Manual de Programación

Víctor D. Cruz González Marco A. Casas Moreno
H. Isaac Téllez Benítez César A. Valenzuela Rauda

Abril 2018

1 Introducción

En el presente documento se detalla la descripción sobre las diferentes clases Java utilizadas para el desarrollo del proyecto capaz de simular una máquina de Turing mediante su descripción, y regresar una cinta finita que muestre las operaciones realizadas por la máquina simulada.

2 Clases

2.1 FeedUTM

La clase *FeedUTM* está dada por el profesor para la ejecución de la simulación. Se utiliza para realizar lecturas de archivos de texto tanto de la descripción de la máquina de Turing, codificada ya sea en binario o en ASCII, así como de la cinta sobre la cuál se realiza la lectura/escritura/recorrido.

Para su operación, el método *main()* de la clase solicita, a través de la terminal, información al usuario sobre la ubicación del archivo que contiene la cinta con los datos de lectura de la máquina y la descripción de la máquina. De igual manera realiza una verificación para comprobar la validez de los datos entregados a la simulación.

Una vez que realiza correctamente las lecturas de los archivos deseados y los transforma en *String* sobre las que Java puede operar, convoca al método *NewTape()* de la clase *UTM* para realizar la transformación de la cinta y almacenarla en un archivo de texto.

2.2 State Table

La clase *State Table* ayuda a *UTM* a transformar la cadena `TT` que contiene la descripción de la máquina de Turing en binario-ASCII en la representación de una tabla de 2 dimensiones, es decir, una matriz compuesta por dos columnas, los datos correspondientes a una entrada `in = 0` y a una entrada `in = 1`, y varias filas, dependiendo del número de estados que el programa detecte en `TT`.

De esta manera, *UTM* podrá identificar cuáles son el siguiente estado y el siguiente caracter que debe escribir en la cinta dado una entrada binaria. *State Table* está diseñada para determinar la cantidad de estados y cuáles son estos al analizar `TT`.

2.2.1 Métodos

2.2.1.1 StateTable(String state)

El constructor de esta clase encuentra el fenotipo de una máquina de estados dado el genotipo, `state`, y lo convierte en una matriz llamada `this.state`, que es un tipo `String[ ]`, con los datos necesarios para visualizarlo como una tabla de 2 dimensiones. Para lo anterior, este método se apoya de otro método llamado `toMatrix()`.

2.2.1.2 getState()

El método regresa una cadena que representa una tabla con todos los estados que reconoció de `state` en los casos que `in = 0` y `in = 1`, con los siguientes atributos:

- **EA** — El número que representa el estado actual, dado que fue invocado por un estado pasado en una unidad de tiempo anterior
- **O** — El bit que debe escribirse en la MT dado que leyó una entrada `in`
- **M** — El movimiento que debe efectuarse al haber escrito el bit `0`, a la Derecha o a la Izquierda
- **SE** — El siguiente estado que debe observar la MT al finalizar con el proceso de lectura y escritura, dado un bit en cierta posición de la cinta

2.2.1.3 toMatrix(String bin)

Esta función convierte la cadena `state` en una tabla de 2 dimensiones, ya que esta cadena guarda cada estado y estos están conformados de 8 bits o 1 byte, los cuales contienen: en el primer bit está el tipo de movimiento que puede realizar una MT, Izquierda (`x = 0`) o Derecha (`x = 1`); el segundo bit contiene el bit que debe escribirse y los restantes son la codificación del siguiente estado en binario.

2.2.1.4 bin2Dec(String bin)

El objetivo de la función `bin2Dec` es convertir una cadena que representa un número en binario en su respectivo número decimal, a través de los métodos algebraicos que permiten transformar una base numérica en otra.

2.2.1.5 nextStep(char car, int state)

La función `nextStep` permite conocer cuáles serán los próximos estado, bit de escritura y movimiento de la cabeza, tal y como se indica en la teoría, puesto que la clase `State Table` usa `this.state` para acceder al *vector* ubicado en la fila `state`. No obstante, la clase deberá discernir entre una entrada binaria llamada `car` y no dar una información que no es requerida, por el momento; así, dependiendo de su valor, la función regresa un intervalo diferente de dicho vector o fila de la *tabla* `this.state`.

2.3 UTM

Esta clase contiene un único método, el cual es estático, llamado *NewTape*, el cual ejecuta la tarea de transformar la cinta de la máquina entregada a partir de la descripción de la máquina de Turing simulada, una posición inicial de la cabeza (pues la cinta no es infinita), así como un número máximo de transiciones para la cabeza dentro de la cinta, de manera tal que garanticemos el término de la función.

2.3.1 Métodos

2.3.1.1 NewTape

El método *NewTape* recibe como parámetros:

- **String TT** — La descripción en binario de la máquina de Turing a simular
- **String Tape** — La cinta sobre la que se realiza la lectura, escritura y movimientos de cabeza de la máquina de Turing a simular.
- **int N** — El número máximo de transiciones que podrá realizar la máquina de Turing antes de detenerse.
- **int P** — La posición inicial de dónde comenzará, en la cinta, la cabeza de la máquina de Turing.

Para realizar su tarea, el método *NewTape* se apoya en la clase *StateTable* y sus métodos; al instanciar un objeto de este tipo, podemos conocer qué hay que escribir en la cinta y cuál será el siguiente estado, pues ambos datos se obtienen mediante el método *nextStep()*. Así, al realizar la simulación, el algoritmo inicia el estado actual de la máquina como el estado "cero", solicita a la clase *StateTable* el valor a escribir y cuál es el siguiente estado. Para ello, si se recibe *I* como siguiente movimiento, se sabe que hay que modificar posición actual a su valor menos uno; por el otro lado, si se recibe *D* como siguiente movimiento, posición actual incrementa en una unidad. El manejo del recorrido dentro de la cinta se realiza de esta forma ya que se emplean el método *substring()* sobre la cadena de la cinta.

Por otra parte, para moverse al estado siguiente se transforma la cadena que se recibe del método *nextStep()* a *integer*, lo que permite usarlo como parámetro

de este mismo método para transitar entre estados. Al detectar que el valor que regresa *nextStep()* es "H", se reconoce que el siguiente estado es *Halt*, por lo que se detiene la ejecución de la simulación y se regresa conserva la cinta transformada, en forma de *String*.

Asimismo, si se supera el número máximo de transiciones de la cabeza, se detiene el recorrido y se registra la cinta transformada en forma de *String*. Los valor de retorno de la función se almacenan en un arreglo de dos espacios: el primero contiene el *String* con la cinta transformada, mientras que el segundo contiene un mensaje para desplegar en terminal que indica el número de transiciones realizadas, si se alcanzó o no el estado HALT, y la productividad de la máquina.