

s

Instituto Politécnico Nacional

Escuela Superior de Computo

Teoría de la computación

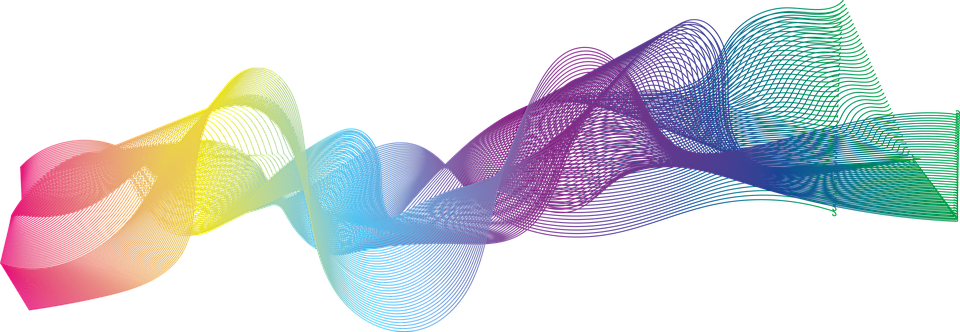
Dra. Sandra Díaz Santiago

**Reporte 3º de laboratorio**

Equipo:

* García González Aarón Antonio
* Osornio Zambrano Alberto Aacini

Marzo 20, 2019



Índice

[Digitalización del ejercicio teórico 3](#_Toc3892579)

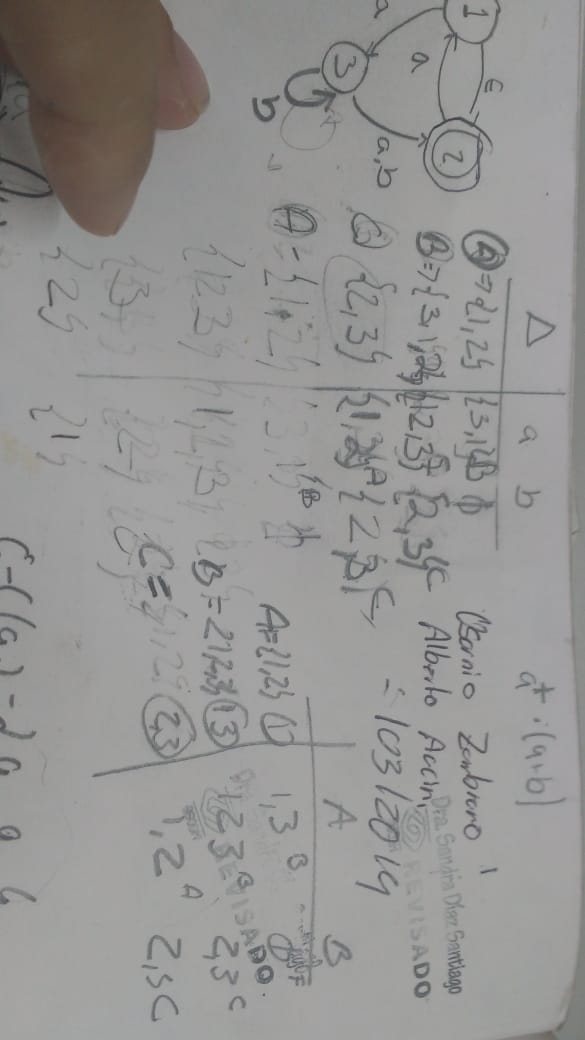
[Desarrollo 3](#_Toc3892580)

[AFD usados para probar el funcionamiento del programa 12](#_Toc3892581)

[Referencias bibliográficas 13](#_Toc3892582)

# Digitalización del ejercicio teórico

García González Aarón Antonio



Osornio Zambrano Alberto Aacini

# Desarrollo

Ejercicio 1 AND a AFD

El código empieza abriendo el archivo donde se indica la tupla del autómata no determinístico (AND), en las primeras líneas se indica los estados, el alfabeto y el estado inicial. En base a estos datos crea un array de dos dimensiones, del tamaño de mxn , donde m es el número de estados, y n es el número de símbolos más uno , que es donde se guardara el estado al que pertenecen las transiciones. Posteriormente lee las siguientes líneas que son la función delta del autómata, cada línea indica el estado y las transiciones acomodadas conforme se especificó el alfabeto y el estado en sus respectivas líneas. Estas líneas las divide con el delimitador “,” y cada ítem resultante es se agrega a la matriz. Se repite esta acción el número de estados que se especificó. Finalmente se lee el archivo donde se especifica los estados de aceptación

f = open("and.csv")

estados=f.readline()

alfabeto=f.readline()

estadoInicial =f.readline()

f.readline() # leer cabecera de la tabla (no se usara)

#quitar salto de inea a fina de cada string

estados = estados[:len(estados) - 1]

alfabeto = alfabeto[:len(alfabeto) - 1]

estadoInicial = estadoInicial[:len(estadoInicial) - 1]

m=len(estados)

n=len(alfabeto)+1 #dejar un espacio mas para la entrada de estado

mAND=[[m] \* n for i in range (m)]# crear esoacio de memoria para mx n

for i in range (m):

linealeida=f.readline()#leer linea

linealeida= linealeida[:len(linealeida) - 1]#quitar salto de linea

x= linealeida.split(",")#dividir por las comas

mAND[i]=x # el array resultante sera nuestro renglon

estadosAceptacion =f.readline()#leer estado de aceptacion

print "Estados Aceptacion"

print estadosAceptacion

f.close();

En la segunda parte del Código se pasara la tabla de transición e un AND a un autómata finito determinístico (AFD). Se inicia creando una lista, donde se guardaran los estados con sus transiciones y una lista de estadoAFD, donde se guardaran los nuevos estados que surjan.

Para esta parte del código se creo una función, al cual se le pasa un estado, y agrega mas ítems en medida que alfabetos haya, cada ítem tendrá los estados que tenga el estado pasado con el símbolo en cuestión, Si hay mas de un estado pasado en el parámetro de la función, los concatenara pero no repitiera estados . Al terminar de calcular el estado (s) lo agrega a la lista de AFD.

#agrega un estado al AFD

def agregarEstadoAFD(strEstado\_s):

aux=[]

aux.append(strEstado\_s)

for k in range( len(aux[0] )): #recorrer cada uno de los estados ingresados

for j in range( len (estados) ):#buscar entre los estados del AND

if mAND[j][0]== aux[0][k]:#el que empiese con el caracter ingresado

for i in range(len (alfabeto) ): #recorer el alfabeto

if len(aux)<= (i+1) : #si no hay entrada en se alfabeto

aux.append(mAND[j][i+1])#pasar todos los estados en el AND #de lo contrario

else:

aux[i+1]=sumNoRepStr(mAND[j][i+1],aux[i+1] )

mAFD.append(aux);

Luego se inicia insertando el estado inicial de AND junto con las transiciones correspondientes, con la función descrita anteriormente. Luego recorrerá la tabla buscando estados en las transiciones que no se hayan indicado su transición correspondiente. Esto también con ayuda de la función descrita arriba. Si se llega al final de la lista es que no hay mas estados por agregar

#crear ADN

mAFD=[]

estadosAFD=[]#aqui se guardaran los estados del AFD

agregarEstadoAFD(estadoInicial) #agregar estado inicial

estadosAFD.append(estadoInicial);

#recorrer todos las tranciciones para los estados

#verificar si el estado ya se agrego

#de lo contrario agregarlo

cont =0

while True: #recorrer cuantos estados haya en AFD

for i in range(1,len(alfabeto)+1): #recorrer cuantos alfabetos existan

if (not existeEstadoAFD(mAFD[cont][i]) and mAFD[cont][i] !="-" ): #si no esta agregado y no es vacio (-)

agregarEstadoAFD(mAFD[cont][i]) #agregar estado

estadosAFD.append(mAFD[cont][i]);

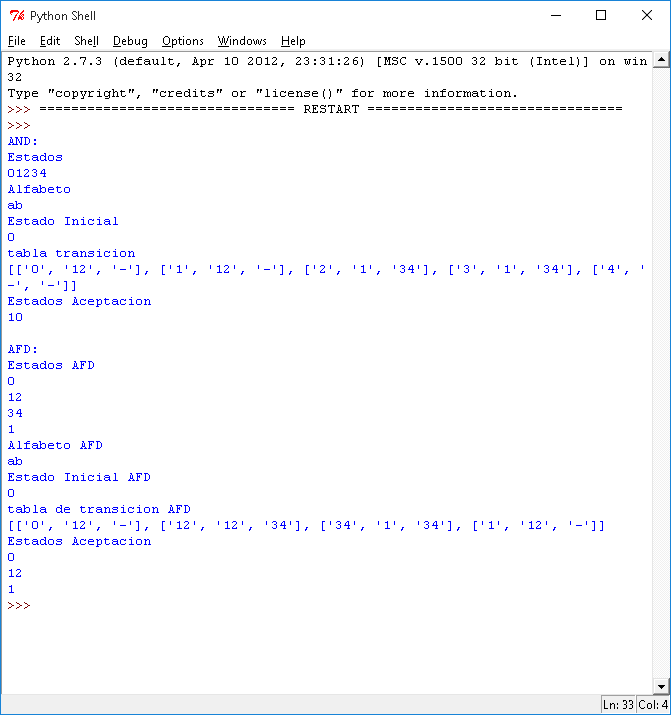
cont=cont+1

if(cont == (len(mAFD)) ): #si ya no hay mas estados salir

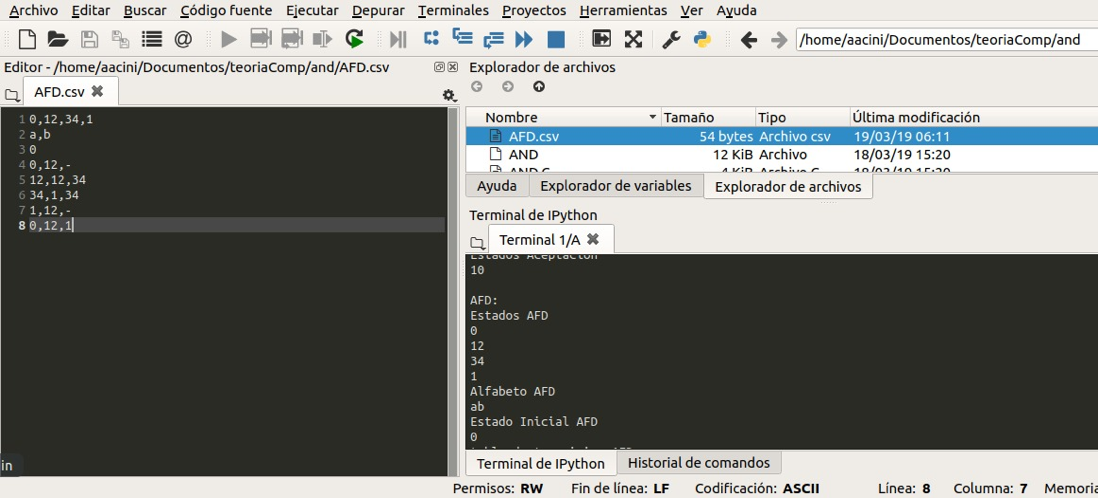
break

Luego se calcula de la lista de estados, los estados de aceptación , buscando los que tengan algún estado de aceptación en sus estados componentes.

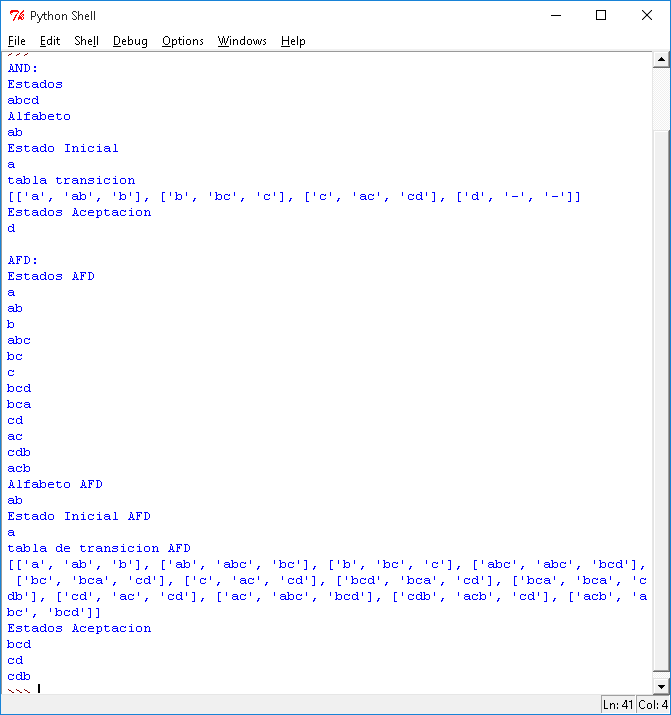
Prueba con el primer autómata no deterministico(and.csv)



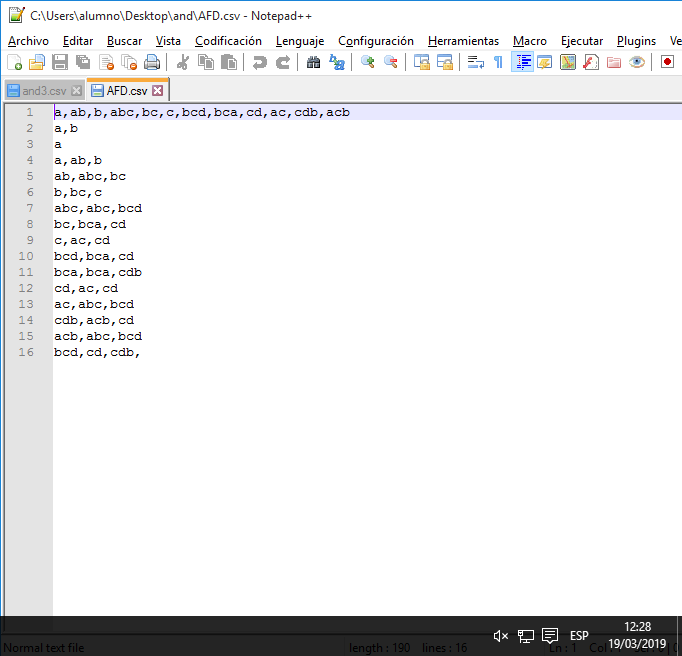
Los resultados se guardan en AFD.csv



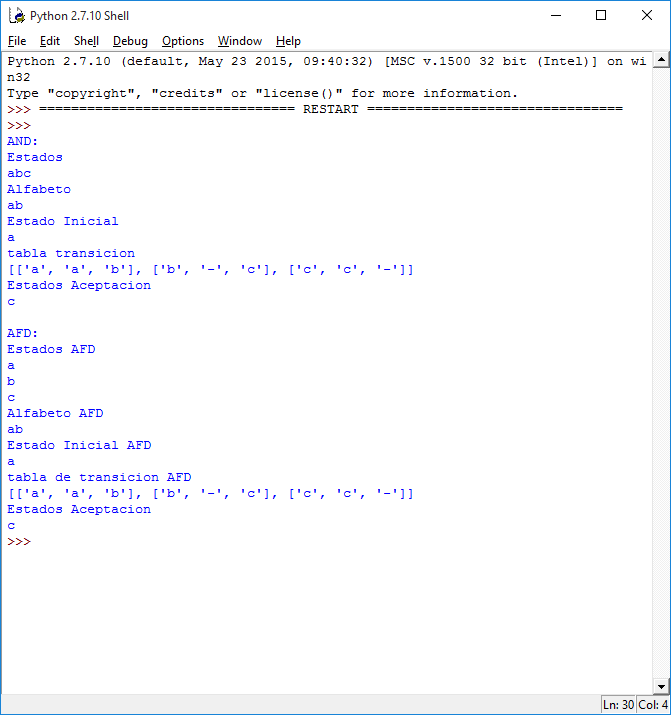
Prueba con AND 2 (AND2.csv)

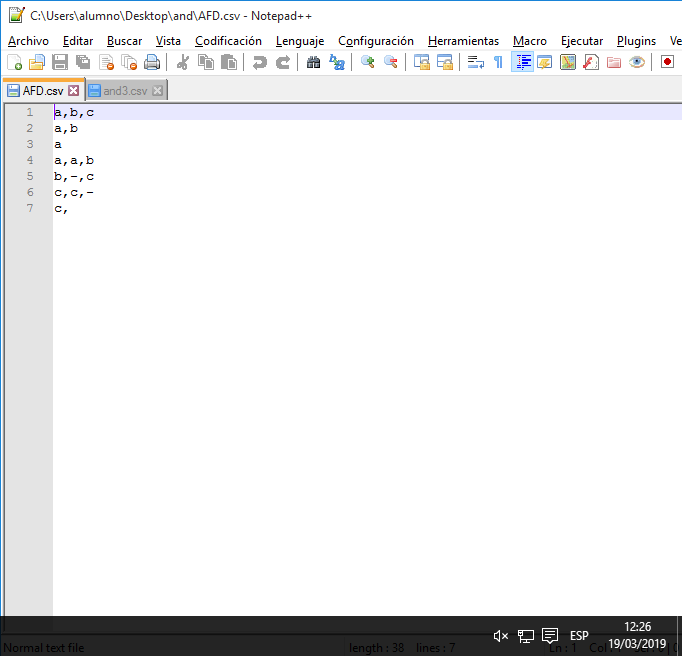


Salida del archivo



Prueba con AND 3 (and3.csv)



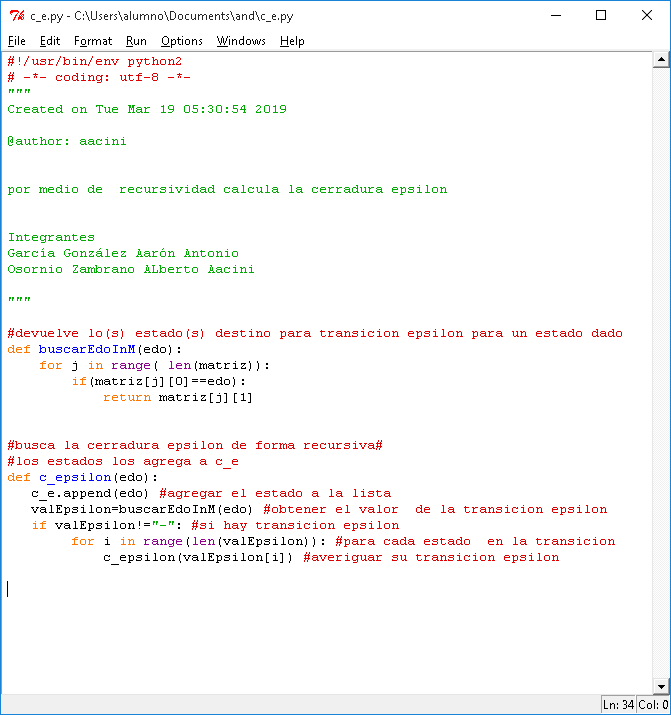


Archivo de salida

Ejercicio 2, cerradura épsilon

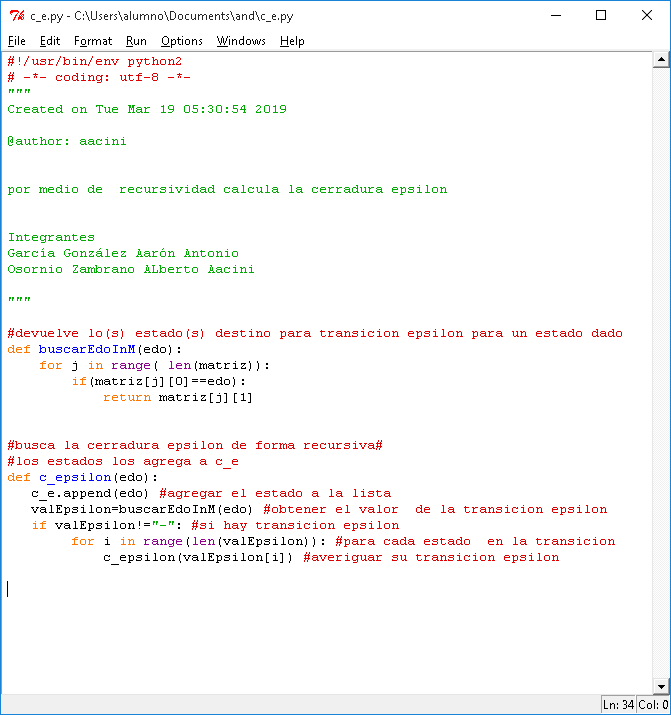
Esta función calcula la cerradura épsilon de un estado de un AND

Para esta parte se crea una función auxiliar para buscar la(s) transición(es) épsilon de un estado, esta los buscara en la matriz



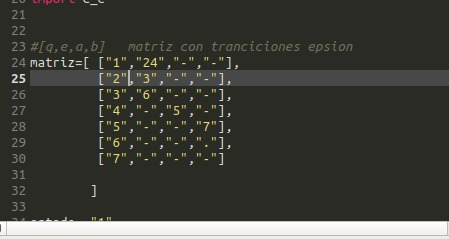
La función c-epsion calcula por medio de recursividad la cerradura épsilon, primero se agrega el edo pasado

Si el estado tiene alguna transición épsilon , calcula la cerradura épsilon de cada estado en dicha trancicion,

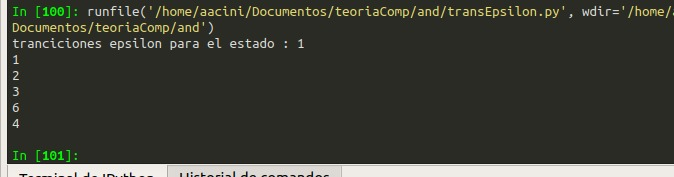


Para este ejercicio usamos la siguiente tabla de transición, el cual esta acomodado en el orden [q,e,a,b]

Donde q es el estado, e es la transición con épsilon, ay b son símbolos del alfabeto



El resultado es



Ejercio 3

Este ejercicio calcula la cerradura épsilon de todos los estados de un AND, y los guarda en un archivo,

El código recorre cada estado de el AND y con ayuda de la función el ejercicio anterior construye una lista con la cerradura espión para cada esta, luego lo guarda en archivo

archivoW = open("c-e.txt","w")

for i in range(len(matriz) ):

c\_e=[]#lista de estados

estadoEnTurno=matriz[i][0]#guardar referencia del estado en turno

c\_epsilon(estadoEnTurno) #calcular cerradura epsilon para el estado

print "c-e("+estadoEnTurno+")=" #imprimir el estado actual

archivoW.write("e-c("+matriz[i][0]+")={") #y en archivo

for j in range(len(c\_e)): #imprimir cada estado resultante

print c\_e[j]

archivoW.write(c\_e[j]) #tambien guardarlo en pantalla

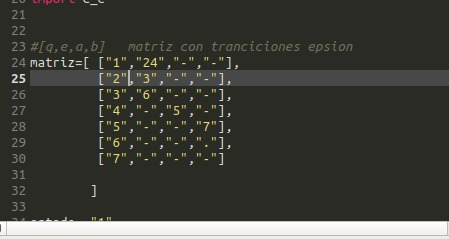
if(j<(len(c\_e)-1)): #si no es el ultimo item

archivoW.write(",") #ponerle una coma

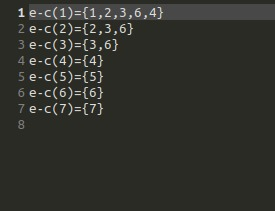
archivoW.write("}\n") #cuando terminen de imprimirse los item cerrrar llaves y dar salto de linea

archivoW.close()

La tabla de transición probada es la misma que el ejercicio anterior

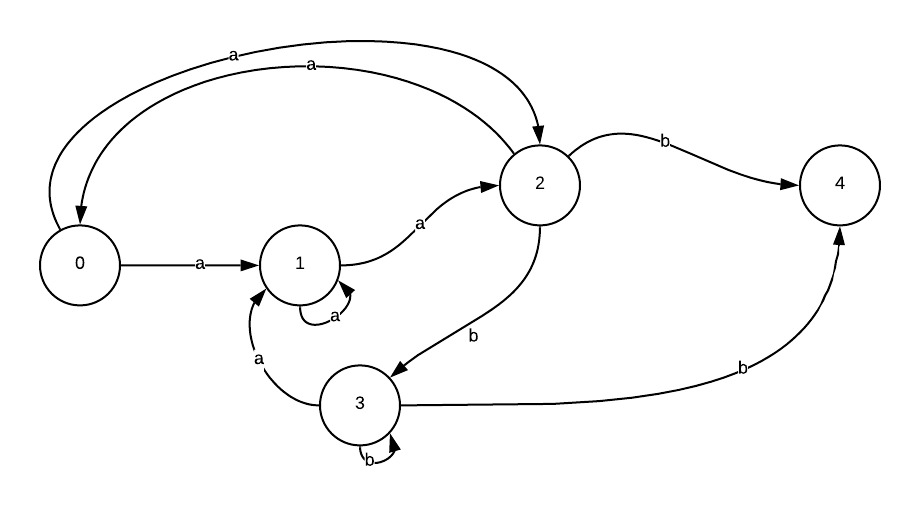


Archivo generado

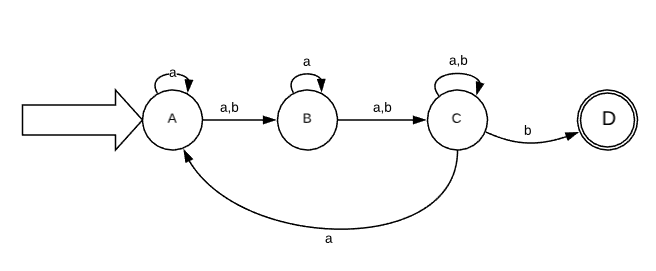


# AFD usados para probar el funcionamiento del programa

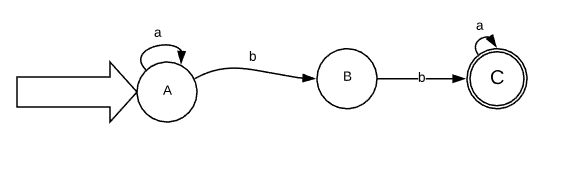
Automata 1



Autómata 2



Autómata 3



# Referencias bibliográficas

* Elisa Viso G. (2008). introducción a la teoría de la computación . México, D.F.: Facultad de ciencias UNAM.
* Franco Martinez E. Teoria Computacional Clase 08 “Automatas Finitos”

[internet] Recuperado el 10/02/2019 de www.eafranco.com/?p=docencia/teoriacomputacional/index.htm

* Kelley Dean. Teoría de autómatas y lenguajes formales. Prentice Hall. España. 1995, 302 págs. ISBN:0­13­ 518705­2.

Hopcroft John, Motwani Rajeev, Ullman Jeffrey. Teoría de autómatas, lenguajes y computación. Addison Wesley, 2008, 452 págs.,ISBN: 978­84­7829­088­8.