Lenguajes de programación - T03: Máquina de Turing para la división binaria de dos números de 3 cifras

Jorge Aurelio Morales Manrique
C.C. 1010075711
jomorales@unal.edu.co

Universidad Nacional de Colombia
Marzo 23 de 2021

5. Manual de usuario

En la carpeta adjunta con el nombre "Código Fuente", se encuentra un proyecto realizado en Visual Studio, el cual es un IDE que provee programas para compilar y ejecutar aplicaciones basadas en C Sharp o dotNET. El proyecto se compone principalmente de dos archivos, **Program.cs** y **mt_description.txt**, en los cuales se encuentra el código en C Sharp y la función de transición de la MT respectivamente. Al ejecutar la aplicación se solicitan dos valores a y b para posteriormente calcular $\frac{a}{b}$. A continuación se lista la serie de pasos necesarios para ejecutar el programa (Windows):

1. Descargar la carpeta **mt_application**
2. Ingresar a la carpeta **bin**, posteriormente a **Debug** y finalmente a **netcoreapp3.1** en donde se podrá encontrar un archivo *exe*.
3. Abrir el archivo encontrado en el paso anterior
4. Ingresar los valores de a y b.

Una vez se ingresan los datos el programa comenzará la ejecución de la MT e imprimirá el resultado.

6. Manual técnico

La estructura interna del programa se divide en dos componentes principales los cuales se describen a continuación

Clase MT

- **Atributos.** Posición en la cinta, estado actual, conjunto de elementos de Σ, cinta, lista de estados de aceptación, función de trasición δ.

```
private int current_pos = 5;
private string current_state = "q0";
private HashSet<char> sigma;
private List<char> tape;
private List<string> final_states;
Dictionary<string, Dictionary<char, Tuple<string, char, char>>> delta;
```

Figura 1: Atributos MT.

- **Constructor.** Inicializa los atributos, teniendo en cuenta la información recibida por parámetro.

```
this.sigma = new HashSet<char>();
this.sigma.Add('0');
this.sigma.Add('1');
this.sigma.Add(' ');

this.tape = new List<char>();
for (int i = 0; i < 5; i++) {
    this.tape.Add(' ');
}
foreach (char bit in a) {
    this.tape.Add(bit);
}
this.tape.Add(' ');
foreach (char bit in b) {
    this.tape.Add(bit);
}
this.tape.Add(' ');

this.delta = new Dictionary<string, Dictionary<char, Tuple<string, char, char>>>();
for (int i = 0; i < n_states; i++) {
    string state = String.Concat("q", i);
    Dictionary<char, Tuple<string, char, char>> edge = new Dictionary<char, Tuple<string, char, char>>();
    this.delta.Add(state, edge);
}
foreach (Tuple<string, char, string, char, char> d_value in d_values) {
    Tuple<string, char, char> change = new Tuple<string, char, char>(d_value.Item3, d_value.Item4, d_value.Item5);
    this.delta[d_value.Item1][d_value.Item2] = change;
}

this.final_states = final_states;
```

Figura 2: Constructor MT.

- **Transición.** Evalua la función de transición y actualiza las variables correspondientes.

```
private void transition() {
    foreach (string state in this.final_states) {
        if (state.Equals(this.current_state)) {
            this.print_result();
            return;
        }
    }
    char character = this.tape[this.current_pos];
    Tuple<string, char, char> action = this.delta[this.current_state][character];
    this.current_state = action.Item1;
    this.tape[this.current_pos] = action.Item2;
    if (action.Item3.Equals('L')) {
        this.current_pos--;
    } else if (action.Item3.Equals('R')) {
        this.current_pos++;
    }
    transition();
}
```

Figura 3: Constructor MT.

Clase Program

- **Lectura de datos.** Obtiene del usuario los valores de a y b. Comprueba que b no sea 0, en cuyo caso termina la ejecución de la aplicación.

```
Console.WriteLine("Turing Machine for Binary Division of 3-Digit Numbers");
Console.WriteLine("Ingrese el valor de a:");
string a = Console.ReadLine();
Console.WriteLine("Ingrese el valor de b:");
string b = Console.ReadLine();
bool one = false;
foreach (char bit in b) {
    if (bit.Equals('1')) {
        one = true;
    }
}
if (!one) {
    Console.WriteLine("No se permite la división por cero.");
    System.Environment.Exit(1);
}
```

Figura 4: Lectura de datos.

- **Lectura de δ y ejecución MT.** Lee un archivo de texto con la información de la función de transición y ejecuta la máquina.

```
string[] lines = File.ReadAllLines("../../../mt_description.txt");
string[] n_s = lines[0].Split(' ');
int n_states = Int32.Parse(n_s[0]);
List<string> final_states = new List<string>();
foreach (string fs in lines[1].Split(' ')) {
    final_states.Add(fs);
}
List<Tuple<string, char, string, char, char>> d_values = new List<Tuple<string, char, string, char, char>>();
for (int i = 2; i < lines.Length; i++) {
    string[] info = lines[i].Split(' ');
    string from = info[0];
    char c1 = info[1].ToCharArray()[0];
    if (c1.Equals('#')) {
        c1 = ' ';
    }
    string to = info[2];
    char c2 = info[3].ToCharArray()[0];
    if (c2.Equals('#')) {
        c2 = ' ';
    }
    char move = info[4].ToCharArray()[0];
    Tuple<string, char, string, char, char> entry = new Tuple<string, char, string, char, char>(from, c1, to, c2, move);
    d_values.Add(entry);
}
MT mt = new MT(n_states, a, b, d_values, final_states);
mt.start();
Console.WriteLine("\nPress any key to close this window...");
Console.ReadKey();
```

Figura 5: Lectura de δ y ejecución MT.