PROYECTO 3

AUTÓMATAS

Matemáticas Computacionales
Dr. Victor de la Cueva

Hecho por:
Rafael Correa A01019498

21/06/2016

Manual de Usuario

Crear un archivo con un formato específico:

- Para definir el conjunto de nodos Q, empezar la línea con ‘q’ seguido del nombre de los estados separados por espacios.
- Para definir el conjunto de entradas S, empezar la linea con ‘e’, seguido de las entradas separadas por espacios.
- Para definir la función □, empezar la línea con ‘d’, seguido de el nodo en que se encuentra, la entrada recibida y los nodos a los que se llega. Por ejemplo, una tabla de transición simple:

	0	1
q1	q1	Q1 q2
q2	q1	\empty

  Se representaría en 4 (o 3 líneas):

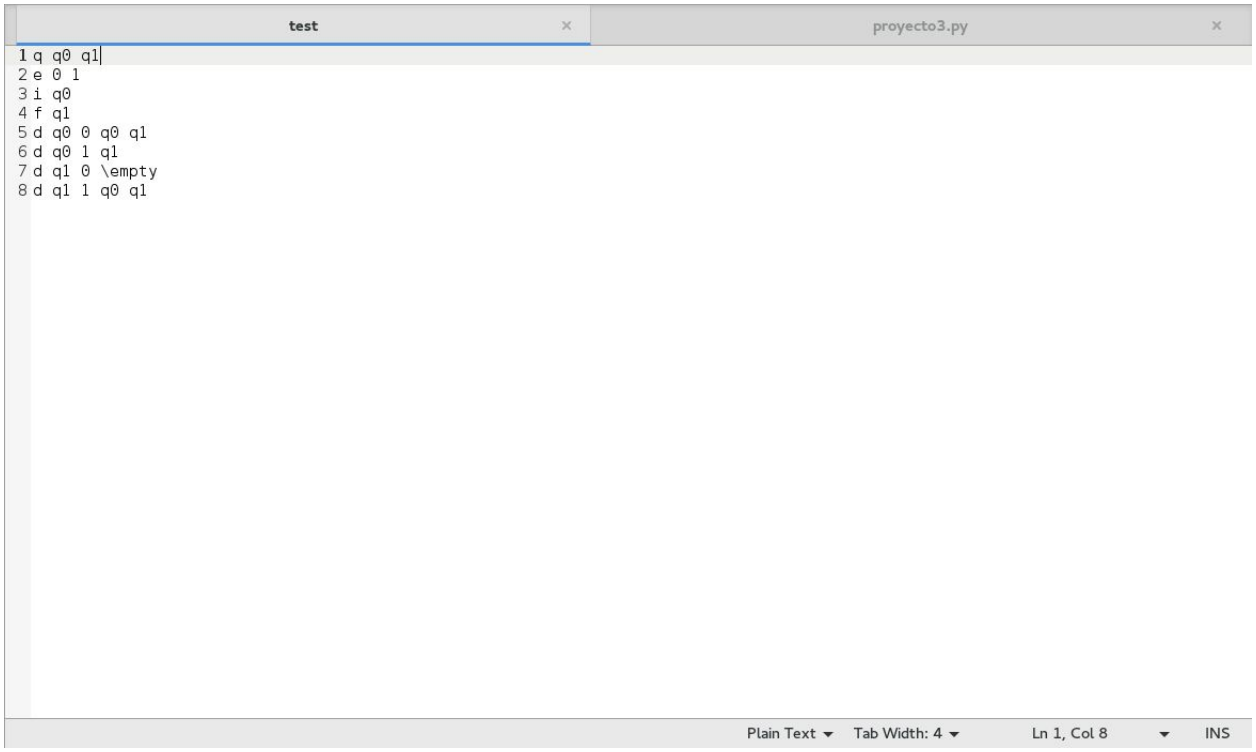
  d q1 0 q1
  d q1 1 q1 q2
  d q2 0 q1
  d q2 1 \empty

  Alternativamente, la última línea se puede omitir.
- Para definir el nodo inicial, empezar la línea con ‘i’, seguido del nombre del nodo inicial
- Para definir el conjunto de estados finales, empezar la línea con ‘f’, seguido de los estados finales separados por espacios.

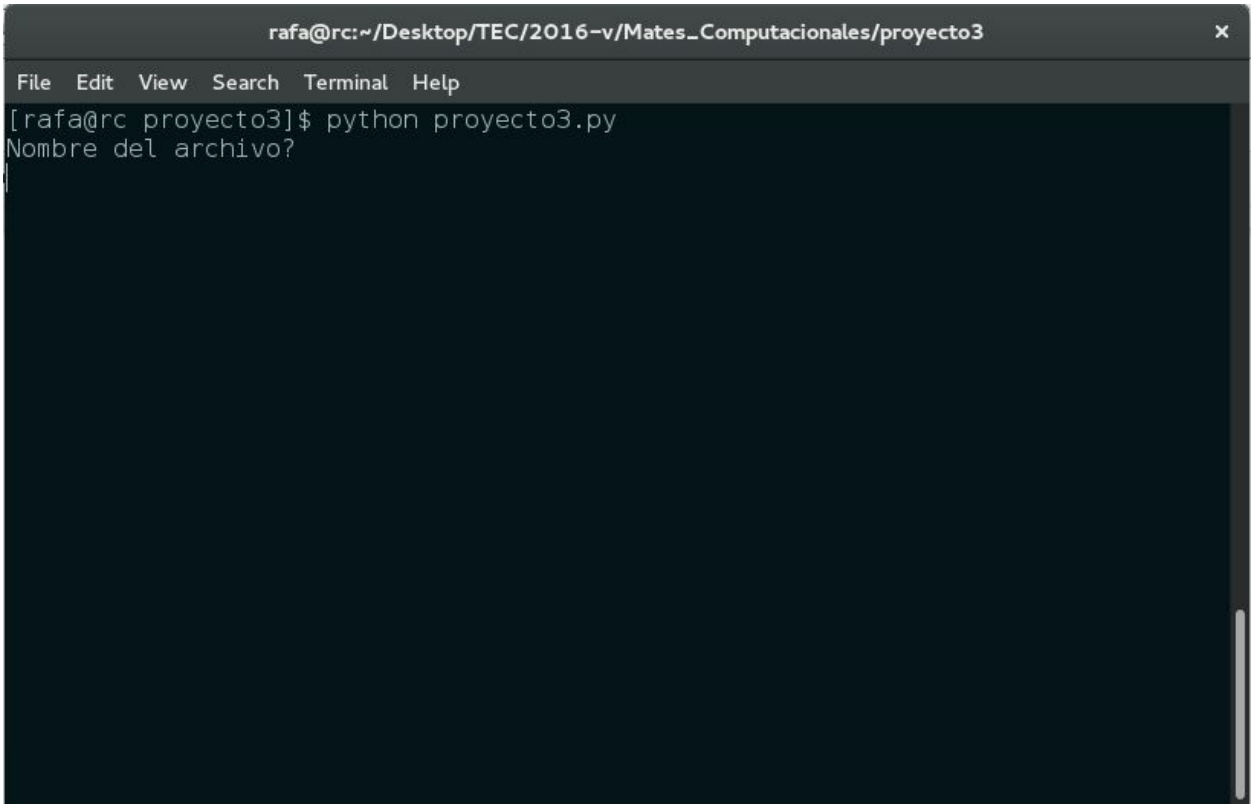
La declaración de cada conjunto puede ir en cualquier orden y en cualquier número de líneas e.j.
‘q q0 q1’ es equivalente a :

- q q0
- q q1

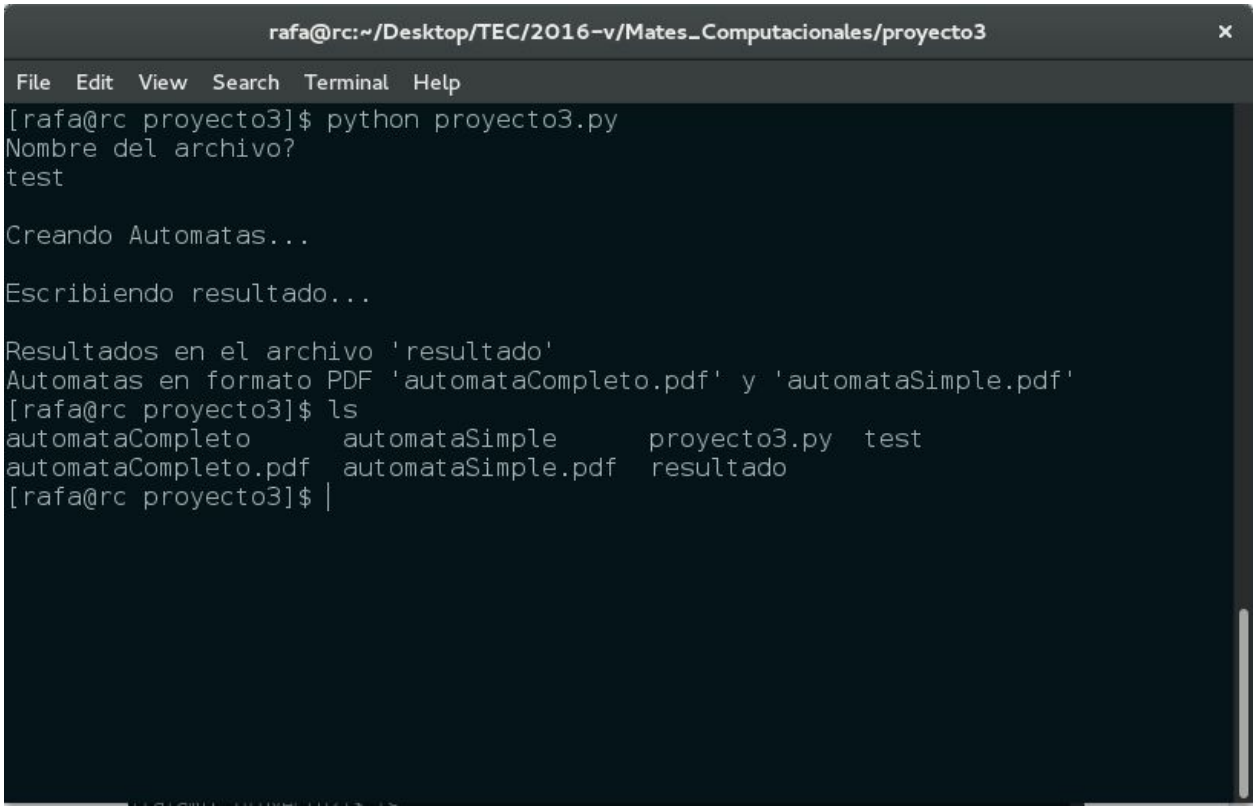
A continuacion, como se debe ingresar el ejemplo definido en las instrucciones del proyecto.



Correr el script de python (con python2.7) y escribir el nombre del archivo. Asegurarse de tener las librerías graphviz e itertools (esta debería venir incluida con la instalación de python) para python:

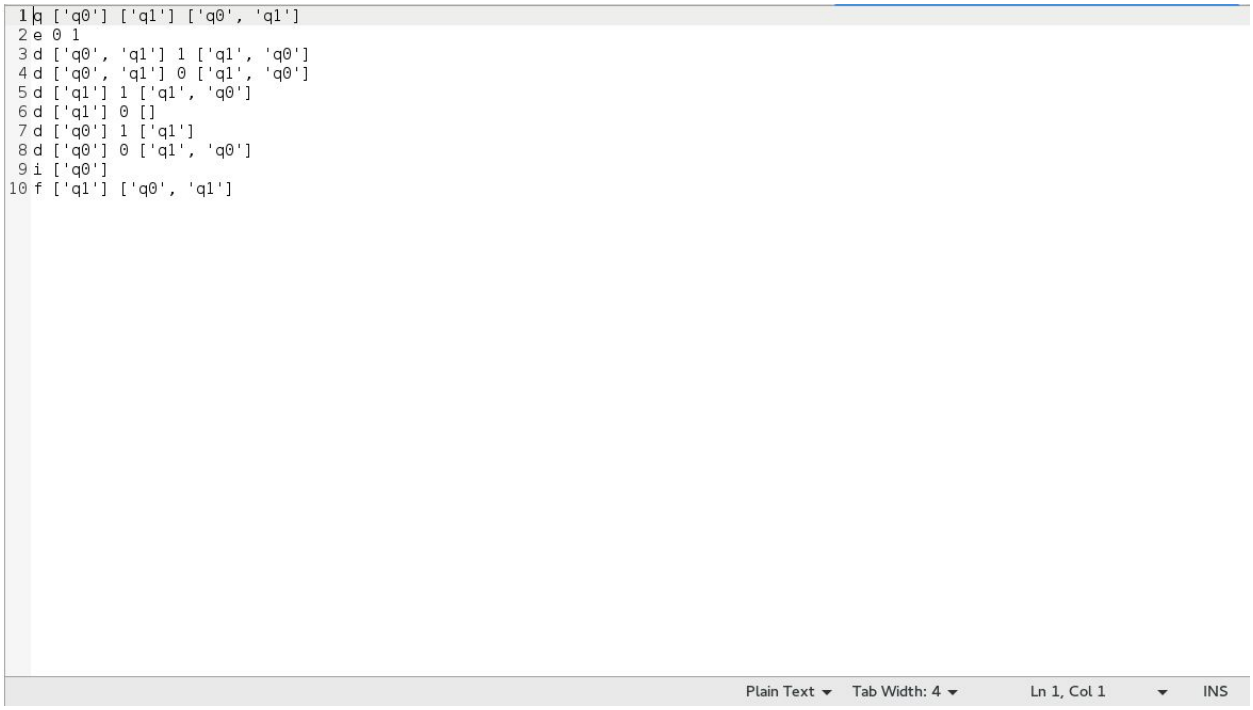


Ingresar el nombre del archivo (en este ejemplo ‘test’) y el programa terminara después de crear tres archivos.



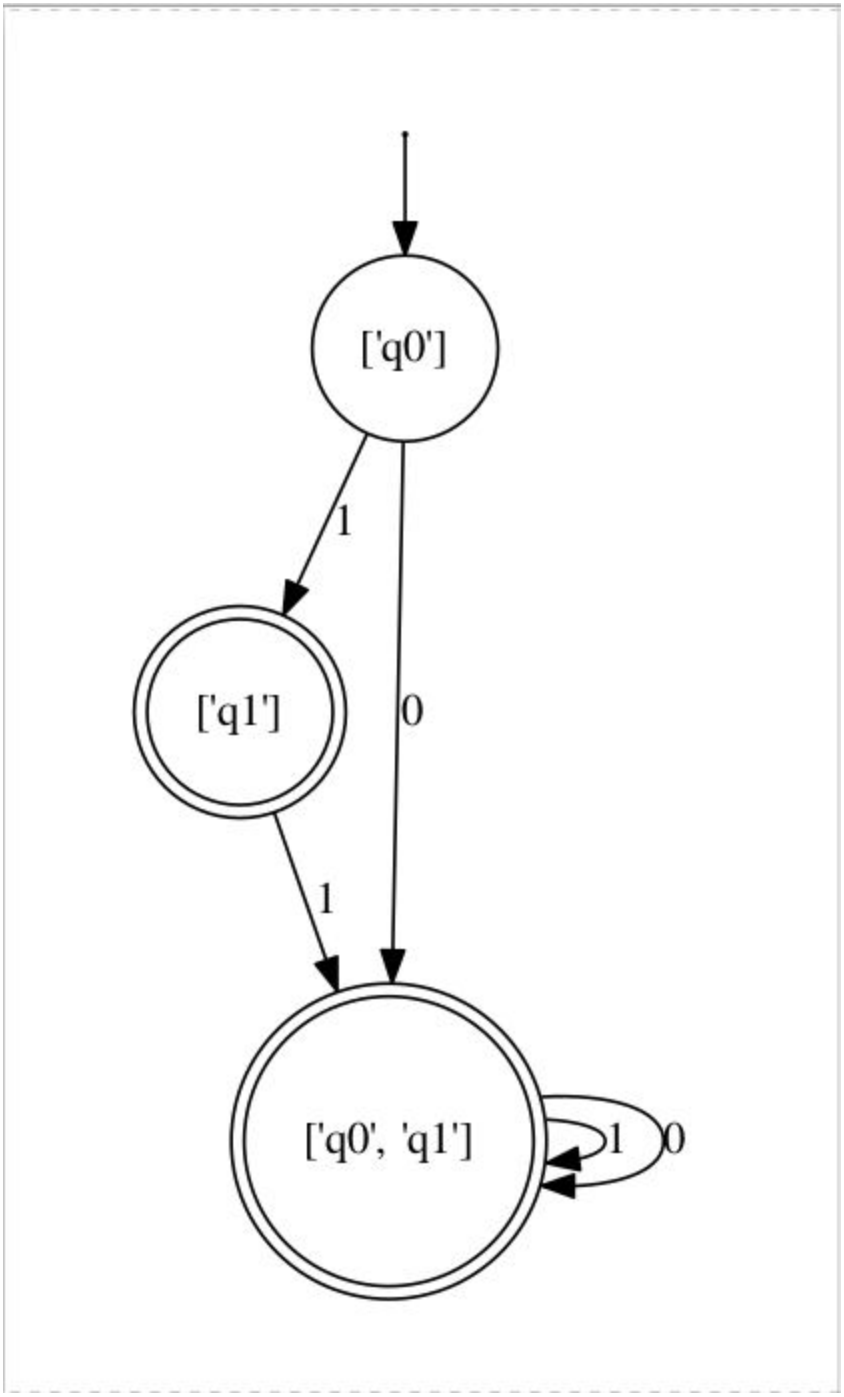
Los tres archivos creados por el programa son:

- resultado: Un archivo de texto con el resultado del DFA con un formato parecido al de la entrada:



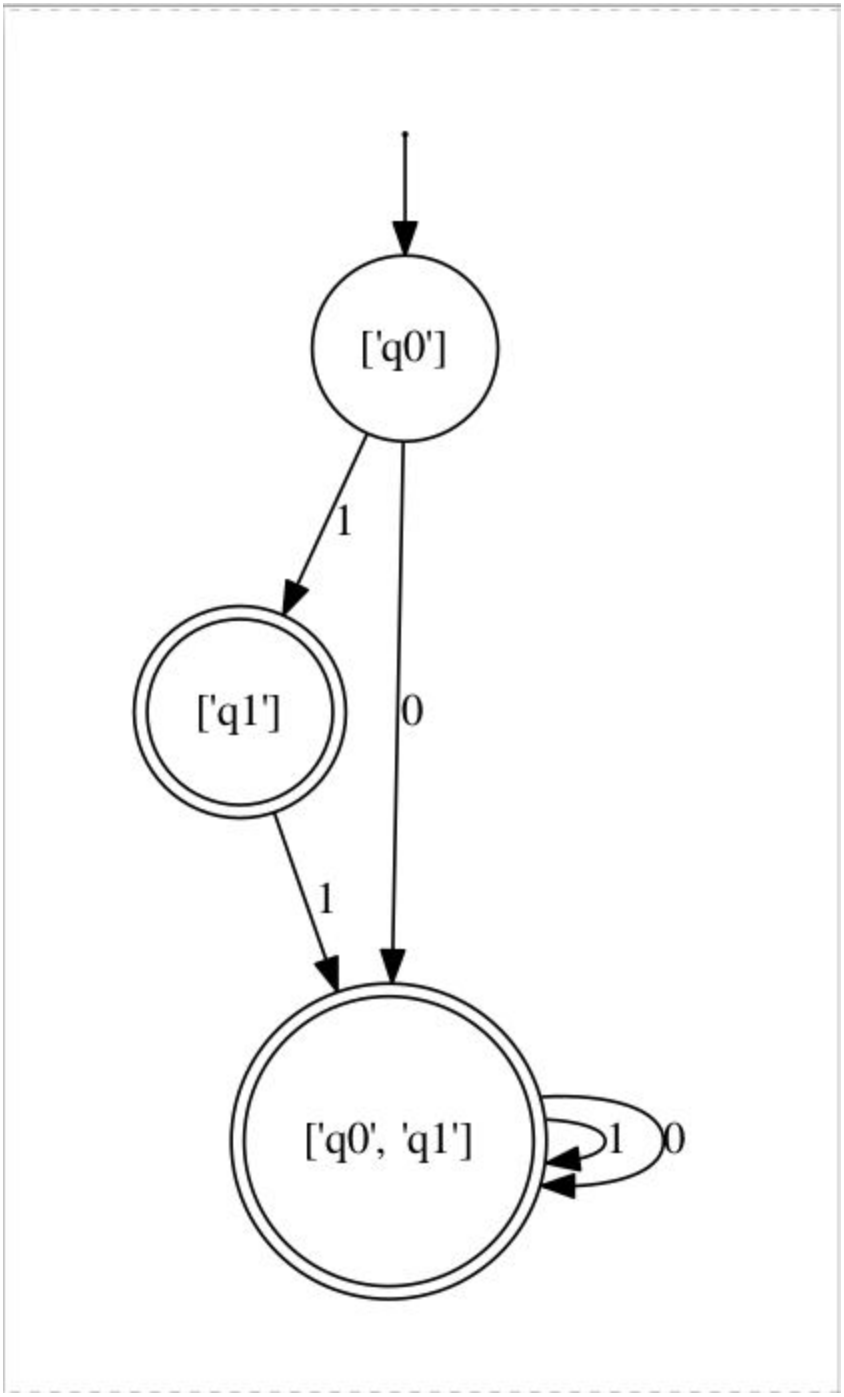
- automataCompleto.pdf: Un archivo en formato pdf que contiene la representación gráfica del autómata, con todos los estados (incluyendo a los que

no se puede llegar):



- automataSimple.pdf: Un archivo en formato pdf que contiene la representación gráfica del autómata, con únicamente los estados posibles dada la tabla de

transición:



En este caso ambos diagramas son idénticos porque existe una transición a todos los nodos posibles.

Descripción Técnica

Para representar la tabla de transición, se utilizó un diccionario de diccionarios, utilizando como llaves el estado donde se encuentra y la entrada. Para accesar los estados a los que se debe ir, simplemente se busca por las llaves, ej. Diccionario[‘[q0]’][‘0’] daria como resultado los estados a los que se puede ir desde q0 cuando se recibe un 0 como input.

```
D = {}
if linea[0] == 'd':
                key = linea[1]
                key2 = linea[2]
                values = linea[3:]
                if empty in values:
                        values.remove(empty)
                tempDict = {key2 : set(values)}
                if key not in D:
                        D[key] = tempDict
                else:
                        D[key].update(tempDict)
```

Para calcular Q’, F’ y □’ se hizo lo siguiente:

Sacar todas las combinaciones posibles entre los estados de Q:

```
#calcular Q'
def calcQ(Q):
        Qf =[]
        for i in xrange(1,len(Q)+1):
                Qf.extend(list(itertools.combinations(Q,i)))
        result = [list(q) for q in Qf]
        return result
```

Encontrar los estados de Q’ que contienen algún estado en F

```
#calcular F'
def calcF(Q, F):
        Ff =[]
        for f in F:
                Ff.extend([q for q in Q if f in q and q not in Ff])
        return Ff
```

Para todo estado compuesto de más de un estado de Q, sacar la unión de sus destinos y agregar el estado compuesto con esta unión a □’

```
#calcular delta'
```

```
def calcD(Q, D):
	D1 = {}
	for q in Q:
		for element in q:
			if element in D:
				if repr(q) not in D1:
					D1[repr(q)] = D[element]
				else:
					A = D1[repr(q)]
					B = D[element]
					D1[repr(q)] = {x: list(set(A[x]).union(B[x])) for
				x in set(A).union(B) if x in A and x in B}

	return D1
```

Para encontrar el autómata simple (el que sólo dibuja los estados a los que se puede llegar), se hizo una función recursiva que empieza desde el nodo inicial y recorre el autómata en todos los caminos posibles con una □ dada.

```
def recorrer(D,F,q,dot, recorrido):
	recorrido.append(repr(q))
	key = repr(q)
	for i in D[key]:
		destino = (D[key][i])
		#sort para que los subconjuntos siempre esten en el mismo orden
		destino = sorted(destino)
		if q in F:
			dot.node(key, peripheries = '2')

		if len(destino):
		#Dibujar la arista...
			dot.edge(key, repr(destino), label = i)

			#Busca si ya recorrio este nodo...
			if repr(destino) not in recorrido:
				#recorre para los destinos
				recorrer(D, F, destino, dot, recorrido)
```