UNIVERSIDAD DEL VALLE DE GUATEMALA

Teoría de la computación

Sección 10

“Proyecto 1”

Javier André Salazar de Léon- 18764

Javier Emilio Alvarez Cifuentes - 18051

Andrea Maria Panigua Acevedo - 18733

Modelo:

Para la creación de este proyecto se utilizaron los algoritmos vistos en clase, tales como el algoritmo de construcción de Thompson, la construcción por subconjuntos, la construcción de un autómata a partir de una expresión regular, así como también la minimización de autómatas.

Todos esto nos sirve para correr simulaciones a partir de una cadena que es ingresada por el usuario, la cual pasa por los algoritmos que fueron realizados para su análisis y luego retorna el resultado de la cadena en los diferentes algoritmos, así como también un archivo del grafo resultante de dicho autómata. Esto con la finalidad de permitir al usuario tener una respuesta de que la cadena testeada si es parte del autómata y si puede ser utilizada como parte de su generación.

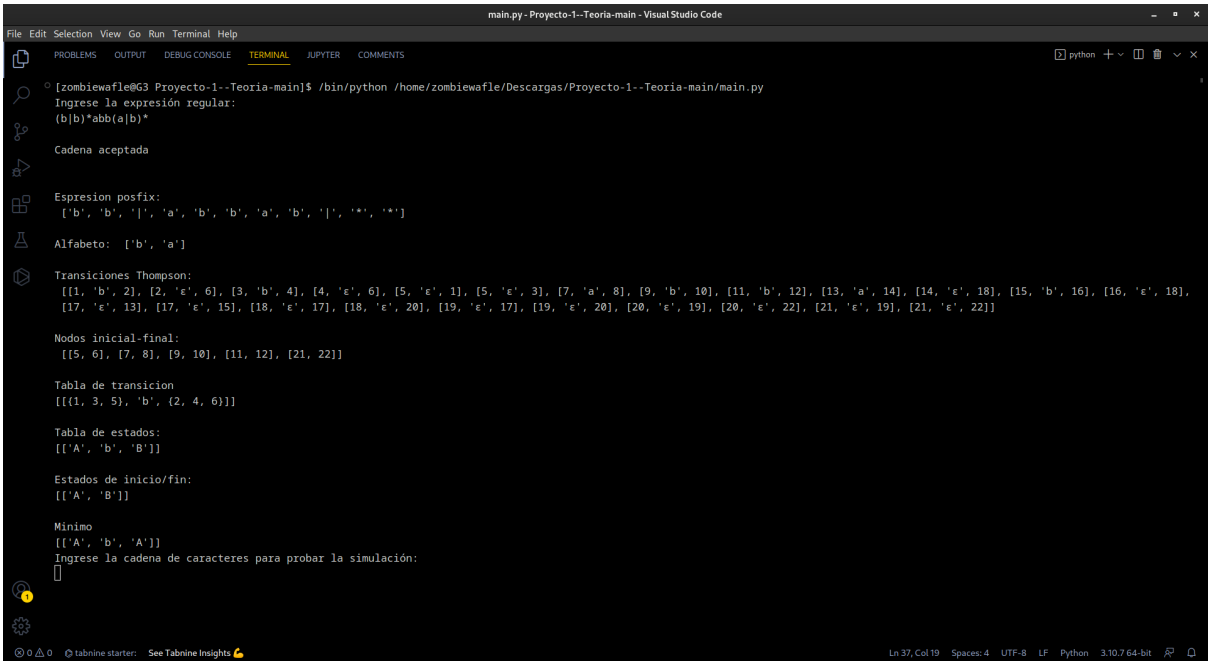
Como equipo decidimos utilizar diferentes librerías que nos ayudarán a la generación de los autómatas, específicamente de los grafos y combinar esto con nuestro conocimiento teórico acerca del tema para poder realizar un programa que sea eficiente y funcional.

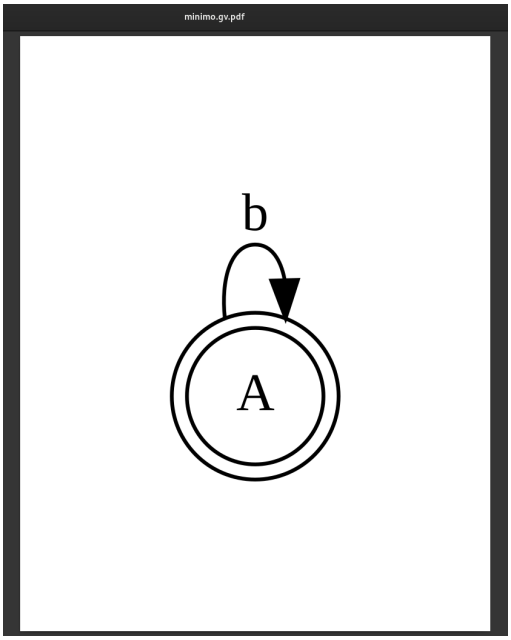
Discusión:

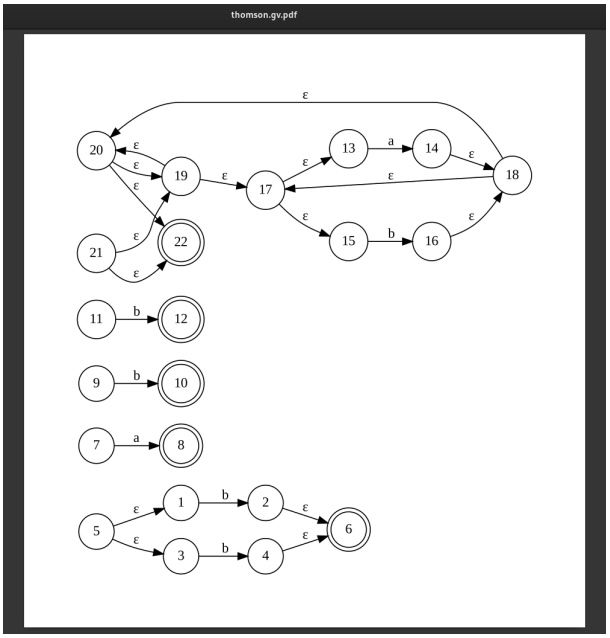
Durante la creación de los distintos programas nos fuimos encontrando con muchos obstáculos, comenzando con la documentación de algunas librerías utilizadas para generar los gráfos, desde dificultades con la instalación de dichas librerías hasta con su ejecución ya que ciertos “paths” se usaban específicamente en Windows o en otros sistemas operativos o incluso la forma en que cada OS maneja las indentaciones en el código. Como recomendación, sería verificar que el OS de los integrantes sea el mismo, que todos tengan la misma versión del lenguaje utilizado (python en nuestro caso). Además recomendamos decidirse por un lenguaje con mucha documentación y ejemplos, como lo son Java y C.

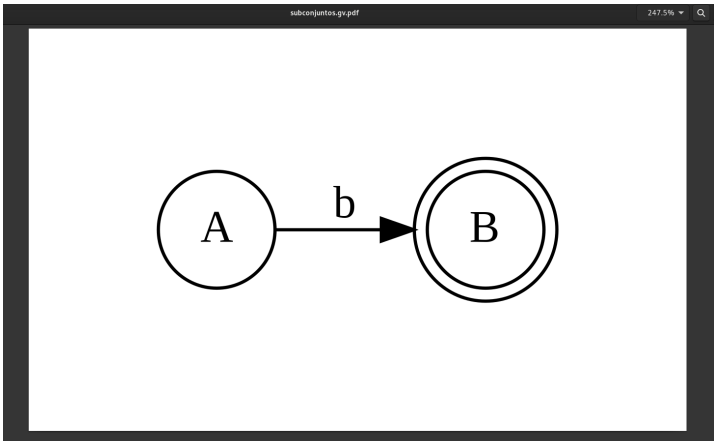
Otros inconvenientes que fuimos encontrando, fueron comprender cómo trasladar un proceso manual como lo es el algoritmo de Yamada-Thompson a código, aún más porque el algoritmo es muy visual y no tanto un proceso mecánico fácilmente programable.

Resultados:


main.py - Proyecto-1--Teoria-main - Visual Studio Code
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER COMMENTS
[zombiewafle@G3 Proyecto-1--Teoria-main]$ /bin/python /home/zombiewafle/Descargas/Proyecto-1--Teoria-main/main.py
Ingrese la expresión regular:
(b|b)*abb(a|b)*
Cadena aceptada
Espresion posfix:
['b', 'b', '|', 'a', 'b', 'b', 'a', 'b', '|', '*', '*']
Alfabeto: ['b', 'a']
Transiciones Thompson:
[[1, 'b', 2], [2, 'ε', 6], [3, 'b', 4], [4, 'ε', 6], [5, 'ε', 1], [5, 'ε', 3], [7, 'a', 8], [9, 'b', 10], [11, 'b', 12], [13, 'a', 14], [14, 'ε', 18], [15, 'b', 16], [16, 'ε', 18],
[17, 'ε', 13], [17, 'ε', 15], [18, 'ε', 17], [18, 'ε', 20], [19, 'ε', 17], [19, 'ε', 20], [20, 'ε', 19], [20, 'ε', 22], [21, 'ε', 19], [21, 'ε', 22]]
Nodos inicial-final:
[[5, 6], [7, 8], [9, 10], [11, 12], [21, 22]]
Tabla de transicion
[[{1, 3, 5}, 'b', {2, 4, 6}]]
Tabla de estados:
[['A', 'b', 'B']]
Estados de inicio/fin:
[['A', 'B']]
Minimo
[['A', 'b', 'A']]
Ingrese la cadena de caracteres para probar la simulación:
Ln 37, Col 19 Spaces: 4 UTF-8 LF Python 3.10.7 64-bit


minimo.gv.pdf
b
A


thomson.gv.pdf
ε
20
19
17
13
a
14
18
15
b
16
21
22
11
b
12
9
b
10
7
a
8
5
1
b
2
3
b
4
6


subconjuntos.gv.pdf
247.5%
A
b
B

Segunda prueba: