Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3725 - Traductores e Interpretadores
Septiembre-Diciembre 2011

Examen Parcial I

(20 puntos)

Carnet:

Nombre:

1. **(2 puntos)** Sea $\Sigma = \{a, b\}$. Escriba una expresión regular que denote el conjunto de palabras sobre Σ^* tales que contengan una cantidad *par* de a y *exactamente* dos b

   Hay una solución directa, que se construye uniendo las diferentes posibilidades derivadas de observar que, si hay dos b entonces debe haber tres segmentos con a los cuales deben combinarse para tener una cantidad total par, esto es

   $$(aa) * b(aa) * b(aa) *$$
   $$+$$
   $$(aa) * ab(aa) * ab(aa) *$$
   $$+$$
   $$(aa) * ab(aa) * b(aa) * a$$
   $$+$$
   $$(aa) * b(aa) * ab(aa) * a$$

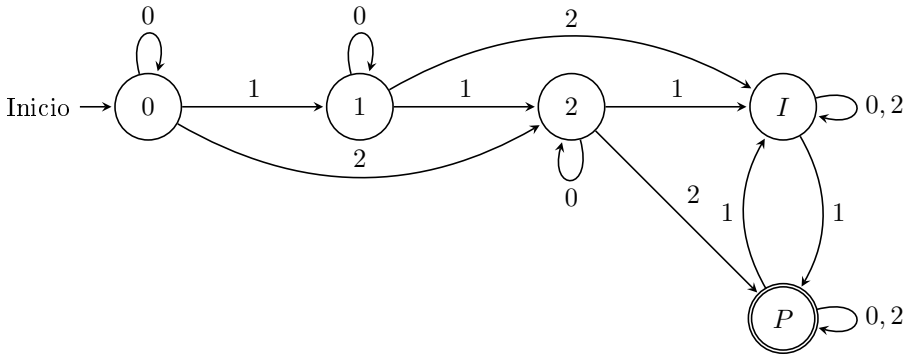
   Con más práctica, o manipulando esa expresión según las identidades del álgebra de expresiones regulares, se puede presentar de manera muy compacta como

   $$(aa) * (ab + ba)(aa) * (ab + ba)(aa)*$$

2. Sea el alfabeto $\Sigma = \{0, 1, 2\}$.

   a) **(2 puntos)** Interpretando las palabras de Σ^* como secuencias de dígitos en base 10, construya un AFD que reconozca el lenguaje de las palabras tales que la suma de sus dígitos sea un número *par* estrictamente mayor a 3. Basta con la representación gráfica del autómata en lugar de escribir la 5-tupla correspondiente.

   Como la palabra está compuesta por dígitos en base 10, la máquina simplemente debe contemplar las sumas acumuladas hasta llegar a un estado final correspondiente al primer número par mayor a 3, y luego oscilar entre suma total par o impar según convenga.



b) **(4 puntos)** Calcule la expresión regular que denote el lenguaje reconocido por el AFD recién construido. *Nota:* si bien no es obligatorio, es conveniente simplificar las expresiones en cada paso de transformación para ahorrar tiempo.

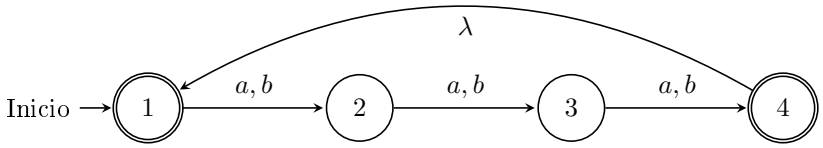
La expresión más simple resultante de aplicar el algoritmo es

$$0*((2+10*1)0*2+(20*1+10*10*1+10*2)(0+2)*1)(0+2+1(0+2)*1)*$$

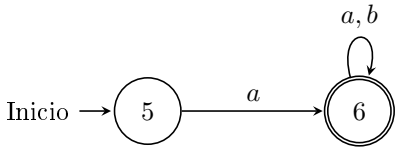
3. Sea el alfabeto $\Sigma = \{a, b\}$.

a) **(0.75 puntos)** Construya sendos autómatas finitos *no-determinísticos,* utilizando $\lambda-$transiciones si le resulta conveniente, que reconozcan las expresiones regulares correspondientes a los conjuntos:

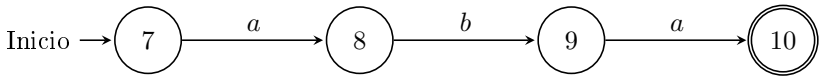
- L_1 de las palabras en Σ^* cuya longitud es múltiplo de 3.



- L_2 de las palabras en Σ^* que comienzan por a.

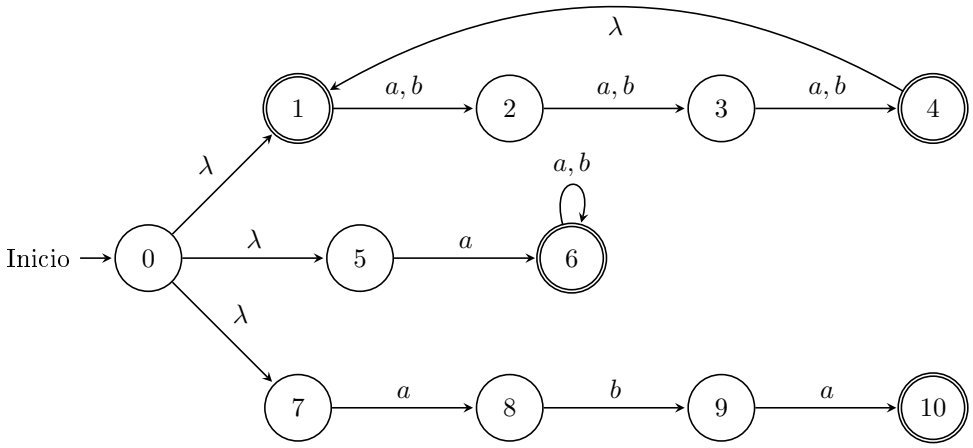


- L_3 de la palabra aba.



Basta con la representación gráfica de cada uno de los autómatas.

b) **(0.25 puntos)** Combine los tres autómatas para crear un AFN-λ que reconozca la unión de los tres lenguajes anteriores, de manera que al procesar alguna cadena de entrada y reconocerla, se pueda saber a cuál de los tres lenguajes pertenece.



c) **(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.

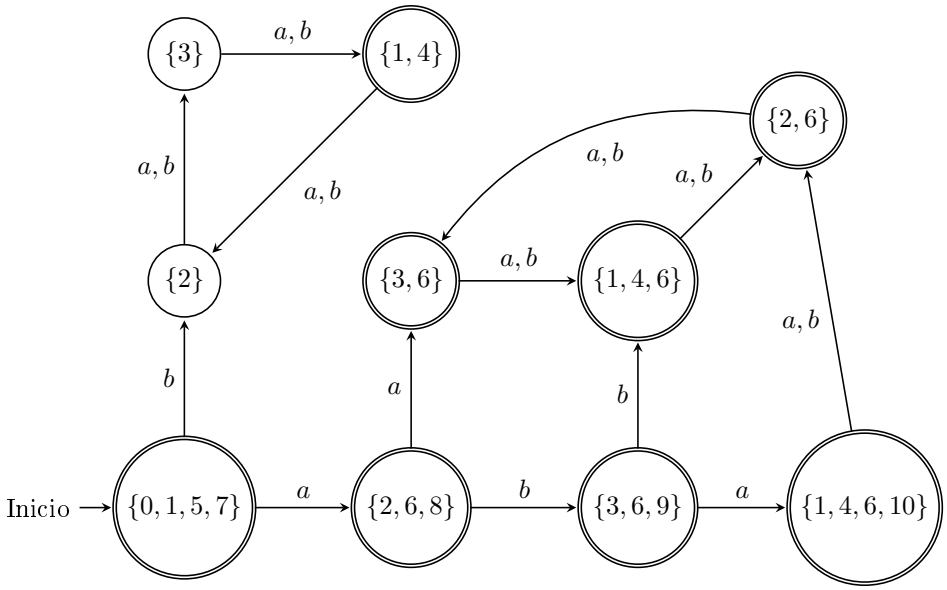
Primero es necesario calcular la λ-clausura para cada uno de los estados del λ-AFN, que por simple inspección puede verse

$$\begin{aligned} \lambda - clausura(0) &= \{0,1,5,7\} \\ \lambda - clausura(4) &= \{1,4\} \\ \lambda - clausura(i) &= \{i\}, 1 \leq i \leq 3 \wedge 5 \leq i \leq 9 \end{aligned}$$

Calculamos entonces la tabla con la Función de Transición Extendida.

t	a	b
0	$\{2,6,8\}$	$\{2\}$
1	$\{2\}$	$\{2\}$
2	$\{3\}$	$\{3\}$
3	$\{1,4\}$	$\{1,4\}$
4	$\{2\}$	$\{2\}$
5	$\{6\}$	$\emptyset$
6	$\{6\}$	$\{6\}$
7	$\{8\}$	$\emptyset$
8	$\emptyset$	$\{9\}$
9	$\{10\}$	$\emptyset$
10	$\emptyset$	$\emptyset$

Construimos el AFD según el algoritmo explicado en clase, partiendo del nuevo estado inicial construido a partir de $\lambda - clausura(0)$.



d) **(1 punto)** Indique a cuál de los lenguajes originales corresponde cada estado final del AFD. En caso de ambigüedad, se prefieren las palabras del conjunto L_1 antes que las palabras del conjunto L_2, y se prefieren las palabras del conjunto L_3 antes que las palabras del conjunto L_1.

1) Si el AFD acepta en el estado $\{0,1,5,7\}$ o en el estado $\{1,4\}$, entonces está aceptando una palabra que corresponde al conjunto (1), i.e. palabras cuya longitud es múltiplo de tres, pues en el λ-AFN original los estados 1 y 4 eran los de aceptación para dicho lenguaje.
2) Si el AFD acepta en el estado $\{2,6,8\}$, en el estado $\{3,6\}$ o en el estado $\{3,6,9\}$, entonces está aceptando una palabra que corresponde al conjunto (2), i.e. palabras que comienzan por a, pues en el λ-AFN original el estado 6 era el de aceptación para dicho lenguaje.

3) Si el AFD acepta en el estado $\{1, 4, 6\}$ entonces está aceptando una palabra que corresponde al conjunto (1), i.e. palabras cuya longitud es múltiplo de tres, pues en el λ-AFN original los estados 1 y 4 eran los de aceptación para el conjunto (1) y el estado 6 era el de aceptación para el conjunto (2), sin embargo la precedencia establecida en el enunciado de la pregunta indica que ante esta ambigüedad debemos preferir al conjunto (1).
4) Si el AFD acepta en el estado $\{1, 4, 6, 10\}$ entonces está aceptando una palabra que corresponde al conjunto (3), i.e. la palabra aba, pues en el λ-AFN original los estados 1 y 4 eran los de aceptación para el conjunto (1), el estado 6 era el de aceptación para el conjunto (2) y el estado 10 era el de aceptación para el conjunto (3), sin embargo las precedencias establecidas en el enunciado de la pregunta indican que ante esta ambigüedad debemos preferir al conjunto (3).

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

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

δ	a	b
q_0	q_5	q_1
q_1	q_4	q_3
q_2	q_2	q_5
q_3	q_3	q_0
q_4	q_1	q_2
q_5	q_5	q_4

Por definición, la clase de equivalencia $\equiv_0$ tiene dos conjuntos, el de estados finales y el de estados no finales, por tanto

$$\equiv_0 \quad = \quad \{\{q_0, q_5\}, \{q_1, q_2, q_3, q_4\}\}$$

Para calcular $\equiv_1$ consideramos:

a) Los estados q_0 y q_5 **si** son equivalentes puesto que $\delta(q_0, a) \equiv_0 \delta(q_5, a) \wedge \delta(q_0, b) \equiv_0 \delta(q_5, b)$.
b) Los estados q_1 y q_2 **no** son equivalentes puesto que $\delta(q_1, b) \not\equiv_0 \delta(q_2, b)$.
c) Los estados q_1 y q_3 **no** son equivalentes puesto que $\delta(q_1, b) \not\equiv_0 \delta(q_3, a)$.
d) Los estados q_1 y q_4 **si** son equivalentes puesto que $\delta(q_1, a) \equiv_0 \delta(q_4, a) \wedge \delta(q_1, b) \equiv_0 \delta(q_4, b)$.
e) Los estados q_2 y q_3 **si** son equivalentes puesto que $\delta(q_2, a) \equiv_0 \delta(q_3, a) \wedge \delta(q_2, b) \equiv_0 \delta(q_3, b)$.

en consecuencia

$$\equiv_1 \quad = \quad \{\{q_0, q_5\}, \{q_1, q_4\}, \{q_2, q_3\}\}$$

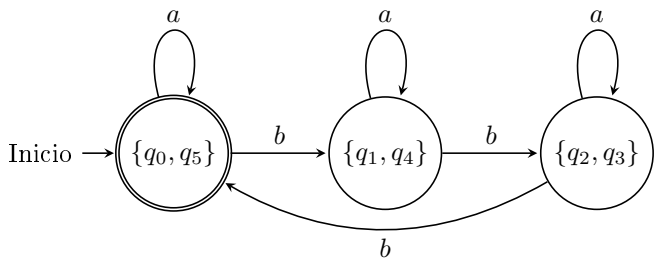
Para calcular $\equiv_2$consideramos:

a) Los estados q_0 y q_5 **si** son equivalentes puesto que $\delta(q_0, a) \equiv_1 \delta(q_5, a) \wedge \delta(q_0, b) \equiv_1 \delta(q_5, b)$.
b) Los estados q_1 y q_4 **si** son equivalentes puesto que $\delta(q_1, a) \equiv_1 \delta(q_4, a)$ y $\delta(q_1, b) \equiv_1 \delta(q_4, b)$.
c) Los estados q_2 y q_3 **si** son equivalentes puesto que $\delta(q_2, a) \equiv_1 \delta(q_3, a)$ y $\delta(q_2, b) \equiv_1 \delta(q_3, b)$.

En consecuencia

$$\equiv_2 \quad = \quad \{\{q_0, q_5\}, \{q_1, q_4\}, \{q_2, q_3\}\}$$

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 $\Sigma = \{a, b\}$ y L el lenguaje de palabras arbitrarias sobre Σ^* tales que tienen la misma cantidad de a y b. Use el Lema de Bombeo de Lenguajes Regulares para demostrar que L no es regular.

   Asumo que L es Lenguaje Regular, entonces existe $M = (Q, \Sigma, \delta, q_0, F)$ con $|Q| = k$ que acepta palabras de L. Por el Lema de Bombeo para Lenguajes Regulares sabemos que $\forall z \in L$ con $|z| \geq k$ 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 $w = a^k b^k \in L$, 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 $p \geq 1$. Así para *cualquier* partición que escojamos, si se bombea v dos o más veces, el segmento de a va a aumentar de longitud mientras el segmento de b permanece intacto, con lo que la primera mitad de la palabra resultante será necesariamente diferente a la segunda mitad y en consecuencia $uv^i w \notin L$ contradiciendo el Lema de Bombeo. Esa contradicción es consecuencia de haber asumido que L era en efecto Lenguaje Regular, por tanto no puede serlo.