Teoría de la Computación 2024

Lab 02

29.julio.2024

1. Implementar en Python un programa que simula un autómata finito determinista (AFN). Para ello, debe implementar funciones que hagan lo siguiente:

   - *transition*(q, a, δ) la cual devuelve el valor de la transición $\delta(q, a)$, para un estado $q \in K$ y un símbolo $a \in \Sigma$.
   - *final_state*(q, w, δ) la cual devuelve el estado q obtenido por el autómata después de terminar de leer la cadena $w \in \Sigma^*$.
   - *derivation*(q, w, δ) la cual derivación de la cadena $w \in \Sigma^*$ desde el estado $q \in K$, esto es, la secuencia ordenada de transiciones obtenidas.
   - *accepted*(q, w, F, δ) la cual devuelve verdadero si la cadena $w \in \Sigma^*$ es aceptada por el autómata partiendo desde el estado s; y falso en caso contrario.

   Su algoritmo debe recibir como inputs lo siguiente: un archivo estructurado (`.json`, `.yml`, `.xml` o similares), donde se indica la estructura del autómata AFD:

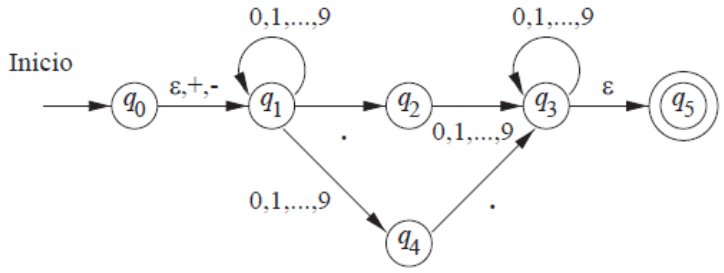
   - Q: una lista finita de todos los estados del autómata,
   - Σ: una lista finita de todos los símbolod admisibles,
   - q_0 el estado inicial,
   - F: una lista que indica los estados de aceptación,
   - δ: la función de transición, en un formato de tabla o lista de triplas $(q, a, q') \in Q \times \Sigma \times Q$, donde

     $$q' = \delta(q, a),$$

     la cual devuelve el valor de la transición $\delta(q, a)$, para un estado $q \in Q$ y un símbolo $a \in \Sigma$.

   Ilustrar el funcionamiento de estas funciones construyendo dos autómatas AFD de su elección.

2. La siguiente figura muestra un $\varepsilon-$AFN que acepta números decimales (con representación finita). En este caso, tenemos



   $M = (K, \Sigma, \delta, s, F)$, con $\Sigma = \{0, 1, \ldots, 9, ., +, -, \varepsilon\}$, $K = \{q_0, q_1, \ldots, q_5\}$, $s = q_0$, $F = \{q_5\}$.

   (a) Convertir el autómata anterior a su AFD equivalente.
   (b) Para el AFD obtenido en (a), mostrar el resultado de la función *derivation* para las siguientes cadenas:

   - $+0.1234567$
   - $1.61 - 8081$
   - $2024.3.3.3$

3. Construya autómatas finitos no-deterministas (AFN) para los siguientes lenguajes sobre $\Sigma = \{0, 1\}$:

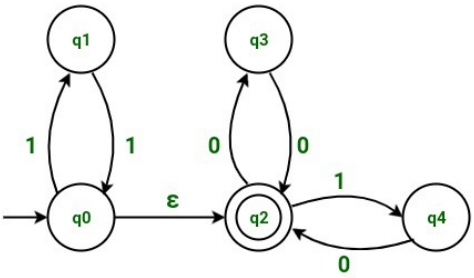
   i) Cadenas con un 1 en la penúltima posición.

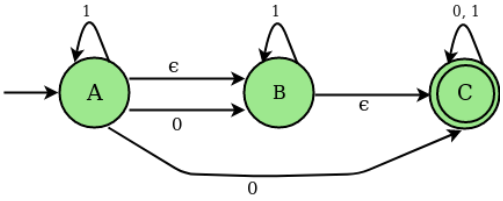
   ii) Cadenas que contengan (al menos) dos 0's consecutivos o dos 1's consecutivos.

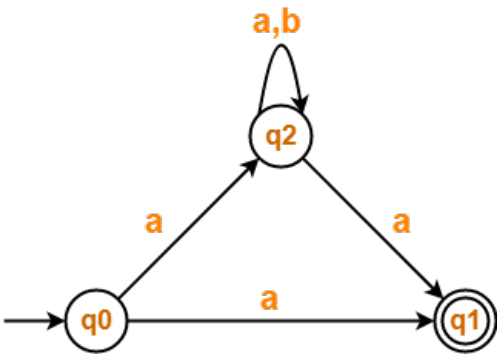
   iii) Cadenas binarias tales que la diferencia (absoluta) entre el número de ceros y el número de unos, es múltiplo de 5.

4. Para cada uno de los autómatas AFN construidos en el ejercicio anterior, construir su AFD equivalente.

5. Construir el AFD equivalente para los siguientes autómatas no deterministas.







6. Para cada una de las siguientes expresiones regulares, construya el árbol sintáctico de la expresión en notación *infix*. Luego, convierta cada regexp a notación *prefix* y *postfix*.

   i) $(0|1)^*11(0|1)^*$

   ii) $a(a + ab^*)^*$

   iii) $a^*b^*c^*$

   iv) $0(0 + 1 + 2)^*2$

   v) $[a - z]^2(\varepsilon + [a - z])[0 - 9]^+@uvg.edu.gt$