Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3725 - Traductores e Interpretadores
Septiembre-Diciembre 2007

Examen Parcial I

(20 puntos)

Carnet:

Nombre:

1. **(1 punto)** Una dirección IP se escribe como una cadena de exactamente cuatro (4) números naturales entre 0 y 255 inclusive, separados entre sí por un punto, e.g. 042.84.126.0 o 69.003.9.000. Dé una expresión regular que denote el conjunto que contiene solamente las direcciones IP válidas. Puede utilizar los operadores clásicos para expresiones regulares y notación de conjuntos, pero **no** puede utilizar las expresiones extendidas de analizadores lexicográficos.

   Defino los conjuntos

   $$\begin{aligned} D &= 0+1+2+3+4+5+6+7+8+9 \\ C &= 0+1+2+3+4 \end{aligned}$$

   entonces puedo escribir una expresión regular para un componente de la dirección IP como

   $$P = (\lambda+0+1)(\lambda+D)D+2(CD+5(C+5))$$

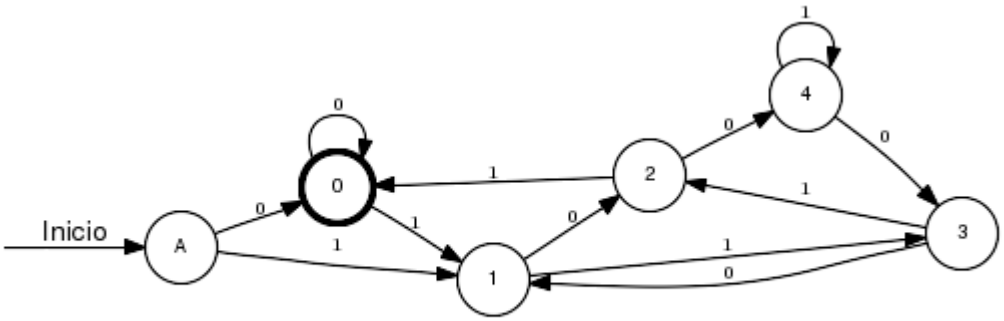
   de manera que la expresión regular requerida es

   $$P.P.P.P$$

2. Sea el alfabeto $\Sigma = \{0,1\}$ e interpretemos las palabras en Σ^* como números binarios leídos de izquierda a derecha, esto es, el primer símbolo de la palabra es el dígito binario más significativo, e.g. 101010 corresponde al 42 decimal, 10100 corresponde al 20 decimal. Evidentemente es posible que las palabras tengan ceros al principio ("ceros a la izquierda"), e.g. 0000101010 corresponde al 42 decimal. Entonces:

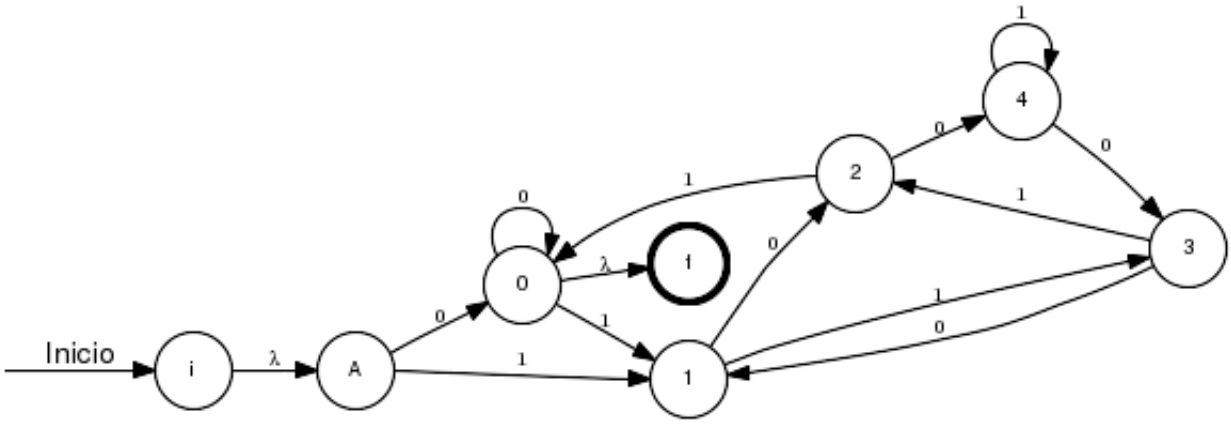
   a) **(3 puntos)** Construya un AFD que reconozca el lenguaje regular de las palabras cuya interpretación como número binario corresponde a un número múltiplo de 5 (cinco). Basta que dé la representación gráfica de su autómata en lugar de la 5-tupla correspondiente.

   Nota: Recordando que en un autómata la "historia" del procesamiento sólo está representada en el estado actual, entonces usaremos un estado para representar el conteo actual módulo 5; esto es, si hemos procesado dígitos que representan el número m el estado actual nos indicará el valor de m módulo 5. Recordando la representación binaria, si tenemos un número m, entonces $m0$ es lo mismo que calcular $2m$ y debemos movernos al estado que represente a $2m$ módulo 5; así mismo $m1$ es lo mismo que calcular $2m+1$ y debemos movernos al estado que represente a $2m+1$ módulo 5. La palabra vacía **no** pertenece al lenguaje, pues no representa ningún número binario, de modo que el estado inicial **no** es final y para lo único que sirve es para establecer el primer contexto (el primer m) pues al recibir un 0 o un 1, iremos respectivamente a los estados 0 y 1. El **único** estado final necesario es el estado 0, pues representa 0 módulo 5. Este razonamiento, que está basado en conocimientos previos de numeración binaria y su naturaleza aritmética y recursiva, **no** es necesario escribirlo en el examen, pues si el autómata está correctamente construido es obvio que se dedujo así.

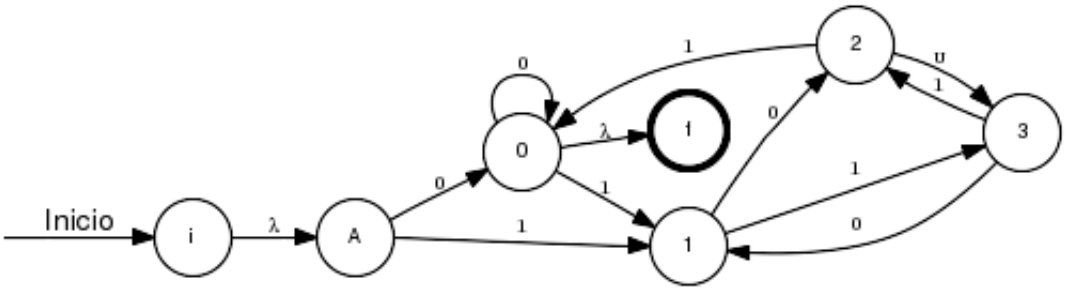


b) **(3 puntos)** Calcule la expresión regular que denote el lenguaje reconocido por el AFD construido.

1) Agregamos un nuevo estado inicial i el cual se conecta al estado 0 usando una λ-transición. Agregamos un nuevo estado final f y conectamos el estado 0 con f usando una λ-transición.



2) Seleccionamos el estado 4 para remoción, preservando los caminos con los cuales contribuye. Desde el estado 2 se puede llegar al estado 3 con la expresión $U = 01 * 0$, por lo tanto



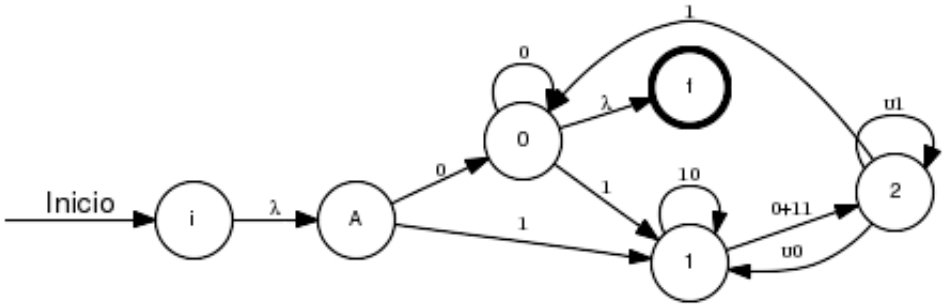
3) Seleccionamos el estado 3 para remoción, preservando los caminos con los cuales contribuye, a saber:

a' Desde el estado 2 se puede llegar al estado 2 con la expresión $U1$.

b' Desde el estado 2 se puede llegar al estado 1 con la expresión $U0$.

c' Desde el estado 1 se puede llegar al estado 2 con la expresión 11.

d' Desde el estado 1 se puede llegar al estado 1 con la expresión 10.



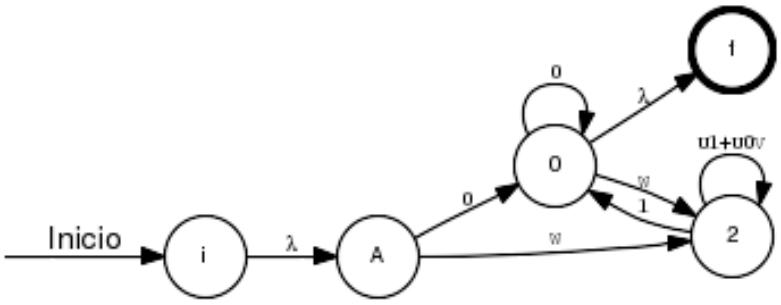
4) Seleccionamos el estado 1 para remoción, preservando los caminos con los cuales contribuye, a saber

a' Desde el estado A se puede llegar al estado 2 con la expresión $1(10)*(0+11)$.

b' Desde el estado 0 se puede llegar al estado 2 con la expresión $1(10)*(0+11)$.

c' Desde el estado 2 se puede llegar al estado 2 con la expresión $U0(10)*(0+11)$

entonces, definimos $V=(10)*(0+11)$ y $W=1V$ para quedar



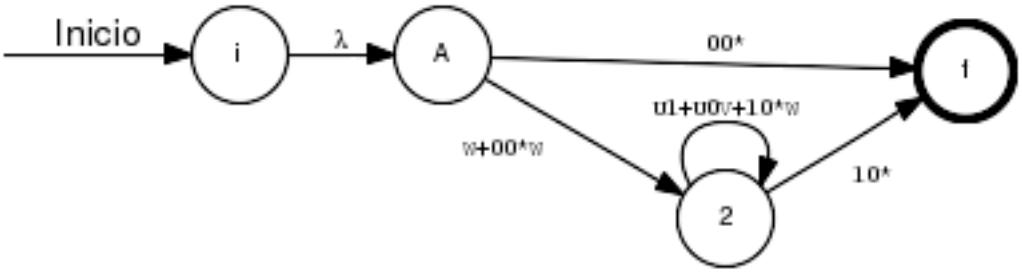
5) Seleccionamos el estado 0 para remoción, preservando los caminos con los cuales contribuye, a saber

a' Desde el estado A se puede llegar al estado f con la expresión $00*$.

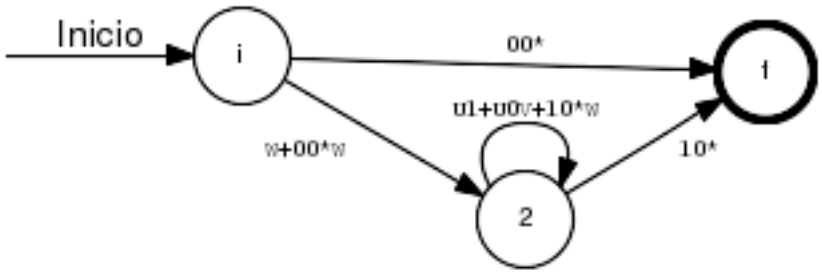
b' Desde el estado A se puede llegar al estado 2 con la expresión $00*W$.

c' Desde el estado 2 se puede llegar al estado f con la expresión $10*$.

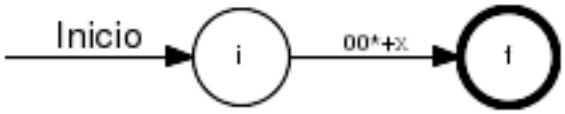
d' Desde el estado 2 se puede llegar al estado 2 con la expresión $10*W$.



6) Seleccionamos el estado A para remoción, preservando los caminos con los cuales contribuye, a saber
   - a' Desde el estado i se puede llegar al estado f con la expresión $00*$.
   - b' Desde el estado i se puede llegar al estado 2 con la expresión $W + 00 * W$



7) Seleccionamos el estado 2 para remoción, preservando los caminos con los cuales contribuye. Desde el esado i se puede llegar al estado f con la expresión $X = (W + 00 * W)(U1 + U0V + 10 * W) * 10*$.



De manera que la expresión regular resultante asociada al autómata es (y es aceptable como respuesta dada la formalidad de la presentación)

$$00 * +X$$

y sustituyendo el valor de X nos queda

$$00 * +(W + 00 * W)(U1 + U0V + 10 * W) * 10*$$

donde aplicamos la distributividad de la concatenación para tener

$$00 * +(\lambda + 00*)W(U(1 + 0V) + 10 * W) * 10*$$

sustituimos los valores de U para tener

$$00 * +(\lambda + 00*)W((01 * 0)(1 + 0V) + 10 * W) * 10*$$

y recordando que $W = 1V$ ahora nos queda

$$00* + (\lambda + 00*)1V((01*0)(1 + 0V) + 10*1V)*10*$$

para terminar sustituyendo el valor de V con

$$00* + (\lambda + 00*)1(10)*(0 + 11)((01*0)(1 + 0(10)*(0 + 11)) + 10*1(10)*(0 + 11))*10*$$

(pero no era necesario llegar hasta ésta expresión si se usan variables).

3. Sea el alfabeto $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, +, -\}$.

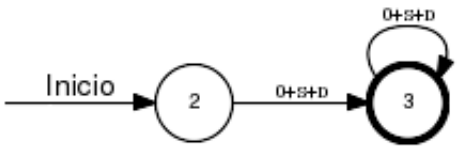
   a) **(0.75 puntos)** Construya sendos autómatas finitos *determinísticos incompletos* que reconozcan las expresiones regulares correspondientes a:

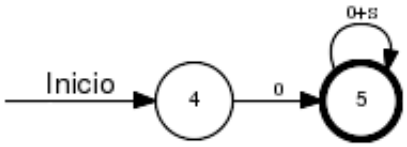
   1) Los números enteros decimales (base 10) que comienzan con cualquier dígito.
   2) Los números enteros octales (base 8) que siempre comienzan con un cero.
   3) Los números enteros hexadecimales (base 16) que comienzan con algún dígito numérico.

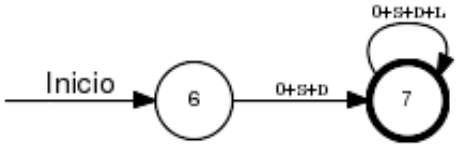
   En los tres casos puede haber ceros a la izquierda. Basta que dé la representación gráfica de cada uno de los autómatas en lugar de la 5-tupla correspondiente.
   Definamos los conjuntos

$$\begin{aligned} S &= 1+2+3+4+5+6+7 \\ D &= 8+9 \\ L &= a+b+c+d+e+f \end{aligned}$$

   entonces los autómatas deseados son (note que el símbolo + en estos autómatas denota la unión de conjuntos regulares)



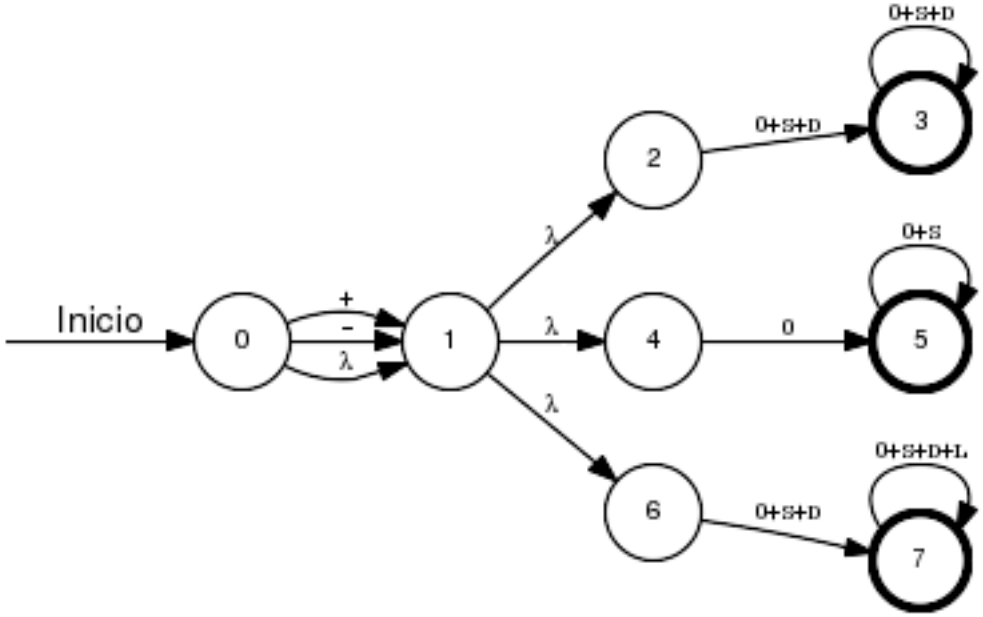




b) **(0.25 puntos)** Construya un autómata finito *determinístico incompleto* que reconozca un signo positivo (+) o negativo (−).

Note que en este autómata los símbolos + y − corresponden al alfabeto y **no** a operaciones sobre conjuntos regulares.

c) **(1 punto)** Combine los cuatro autómatas para crear un AFN-λ que reconozca la unión de los tres lenguajes correspondientes a los números, de manera que al procesar alguna cadena de entrada y reconocerla, se pueda saber a cuál de los tres lenguajes pertenece. Adicionalmente, el AFN-λ debe considerar la posibilidad de que el signo aparezca *opcionalmente* al principio de cualquier número.



Note que en este autómata los símbolos + y − utilizados en las transiciones entre los estados 0 y 1 corresponden a los símbolos del alfabeto, mientras que el símbolo + utilizando en el resto de las transiciones corresponden a la unión de conjuntos regulares.

El autómata creado es una máquina de Moore, pues dependiendo del estado final puede saberse cual lenguaje a reconocido, en concreto

1) Si acepta en el estado **3** entonces se trata de un número entero decimal, posiblemente con signo.
2) Si acepta en el estado **5** entonces se trata de un número entero octal que comienza por cero, posiblemente con signo.
3) Si acepta en el estado **7** entonces se trata de un número entero hexadecimal que comienza con un dígito, posiblemente con signo.

d) **(4 puntos)** Convierta el AFN-λ construido en un AFD. Presente el procedimiento detallado de cálculo de las λ-clausuras y la construcción de la función de transición extendida. Puede utilizar los operadores clásicos para expresiones regulares y notación de conjuntos para simplificar su trabajo, pero **no** puede utilizar las expresiones extendidas de analizadores lexicográficos.

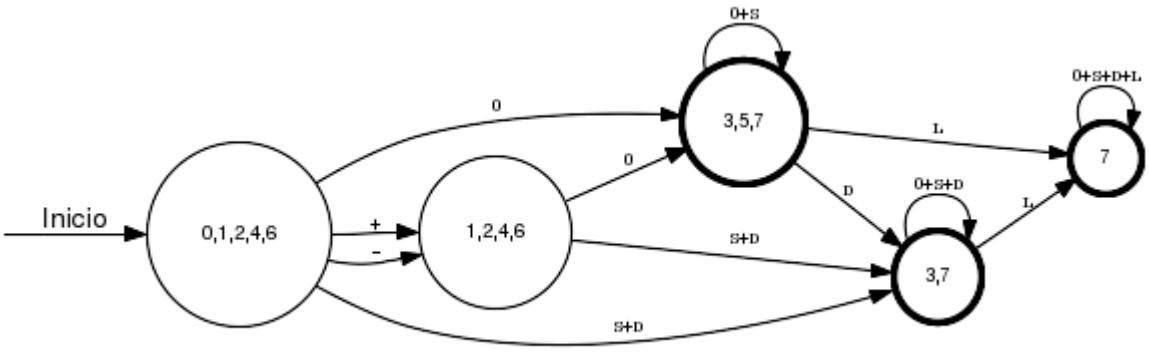
Primero es necesario calcular la λ-clausura para cada uno de los estados del AFN-λ, que por simple inspección puede verse

$$\begin{aligned} \lambda - clausura(0) &= \{0,1,2,4,6\} \\ \lambda - clausura(1) &= \{1,2,4,6\} \\ \lambda - clausura(i) &= \{i\}, 2 \leq i \leq 7 \end{aligned}$$

Calculamos entonces la tabla con la Función de Transición Extendida. Nótese el uso de notación de conjuntos aprovechando los conjuntos definidos al principio, para abreviar el trabajo

	$-$	$+$	0	S	D	L
0	$\{1,2,4,6\}$	$\{1,2,4,6\}$	$\{3,5,7\}$	$\{3,7\}$	$\{3,7\}$	$\emptyset$
1	$\emptyset$	$\emptyset$	$\{3,5,7\}$	$\{3,7\}$	$\{3,7\}$	$\emptyset$
2	$\emptyset$	$\emptyset$	$\{3\}$	$\{3\}$	$\{3\}$	$\emptyset$
3	$\emptyset$	$\emptyset$	$\{3\}$	$\{3\}$	$\{3\}$	$\emptyset$
4	$\emptyset$	$\emptyset$	$\{5\}$	$\emptyset$	$\emptyset$	$\emptyset$
5	$\emptyset$	$\emptyset$	$\{5\}$	$\{5\}$	$\emptyset$	$\emptyset$
6	$\emptyset$	$\emptyset$	$\{7\}$	$\{7\}$	$\{7\}$	$\emptyset$
7	$\emptyset$	$\emptyset$	$\{7\}$	$\{7\}$	$\{7\}$	$\{7\}$

Como **no** se solicita un AFD *completo* podemos omitir el estado correspondiente al vacío. Así mismo, note que los símbolos + y − en las transiciones desde el estado $\{0,1,2,4,6\}$ hasta el estado $\{1,2,4,6\}$ corresponden a símbolos del alfabeto, mientras que el símbolo + utilizado en el resto de las transiciones corresponde a la unión de conjuntos regulares, aprovechando los conjuntos definidos inicialmente



e) **(1 punto)** Indique a cuál de los lenguajes originales corresponde cada estado final del AFD. En caso de ambigüedad, se prefieren los números octales a los decimales, que a su vez se prefieren a los números hexadecimales.

1) Si el DFA acepta en el estado $\{7\}$ entonces está aceptando una palabra que corresponde a un número **hexadecimal**, pues en el NFA-λ original el estado 7 era el de aceptación para dicho lenguaje.
2) Si el DFA acepta en el estado $\{3,7\}$ entonces está aceptando una palabra que corresponde a un número **decimal**, pues en el NFA-λoriginal el estado 3 era el de aceptación para los números decimales

y el estado 7 era el de aceptación para los números hexadecimales, sin embargo la precedencia establecida en el enunciado de la pregunta establece que ante esta ambigüedad debemos preferir a los decimales.

3) Si el DFA acepta en el estado $\{3, 5, 7\}$ entonces está aceptando una palabra que corresponde a un número **octal**, pues en el NFA-λ original el estado 3 era el de aceptación para los números decimales, el estado 5 era el de aceptación para los números octales y el estado 7 era el de aceptación para los números hexadecimales. Ante tal ambigüedad la precedencia establecida en el enunciado indica que se prefieren decimales sobre hexadecimales, y octales sobre decimales.

4. **(3 puntos)** Construya el AFD mínimo equivalente, mostrando y justificando la construcción de los $\equiv_i$ necesarios, para el AFD definido por la 5-tupla

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}, \{a, b\}, \delta, q_0, \{q_2, q_5, q_8\})$$

δ	a	b
q_0	q_1	q_4
q_1	q_2	q_5
q_2	q_3	q_7
q_3	q_4	q_7
q_4	q_5	q_8
q_5	q_6	q_1
q_6	q_7	q_1
q_7	q_8	q_2
q_8	q_0	q_4

Por definición, la clase de equivalencia $\equiv_0$ tiene dos conjuntos, el de estados iniciales y el de estados finales, por tanto

$$\equiv_0 \quad = \quad \{\{0, 1, 3, 4, 6, 7\}, \{2, 5, 8\}\}$$

Para calcular $\equiv_1$ consideramos:

a) Los estados 0 y 1 **no** son equivalentes puesto que $\delta(0, a) \not\equiv_0 \delta(1, a)$.
b) Los estados 0 y 4 **no** son equivalentes puesto que $\delta(0, a) \not\equiv_0 \delta(4, a)$.
c) Los estados 0 y 7 **no** son equivalentes puesto que $\delta(0, a) \not\equiv_0 \delta(7, a)$.
d) Los estados 0, 3 y 6 **si** son equivalentes puesto que $\delta(0, a) \equiv_0 \delta(3, a) \equiv_0 (6, a)$ y $\delta(0, b) \equiv_0 \delta(3, b) \equiv_0 (6, b)$.
e) Los estados 1, 4 y 7 **si** son equivalentes puesto que $\delta(1, a) \equiv_0 \delta(4, a) \equiv_0 (7, a)$ y $\delta(1, b) \equiv_0 \delta(4, b) \equiv_0 (7, b)$.
f) Los estados 2, 5 y 8 **si** son equivalentes puesto que $\delta(2, a) \equiv_0 \delta(5, a) \equiv_0 (8, a)$ y $\delta(2, b) \equiv_0 \delta(5, b) \equiv_0 (8, b)$.

en consecuencia

$$\equiv_1 \quad = \quad \{\{0, 3, 6\}, \{1, 4, 7\}, \{2, 5, 8\}\}$$

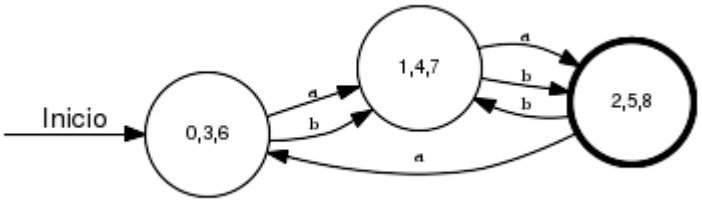
Para calcular $\equiv_2$ consideramos:

a) Los estados 0, 3 y 6 **si** son equivalentes puesto que $\delta(0, a) \equiv_1 \delta(3, a) \equiv_1 (6, a)$ y $\delta(0, b) \equiv_1 \delta(3, b) \equiv_1 (6, b)$.
b) Los estados 1, 4 y 7 **si** son equivalentes puesto que $\delta(1, a) \equiv_1 \delta(4, a) \equiv_1 (7, a)$ y $\delta(1, b) \equiv_1 \delta(4, b) \equiv_1 (7, b)$.
c) Los estados 2, 5 y 8 **si** son equivalentes puesto que $\delta(2, a) \equiv_1 \delta(5, a) \equiv_1 (8, a)$ y $\delta(2, b) \equiv_1 \delta(5, b) \equiv_1 (8, b)$.

en consecuencia

$$\equiv_2 \quad = \quad \{\{0, 3, 6\}, \{1, 4, 7\}, \{2, 5, 8\}\}$$

Como $\equiv_1 = \equiv_2$ hemos llegado al punto fijo de las clases de equivalencia. Cada una de las clases de equivalencia representará uno de los estados. Así, el AFD mínimo resultante será



5. **(3 puntos)** Sea el lenguaje $L = \{a^n b^m | n > m, n \geq 0\}$. Use el Lema de Bombeo para Lenguajes Regulares para demostrar que L no es regular.

   Asumo que L es Lenguaje Regular, entonces existe $M = (Q, \Sigma, \sigma, q_0, F)$ con $|Q| = k$ que acepta palabras de L. Por el Lema de Bombeo para Lenguajes Regulares sabemos que $\forall z \in L$ siempre se puede escribir $z = uvw$ tal que $|uv| \leq k, |v| > 0$ y luego $\forall i \geq 0$ se cumple $uv^i w \in L$.

   Consideremos la palabra $w = a^{k+1}b^k \in L$, entonces cualquier partición de w que cumpla con las condiciones del Lema de Bombeo debe tener necesariamente $uv = a^j$ con $1 \leq j \leq k$, más aún $v = a^p$ con $p > 1$. Así, para *cualquier* partición que escojamos, si se bombea v **cero** veces, el segmento de a va a perder al menos **una** a con lo que la cantidad de a en la palabra bombeada será igual o menor a la cantidad de b y en consecuencia $\notin L$ contradiciendo el Lema de Bombeo. Esa contradicción es consecuencia de haber asumido que L era en efecto Lenguaje Regular, por tanto no puede serlo.