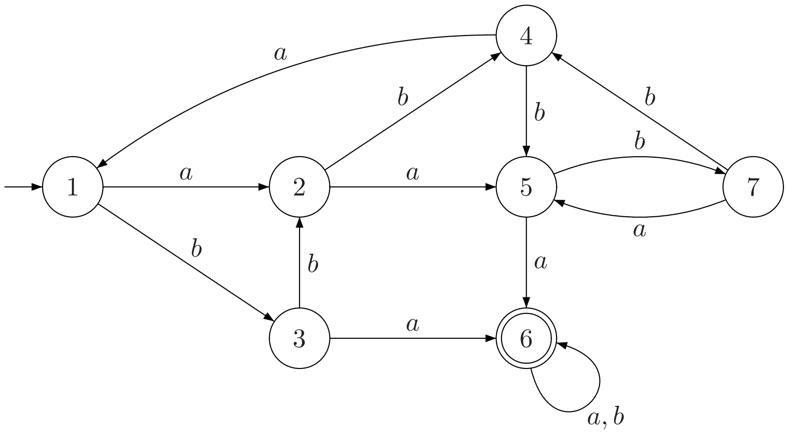
Teoría de Autómatas y Lenguajes Formales
Ingeniería Técnica en Informática de Sistemas
Facultad de Informática, UCM. Examen final, Septiembre 2005

1. **(1.5 puntos)** Aplica un algoritmo automatizable para minimizar el siguiente AFD. Para cada pareja que marques como distinguible, la notación que utilices debe permitir averiguar claramente *por qué* y *en qué momento* consideraste ese par de estados como distinguible. Si la notación que utilizas no es alguna de las utilizadas en clase, explícala.



2. **(1 punto)** Aplica algún algoritmo automatizable para hallar un AFN-λ que reconozca el lenguaje caracterizado por la expresión regular $(a^*b^*(c+d))^* + (ab)^*$.

3. **(3 puntos)** Para cada uno de los siguientes lenguajes, indica si es regular o no, así como si es incontextual o no.

   (1) Si consideras que es regular e incontextual, muestra una expresión regular que lo caracterice.

   (2) Si consideras que es no es regular pero es incontextual, demuestra que no es regular aplicando el lema del bombeo para lenguajes regulares **o bien** mostrando un conjunto infinito de cadenas distinguibles dos a dos (y mostrando por qué cualquier pareja del conjunto es distinguible). **Además**, muestra una gramática incontextual que caracterice el lenguaje.

   (3) Si consideras que no es regular ni incontextual, demuestra que no es incontextual aplicando el lema del bombeo para lenguajes incontextuales.

   (4) Si consideras que es regular y no incontextual, escribe cincuenta veces *"No he estudiado mucho"* en una hoja aparte.

   Los lenguajes a analizar son los siguientes:

   (a) $L_1 = \{a^i b^j | i$ y j se diferencian como mucho en 2 unidades (a favor de cualquiera)$\}$

   (b) $L_2 = \{a^i b^j c^k | i, j$ y k son todas impares a la vez o todas pares a la vez$\}$

   (c) $L_3 = \{a^i b^j c^k | i+j+k$ es un cuadrado perfecto$\}$

   (d) $L_4 = \{a^i b^j c^k | i \geq j + k\}$

4. **(1 punto)** Muestra un autómata con pila determinista por estado final que reconozca el lenguaje $L = \{c^n(ba)^m | n > m \geq 0\}$.

5. **(2 puntos)** Consideremos el alfabeto $\Sigma = \{a, b\}$. Sea L el conjunto de todas las palabras que se pueden crear con dicho alfabeto tales que la diferencia entre el número de a's y el número de b's es impar (a favor de cualquiera de las dos cantidades). Por ejemplo, $bbaabaabbbb \in L$ porque hay 4 a's y 7 b's, por lo que la diferencia entre ambas cantidades es impar (3). Además, $ababab baa \in L$ porque hay 5 a's y 4 b's, por lo que la diferencia es impar (1). Por otro lado, $abbabaaa \notin L$ y $\lambda \notin L$ porque la diferencia es par en ambos casos (2 y 0 respectivamente).

   Crea una Máquina de Turing que reconozca el lenguaje L. Dado que sólo queremos una MT que reconozca el lenguaje, no estaremos interesados en que el contenido final de la cinta sea uno determinado, ni tampoco en que el cursor apunte al final a una casilla determinada de la cinta. Lo único que nos interesa es que la MT pare en un estado de aceptación si la palabra introducida está en L, y que pare en un estado de no aceptación en caso contrario.

6. **(1.5 puntos)** Responde a las siguientes cuestiones:

   (a) Sea Q el conjunto de todos los autómatas finitos deterministas tales que no hay ninguna transición en la que el estado de origen y el de destino coincidan (es decir, tales que no hay bucles de un solo paso). Sea LQ el conjunto de todos los lenguajes que pueden reconocerse con alguna máquina en Q. ¿Se cumple $LQ = LR$? Si es cierto, explica por qué la restricción considerada no impide reconocer ningún lenguaje de LR. Si no es cierto, muestra un lenguaje L tal que $L \in LR$ y $L \notin LQ$.

   (b) Sea M el conjunto de todos los autómatas con pila que jamás almacenan más de 10 elementos en su pila. Es decir, no existe ninguna entrada tal que, en algún momento, la pila del autómata contenga 11 símbolos o más. Sea LM el conjunto de todos los lenguajes que pueden reconocerse con alguna máquina en M.

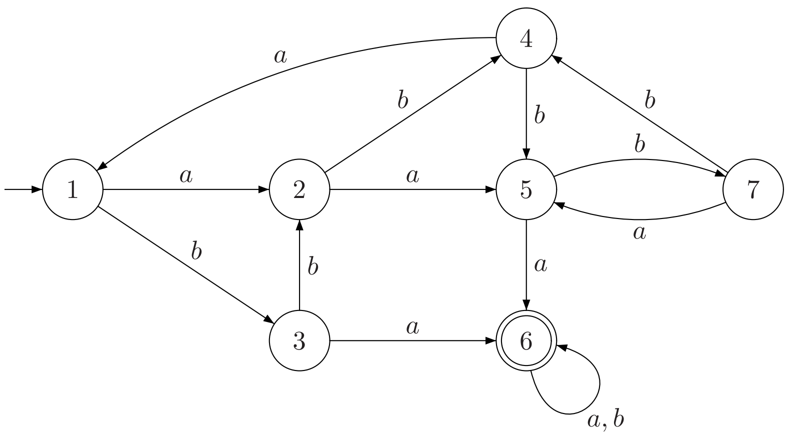
   (b.1) Si existe, muestra un lenguaje L_1 tal que $L_1 \in LI$ pero $L_1 \notin LM$, y explica por qué L_1 no pertenece a LM. Si no existe tal lenguaje, explica por qué.

   (b.2) Si existe, muestra un lenguaje L_2 tal que $L_2 \in LI$ y $L_2 \in LM$, y explica por qué L_2 pertenece a LM. Si no existe tal lenguaje, explica por qué.

   (b.3) Si existe, muestra un lenguaje L_3 tal que $L_3 \in LM$ pero $L_3 \notin LR$, y explica por qué L_3 pertenece a LM pero no a LR. Si no existe tal lenguaje, explica por qué.

Solución

1. Autómata inicial:



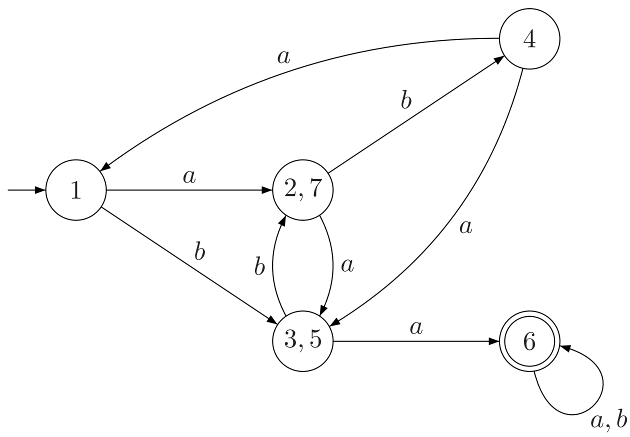
Utilizaremos el método de las listas visto en clase. La notación usada será también la utilizada en clase.

2	3,4 (2)					
3	2,6 (9)	2,4 (7)				
4	1,2 (5)	4,5 (3)	1,6 (1)			
5	2,6 (9)	4,7 (8)		1,6 (1)		
6	X (1)	X (9)	X (18)	X (4)	X (19)	
7	3,4 (2)		2,4 (7)	4,5 (3)	4,7 (8)	X (10)
	1	2	3	4	5	6

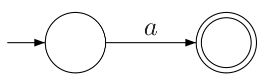
$$
\begin{array}{rlcl}
1: & (1,6) & \rightarrow & [(3,4),(4,5),(4,6)] \\
2: & (3,4) & \rightarrow & [(1,2),(1,7)] \\
3: & (4,5) & \rightarrow & [(2,4),(4,7)] \\
4: & (4,6) & \rightarrow & [(2,6),(6,7)] \\
5: & (1,2) & \rightarrow & [(1,4)] \\
6: & (1,7) & \rightarrow & [] \\
7: & (2,4) & \rightarrow & [(2,3),(3,7)] \\
8: & (4,7) & \rightarrow & [(2,5),(5,7)] \\
9: & (2,6) & \rightarrow & [(1,3),(1,5),(1,6),(3,6)] \\
10: & (6,7) & \rightarrow & [(5,6)] \\
11: & (1,4) & \rightarrow & [] \\
12: & (2,3) & \rightarrow & [(1,3)] \\
13: & (3,7) & \rightarrow & [(1,5)] \\
14: & (2,5) & \rightarrow & [(1,2),(1,7),(3,4)] \\
15: & (5,7) & \rightarrow & [(4,5)] \\
16: & (1,3) & \rightarrow & [] \\
17: & (1,5) & \rightarrow & [(2,4),(4,7)] \\
18: & (3,6) & \rightarrow & [(1,6)] \\
19: & (5,6) & \rightarrow & [(2,3),(2,5),(2,6),(3,7),(5,7),(6,7),(4,6)]
\end{array}
$$

Llegados a este punto, ya no quedan pares sin tratar.

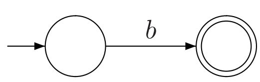
Al terminar, sólo los pares $(3,5)$ y $(2,7)$ permanecen como indistinguibles. Por tanto, el AFD mínimo resultante es el siguiente:



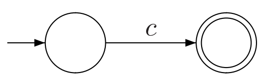
2.
- a:



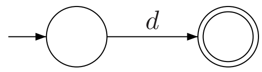
- b:



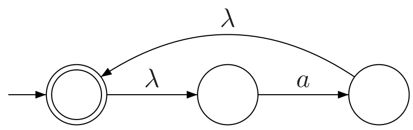
- c:



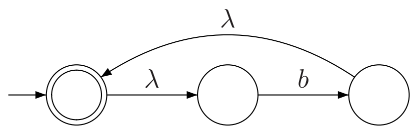
- d:



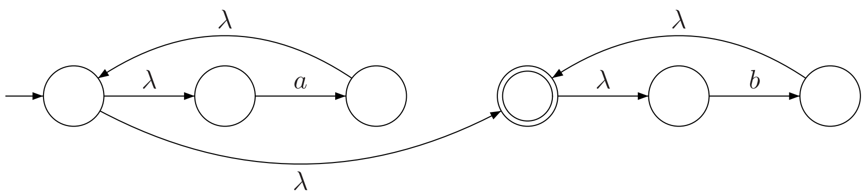
- a^*:



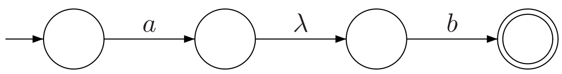
- b^*:



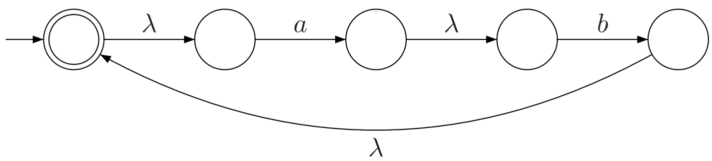
- a^*b^*:



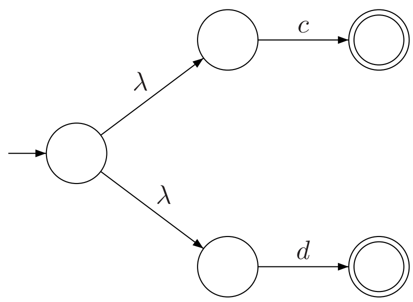
- ab:



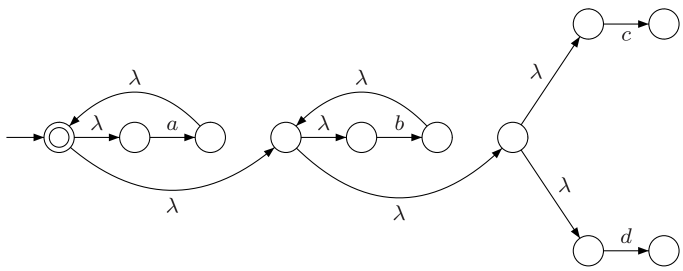
- $(ab)^*$:



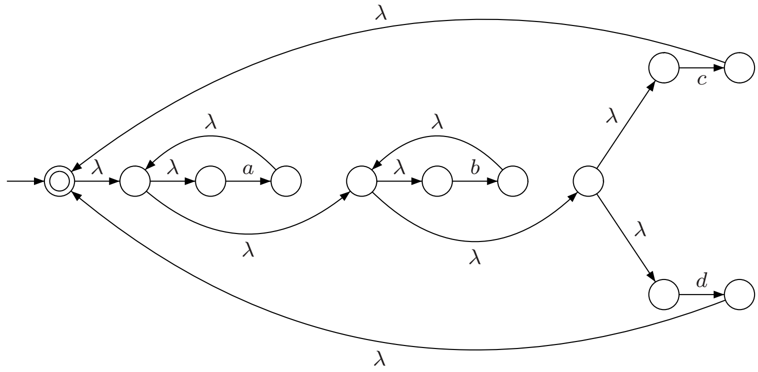
- $c + d$:



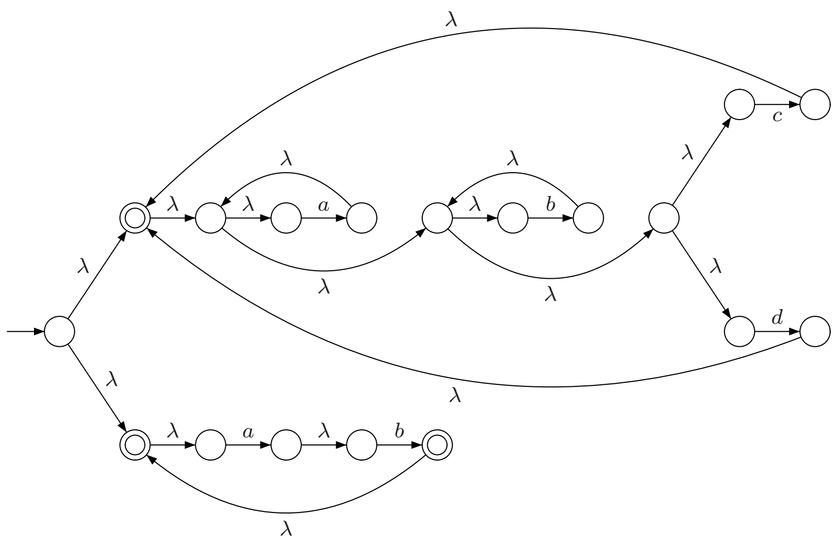
- $a^*b^*(c+d)$:



- $\big(a^*b^*(c+d)\big)^*$:



- $\big(a^*b^*(c+d)\big)^* + (ab)^*$:



3. (a) $L_1 = \{a^i b^j | i$ y j se diferencian como mucho en 2 unidades (a favor de cualquiera)$\}$
Es un lenguaje *no regular* e *incontextual.*

   - Veamos que no es regular por medio del conjunto infinito: Consideremos el conjunto $A = \{aa, aaa, aaaa, aaaaa, \ldots\}$. Dos elementos cualesquiera siguen la forma $w_1 = a^i$ y $w_2 = a^j$ con $i \neq j$. Veamos que w_1 y w_2 son distinguibles en L_1.
     - Si $i > j$ entonces consideremos la continuación b^{j-2}. Tenemos $w_1 b^{j-2} = a^i b^{j-2} \neq L_1$, pues por $i > j$ se cumple que i y $j-2$ se diferencian en al menos 3 unidades. Sin embargo, $w_2 b^{j-2} = a^j b^{j-2} \in L_1$, pues ahora la diferencia es sólo 2.
     - Si $i < j$, tomemos b^{i-2}. $w_1 b^{i-2} = a^i b^{i-2} \in L$. Sin embargo, $w_2 b^{i-2} = a^j b^{i-2} \neq L$ pues ahora la diferencia es, al menos, 3.
   - También podríamos haber demostrado que L_1 no es regular usando el lema del bombeo para lenguajes regulares:
     - Una vez fijada cierta n, tomemos la palabra $w = a^{n+2} b^n$. Se cumple $|w| \geq n$ y $w \in L_1$. Veamos si es posible descomponer w en 3 pedazos $w = xyz$ tales que:

       (1 $|xy| \leq n$

       (2 $y \neq \lambda$

       (3 $\forall i \geq 0 : xy^i z \in L_1$
     - Por (1), y sólo contiene a's y por (2), contiene al menos una. Entonces, la palabra xy^2z contendrá *más de* $n+2$ a's y *exactamente* n b's. Por tanto, $xy^2z \notin L_1$ y la condición (3) no se cumple.
   - Como dijimos, L_1 es incontextual. Una posible gramática incontextual para L_1 es la siguiente, donde el símbolo inicial es I:

$$I \rightarrow aIb \mid \lambda \mid a \mid b \mid aa \mid bb$$

(b) $L_2 = \{a^i b^j c^k | i, j$ y k son todas impares a la vez o todas pares a la vez $\}$
L_2 es *regular* e *incontextual.* Una posible expresión regular para L_2 es:

$$(aa)^*(bb)^*(cc)^* + a(aa)^*b(bb)^*c(cc)^*$$

(c) $L_3 = \{a^i b^j c^k | i + j + k$ es un cuadrado perfecto $\}$
L_3 es *no regular* y *no incontextual.*
Probamos que no es incontextual usando el lema del bombeo para lenguajes incontextuales:

   - Tomemos una cierta n. Ahora podríamos tomar una palabra que contuviera a's, b's y c's en una cantidad adecuada, pero una palabra con sólo un tipo de símbolos (por ejemplo, a's) servirá y será más sencilla.
   - Tomemos $z = a^{n^2}$. Se cumple $|z| \geq n$ y $z \in L_3$.
   - Veamos si podemos descomponer z en 5 pedazos $z = uvwxy$ tales que:
     1) $|vwx| \leq n$
     2) $vx \neq \lambda$
     3) $\forall i \geq 0 : uv^i wx^i y \in L_3$
   - Por (1), vx tiene como mucho n a's, y por (2) tiene al menos una a. Dado que z sólo contiene a's, no nos importa dónde están esas a's (por eso no hará falta hacer una distinción de casos como en otras ocasiones).
   - Si deseáramos añadir a's a a^{n^2} y aún así obtener una palabra en L_3, deberíamos añadir tantas como para crear $a^{(n+1)^2}$. Se cumple $(n+1)^2 = n^2 + 2n + 1$, por lo que necesitaríamos $2n + 1$ a's de más.
   - Sin embargo, uv^2wx^2z tendrá como mucho $n^2 + n$ a's en total, pues la v y la x extras aportan como mucho n a's. Es decir, como mucho obtendremos n a's de más, pero hacían falta $2n + 1$. Por tanto, $uv^2wx^2z \notin L_3$ y la condición (3) es falsa.

(d) $L_4 = \{a^i b^j c^k | i \geq j + k\}$

L_4 es *no regular* e *incontextual*.

- Vemos que no es regular por el lema del bombeo para lenguajes regulares:
  - Fijada n, tomamos $w = a^{2n} b^n c^n$. Se cumple $w \in L_4$ y $|w| \geq n$. Probemos que no podemos descomponer w en xyz con

    (1 $|xy| \leq n$

    (2 $y \neq \lambda$

    (3 $\forall i \geq 0 : xy^i z \in L_1$
  - Por (1), y sólo tiene a's, y por (2) tiene al menos una. Entonces, $xy^0 z = xz$ tendrá *menos de* $2n$ a's, pero la suma de b's y c's seguirá siendo $2n$. Por tanto, $xy^0 z \notin L_4$ y la condición (3) no se cumple.
- También podríamos haber probado que L_4 no es regular por medio de un conjunto infinito adecuado:
  - Sea $A = \{\lambda, a, aa, aaa, aaaa, \ldots\}$. Dos palabras de A cualesquiera siguen la forma $w_1 = a^i$ y $w_2 = a^j$ con $i \neq j$.
  - Si $i > j$, consideremos la continuación común b^i. Se cumple $w_1 b^i = a^i b^i \in L_4$, pero $w_2 b^i = a^j b^i \notin L_4$, pues en este último caso hay más b's y c's (i) que a's (j).
  - Si $i < j$, tomamos b^j. Se cumple $w_1 b^j \notin L_4$, pero tenemos $w_2 b^j = a^j b^j \in L_4$.
  - Por tanto, cualesquiera dos elementos de A son distinguibles entre sí en el lenguaje L_4.
- Como dijimos, L_4 es incontextual. Una posible gramática incontextual para L_4 es la siguiente, donde el símbolo inicial es I:

$$\begin{aligned} I &\rightarrow AP \\ A &\rightarrow \lambda \mid aA \\ P &\rightarrow aPc \mid Q \\ Q &\rightarrow aQb \mid \lambda \end{aligned}$$

  La GI anterior surge de la siguiente descomposición de L_4:

$$L_4 = \{a^i b^j c^k | i \geq j + k\} = \underbrace{\{a^m\}}_{A} \underbrace{\{a^k \underbrace{a^j b^j}_{Q} c^k\}}_{P}$$

4. $L = \{c^n (ba)^m | n > m \geq 0\}$

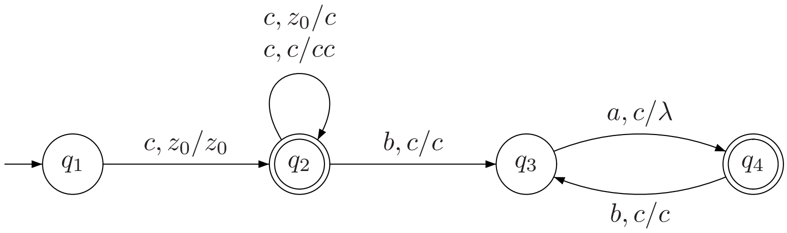
Recordemos que para que un AP sea *determinista*, nunca debe haber más de una transición disponible a la vez. Esto podría ocurrir:

- si desde un estado hay dos transiciones con el mismo símbolo de entrada y la misma cima.
- si desde un estado hay dos transiciones con la misma cima y el símbolo de alguna es λ.

Nuestro APD funcionará como sigue:

- Tomaremos la primera c sin apilarla. Dado que se pide $n > m$, haremos que las c's de la pila sean *las que podrían emparejarse con cadenas ba's*, no las que han llegado desde el principio. Mientras se apilas la 2ª c y siguientes, estaremos en aceptación (es un estado diferente al inicial).
- Después, cada ba restará una c. Para ello, quitaremos la c al tomarse la a. Justo en esos casos, aceptaremos.
- Si las ba's exceden a las c's apiladas: no usaremos ninguna transición especial para eso. Así, la única rama del APD morirá y no aceptaremos.

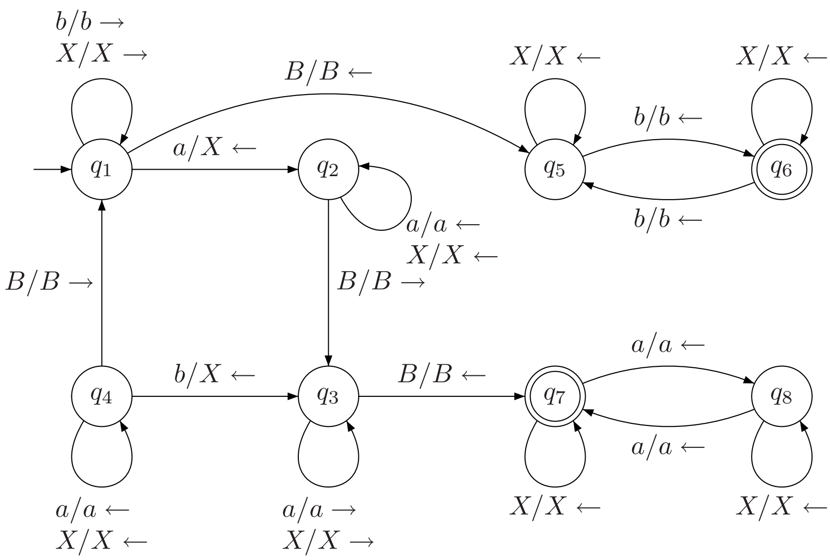
Símbolo inicial de la pila: z_0.



5. Veamos tres posibles MT que reconocen L.

   - Para la primera, la estrategia será la siguiente:
     - Repetiremos el siguiente bucle:
       - Buscaremos la 1ª a, la cambiaremos por X y volveremos al principio.
       - Buscaremos la 1ª b, la cambiaremos por X y volveremos al principio.

       hasta que *no* hallemos la a o la b que buscábamos (e.d., veo B).
       - Si era a: comprobaremos si las b's restantes son impares.
       - Si era b: comprobaremos si las a's restantes son impares.



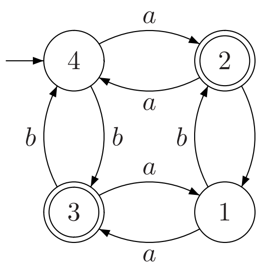
(si encontramos B en q_6 ó q_7 pararemos y aceptaremos).

La imparidad de las a's sobrantes se controla *al revés* que la imparidad de las b's sobrantes (e.d., estado de aceptación intercambiado). El motivo es que una de las a's sobrantes se transformará en$X$ en la transición de q_1 a q_2.

   - Veremos ahora una MT alternativa que también resuelve lo que pide el ejercicio. Resulta que para saber si la diferencia entre las a's y las b's es impar basta con recordar *si las a's recibidas son pares o no* y *si las b's recibidas son pares o no*, y considerar ambas condiciones juntas. De hecho, no nos importa si hay más a's que b's o viceversa:

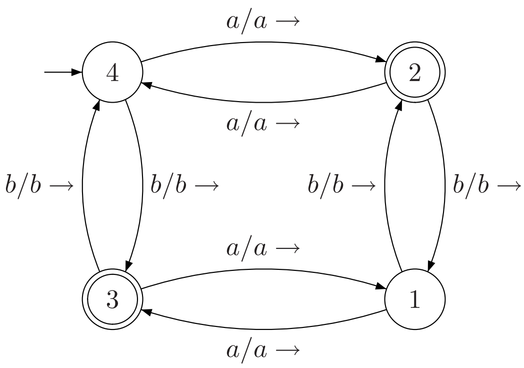
     1.- impares a's e impares b's: diferencia par.
     2.- impares a's y pares b's: diferencia impar.
     3.- pares a's e impares b's: diferencia impar.
     4.- pares a's y pares b's: diferencia par.

Por todo lo dicho, L *es regular* y hay un AFD que lo reconoce:



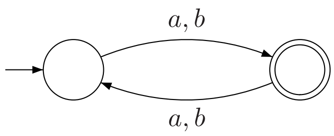
(el nº de estado indica qué combinación de las 4 de arriba representa).

Desgraciadamente, eso no es una MT sino un AFD. Sin embargo, transformar el AFD a MT es muy sencillo: basta con recorrer la cinta de izquierda a derecha sin modificarla mientras cambiamos el estado como indicaba el AFD. El resultado es la siguiente MT:

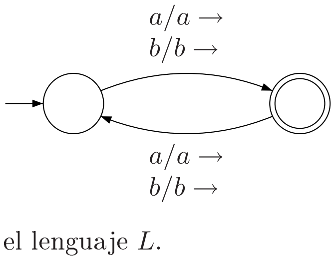


- Cuando la MT encuentre un blanco parará, pues ninguna transición toma un blanco de la cinta. Si para en 2 ó en 3 aceptará, y si para en 4 ó en 1 rechazará.

- No obstante, existe un AFD (y una MT) *aún más simple*, Se cumple $w \in L \Leftrightarrow |w|$ impar. De hecho, si minimizamos el AFD que vimos anteriormente, obtenemos



por lo que la MT

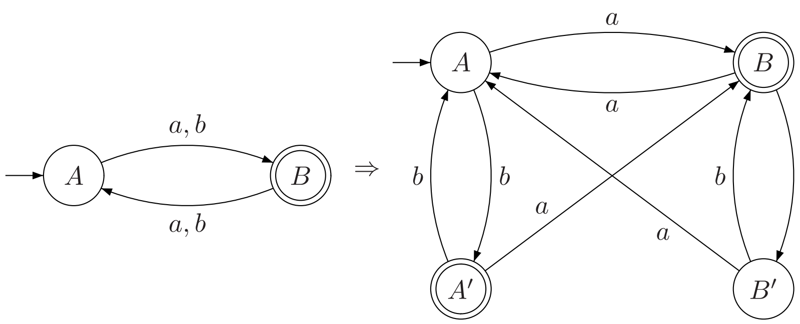


es válida para reconocer el lenguaje L.

6. (a) $LQ = LR$. Dado un AFD en el que hay transiciones de algunos estados a sí mismos, podemos transformarlo en un AFD que es equivalente pero no tiene ese tipo de transiciones. Por cada estado q que tiene transiciones a sí mismo, hacemos lo siguiente:

- Creamos un nuevo estado q' (aceptará o no igual que q).
- Eliminamos todas las transiciones de q a sí mismo, y creamos nuevas transiciones de q a q' y de q' a q con los mismos símbolos que tenían esas transiciones.
- Por cada transición que ya existiera en el AFD de q a cualquier otro estado q'', unimos q' con q'' con una nueva transición etiquetada por el mismo símbolo.

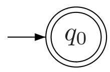
Ejemplo:



Desde cualquier lenguaje L en LR, existe un AFD que lo reconoce. Este AFD puede transformarse como hemos dicho. Por tanto, $L \in LQ$. Por otro lado, si $L' \in LQ$ entonces es obvio que $L' \in LR$. Por tanto, $LQ = LR$.

(b) (b.1) $PAL \in LI$ pero $PAL \notin LM$. El lenguaje PAL no está en LM pues para reconocer palabras de longitud mayor de 21 el AP correspondiente deberá apilar al menos 11 símbolos (toda la primera mitad de la palabra).

(b.2) $\{\lambda\} \in LI\}$ y $\{\lambda\} \in LM$, pues para reconocer el lenguaje $\{\lambda\}$ no es necesario apilar nada.
AP que reconoce $\{\lambda\}$:



(b.3) No existe ningún lenguaje L tal que $L \in LM$ pero $L \notin LR$. De hecho, $LM = LR$. Tener una pila que sólo puede almacenar 10 símbolos no permite reconocer ningún lenguaje que no pudiera reconocerse sin pila alguna.

- Si tenemos una pila de tamaño finito, *el número de contenidos posibles de la pila es finito*, pues el alfabeto de la pila también es finito.
- Como hay finitas pilas posibles, podemos hacer que los estados no sólo representen el estado del AP, sino también su pila, a base de añadir muchísimos estados nuevos y las correspondientes transiciones (¡finitas!).