ILI-255: Informática Teórica

Tarea #1

“"It’s not my fault"”

Roberto Fuentes
201173037-2

10 de Abril 2017

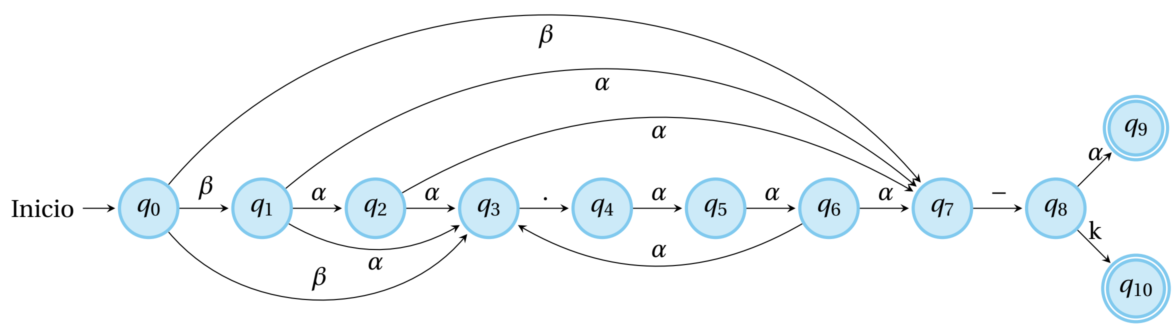
Desarrollo

1. Pregunta 1:

   a) Nos piden una expresión regular que permita identificar un RUT válido en el formato de chileno. Para esto, Luego, definiremos el alfabeto "α"para denotar los numeros de un digito (0,1,2,...,9) y definiremos el alfabeto β para denotar los numeros de un digito sin el 0 (1,2,...,9). Un RUT debe terminar con (α |k)". Sabiendo esto, un rut puede tener el siguiente formato: $número$ - ($número$ | k), $número$ $numero$ - ($número$ |k), $número$ $numero$ $numero$ - ($número$ | k) y finalmente cualquiera de estas 3 opciones seguido de (. $número$ $numero$ $numero$) seguido de ($número$ | k). Una expresión regular para el RUT chileno seria entonces:

   $$\textbf{\textit{RE}}: (\beta \mid \beta\alpha \mid \beta\alpha\alpha)(.\alpha\alpha\alpha)^* - (\alpha \mid \text{k})$$

   b) Un DFA que puede representar esta expresión regular es la siguiente:



2. Pregunta 2:

   Aplicando el algoritmo ϵ - closure a nuestro NFA nos queda lo siguiente:

   $\epsilon - closure\ \{a\} = \{a\}$
   $0: a, b, c, d, e$
   $1: e, d$
   $A = \{a\}$
   $B = \{a, b, c, d, e\}$
   $C = \{d, e\}$

B:

$\epsilon - closure\ \{a, b, c, d, e\} = \{a, b, c, d, e\}$
$0 : a, b, c, d, e$
$1 : b, d, e$
$D = \{b, e, d\}$

C:

$\epsilon - closure\ \{d, e\} = \{d, e\}$
$0 : e$
$1 : \emptyset$
$e = \{b, e, d\}$

D:

$\epsilon - closure\ \{b, e, d\} = \{b, e, d\}$
$0 : c, e$
$1 : e$
$G = \{c, e\}$

E:

$\epsilon - closure\ \{e\} = \{e\}$
$0 : \emptyset$
$1 : \emptyset$

F

$\epsilon - closure\ \{\emptyset\} = \{\emptyset\}$

G:

$\epsilon - closure\ \{c, e\} = \{c, e\}$
$0 : \emptyset$
$1 : b$
$H : \{b\}$

H:

$\epsilon - closure\ \{b\} = \{b\}$
$0 : c$
$1 : e$
$I : \{c\}$

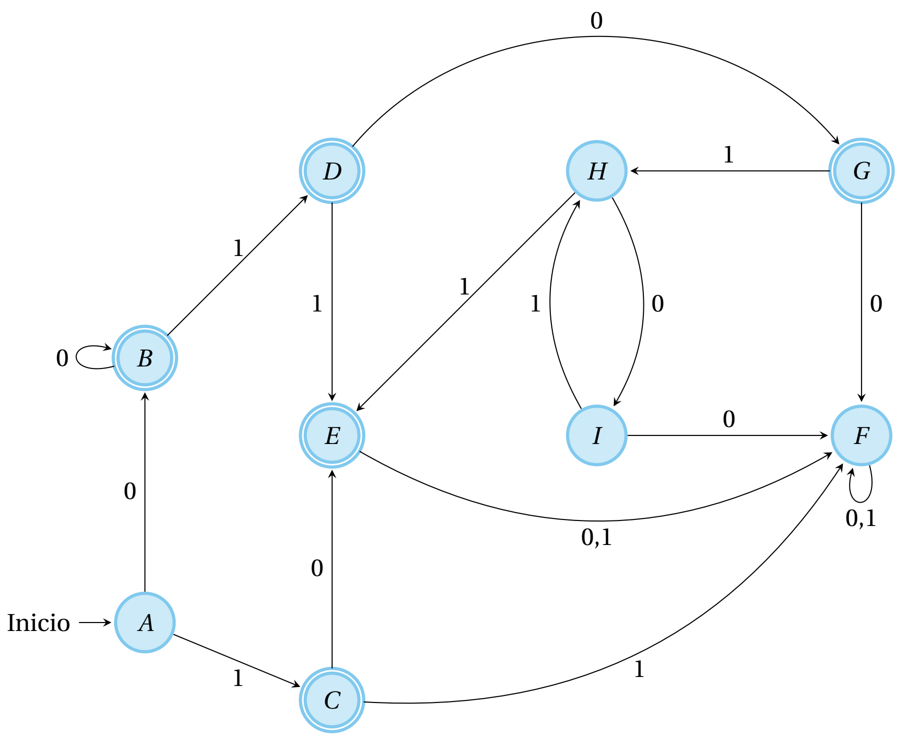
I:

$\epsilon - closure\ \{c\} = \{c\}$
$0 : \emptyset$
$1 : b$
$H : \{b\}$

Luego construimos la tabla de transiciones:

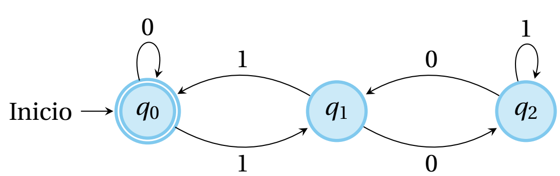
DFA	**NFA**	**0**	**1**
A	{a}	B	C
B	{a,b,c,d,e}	B	D
C	{d,e}	E	F
D	{d,e,b}	G	E
E	{e}	F	F
F	{∅}	F	F
G	{e,c}	F	H
H	{b}	I	E
I	{c}	F	H

Finalmente el DFA nos queda de la siguiente forma:



3. Pregunta 3:

Primero creamos un automata cuyo $output$ sean números divisibles por 3. El automata nos queda de la siguiente forma:



Este grafo nace de lo siguiente:

Consideramos que un numero puede ser escrito de la siguiente forma: $num = 3 * a + b$, donde b es el resto. Queremos que cada numero que acepte el automata tenga b = 0, por lo que armamos tres estados q_1,q_2 y q_3, siendo estos los restos 0,1,2. Analizamos entonces los arcos:

a) Al empezar en el estado q_0 y leer un 0, nos quedamos en el estado a_0, y el numero 0 es divisible por 3.

b) Estando en el estado q_0 y leyendo un 1, nos vamos al estado q_1. Esto lo hacemos ya que el número que se forma (1) en decimal nos da un resto de 1.

c) Cuando estamos en el estado q_1 y leemos un 0, nos movemos al 2. Esto ocurre ya que el número en binario generado (10) en decimal nos da un resto de 2.

d) Cuando estamos en el estado q_1 y leemos un 1, nos vamos al estado q_0. Esto ocurre ya que el binario que nos entrega el automara (11) en decimal nos da un resto de 0.

e) Cuando estamos en el estado q_2 y leemos un 0, nos vamos al estado q_1. Esto ocurre ya que el binario generado (100) en decimal nos entrega un resto de 1

f) Cuando estamos en el estado q_2 y leemos un 1, nos quedamos en el estado 2. Esto ocurre porque el binario generado (101) en decimal nos entrega un resto de 2

Por lo que la tabla de transiciones debe ser la siguiente:

Estado	0	1
q_0	0	1
q_1	2	0
q_2	1	2

Quedando el grafo resultante que se propuso anteriormente. El problema que tenemos con este DFA es que no incluye la condicion de que la cantidad de 0's y 1's sea par, ya que tanto en el nodo q_0 como el nodo q_1 se pueden repetir una cantidad impar de 0's y 1's. Para esto, tomemos el caso de q_0. Cambiamos la condicion de repetir muchas veces el 0, yendo con un 0 a otro estado q_3 y devolviendonos al mismo estado q_0. Asi, nos aseguramos de que la cantidad de 0 siempre sea multiplo de 2 (es decir par), y realizamos lo mismo con el nodo q_2. Haciendo esto nos queda: