Instituto Politécnico Nacional
Escuela Superior de Cómputo
Teoría Computacional
Práctica 5 : Autómata finito determinista.
Alumno: AbrMa
Profesor: Jorge Luis Rosas Trigueros
Realización: 08/03/2019
Entrega: 15/03/2019

1 Marco teórico.

Formalmente, un *autómata finito determinista* M es una colección de cinco elementos.

1. Un alfabeto de entrada Σ.
2. Una colección finita de estados Q.
3. Un estado inicial s.
4. Una colección F de estados finales o de aceptación.
5. Una función $\delta : Q \times \Sigma \rightarrow Q$ que determina el único estado siguiente para el par (q_i, σ) correspondiente al estado actual y la entrada.

Generalmente el término *autómata finito determinista* se abrevia como AFD. Usaremos $M = (Q, \Sigma, s, F, \delta)$ para indicar el conjunto de estados, el alfabeto, el estado inicial, el conjunto de estados finales, y la función asociada con el $AFDM$.

Por ejemplo, el AFD correspondiente al ejemplo anterior se representa mediante $M = (Q, \Sigma, s, F, \delta)$ donde

$$Q = \{q_0, q_1, q_2\}$$
$$\Sigma = \{a, b\}$$
$$s = q_0$$
$$F = \{q_0\}$$

δ se define en la siguiente tabla

δ	a	b
q_0	q_1	q_2
q_1	q_2	q_0
q_2	q_2	q_2

La característica principal de un AFD es que δ es una *función*. Por tanto, δ se debe definir para todos los pares (q_i, σ) de $Q \times \Sigma$. Eso significa que sea cual sea estado actual del carácter de la entrada, *siempre* hay un estado siguiente, y éste es único. Por tanto, para un par (q_i, σ) hay *uno y solo un valor* de la función (estado siguiente), $\delta(q_i, \sigma)$. En otras palabras el estado siguiente está determinado por la información que proporciona el par (q_i, σ).

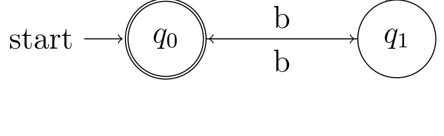
Se puede crear un diagrama de transición a partir de la definición de un AFD. Primero, creamos y etiquetamos un nodo para cada estado. Entonces, para cada celda q_j de la fila correspondiente al estado q_i trazamos una arista desde q_i a q_j , etiquetada con el carácter de entrada asociado a q_j. Finalmente, se marca el nodo s con una flecha , y se trazan unos círculos en todos los nodos de F para indicar cuáles son los estados de aceptación. Por tanto, el diagrama de transición para el AFD $M = (Q, \Sigma, s, F, \delta)$, donde

$$Q = \{q_0, q_1\}$$
$$\Sigma = \{a, b\}$$
$$s = q_0$$
$$F = \{q_0\}$$

y δ se representa mediante la tabla siguiente

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_0

se muestra en la siguiente figura



2 Desarrollo de la práctica.

Realizamos autómatas que verificaran lo siguiente:

1. Cadenas que tengan un número par de `a`.

   (a) Declaramos un lenguaje $\Sigma = \{a, b\}$
   `Sigma = ['a','b'] .`

   (b) Declaramos una colección finita de estados $Q = \{q_o, q_1\}$
   `Q = ['q0','q1'].`

   (c) Declaramos un estado inicial $s = q_0$
   `s = 'q0'.`

   (d) Declaramos un estado final $F = q_0$
   `F = ['q0'].`

   (e) Finalmente declaramos δ de la siguiente manera
   `delta = { ('q0','a'):'q1', ('q0','b'):'q0', ('q1','a'):'q0', ('q1','b'):'q1' } .`

   (f) Ingresamos dentro de una lista ejemplos
   `ejemplos = ['aaa','abab'].`

   (g) Definimos nuestra función para cambiar de estados

   - `def cambioEstado(q,s):`

   - `    global delta`
   - `    for c in w:`
   - `      estado = cambioEstado(estado,c)`
   - `    if estado in F:`
   - `      print(w + ' si está en el lenguaje')`
   - `    else:`
   - `      print(w + ' no está en el lenguaje')`

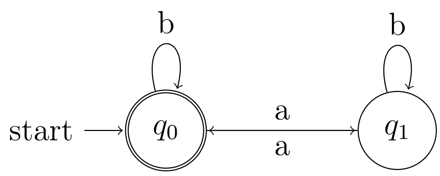


Figure 1: Problema 1

2. Cadenas que tengan un número impar de `x` y cadenas que tengan un número impar de `y` .

   (a) Declaramos un lenguaje $\Sigma = \{x, y\}$
   `Sigma = ['x','y']` .

   (b) Declaramos una colección finita de estados $Q = \{q_o, q_1, q_2, q_3\}$
   `Q = ['q0','q1','q2','q3']`.

   (c) Declaramos un estado inicial $s = q_0$
   `s = 'q0'`.

   (d) Declaramos un estado final $F = q_3$
   `F = ['q3']`.

   (e) Finalmente declaramos δ de la siguiente manera
   `delta = { ('q0','x'):'q2', ('q0','y'):'q1', ('q1','x'):'q3', ('q1','y'):'q0', ('q2','x'):'q0', ('q2','y'):'q3', ('q3','x'):'q1', ('q3','y'):'q2'}` .

   (f) Ingresamos dentro de una lista ejemplos
   `ejemplos = ['xy','xyxy', 'yx', 'xyxyxy','xyxyxyxy', 'xxxyyy']`.

   (g) Definimos nuestra función para cambiar de estados

   - `def cambioEstado(q,s):`

   - `    global delta`
   - `    for c in w:`
   - `      estado = cambioEstado(estado,c)`
   - `    if estado in F:`
   - `      print(w + ' si está en el lenguaje')`
   - `    else:`
   - `      print(w + ' no está en el lenguaje')`

3. Cadenas que sobre $\Sigma = \{0, 1, 2\}$ que no contengan la subcadena `'12'` .

   (a) Declaramos un lenguaje $\Sigma = \{0, 1, 2\}$
   `Sigma = ['0','1','2']` .

   (b) Declaramos una colección finita de estados $Q = \{q_o, q_1, q_2\}$
   `Q = ['q0','q1','q2']]`.

   (c) Declaramos un estado inicial $s = q_0$
   `s = 'q0'`.

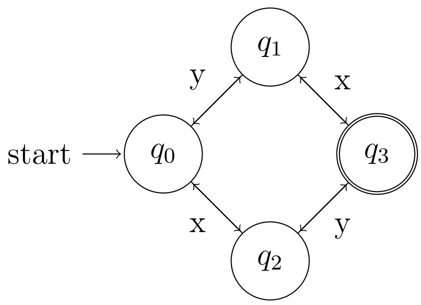


Figure 2: Problema 2

(d) Declaramos dos estados finales $F = \{q_0, q_1\}$
`F = ['q0','q1'].`

(e) Finalmente declaramos δ de la siguiente manera
`delta = ('q0','0'):'q0', ('q0','2'):'q0', ('q0','1'):'q1', ('q1','0'):'q0', ('q1','1'):'q1', ('q1','2'):'q2', ('q2','0'):'q2', ('q2','1'):'q2', ('q2','2'):'q2' .`

(f) Ingresamos dentro de una lista ejemplos
`ejemplos = ['', '12', '21', '111111112', '2222222222221', '012', '102', '21002120', '02000211', '12002200', '21021202', '12210011' '212'].`

(g) Definimos nuestra función para cambiar de estados

- `def cambioEstado(q,s):`

- `    global delta`
- `    for c in w:`
- `      estado = cambioEstado(estado,c)`
- `    if estado in F:`
- `      print(w + ' si está en el lenguaje')`
- `    else:`
- `      print(w + ' no está en el lenguaje')`

3 Material y equipo.

- Computadora.
- Sistema operativo: Ubuntu 18.04.2 LTS.

- Editor de texto: Vim.
- Lenguaje: Python 3.

4 Conclusiones y recomendaciones.

El diseño de autómatas en un principio parece difícil, sin embargo, con la realización de los ejercicios de la práctica entendí su relación con las *expresiones regulares*, y esto me facilito pasar de estas a los AFD. También es posible ver que estos cumplen con el *teorema fundamental de la programación*.

References

[1] Dean Kelley. (1995). Teoría de autómatas y lenguajes formales. Madrid: Pearson.

[2] Mark Lutz. (2013). Learning Python. Estados Unidos: O'Reilly.

[3] Python Software Foundation. (2019). Python 3.7.2 documentation. 20 Feb 2019, de Python Software Foundation Sitio web: https://docs.python.org/3/