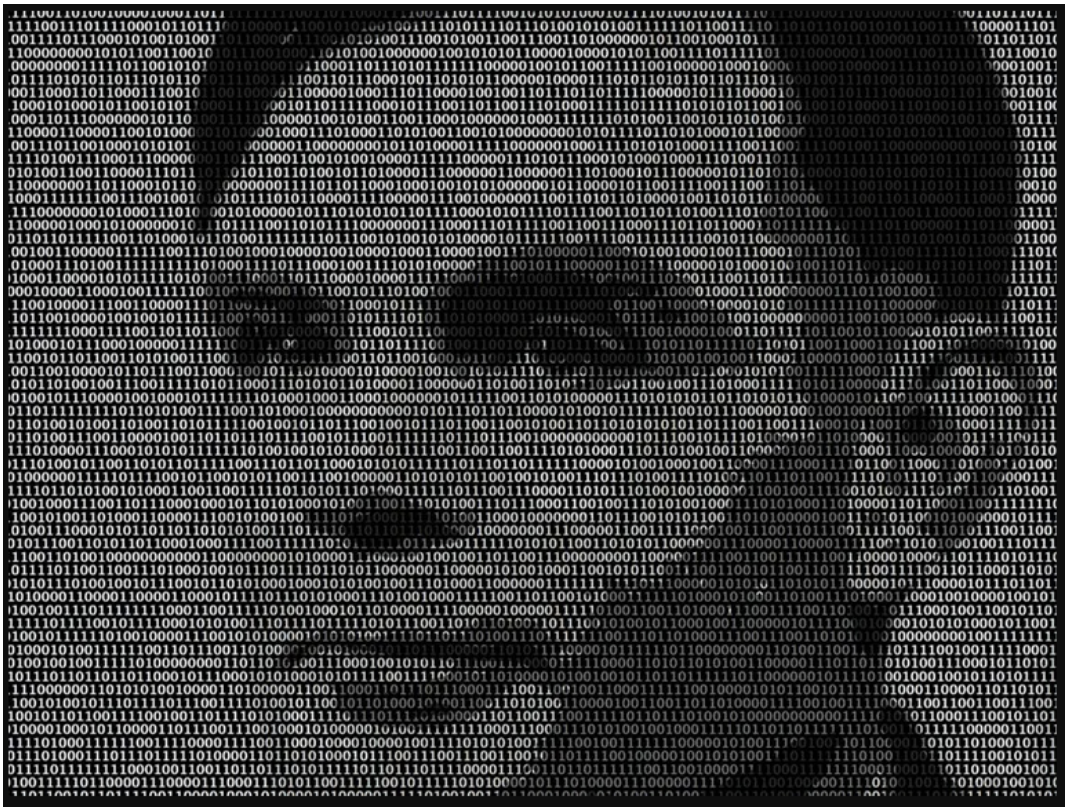
Universidad Austral de Chile
Conocimiento y Naturaleza

Ingeniería Civil en INFORMÁTICA UACh

Simulación de una máquina de Turing

Teoría de Autómatas - INFO139



Profesora: Maria Eliana de la Maza

Integrantes:

- Nicolás Robledo
- Francisco Aranda
- Rodrigo Almonacid

Fecha de entrega: 18/06/2021

Índice

Introducción **3**

Desarrollo **4**

Especificaciones del programa **10**

Nombre del programa 10

Lenguaje de implementación 10

Requerimientos mínimos del sistema 10

Indicaciones de cómo ejecutar el programa 10

Ejemplo de funcionamiento 11

Conclusiones **12**

Bibliografía **12**

1. Introducción

¿Qué es una Máquina de Turing?

Una máquina de Turing o MT es un dispositivo que manipula símbolos sobre una tira de cinta de acuerdo con una tabla de reglas.

En esta actividad, se nos propuso el desafío de diseñar una máquina de Turing o MT en un lenguaje de programación de nuestra elección, la cual teniendo de ingreso:

- Estado inicial
- Estado final
- Transiciones de la máquina de Turing
- Palabra de entrada

El programa deberá ser capaz de recibir una palabra y revisar si esta pertenece o no a la máquina de Turing descrita en las transiciones dadas anteriormente por el usuario.

Para simular una máquina eficiente deberemos elegir las estructuras de datos adecuadas que nos facilite el lenguaje, por ello decidimos usar **Python,** ya que es uno de los lenguajes de programación más amigables con el usuario y ya lo hemos practicado en nuestro primer año.

2. Desarrollo

En esta sección veremos el **qué se hizo** y el **cómo se hizo** a lo largo del desarrollo de esta actividad.
Entonces, **¿qué se hizo?** Desde un punto de vista general, dividimos el funcionamiento de la máquina de Turing en distintas subfunciones que fueron programadas en Python:

transitions(): se encarga de almacenar todas las transiciones ingresadas por el usuario.

createTape(): se encarga de crear una lista enlazada que simula una cinta que contiene blancos **'B'** y la palabra ingresada.

TM(): su objetivo es recorrer la cinta creada anteriormente con ***createTape()*** siguiendo las transiciones ingresadas por el usuario almacenadas en ***transitions(),*** esto significa desplazarse en la lista espacios a la izquierda o a la derecha según lo indiquen las transiciones. Su finalidad es imprimir en pantalla si la palabra ingresada pertenece al lenguaje especificado o no.

run(): se le pedirá al usuario ingresar un estado inicial, después un estado final, luego se invocará la función ***transitions()*** para así obtener las transiciones de la máquina de Turing y finalmente se le pedirá al usuario que ingrese una palabra para verificar si ésta pertenece al lenguaje especificado o no invocando a la función ***TM(),*** esta última función invocará la función ***createTape()*** para transformar la palabra ingresada en una lista y así simular una cinta.

¿Cómo se hizo? A continuación explicaremos el funcionamiento del programa desde un punto de vista más técnico:

transitions():

1. Despliega un mensaje en pantalla con algunas instrucciones de ingreso que se ingresarán.
2. Pedirá al usuario que ingrese cada parámetro de cada transición de la forma $d(q, x) = (p, Y, L)$, en donde:

   q: estado actual, denotado por un entero o un string a elección
   x: caracter o char actual en la cinta.
   p: estado al que cambiará, denotado por un entero o un string a elección.
   Y: se cambia el carácter **'x'** por **'Y'**. El símbolo blanco se denota con el carácter **'B'**.
   L: debe ingresar un carácter con la letra 'L' o 'I' que indicará la dirección a moverse en la cinta.
3. Cada transición se almacenará en estructuras de datos llamadas diccionarios que almacenan los parámetros ingresados por el usuario anteriormente.
4. Finalmente, se retorna una variable diccionario 'd' que contiene toda la información de las transiciones.

```
def transitions():
    #instrucciones
    print("\nProcederá a escribir transiciones de la forma d(q,x)=(p,Y,L) en donde:")
    print(" - q: estado actual")
    print(" - x: símbolo actual")
    print(" - p: se pasa de estado 'q' a estado 'p'")
    print(" - Y: se cambia el símbolo 'x' por 'Y'. El símbolo blanco se denota con la letra 'B'")
    print(" - L: se mueve el cabezal en la dirección 'L', en donde 'L' es 'I' o 'D' (izquierda o derecha)");

    #ingresa el número de transiciones
    while True:
        try:
            nTransitions = int(input('\nIngrese el número de transiciones: '))
            break
        except ValueError:
            print('Error. Ingreso no válido.')

    #inicializamos variables
    count = 1
    d={}

    for i in range(nTransitions):
        print('\n----- Transición',count,'-----')
        q = input('Ingrese q: ')
        x = input('Ingrese x: ')
        p = input('Ingrese p: ')
        Y = input('Ingrese Y: ')
        L = input('Ingrese L: ')
        print('d(',q,',',x,') = (',p,',',Y,',',L,')')

        d[(q,x)] = [p,Y,L]
        count = count+1

    #retorna un diccionario con las transiciones
    return d
```

fig.1: algoritmo de la función transitions()

createTape():

1. Acepta como parámetro de entrada un string con la palabra a revisar.
2. Almacena cada caracter del string en una lista.
3. Une la lista creada anteriormente con otra lista que contiene 100 caracteres ‘B’ a cada lado, para así simular una cinta con la palabra de entrada y blancos.
4. Retorna la lista con char creada anteriormente.

```
def createTape(word):
    #inicializamos variables
    tape = []
    blanks = []

    for i in word:
        tape.append(i)

    for i in range(100):
        blanks.append('B')

    tape = blanks + tape + blanks

    return tape
```

fig.2: algoritmo de la función createTape()

TM():

1. Acepta como parámetro de entrada: initial_state, final_state, d, word, en donde:
   initial_state: string con estado inicial ingresado por el usuario.
   final_state: string con estado final ingresado por el usuario.
   d: diccionario con las transiciones ingresadas por el usuario.
   word: string con la palabra ingresada por el usuario.
2. Se llama a la función ***createTape()*** vista anteriormente para transformar el string ‘word’ a una lista.
3. Utilizamos una variable llamada ‘key’ para almacenar el estado inicial y el primer caracter de la palabra word.
4. Accedemos a la tercera variable del diccionario en la posición ‘key’ para así obtener la dirección a la cual moverse.
5. Nos movemos un espacio a la izquierda o derecha en la lista ‘tape’ según se indica en el paso anterior.
6. Almacenamos la siguiente transición en la variable ‘next_key’.
7. Repetimos los pasos anteriores en un bucle while hasta que nos topemos con el estado final o nos encontremos con alguna dirección que no exista en los diccionarios creados por el usuario.

8. Si nos topamos con el estado final, entonces se imprime en pantalla un mensaje diciendo que la palabra SÍ pertenece al lenguaje especificado.
9. Si nos topamos con algún error, esto significa el encontrarse con alguna dirección que no existe, entonces se imprimirá en pantalla un mensaje diciendo que la palabra NO pertenece al lenguaje especificado.

```
def TM(initial_state, final_state, d, word):
    try:
        #inicializamos variables
        pos = 100;
        tape = createTape(word)
        key = (initial_state, tape[pos])

        #actualizamos la cinta
        tape[pos] = d[key][1]

        #nos movemos un espacio a la izquierda
        if(d[key][2] == 'I' or d[key][2] == 'i'):
            pos = pos-1;

        #nos movemos un espacio a la derecha
        if(d[key][2] == 'D' or d[key][2] == 'd'):
            pos = pos+1;

        next_key = (d[key][0],tape[pos])

        while d[next_key][0] != final_state:
            key = next_key

            #actualizamos la cinta
            tape[pos] = d[key][1]
            #print('cinta:', tape)

            #nos movemos un espacio a la izquierda
            if(d[key][2] == 'I' or d[key][2] == 'i'):
                pos = pos-1;

            #nos movemos un espacio a la derecha
            if(d[key][2] == 'D' or d[key][2] == 'd'):
                pos = pos+1;

            next_key = (d[key][0],tape[pos])

            if(d[next_key][0] == final_state):
                print('La palabra', word, 'SI pertenece al lenguaje especificado.')

    except KeyError:
        print('La palabra', word, 'NO pertenece al lenguaje especificado.')
```

fig.3: algoritmo de la función TM()

run():

1. Le pide al usuario un input con el estado inicial y lo almacena en initial_state.
2. Le pide al usuario un input con el estado final y lo almacena en final_state.
3. Luego invoca a la función transitions() y almacena las transiciones en un diccionario ‘d’.
4. Finalmente, le pide al usuario un input para almacenar la palabra que se va a revisar.
5. El programa le seguirá pidiendo palabras para revisar hasta que éste escriba ‘exit’.

```
def run():
    ########## PASO 1: INGRESAR ESTADO INICIAL ###################
    initial_state = input('Ingrese el estado inicial: ')

    ########## PASO 2: INGRESAR ESTADO FINAL ######################
    final_state = input('Ingrese el estado final: ')

    ########## PASO 3: INGRESAR TRANSICIONES ######################
    d = transitions()

    ########## PASO 4: INGRESAR PALABRAS ##########################
    while True:
        word = input("\nIngrese palabra ('exit' para salir): ")
        if word == 'exit':
            break
        TM(initial_state, final_state, d, word)

    print('\nPrograma finalizado.')
    exit()
```

fig.4: algoritmo de la función run()

Luego, se procedió a crear una interfaz gráfica de dicho programa para así facilitar el uso al usuario:

Hecho todo lo anterior mencionado nuestro programa era completamente funcional pero no tenía una interfaz gráfica, para ello tomamos la decisión de editar algunas funciones del código para que funcionara correctamente, algunas de los cambios que realizamos fueron:

- ***transitions():*** Anteriormente con un Input recibia los datos del usuario de las transiciones, para implementarlo en la Interfaz Grafica cambiamos la función para que solo recibiera los datos de la transición (q,x,p,Y,L) y los ingresara en el diccionario.

- ***createTape():*** Dado que cambiamos algunas variables para poder utilizarlas en la Interfaz, aquí tuvimos que agregarle una función que convirtiera la palabra de StringVar a String para poder convertirla en una lista y así poder trabajar con la cinta
- ***TM():*** Se dejó prácticamente igual a excepción de las "print" en los que se daba la información de si la palabra pertenece o no pertenece a la Máquina de Turing. Los reemplazamos por cuadros emergentes de texto.

La interfaz la diseñamos con la librería tkinter, con la cual diseñamos los cuadros de Texto, de ingreso de datos, los encabezados y los botones.

También utilizamos una función llamada show_frame() para avanzar de "pagina" en el programa, dichas paginas son:

- ***Frame1:*** Se solicita el ingreso del Estado Inicial, Estado Final y número de transiciones.

- ***Frame2:*** Se dan instrucciones de cómo se deben ingresar las transiciones para después solicitar su ingreso de la siguiente forma:
    - Ingrese q:
    - Ingrese x:
    - Ingrese p:
    - Ingrese Y:
    - Ingrese L:

- ***Frame3:*** Es solo una casilla en la que se solicita el ingreso de la palabra, se pueden ingresar varias palabras o ingresar una nueva MT.

3. Especificaciones del programa

3.1. Nombre del programa:

- Tarea_Automatas.exe (con GUI)
- Tarea MT.exe

3.2. Lenguaje de implementación: Python 3.8.5

3.3. Requerimientos mínimos del sistema:

- Procesador: Intel Atom® processor o Intel® Core™ i3 processor
- Disk space: 20 mb.
- Operating systems: Windows* 7 o versiones más recientes, macOS, y Linux
- Python* versiones: 2.7.X, 3.6.X

3.4. Indicaciones de cómo ejecutar el programa:

1. Abrir el ejecutable con nombre 'Tarea_Automatas.exe'.
2. Ingresar **estado inicial**, este puede ser un número o un string. Lo importante es ser consistente con el tipo de dato que se ingresará.
3. Ingresar **estado final**, al igual que el estado inicial, este puede ser un número o un string. Lo importante es ser consistente con el tipo de dato que se ingresará.
4. Ingresar el número de transiciones.
5. Ingresar cada parámetro de cada transición de forma **d(q,x)=(p,Y,L),** esto es:
    - **q:** estado actual.
    - **x:** símbolo actual.
    - **p:** se pasa de estado **'q'** a estado **'p'**.
    - **Y:** se cambia el símbolo **'x'** por **'Y'**. El símbolo blanco se denota con la letra **'B'**.
    - **L:** se mueve el cabezal en la dirección **'L'**, en donde **'L'** es **'I'** o **'D'** (izquierda o derecha).

3.5. Ejemplo de funcionamiento: $L = \{(abc)^{n} / n > 0\}$

- **Estado Inicial:** 0
- **Estado final:** 4
- **Número de transiciones:** 5
- **Transiciones ‘d’:**
  - d[('0','a')] = ['1','a','D']
  - d[('1','b')] = ['2','b','D']
  - d[('2','c')] = ['3','c','D']
  - d[('3','a')] = ['1','a','D']
  - d[('3','B')] = ['4','B','I']

```
Ingrese el estado inicial: 0
Ingrese el estado final: 4
Ingrese el número de transiciones: 5

----- Transición 1 -----
Ingrese q: 0
Ingrese x: a
Ingrese p: 1
Ingrese Y: a
Ingrese L: D
d( 0 , a ) = ( 1 , a , D )

----- Transición 2 -----
Ingrese q: 1
Ingrese x: b
Ingrese p: 2
Ingrese Y: b
Ingrese L: D
d( 1 , b ) = ( 2 , b , D )

----- Transición 3 -----
Ingrese q: 2
Ingrese x: c
Ingrese p: 3
Ingrese Y: c
Ingrese L: D
d( 2 , c ) = ( 3 , c , D )

----- Transición 4 -----
Ingrese q: 3
Ingrese x: a
Ingrese p: 1
Ingrese Y: a
Ingrese L: D
d( 3 , a ) = ( 1 , a , D )

----- Transición 5 -----
Ingrese q: 3
Ingrese x: B
Ingrese p: 4
Ingrese Y: B
Ingrese L: I
d( 3 , B ) = ( 4 , B , I )

Ingrese palabra ('exit' para salir): abc
La palabra abc SI pertenece al lenguaje especificado.

Ingrese palabra ('exit' para salir): 10101010
La palabra 10101010 NO pertenece al lenguaje especificado.

Ingrese palabra ('exit' para salir):
La palabra   NO pertenece al lenguaje especificado.

Ingrese palabra ('exit' para salir): exit

Programa finalizado.
```

fig. 5: ejemplo de máquina de Turing sin GUI

4. Conclusiones

Se pudo realizar de manera satisfactoria la simulación de una máquina de Turing simple, pero no sin tener algunas limitaciones, ya que por ejemplo, es responsabilidad del usuario el que el ingreso de datos sea consistente a todo momento, si ingresa como estado inicial ‘q0’, es necesario que siga utilizando la misma notación ‘q’ para que los datos tengan sentido.

5. Bibliografía

- Diccionarios: https://www.programiz.com/python-programming/dictionary
- Interfaz gráfica: https://www.youtube.com/watch?v=hTUJC8HsC2I
- https://es.wikipedia.org/wiki/M%C3%A1quina_de_Turing