Universidad del Valle de Guatemala
Facultad de Ingeniería
Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Laboratorio No. 4

Para los siguientes incisos, siga estas instrucciones:

- Cree un repositorio **privado** en GitHub.com para alojar el código correspondiente para este laboratorio.
- Añada los usuarios de GitHub de su catedrático y de su(s) auxiliar(es) con rol de *collaborator*. Estos se encuentran en la sección de información del curso en Canvas.
- Coloque todas las instrucciones que considere pertinentes en un archivo denominado README.md en la raíz del repositorio.
- Para los incisos que no involucren la generación de código, cree una carpeta por separado en su repositorio y coloque las respuestas en un documento con formato PDF
- Grabe un video de no más de 10 minutos donde muestre la ejecución de sus programas para el ejercicio 1. Súbalo a YouTube como video no listado y adjúntelo en el README de su repositorio.

Ejercicio No. 1 (90%) – En laboratorios anteriores implementó un balanceo de expresiones regulares, luego implementó Shunting Yard para convertir una Expresión Regular en notación infix a postfix y por último creó un árbol abstracto sintáctico que representa la expresión en postfix de forma visual.

Ahora, utilizando todos estos elementos, deberá construir (con base en el árbol generado) el AFN resultante de aplicar el algoritmo de Thompson al árbol construido y mostrar su dibujo en pantalla, así como deberá también simular el AFN para reconocer cadenas de la expresión regular asociada.

Especificación del funcionamiento del programa

- Entrada
  - Una expresión regular r.
  - Una cadena w.
- Salida
  - Por cada AFN generado a partir de r:
    - Una imagen con el Grafo correspondiente para el AFN generado, mostrando el estado inicial, los estados adicionales, el estado de aceptación y las transiciones con sus símbolos correspondientes.
    - La simulación del AFN al colocar la cadena w: el programa debe indicar si $\mathrm{w} \in \mathrm{L(r)}$ con un "sí" en caso el enunciado anterior sea correcto, de lo contrario deberá mostrar un "no".
- Su programa debe leer un archivo de texto y procesar cada línea en este archivo. Cada línea deberá de tener una expresión regular, correspondiente a la siguiente lista:
  - $(a*|b*)+$
  - $((\varepsilon|a)|b*)*$
  - $(a|b)*abb(a|b)*$
  - $0?(1?)?0*$
- Muestre la ejecución completa de su programa y explique brevemente en el video su código.

Ejercicio No. 2 (10%) – Utilice el *Pumping Lemma* para demostrar que el Lenguaje $A = \{yy \mid y \in \{0,1\}^*\}$ no es regular.

- y son todas las cadenas que pueden ser generadas con 0's y 1's.
- El lenguaje está conformado entonces por todas las cadenas y seguidas de la misma y. Por ejemplo, si $y = 01$ entonces una cadena parte del lenguaje será $yy = 0101$.
- Tome como base de su demostración que $S = 0^P 10^P 1$, con P = pumping length.

Ejercicio No. 2 (10%) – Utilice el *Pumping Lemma* para demostrar que el Lenguaje $A = \{yy \mid y \in \{0,1\}^*\}$ no es regular.

- y son todas las cadenas que pueden ser generadas con 0's y 1's.
- El lenguaje está conformado entonces por todas las cadenas y seguidas de la misma y. Por ejemplo, si $y = 01$ entonces una cadena parte del lenguaje será $yy = 0101$.
- Tome como base de su demostración que $S = 0^P 10^P 1$, con P = pumping length.

$W = 0^P 10^P 1$

$x = 0^P$

$y = 10^P$

$z = 1$

Tenemos $0^P 10^P 1 \Rightarrow 0^{2P} 11$, Pero sabemos que $|xyz| \leq P$, Pero $|10^P 10^P| \leq 2P$, por lo tanto no es un "regex" ☐