Instituto Tecnológico de Costa Rica

IC-2001 Estructuras de Datos, grupo 03

Bachillerato de Ingeniería en Computación

Prof. Mauricio Avilés

Proyecto Programado II - Máquina de Turing Genérica

Documentación del proyecto

Estudiantes: Gabriela Garro - 2014107501

Geovanni Villalobos - 2014116027

Daniel Ulate - 2014082275

Índice

Introducción

La máquina de Turing es un dispositivo hipotético que manipula símbolos en una tira de éstos, de acuerdo con una serie de reglas. A pesar de su simplicidad, una máquina de Turing puede ser adaptada para simular la lógica de cualquier algoritmo computacional, y es particularmente útil para explicar las funciones del CPU dentro de una computadora.

Fue inventada en 1936 por Alan Turing, el cual la llamó “a-machine”, diminutivo para “máquina automática”. La intención de ésta máquina no es precisamente ser una tecnología computacional práctica, sino un dispositivo hipotético que representa a una máquina que computa. Las máquinas de Turing simplemente ayudan a los científicos de la computación a entender los límites de la computación mecánica.

Una descripción informal podría sintetizarse en que la máquina de Turing modela matemáticamente una máquina que opera mecánicamente sobre una tira o cinta, la cual no es precisamente infinita, pero es lo suficientemente larga para poder abarcar los movimientos de la máquina hacia ambos extremos (es extensible arbitrariamente). En ésta tira hay símbolos, separados por celdas, los cuales la máquina puede leer y sobre escribir, uno a la vez, usando una “cabeza lectora”. La operación está determinada solamente por una serie de instrucciones elementales finitas, tales como “en el estado 34, si el caracter es 1, escriba 0; si el caracter es 0, escriba 1; cambie a estado 27”.

Una forma matemática de representar visualmente el funcionamiento de una máquina de Turing, es por medio de grafo dirigido y etiquetado; de la misma forma que se representa un autómata. En la computación, un grafo ayuda para verificar la complejidad de un modelo computacional. Éste modelo explica la configuración inicial de la máquina y qué pasos se pueden tomar para continuar la computación, hasta que eventualmente se para; implementando lógicamente así las múltiples interpretaciones abstractas que se le han dado a las instrucciones de una máquina de Turing.

Una máquina de Turing que es capaz de simular el funcionamiento de cualquier otra máquina de Turing es llamada Máquina de Turing Universal (UTM), y el desarrollo de una éstas, llamada en nuestro caso “Máquina de Turing Genérica”, es el objetivo del presente proyecto. Ésta es capaz de simular una máquina con instrucciones arbitrarias y una tira arbitraria, especificadas por el usuario.

Presentación y análisis del problema

¿Qué es lo que hay resolver?

1. Leer archivo de texto.
    a. Leer archivo línea por línea.
    b. Validar cada línea y separar sus contenidos.
    c. Guardar los contenidos en una clase.
2. Dibujar en pantalla el grafo que representa a la máquina.
    a. Marcar de una forma distinta al estado actual.
3. Crear una clase grafo que maneje el funcionamiento de la máquina.
    a. Debe de ser dirigida y etiquetada.
4. Manejo de la tira.
    a. Crear una lista.
    b. Validar los caracteres del string entrados por el usuario con los símbolos de máquina.
    c. Dibujar en pantalla los contenidos de la tira y la posición actual de la cabeza.
5. Ejecutar un paso de la máquina.
    a. Obtener símbolo actual.
    b. Cambiar el estado actual de la máquina. Actualizar el estado en pantalla si éste ha cambiado.
    c. Escribir un símbolo en la tira.
    d. Mover la cabeza.

¿Cómo se va a resolver el problema?

1. Crear un programa en consola que pida el nombre del archivo de texto desde el cual se va a cargar la definición formal de la máquina.
2. Crear una clase que reciba el nombre de un archivo de texto y lo lea línea por línea. Dependiendo del primer caracter de la línea, asigna los valores enlistados en el resto de ésta dentro de un objeto contenedor de la definición formal.
3. Crear una clase grafo que representa lógicamente el funcionamiento de la máquina y maneje las transiciones.
4. Crear una clase que a partir del número de estados y las relaciones que hay entre ellos, dibuje en pantalla un grafo que represente el funcionamiento de la máquina.
5. Crear una clase que represente a la tira de la máquina de Turing y que ésta se dibuje en pantalla.
6. Ejecutar un paso de la máquina:
    a. Obtener el símbolo actual.
    b. Cambiar el estado actual de la máquina. Actualizar el estado en pantalla si éste ha cambiado.
    c. Escribir el símbolo indicado por la transición en donde está ubicada la “cabeza lectora” de la tira.
    d. Mover la cabeza.

Análisis crítico de la implementación.

El proyecto fue dividido en tres partes fundamentales, las cuales se detallan a continuación:

1. Las clases que se encargan de cargar desde el archivo de texto la definición formal de la clase y de almacenarla.
2. Las clases asociadas al TDA de grafo, las cuales indican lógicamente las relaciones entre cada estado de la máquina.
3. La representación en pantalla del grafo y de la tira y cómo éstos cambian con cada paso de la máquina.

La última parte del proyecto consistió en agrupar el funcionamiento de éstas tres partes para crear un simulador de máquina de Turing funcional.

En el caso de las clases que cargan los datos de la máquina desde el archivo de texto, se implementó en primera instancia la clase LectorDeArchivos, la cual toma el nombre de un archivo, lo abre y lo analiza línea por línea. De cada línea, analiza el primer caracter, y a partir de éste identifica qué característica de la máquina la línea en cuestión describe. Delega, de ésta forma, a otros de sus métodos, el manejo del string. Ésto incluye separarlo por sus comas y guardar cada “elemento” en los atributos de la clase VirtualTuringMachine, la cual es creada dentro de LectorDeArchivos. A la hora de cargar los datos dentro de la VTM se hacen las validaciones (que los estados de las transiciones existan, por ejemplo).

La clase LectorDeArchivos es capaz de devolver la instancia de VirtualTuringMachine que ha creado.

El objetivo de la clase VirtualTuringMachine es actuar de contenedor de la definición formal de la máquina de Turing con la que se está trabajando, y sirve de medio de comunicación de datos entre la lectura de los archivos y el grafo, además de servir de medio de fácil acceso a la lista de características de la máquina. Se compone de varias listas de Strings que contienen los estados, los símbolos de entrada, los símbolos de máquina, las transiciones (denotadas con la clase Transición), estados de rechazo y de aceptación. Además, contiene dos atributos que indican el caracter en blanco y el estado inicial respectivamente. De forma análoga, la clase Transición almacena el estado de origen, el caracter leído, el estado de destino, el caracter escrito y el movimiento de la cabeza que se debe de hacer.

VirtualTuringMachine, se encarga de llevar a cabo cada paso de la máquina de Turing durante la ejecución. De acuerdo a el estado actual de la máquina y el símbolo leído, la máquina pasa a otro estado, escribe lo que indica la transición y se mueve en la tira. O termina la ejecución si es necesario.

En el caso del grafo, se depende de dos clases: Graph y Edge. Graph es la clase

que se encarga de llevar los datos y métodos del grafo como tal. Edge es una clase creada para el manejo de las transiciones de un vértice del grafo al siguiente. Graph guarda los vértices en un arreglo de string, donde cada elemento es el nombre de un vértice. Se creó un método que, al recibir un string, retorna el índice del arreglo de nombres en el que dicho string se encuentra.

El grafo se implementó mediante la utilización de una lista de adyacencia. Esta corresponde a un arreglo de *n* Listas Enlazadas con Edges como elementos; y *n* siendo la cantidad de vértices con los que cuenta el grafo. El grafo se instancia a partir de una referencia a la VirtualTuringMachine, ya que obtiene todos los datos para crear sus Edges de las listas que contienen los diferentes elementos lógicos de la máquina de Turing.

La clase Edge tiene varios atributos acerca de la transición: carácter leído, carácter escrito, nombre del estado de destino y dirección en que se mueve la cabeza lectora de la VTM.

La clase que a partir de datos específicos del grafo, logra dibujar en pantalla, fue brindada por el profesor. Los cambios que se realizaron sobre ésta clase fueron:

1. La ventana para dibujar se inicializa en el programa principal, en vez de dentro de la clase Graficador. EL graficador asume que dicha ventana ya se inició y lo que hace es limpiarla para dibujar la máquina.
2. Se agregó un método que se encarga de dibujar la tira en pantalla. A este método solamente se le pasa un string que es el que corresponde a la tira y el solo dibuja dicho string.

En la parte de la tira, se implementó una lista enlazada para su simulación, con el objetivo de ahorrar memoria. Para lograrlo, se utilizó una lista simplemente enlazada, la cual, en comparación con un arreglo común, es de tamaño variable, y en comparación con una lista doblemente enlazada, ahorra el espacio del puntero al nodo anterior.

La tira se divide en dos clases; la lista enlazada (LinkedList) y la clase Tira propiamente, donde se incluyen todos los métodos para su correcto funcionamiento. La tira funciona de manera simple; con un string que se recibe en consola por parte del usuario. Se muestra en la parte inferior de la pantalla, debajo del grafo. Se marca la posición actual de la cabeza lectora con corchetes cuadrados en el símbolo actual.

Conclusiones

1. Se utilizaron diferentes clases con labores individuales para la implementación de la máquina.
2. La clase LectorDeArchivos se encarga de realizar la lectura del archivo de texto con la definición, línea a línea, de la máquina de Turing.
3. LectorDeArchivos carga los datos de la máquina en una instancia de tipo VirtualTuringMachine.
4. VirtualTuringMachine es la clase encargada de la contención lógica de los elementos que definen la máquina de Turing.
5. VirtualTuringMachine lleva a cabo la ejecución de cada paso de la máquina durante la ejecución.
6. La VirtualTuringMachine determina si la transición se lleva a cabo o ya se detiene el funcionamiento, y se lo indica al programa principal.
7. La clase Graph lleva el control de las transiciones de un estado a otro mediante una lista de adyacencia de instancias de la clase Edge, que guardan los datos de cada transición.
8. El grafo de tipo Graph es instanciado a partir de los datos contenidos dentro de la VirtualTuringMachine.
9. La clase TIra maneja la tira que se le inserta a la máquina mediante una Lista Enlazada donde cada elemento es uno de los caracteres de la tira.
10. El graficador es la clase que se encarga de dibujar la máquina de turing y la tira en pantalla, utilizando la librería WinBGim.

Recomendaciones

1. Encapsular el funcionamiento de cada sección del proyecto en orden de facilitar los cambios posteriores que se tengan que hacer en el diseño. En el caso de las clases que dibujan en pantalla, se debe de encapsular su funcionamiento lo más posible, si es que se necesita hacer una misma acción en la pantalla más de una vez (por ejemplo, pintar todo de negro y volver a dibujar).
2. Se pudo haber implementado que la clase Lector de Archivos cargue los datos de la máquina de Turing directamente del grafo, para que no hayan algunos datos repetidos entre las clases VirtualTuringMachine y Graph.
3. Implementar la lista de símbolos que representa a la lista como un lista doblemente enlazada, en orden de que la navegación hacia adelante y hacia atrás en la lista sea más rápida.
4. Agregar una funcionalidad de poder cargar la definición de otra máquina de Turing, para no tener la necesidad de cerrar el programa para probar otra.
5. Agregar una funcionalidad de poder probar con otra tira la máquina, sin tener la necesidad de cerrar el programa para lograrlo.
6. Validar si el archivo de texto ingresado a la máquina existe.
7. Validar si el string con el nombre del archivo de texto ingresado por el usuario contiene la extensión; porque el método leerArchivo(string) de la clase LectorDeArchivos necesita que el string no contenga la extensión.
8. Reemplazar el ingreso del nombre del archivo en consola por el usuario por un ingreso por medio de una ventana del navegador de archivos del sistema operativo.
9. Validar y notificar al usuario si existe alguna anomalía con la definición de máquina de Turing que intenta cargar.
10. Utilizar una biblioteca gráfica más avanzada y con más funciones que WinBGIM, en orden de facilitar la representación del grafo y la tira en pantalla.
11. Se recomienda modificar la clase Graficador para que pueda dibujar cuando más de una transición que van en la misma dirección a partir del mismo estado. Esto podría hacerse utilizando un estilo Bucket para almacenar las diferentes transiciones que tienen el mismo origen y destino.

Referencias

1. Byrnes, J. (1994). K-graph machines: generalizing Turing's machines and arguments.
2. C++ Reference. string::c_str. Recuperado el 08 de Noviembre del 2014 de: http://www.cplusplus.com/reference/string/string/c_str/
3. C++ Reference. Input/Output with files. Recuperado el 08 de Noviembre del 2014 de: http://www.cplusplus.com/doc/tutorial/files/
4. C++ Reference. string::at. Recuperado el 08 de Noviembre del 2014 de: http://www.cplusplus.com/reference/string/string/at/
5. C++ Reference. string::erase. Recuperado el 08 de Noviembre del 2014 de: http://www.cplusplus.com/reference/string/string/erase/
6. Estructuras de datos: listas enlazadas, pilas y colas. recuperado el 05 de noviembre del 2014 de: http://www.calcifer.org/documentos/librognome/glib-lists-queues.html
7. How to include “graphics.h” in code blocks. Recuperado el 10 de noviembre del 2014 de: http://rekursed.blogspot.com/2012/03/how-to-include-graphicsh-in-codeblocks.html
8. Problems with graphics.h and winbgim.h; redefinition of 'int right'. Recuperado el 9 de noviembre del 2014 de: http://forums.devshed.com/beginner-programming-16/graphics-winbgim-redefinition-int-842275.html