



UNIVERSIDAD  
DE SANTIAGO  
DE CHILE

UNIVERSIDAD DE SANTIAGO DE CHILE

FUNDAMENTOS DE CIENCIA DE LA COMPUTACIÓN

---

# Laboratorio 1

---

*Autor:*

Brady Cardenas  
David Sanhueza Andrés

28 de Marzo de 2017

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Funcionamiento</b>	<b>3</b>
<b>3. Especificaciones</b>	<b>3</b>
<b>4. Algoritmos</b>	<b>4</b>
<b>5. Ejemplo de Uso</b>	<b>5</b>
5.1. Función de Transición de un DFA . . . . .	5
5.2. Matriz de la Función de Transición . . . . .	5
5.3. Estados de Aceptación . . . . .	5
5.4. String de Entrada . . . . .	6
5.5. Verificación Lenguaje Aceptado . . . . .	6
<b>6. Curvas de Desempeño de Resultados</b>	<b>7</b>
<b>7. Conclusiones</b>	<b>8</b>
<b>8. Plataforma Computacional</b>	<b>8</b>
<b>9. Bibliografía</b>	<b>8</b>

## 1. Introducción

Un autómata finito es una lista de cinco objetos: conjunto de estados, alfabeto de entrada, reglas para mover, estado inicial y estados aceptados. En el lenguaje matemático una lista de cinco elementos a menudo se llama una 5-tupla. Por lo tanto, definimos un autómata finito como una 5-tupla que consiste en estas cinco partes. Usamos algo llamado una función de transición, frecuentemente denotada  $\delta$ , para definir las reglas para mover. Si el autómata finito tiene una flecha desde un estado  $x$  a un estado  $y$  marcado con el símbolo de entrada 1, significa que, si el autómata está en el estado  $x$  cuando lee un 1, entonces se mueve al estado  $y$ . Podemos indicar lo mismo con la función de transición diciendo que  $\delta(x, 1) = y$ . Esta notación es una especie de escritura abreviada matemática. Juntándolo todo llegamos a la definición formal de autómatas finitos.

### Definición

Un **Autómata finito** es una 5-tupla  $(Q, \Sigma, \delta, q_0, F)$  donde

1.  $Q$  es un conjunto finito llamado **estados**,
2.  $\Sigma$  es un conjunto finito llamado **alfabeto**,
3.  $\delta : Q \times \Sigma \rightarrow Q$  es la **función de transición**,
4.  $q_0 \in Q$  es el **estado inicial**, y
5.  $F \subseteq Q$  es el **conjunto de estados aceptados**.

Se nos presenta el siguiente problema a resolver:

1. Emular la función de transición, dados un autómata finito y un string de entrada
2. Implementar en lenguaje C un algoritmo que de entrada reciba un autómata finito y un string de entrada y este realice la descripción instantánea
3. La salida en la descripción instantánea sea en lenguaje L<sup>A</sup>T<sub>E</sub>X

Teniendo en cuenta cada uno de los términos mencionados en el fragmento anterior, en este informe se aprecia una de las muchas formas para poder dar una solución a la problemática presentada anteriormente. Se explicará la lógica utilizada y su implementación en C, lenguaje utilizada para desarrollar el problema.

## 2. Funcionamiento

Para desarrollar la Emulación de la función de transición de un DFA, el usuario deberá ingresar por consola la cantidad de estados del DFA. Este dato será ingresado de la siguiente forma :

```
#user@user-desktop: ./laboratorio [X]
```

- Información Entregada:
  - Matriz de la Función de Transición
  - Cantidad de Números del String de Entrada
  - Salida en Formato latex del Recorrido de la Función de Transición
  - Resultado de la Función de Transición



## 3. Especificaciones

- El programa recibe un número, que corresponde a la cantidad total de estados del DFA.
- Se genera un número aleatorio, que determina la cantidad de estados finales.
- Se genera un arreglo para guardar los estados finales.
- Se genera la matriz de la función de Transición  $M_{n*3}$ .
- Se llena la matriz con números aleatorios.
- Se genera un número aleatorio, que es el largo del string de entrada.
- Se llena el string de entrada con números aleatorios entre 0 y 1.
- Se compara si el último elemento del string está contenido en alguno de los estados finales.

## 4. Algoritmos

---

**Algorithm 1** Estados Finales

---

**Input:**  $n$ , Cantidad de Estados Finales.**Output:** Conjunto de Estados Finales

```
1: for  $i \leftarrow 0$  to  $n$  do  
2:   finalstates[]  $\leftarrow$  rand(1,n)  
3: end for
```

---

---

**Algorithm 2** String de Entrada

---

**Input:**  $n$ , Largo del String de Entrada.**Output:** String de Entrada

```
1: for  $i \leftarrow 0$  to  $n$  do  
2:   string[i]  $\leftarrow$  rand(0,1)  
3: end for
```

---

---

**Algorithm 3** Verificación Lenguaje Aceptado

---

**Input:**  $n$ , Cant. Estados Finales -  $k$ [], Estados Finales -  $j$ , Último Dato del String.**Output:** String de Entrada

```
1: for  $i \leftarrow 0$  to  $n$  do  
2:   if  $j = k[i]$  then  
3:     return 1  
4:   end if  
5: end for
```

---

## 5. Ejemplo de Uso

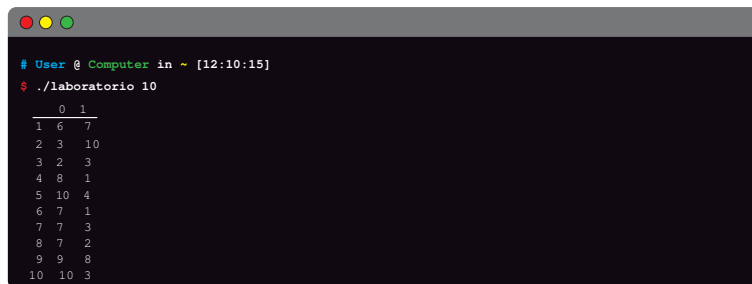
### 5.1. Función de Transición de un DFA

Queremos verificar si un lenguaje es aceptado por una Autómata de 10 Estados. El programa se ejecutará de la siguiente manera:



```
# User @ Computer in ~ [12:10:15]
$ ./laboratorio 10
```

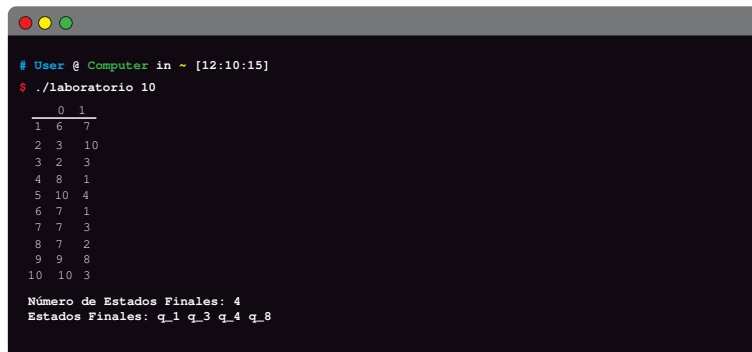
### 5.2. Matriz de la Función de Transición



```
# User @ Computer in ~ [12:10:15]
$ ./laboratorio 10

  0  1
1  6  7
2  3 10
3  2  3
4  8  1
5 10  4
6  7  1
7  7  3
8  7  2
9  9  8
10 10  3
```

### 5.3. Estados de Aceptación



```
# User @ Computer in ~ [12:10:15]
$ ./laboratorio 10

  0  1
1  6  7
2  3 10
3  2  3
4  8  1
5 10  4
6  7  1
7  7  3
8  7  2
9  9  8
10 10  3

Número de Estados Finales: 4
Estados Finales: q_1 q_3 q_4 q_8
```

## 5.4. String de Entrada

```
# User @ Computer in ~ [12:10:15]
$ ./laboratorio 10

  0 1
1 6 7
2 3 10
3 2 3
4 8 1
5 10 4
6 7 1
7 7 3
8 7 2
9 9 8
10 10 3

Número de Estados Finales: 4
Estados Finales: q_1 q_3 q_4 q_8

Cantidad de Números del String de Entrada: 11
String de Entrada: 10001110101
```

## 5.5. Verificación Lenguaje Aceptado

```
# User @ Computer in ~ [12:10:15]
$ ./laboratorio 10

  0 1
1 6 7
2 3 10
3 2 3
4 8 1
5 10 4
6 7 1
7 7 3
8 7 2
9 9 8
10 10 3

Número de Estados Finales: 4
Estados Finales: q_1 q_3 q_4 q_8

Cantidad de Números del String de Entrada: 11
String de Entrada: 10001110101

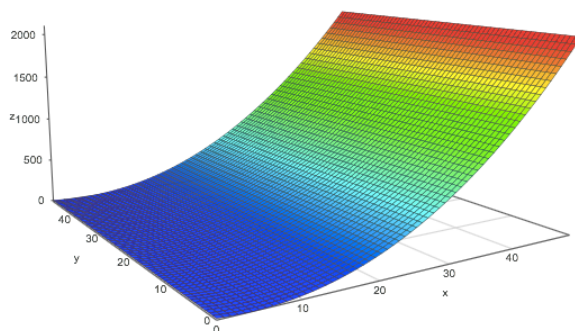
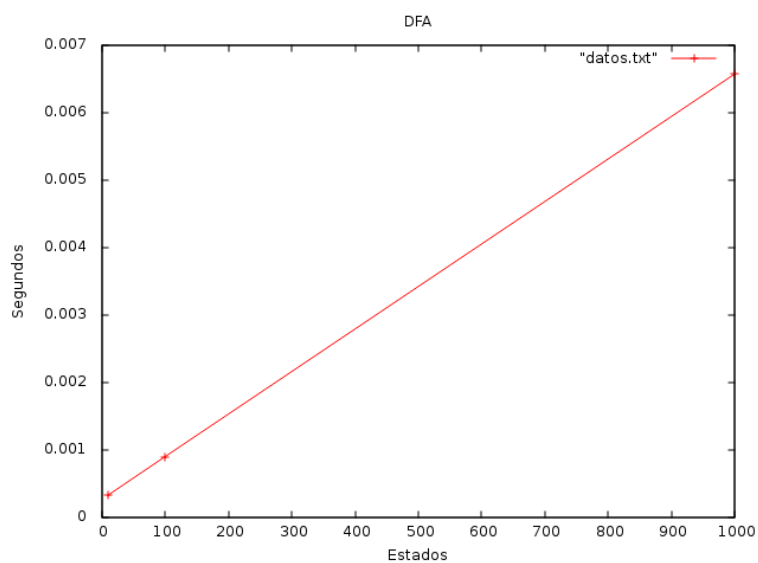
Formato Latex
$q_110001110101 \vdash 1q_70001110101 \vdash 10q_7001110101 \vdash ... $

Último Estado: q_3

STRING ACEPTADO
```

## 6. Curvas de Desempeño de Resultados

El programa tiene un desempeño Constante, como se muestra en los gráficos.





## 7. Conclusiones

Los autómatas finitos son buenos modelos para computadoras con una cantidad de memoria extremadamente limitada. ¿Qué puede hacer una computadora con una memoria tan pequeña? Muchas cosas útiles! De hecho, interactuamos con tales computadoras todo el tiempo, ya que se encuentran en el corazón de varios dispositivos electromecánicos.

## 8. Plataforma Computacional

El programa fue ejecutado en un computador con las siguientes características:

- Procesador: Intel(R) Core i3 2.40GHz
- Memoria RAM: 8 GB
- Sistema Operativo: Debian GNU/Linux 8 (jessie) 64-bit

## 9. Bibliografía

Michael Sipser. (2006). Introduction to the Theory of Computation. USA: Thomson Course Technology.