Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información.
CI–3725 – Traductores e Interpretadores
Octubre 2013 – Enero 2014

Examen I

(25 puntos)
Solución

0. (12 puntos) Considere cuidadosamente cada uno de los siguientes enunciados y seleccione exactamente una respuesta de entre las opciones disponibles, que haga cierto el enunciado en cuestión. Cada respuesta correcta **suma un punto y medio (1.5)**, sin embargo **cuatro (4) respuestas incorrectas cancelan una correcta**. Tiene la opción de contestar la opción *No sabe / No contesta*, lo cual anula la puntuación de dicha respuesta, mas tampoco es penalizada. Cualquier enunciado que no sea respondido, o que sea respondido con dos o más opciones marcadas, se considerará incorrecto.

   (a) Considerando un alfabeto $\Sigma = \{a, b, c\}$. ¿Cuál de los siguientes lenguajes es regular?

      i. $\{a^n\ b^n\ c^n \mid n \geq 0\}$

         No puede ser, ya que se debe recordar la cantidad de a's vistas para poder saber cuantas b's y c's deben leerse a continuación.

      ii. $\{w\ w \mid w \in \Sigma^*\}$

         No puede ser, ya que se debe recordar la cadena w para poder leerla nuevamente a continuación.

      iii. $\{c\ b\ a^n\ a^n\ b\ c \mid n \geq 0\}$

         Si, ya que $a^n\ a^n = a^{n+n} = a^{2*n}$. Esto es, cantidad par de a's. Por tanto el lenguaje anterior tiene expresión regular: $cb(aa)*bc$.

      iv. $\{y\ w\ y \mid w \in \Sigma^* \wedge y \in \Sigma^*\}$

         No puede ser, ya que se debe recordar la cadena y para poder leerla nuevamente a continuación.

      v. *No Sabe / No Contesta.*

   (b) *Siempre* podemos asegurar que:

      i. Si un AFD M reconoce un lenguaje L, entonces L es regular.

         Si, la clase de lenguajes reconocida por AFD's es igual a la de los lenguajes regulares.

      ii. Si una Gramática G genera un lenguaje L, entonces L es regular.

         No necesariamente, esto sería cierto solo si G es una gramática regular.

      iii. Si un lenguaje L es representado por una Expresión Regular e, entonces *debe* existir otro lenguaje $L' \subset L$, tal que e represente a L' también.

         No, una expresión regular representa a un y solo un lenguaje.

      iv. Para cualquier lenguaje regular L. Un AFD mínimo, que reconoce L, tiene menos estados que cualquier otro autómata (AFD ó AFND) que reconozca L.

         No, es posible que existan AFND's, que reconozcan L, con menos estados que el AFD mínimo.

      v. *No Sabe / No Contesta.*

(c) Considere la expresión regular sobre el alfabeto $\Sigma = \{0, 1\}$, $e = 01(0 + 1)* 11$. ¿Qué podemos asegurar sobre e?

i. $sem(e) = \emptyset$

No, $sem(e)$ no es vacío ya que existen frases en el lenguaje que representa e.

ii. $D_0 sem(e) = \emptyset$

No, $D_0 sem(e)$ no es vacío ya que existen frases en el lenguaje $D_0 sem(e) = sem(D_0(e)) = sem(1(0 + 1) 11)$.*

iii. $D_1 sem(e) = \emptyset$

Si, $D_1 sem(e)$ es vacío ya que no existen frases en el lenguaje $D_1 sem(e) = sem(D_1(e)) = \emptyset$.

iv. Ninguna de las anteriores.

No, ya que la opción (iii) es correcta.

v. *No Sabe / No Contesta.*

(d) Para probar que un lenguaje L no es regular, usando el Lema de Bombeo, debemos:

i. Hallar una frase cualquiera $w \in L$ que no se pueda particionar en $w = xyz$.

No, cualquier frase siempre se puede particionar en tres partes. Paricularmente $w = w\lambda\lambda$.

ii. Demostrar que para toda frase $w \in L$, la misma no se puede particionar en $w = xy^i z$ para ningún $i \geq 0$.

No, cualquier frase siempre se puede particionar en tres partes. Paricularmente $w = w\lambda^i\lambda$, note que en este caso la i es indiferente.

iii. Dada una frase cualquiera w, de tamaño n, hallar una partición de la frase $w = xyz$ (con $|y| \geq 1 \wedge |xy| \leq n$) tal que $xy^i z \in L$, para cualquier $i \geq 0$.

No, esto sería consistente con el lenguaje siendo regular.

iv. Para un número natural n, hallar una frase cualquiera $w \in L$ (con $|w| \geq n$) tal que para cualquier partición $w = xyz$ (con $|y| \geq 1 \wedge |xy| \leq n$), exista un número natural i donde $xy^i z \notin L$.

Si, la suposición de regularidad se contradice al mostrar que el autómata supuesto de n estados es imposible de construir.

v. *No Sabe / No Contesta.*

(e) Considere un grafo de expresiones $M = (\Sigma, Q, \delta, q_0, F)$ con $\delta : Q \times ER_\Sigma \to Q$. Podemos decir que M es determinista si y sólo si:

i. $(\forall q, x, y : q \in Q \wedge x, y \in ER_\Sigma : \delta_m(q, x) = \delta_m(q, y) \Rightarrow sem(x) = sem(y))$

No, esta fórmula plantea (si se toma un contrapositivo) que si la semántica de dos expresiones regulares es diferente, entonces deben ir a estados diferentes (partiendo de un mismo estado) y este no es el concepto de determinismo.

ii. $(\forall q, q', x : q, q' \in Q \wedge x \in ER_\Sigma : \delta_m(q, x) \neq \delta_m(q', x))$

No, dos estados diferentes pueden llegar al mismo estado con la misma expresión y aún tener un grafo de expresiones determinista.

iii. $(\forall q, x, y : q \in Q \wedge x, y \in ER_\Sigma : \delta_m(q, x) \neq \delta_m(q, y) \Rightarrow sem(x) \cap sem(y) = \emptyset)$

Si, esta fórmula plantea (si se toma un contrapositivo) que si la semántica de dos expresiones regulares tiene elementos en común, entonces deben ir al mismo estado (partiendo de un mismo estado).

iv. Siempre es determinista.

No, cualquier AFND se puede reinterpretar directamente como un grafo de expresiones. Estos grafos no serían deterministas.

v. *No Sabe / No Contesta.*

(f) Sea el siguiente AFD, representado en forma de *tabla de transiciones*:

	a	b
q_0	q_1	q_2
q_1	q_0	q_2
q_2	q_0	q_0

Donde q_0 es el estado inicial y q_2 es el único estado final. Como podemos observar, el autómata no acepta frases de tamaño cero (es decir, λ). Acepta una sola frase de tamaño 1, una sola frase de tamaño 2 y tres frases de tamaño 3. ¿Cuantas frases de tamaño 7 acepta este autómata?

Sugerencia: Enumerar las 2^7 posibles frases de tamaño 7 y verificar cada una es una estrategia válida, pero seguramente tome mucho tiempo. En vez de eso, intente hallar una recurrencia $cant(n)$ sobre las frases que el autómata acepta, de tamaño n. Puede serle útil dibujar el diagrama de transiciones para el autómata primero.

i. 52
ii. 43
iii. 21
iv. ∞
v. *No Sabe / No Contesta.*

Note que los primeros valores de la recurrencia que sugerida ya le fueron dados:

- $cant(0) = 0$
- $cant(1) = 1$
- $cant(2) = 1$
- $cant(3) = 3$

Ahora para calcular la cantidad de palabras que acepta con longitud mayor, lo que podemos es ver como depende dichas palabras de palabras anteriores.

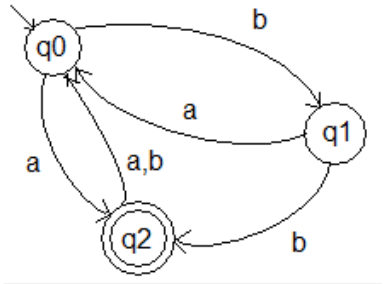


Diagrama de Transiciones

Note que si una palabra w comienza con ab, aa o ba el autómata vuelve al estado inicial con la resto de la frase por consumir. Por ende, si el autómata acepta n frases de tamaño k, sabemos que aceptará $3 * n$ frases de tamaño $k + 2$.

Note también que si una palabra w comienza con bba o bbb el autómata igualmente vuelve al estado inicial con la resto de la frase por consumir. Por ende, si el autómata acepta n frases de tamaño k, sabemos que aceptará $2 * n$ frases de tamaño $k + 3$.

No hace falta verificar palabras de mayor longitud que esto, ya que cualquier otro camino en el autómata no sería simple (pasaría por algún nodo más de una vez).

Con esta información a la mano, podemos plantear una recurrencia:

$$cant(n) = 3 \times cant(n-2) + 2 \times cant(n-3)$$

Podemos ahora usar los casos bases que tenemos e ir calculando la solución iterativamente.

- $cant(4) = 3 \times cant(2) + 2 \times cant(1) = 3 \times 1 + 2 \times 1 = 5$
- $cant(5) = 3 \times cant(3) + 2 \times cant(2) = 3 \times 3 + 2 \times 1 = 11$
- $cant(6) = 3 \times cant(4) + 2 \times cant(3) = 3 \times 5 + 2 \times 3 = 21$
- $cant(7) = 3 \times cant(5) + 2 \times cant(4) = 3 \times 11 + 2 \times 5 = 43$

(g) La tres componentes del *front–end* en el proceso de traducción/interpretación, son:

i. Análisis Léxico, Análisis Sintáctico y Análisis de Contexto.

Si, estas son las tres componentes del front-end.

ii. Análisis Léxico, Análisis Sintáctico y Análisis Pragmático.

No, la pragmática tiene que ver con el uso que se de al lenguaje y eso no es parte del proceso de traducción/interpretación.

iii. Análisis Léxico, Análisis de Contexto y Optimización de Código.

No, la optimización de código corresponde a una (o más) etapas del back-end.

iv. Análisis Léxico, Análisis Pragmático y Optimización de Código.

No, por la razones expuestas en las opciones (ii) y (iii).

v. *No Sabe / No Contesta.*

(h) Si L, S y T son lenguajes regulares, entonces podemos asegurar *siempre* que:

i. $L \cup S \cup T$ es un lenguaje regular.

Si, la unión es cerrada en los lenguajes regulares.

ii. $(L \cap S) - T$ es un lenguaje regular.

Si, la intersección y la resta son cerradas en los lenguajes regulares.

iii. $L \cdot (S \cap T)^*$ es un lenguaje regular.

Si, la intersección, la concatenación y la estrella de Kleene son cerradas en los lenguajes regulares.

iv. Todas las anteriores.

Si, pues las tres opciones planteadas son lenguajes regulares.

v. *No Sabe / No Contesta.*

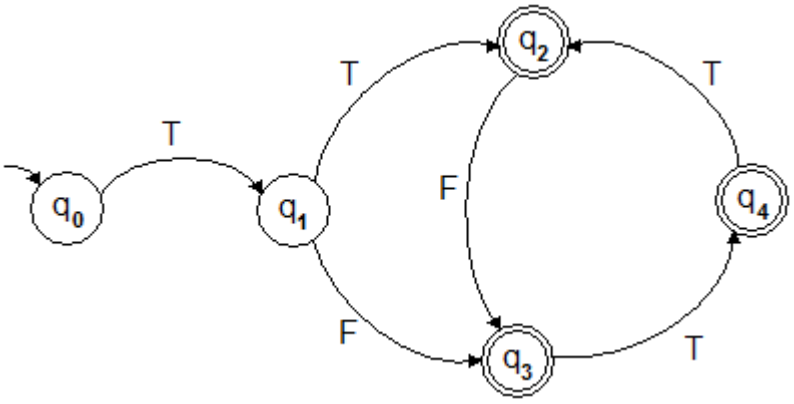
1. (5 puntos) Considere el lenguaje, sobre el alfabeto $\Sigma = \{T, F\}$, de frases w tal que:

   - $|w| \geq 2$
   - w empieza con T.
   - "TT" y "TF" están en el lenguaje.
   - Para tres símbolos a, b y c cualesquiera, que estén ***contiguos*** en w (con $|w| \geq 3$), se debe cumplir que $c = \neg(a \equiv b)$. – Esto interpretando T como $true$ y F como $false$

   Ejemplo de frases que están en el lenguaje propuesto:

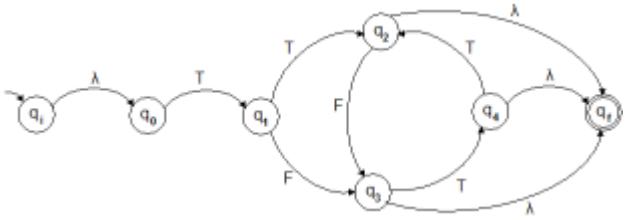
   "TF" "TTF" "TFTTF" "TTFTTFTTFT" "TFT"

   a) (3 pts) Construya un AFND (o un AFD si lo desea), que reconozca el lenguaje en cuestión.

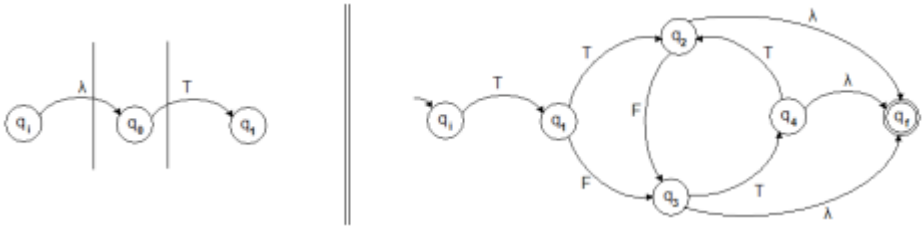


   b) (2 pts) Extraiga del autómata planteado una expresión regular para el mismo lenguaje. Debe mostrar cada uno de los pasos de eliminación de estados que plantea el *algoritmo de las ventanas*.

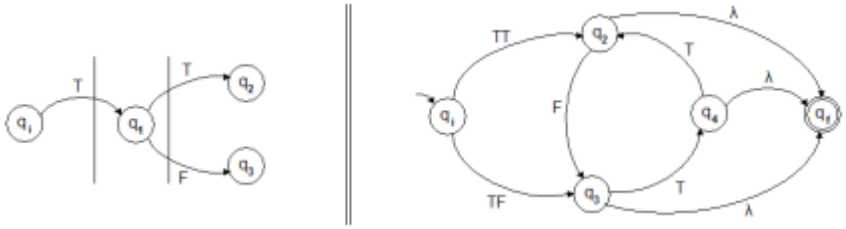
      0) Primeramente, aumentamos el autómata con un nuevo estado inicial y un nuevo estado final que reemplaza a los originales.



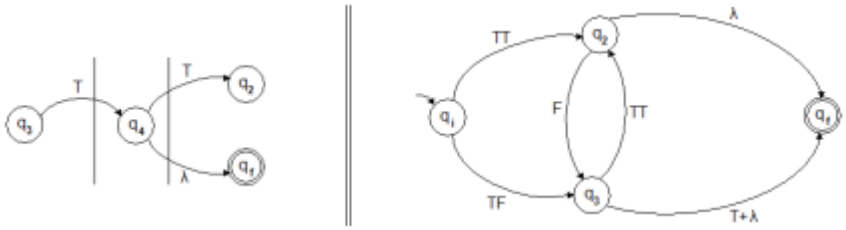
      1) Ahora eliminamos el estado q_0.



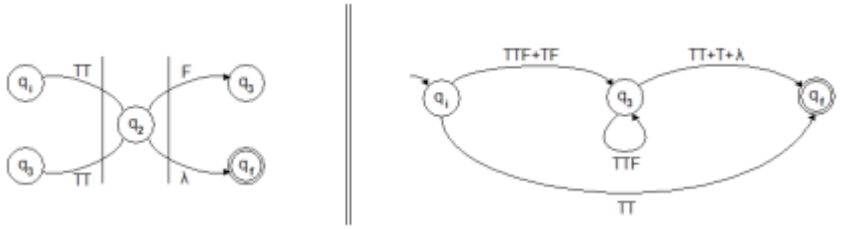
2) Ahora eliminamos el estado q_1.



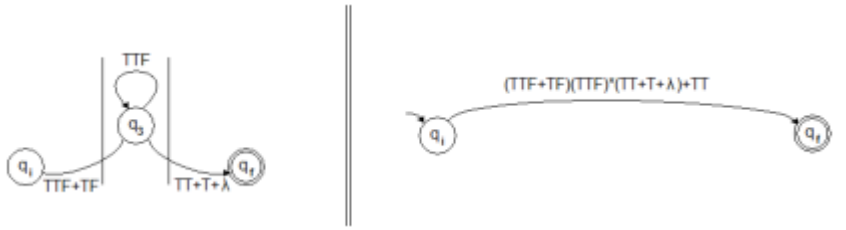
3) Ahora eliminamos el estado q_4.



4) Ahora eliminamos el estado q_2.



5) Ahora eliminamos el estado q_3.



La expresión resultante es por tanto:

$$(TTF + TF)(TTF) * (TT + T + \lambda) + TT$$

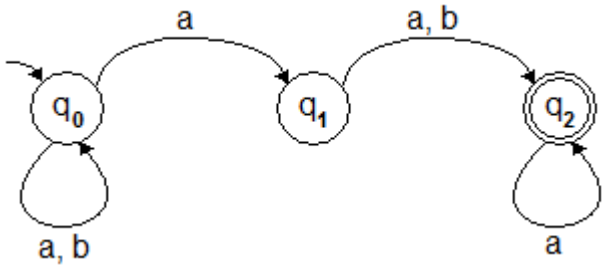
2. (5 puntos) Considere la expresión regular, sobre el alfabeto $\Sigma = \{a, b\}$, $e = (a + b)* a(b + a)a*$.

   Se desea que construya un AFD mínimo que reconozca $sem(e)$. Para hacerlo, puede utilizar el algoritmo de Hopcroft o de Brzozowski (derivadas).

   - Si utiliza el algoritmo de Hopcroft, se le evaluará de la siguiente manera:
     - (1 pt) $ER \rightarrow AFND$.
     - (2 pts) $AFND \rightarrow AFD$
       – *Debe mostrar a que conjunto de estados del $AFND$ corresponde cada estado de su AFD.*
     - (2 pts) $AFD \rightarrow AFD_{min}$
       – *Debe incluir el cálculo de las clases de equivalencia correspondientes.*
   - Si utiliza el algoritmo de Brzozowski, se le evaluará de la siguiente manera:
     - (3 pts) Hallar el conjunto de derivadas diferentes
       – *Debe mostrar el cálculo de las mismas.*
     - (2 pts) Construcción del AFD inducido por las derivadas halladas.

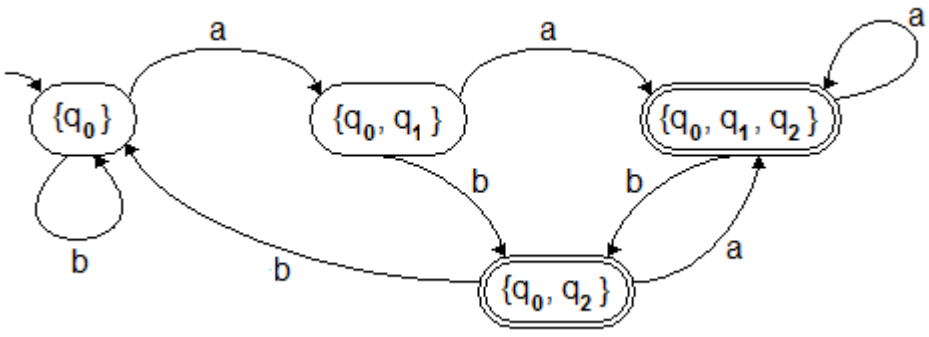
Solución mediante el algoritmo de Hopcroft:

a) Un posible AFND para la expresión planteada sería el siguiente:



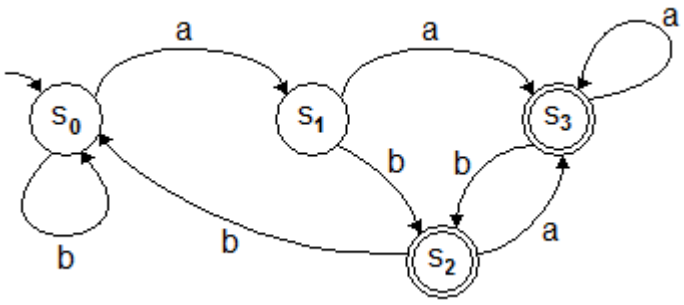
Si usted construyó su autómata a partir de las reglas de construcción de $ER \rightarrow AFND$, su respuesta es correcta. Sin embargo, si usted propuso un AFND similar al que se presenta aquí, se considerará correcta igualmente.

b) Al aplicar el algoritmo de las clausuras (alcance de cada estado, con cada símbolo del alfabeto), tenemos:



c) Primeramente haremos el siguiente renombramiento de los estados:

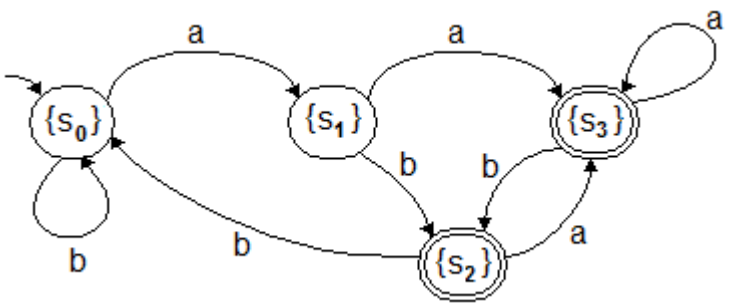
- $s_0 = \{q_0\}$
- $s_1 = \{q_0, q_1\}$
- $s_2 = \{q_0, q_2\}$
- $s_3 = \{q_0, q_1, q_2\}$



Comenzaremoa ahora a plantear las clases de equivalencia.

$$
\begin{array}{lcll}
\equiv_0 & = & \{\{s_0, s_1\}, \{s_2, s_3\}\} & \\
\equiv_1 & = & \{\{s_0\}, \{s_1\}, \{s_2\}, \{s_3\}\} & \text{ya que } \delta(s_0, a) \not\equiv_0 \delta(s_1, a) \wedge \delta(s_2, b) \not\equiv_0 \delta(s_3, b) \\
\equiv_2 & = & \{\{s_0\}, \{s_1\}, \{s_2\}, \{s_3\}\} & \text{como } (\equiv_1) \ = \ (\equiv_2), \text{ es la definitiva.}
\end{array}
$$

Como las clases de equivalencia corresponden a una partición en exclusivamente conjuntos unitarios, el autómata calculado ya era mínimo. De todas formas podemos construir el AFD mínimo, usando las clases de equivalencia calculadas.



Solución mediante el algoritmo de Bzozowski:

a) Primeramente se calculan todas las derivadas diferentes.

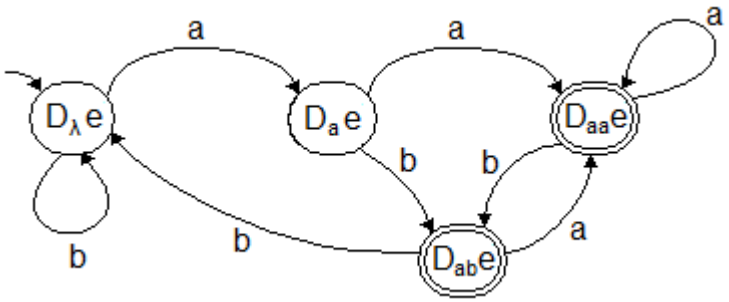
$$
\begin{aligned}
D_\lambda e &= e \quad \checkmark \\
D_a e &= D_a((a+b)*\, a(b+a)a*) \\
&= D_a((a+b)*)a(b+a)a* + D_a(a(b+a)a*) \\
&= (a+b)*\, a(b+a)a* + (b+a)a* \\
&= e + (b+a)a* \quad \checkmark \\
D_b e &= D_b((a+b)*\, a(b+a)a*) \\
&= D_b((a+b)*)a(b+a)a* + D_b(a(b+a)a*) \\
&= (a+b)*\, a(b+a)a* + \emptyset \\
&= e \\
&= D_\lambda e \quad \square \\
D_{aa} e &= D_a(D_a e) \\
&= D_a(e + (b+a)a*) \\
&= D_a e + D_a((b+a)a*) \\
&= e + (b+a)a* + a* \quad \checkmark \\
D_{ab} e &= D_b(D_a e) \\
&= D_b(e + (b+a)a*) \\
&= D_b e + D_b((b+a)a*) \\
&= e + a* \quad \checkmark \\
D_{ba} e &= D_a(D_b e) \\
&= D_a e \quad \square \\
D_{bb} e &= D_b(D_b e) \\
&= D_b e \quad \square \\
D_{aaa} e &= D_a(D_a a e) \\
&= D_a(e + (b+a)a* + a*) \\
&= D_a e + D_a((b+a)a*) + D_a a* \\
&= e + (b+a)a* + a* + a* \\
&= e + (b+a)a* + a* \\
&= D_{aa} e \quad \square \\
D_{aab} e &= D_b(D_a a e) \\
&= D_b(e + (b+a)a* + a*) \\
&= D_b e + D_b((b+a)a*) + D_b a* \\
&= e + a* + \emptyset \\
&= e + a* \\
&= D_{ab} e \quad \square \\
D_{aba} e &= D_a(D_a b e) \\
&= D_a(e + a*) \\
&= D_a e + D_a a* \\
&= e + (b+a)a* + a* \\
&= D_{aa} e \quad \square
\end{aligned}
$$

$$
\begin{aligned}
D_{abb}e &= D_b(D_abe) \\
&= D_b(e + a*) \\
&= D_be + D_ba* \\
&= e + \emptyset \\
&= e \\
&= D_\lambda e \quad \square
\end{aligned}
$$

b) Como podemos observar, resultan 4 derivadas diferentes:

- $D_\lambda e = e$
- $D_a e = e + (b + a)a*$
- $D_{aa} e = e + (b + a)a* + a*$
- $D_{ab} e = e + a*$

Con esto, podemos crear el autómata inducido por las derivadas calculadas:



Los estados correspondientes a $D_{aa}e$ y $D_{ab}e$ son finales, ya que sus semánticas respectivas contienen λ.

3. (3 puntos) Se define la operación $Strip(w)$, sobre frases w en Σ de la siguiente manera:

$$\begin{array}{rcll} Strip(\lambda) & = & \lambda & \\ Strip(a) & = & a & \text{si } a \in \Sigma \\ Strip(abw) & = & a\ Strip(w) & \text{si } a, b \in \Sigma \wedge w \in \Sigma^* \end{array}$$

Tal operación se puede generalizar a lenguajes:

$$Strip(L) = \{Strip(w) \mid w \in L\}$$

Por ejemplo: $Strip(\{aaa, ba, \lambda, abab, aaaab\}) = \{aa, b, \lambda, aab\}$, ya que:

$$\begin{array}{rcl} Strip(aaa) & = & aa \\ Strip(ba) & = & b \\ Strip(\lambda) & = & \lambda \\ Strip(abab) & = & aa \\ Strip(aaaab) & = & aab \end{array}$$

Note que $Strip(w)$ lo que hace es tomar la frase w y quedarse únicamente con aquellos símbolos que estén en posiciones pares (tomando la primera posición como posición cero).

Sea L un lenguaje regular. ¿Es $Strip(L)$ un lenguaje regular? Si lo és, argumente a favor (construyendo un autómata finito, expresión regular, gramática regular o usando propiedades de clausura). Si no lo es, demuéstrelo (usando el Lema de Bombeo o propiedades de clausura).

Nota: *Solamente debe proponer la Expresión Regular, Gramática Regular o Autómata Finito, no debe demostrar formalmente la igualdad del lenguaje que representa/genera/reconoce éste con el lenguaje propuesto.*

Como L es regular, debe tener un AFD que lo reconozca.

Esto es, debe existir $M = (\Sigma, Q, \delta, q_0, F)$, tal que $\mathcal{L}(M) = L$.

Se construirá a partir del mismo un AFND (nótese que será no determinista):

$$M_{Strip} = (\Sigma, Q_{Strip}, \delta_{Strip}, q_{0\,Strip}, F_{Strip})$$

Como este operador busca tomar un caracter, saltar el siguiente y así sucesivamente, lo que debe hacerse es saltar dos veces por la función de transición original ignorando el caracter con el que se hace el segundo movimiento. El resto de los elementos de la 5-tupla correspondiente a M_{Strip} pueden ser idénticos a los de M (Debe recordarse que el conjunto de estados de un autómata no necesariamente debe ser alcanzable desde el estado inicial).

- $Q_{Strip} = Q$
- $\delta_{Strip}(q, a) = \{\delta(\delta(q, a), x) \mid x \in \Sigma\}$ con $q \in Q_{Strip} \wedge a \in \Sigma$
- $q_{0\,Strip} = q_0$
- $F_{Strip} = F$

Como hemos definido un AFND para el lenguaje $Strip(L)$, podemos argumentar que L es regular. Si se buscara ser más formal, tal intuición debe ser luego respaldada por una demostración (posiblemente por doble contención) de $\mathcal{M}_{Strip} = L$. Sin embargo, tal demostración sale del alcance de la pregunta en cuestión.