luego le toca a q_3

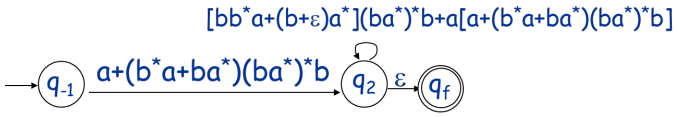
[1] Usando el algoritmo que vimos en clases
[2] Español que su abuelita entendería ©



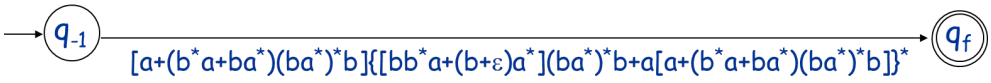
y a q_4



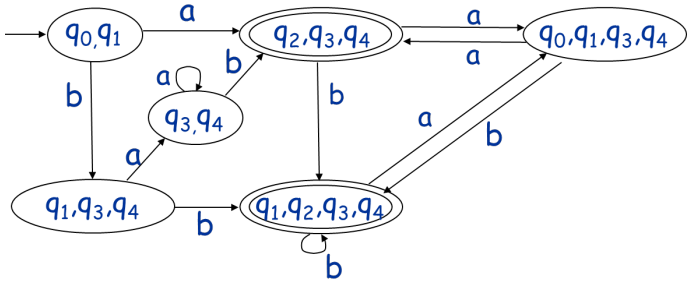
Ahí la cosa se pone un poco fea, pero en fin. Le toca a q_0:



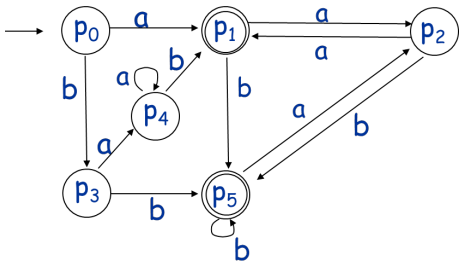
y finalmente...



(b) Se obtiene



(c) Primero que nada renombraré por comodidad:



Ahora hacemos la tabla, y marcamos con ε los pares de aceptación/no-aceptación:

p_1	ε				
p_2		ε			
p_3		ε			
p_4		ε			
p_5	ε		ε	ε	ε
	p_0	p_1	p_2	p_3	p_4

Y ahora revisamos (iré por filas):

- Del par (p_0, p_2) con a pasamos a p_1 en ambos casos (lo que jamás nos hará marcar) y con b al par (p_3, p_5), marcado con ε. Así que marcamos con b.
- Del par (p_0, p_3) con a pasamos a p_1 y a p_4, que está marcado con ε así que marcamos con a.
- Del par (p_2, p_3) con a pasamos a p_1 y a p_4, que está marcado con ε así que marcamos con a.
- Del par (p_0, p_4) con a pasamos a p_1 y a p_4, que está marcado con ε así que marcamos con a.
- Del par (p_2, p_4) con a pasamos a p_1 y a p_4, que está marcado con ε así que marcamos con a.
- Del par (p_3, p_4) con a pasamos a p_4 en ambos (así que eso no nos hará marcar nunca), y con b al par (p_1, p_5), no marcado. Así que al menos de momento no se marca.
- Del par (p_1, p_5) con a pasamos a p_2 en ambos casos, y con b a p_5 en ambos casos. Ergo, es inmarcable.

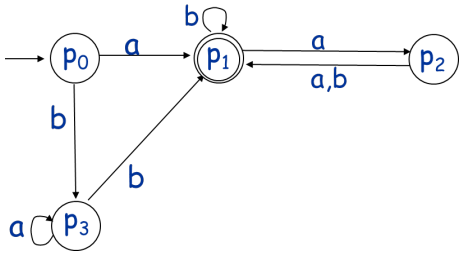
La tabla pasa a ser

p_1	ε				
p_2	b	ε			
p_3	a	ε	a		
p_4	a	ε	a		
p_5	ε		ε	ε	ε
	p_0	p_1	p_2	p_3	p_4

Revisamos de nuevo (saltándonos los que se marcaron, y los declarados inmarcables):

- Del par (p_3, p_4) con a pasamos a p_4 en ambos (así que eso no nos hará marcar nunca), y con b al par (p_1, p_5), no marcado. Así que al menos de momento no se marca.
- Del par (p_1, p_5) con a pasamos a p_2 en ambos casos, y con b a p_5 en ambos casos. Ergo, es inmarcable.

Como no marcamos nada en esta segunda pasada, terminamos. Los estados equivalentes son $p_1 \equiv p_5$, y $p_3 \equiv p_4$. El AFD mínimo queda



(d) El lenguaje está formado por tres grupos de palabras:

- Palabras formadas sólo por una cantidad impar de a's.
- Palabras que parten con a, contienen alguna b, y después de la última b tienen una cantidad par de a's.
- Palabras que parten con b, contienen al menos dos b, y después de la última b tienen una cantidad par de a's.

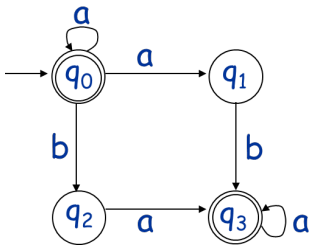
2. **[16 = 10 + 6 pt]** Considerando la siguiente expresión regular: $a^*(ab + \varepsilon + ba)a^*$

(a) Construya un AFND+ε equivalente, usando a lo más 4 estados

(b) Describa el lenguaje en e.q.s.a.e..

R:

(a) Lo más simple creo que sería



(b) Son palabras sin b, o palabras con una sola b y al menos una a.

3. **[15 = 3×5 pt]** El propósito de esta pregunta es demostrar que si es que un lenguaje es regular, entonces existe un algoritmo para determinar acaso es *exigente*[3]. Lo haremos por partes (aunque si a alguien se le ocurre una aproximación más simple que se ahorre estos pasos, tiene puntaje completo y mis felicitaciones). Nótese que las partes son independientes, así que pueden hacer alguna sin tener las previas.

   (a) **[Bonus, 6pt]** Describa un algoritmo *suf* para obtener, a partir de un AFD M, un AFD $suf(M)$ cuyo lenguaje sean todos los sufijos propios de palabras de $L(M)$.

   (b) Suponiendo que disponemos de *suf*, describa un algoritmo *pref* que entregue, dado un AFD M, un AFD $pref(M)$ cuyo lenguaje sean los prefijos propios de palabras de $L(M)$.

   (c) Suponiendo que disponemos de *suf* y de *pref*, describa un algoritmo *subp* que entregue, dado un AFD M, un AFD $subp(M)$ cuyo lenguaje sean todas las subpalabras propias de palabras de $L(M)$.

   (d) Suponiendo que disponemos de *suf*, *pref* y *subp*, muestre que es posible construir un algoritmo que decida, dado un AFD, acaso su lenguaje es exigente.

   R:

   (a) Lo que queremos es saltarnos una o más letras. Haremos esto creando un nuevo estado de inicio, y enviando flechas ε desde él a todos los estados de M... salvo que en M el estado de inicio no haya tenido flechas de entrada. En ese caso lo borramos, y mandamos las flechas ε a todos los demás.

   (b) Podemos usar el anterior. Dado M con lenguaje L, obtenemos un AFD M_R cuyo lenguaje sea L^R (dijimos en clase como lo hacemos: normalizamos, invertimos las flechas, e intercambiamos el estado de inicio con el de aceptación... En este caso además pasaríamos el resultado a determinista). Luego aplicamos a M_R el algoritmo de (a), obteniendo un autómata para los sufijos propios de L_R... que no son otra cosa que los prefijos propios de L, transpuestos. Así que transponemos de nuevo, como dijimos antes. Y listo.

   (c) Aquí la clave es notar que el conjunto de subpalabras propias lo podemos descomponer en tres grupos: las que son prefijos propios, las que son sufijos propios, y las que están "dentro" (les hemos cortado cosas por las dos puntas). Este último grupo lo podemos obtener si tomamos todos los sufijos propios de los prefijos propios (o, equivalentemente, si tomamos los prefijos propios de los sufijos propios). Así que para obtener $subp(M)$ aplicamos lo de (a) para obtener $suf(M)$, lo de (b) para obtener $pref(M)$, y luego de nuevo lo de (b) pero aplicándolo a $suf(M)$, obteniendo un tercer autómata $sufpref(M)$. Luego creamos el AF que resulta de unir esos tres, y lo pasamos a determinista.

   (d) Ser exigente se traduce en que el lenguaje de $subp(M)$ no puede tener ninguna palabra que esté en $L(M)$. Aplicamos lo de (c) para obtener el AFD $subp(M)$, lo intersectamos con M (sabemos hacer eso, con la construcción de autómata producto) y vemos acaso el resultado tiene lenguaje $\emptyset$.

[3] En el sentido definido en la tarea 1: un lenguaje es exigente si para cualquier palabra del lenguaje, ninguna de sus subpalabras propias está en el lenguaje.

4. **[36 = 4×9 pt]** ¿Regular o no? Determine la respuesta en cada caso y demuéstrelo.

   (a) $L_1 = \{w \in \{a,b\}^* : |w|$ es par y la cantidad de a's en la primera mitad es igual a la cantidad de b's en la segunda $\}$.

   (b) $L_2 = \{w \in \{a,b\}^* : |w| \geq 6$ y la cantidad de a en sus primeras 3 letras es igual a la cantidad de b en las últimas 3 $\}$.

   (c) $L_3 = \{w \in \{0,1,2,3,4,5,6,7,8,9\}^* : w$ es la edad, en horas, de algún humano vivo al momento de entregar la tarea $\}$.

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

   R:

   (a) NO REGULAR. Sea N la constante de bombeo, y tomemos un clásico $w = a^N b^N$, que cumple con la condición así que está en L_1. El lema nos garantiza una descomposición $w = xyz$ donde $|xy| < N$, por lo que $x = a^p$, $y = a^q$, $z = a^{N-p-q}b^N$, para algún $p \geq 0$, $q \geq 1$ (esto último porque también se garantiza $y \neq \varepsilon$). Bombeemos hacia abajo, con $k = 0$. La palabra bombeada queda $xy^0z = xz = a^p a^{N-p-q}b^N = a^{N-q}b^N$. Si q fuese impar, esto nos quedaría de largo impar así que salimos de L_1. Si fuese par, entonces sería $q = 2r$ para algún $r \geq 1$. La palabra bombeada, $a^{N-2r}b^N$, sería de largo $2N - 2r$, por lo que cada mitad es de largo $N - r$. La segunda mitad, entonces, sólo tendrá b, y primera mitad no podrá tener esa cantidad de a, pues no hay tantas! Así que también salimos del lenguaje. Como no se cumple el lema de bombeo, el lenguaje no es regular.

   (b) Regular. Una posible expresión regular sería $aaa(a+b)^*bbb+ (aab + aba + baa)(a + b)^*(bba + bab + abb)+ (bba + bab + abb)(a + b)^*(aab + aba + baa)+ bbb(a + b)^*aaa$.

   (c) REGULAR, porque es finito. Y no hace falta dar cotas para la cantidad de horas que alguien pueda vivir: basta con notar que la cantidad de humanos vivos en un momento dado es finita, así que la colección de sus edades es un conjunto finito también.

   (d) NO REGULAR. Esa a del comienzo complica el bombeo, pero podemos considerar L_4^R (la transposición), y como la transposición preserva la regularidad, si L_4^R no es regular entonces L_4 tampoco. Y ahí queda más cómodo; además, multiplicaré la condición por 2 (que no la afecta): $L_4^R = \{w \in \{a,b\}^* : w = a^m b^n a$ para $0 \leq 2m < n\}$.

   Sea N la constante de bombeo. Podemos tomar, por ejemplo, $w = a^N b^{2N+1}a$, que está en L_4^R. El lema nos garantiza una descomposición $w = xyz$ donde $|xy| < N$, por lo que $x = a^p$, $y = a^q$, $z = a^{N-p-q}b^{2N+1}a$, para algún $p \geq 0$, $q \geq 1$ (esto último porque también se garantiza $y \neq \varepsilon$). Si bombeamos con $k = 2$, la palabra bombeada será $xy^2z = a^p a^{2q} a^{N-p-q}b^{2N+1}a = a^{N+q}b^{2N+1}a$, que se sale de L_4^R. ¿Por qué? Porque ahí $m = N + q$, $n = 2N + 1$, así que $2m = 2N + 2q > 2N + 1 = n$ (y la desigualdad del medio se la debemos a que $q \geq 1$).