Universidad Nacional Autónoma de México
Licenciatura en Ciencias de la Computación, Facultad de Ciencias
Autómatas y Lenguajes Formales 2022-1

Noé S. Hernández
Alan E. Arteaga

Práctica 2
Fecha de entrega: domingo 23 de enero.

1 Objetivo

El alumno pondrá en práctica sus conocimientos acerca de las máquinas de Turing, programando un simulador que permita leer y simular descripciones de dichas máquinas.

2 Introducción

En el material del curso se ha presentado ya el modelo de cómputo conocido como *"máquina de Turing"*; este modelo es el más expresivo de los que se han visto a lo largo del curso.

Para esta práctica, se deberá idear la implementación de un *algoritmo universal* que leerá la descripción de una máquina de Turing y la ejecutará.

La práctica se dividirá en dos partes principales:

- Idear y concretar la implementación del *algoritmo universal*.
- Solucionar un problema mediante una descripción de una Máquina de Turing, la cual correrán sobre su implementación del *algoritmo universal*.

3 Especificación

3.1 El programa

- **Entrada:** El programa recibirá dos cadenas por la entrada estándar:
  - El nombre de un archivo con la descripción de una Máquina de Turing M a simular.
  - Una cadena w que se encuentra en la cinta al iniciar la ejecución de M
- **Salida:** El programa deberá imprimir las configuraciones correspondientes a la ejecución de M sobre w, además, el programa deberá mostrar un texto indicando si w es aceptada o rechazada, en caso de que M se detenga (ten en cuenta que la máquina M puede no detenerse, y de hecho es un comportamiento necesario de poder simular).

3.2 El archivo de entrada

El archivo con la especificación de la máquina M deberá ser un archivo JSON con los siete elementos de la tupla $M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, F)$.

Un archivo JSON cuenta con dos estructuras: objetos y arreglos. Los objetos consisten en una serie de parejas "**llave:valor**" separadas por una coma y rodeadas por llaves.

Los arreglos, por su parte, consisten de corchetes con "**valores**" separados por una coma. Estos elementos pueden ser anidados de manera que los elementos "**valor**" pueden representar de nuevo objetos y arreglos.

Hay bibliotecas para trabajar con archivos JSON en distintos lenguajes de programación. Algunos ejemplos son:

- Para Go: Mediante el paquete "`json`" (incluido ya en varias instalaciones de Go).
- Para Python: Incluido en la biblioteca estándar (mediante `import json`).
- Para Java: Mediante "json-simple" (https://code.google.com/archive/p/json-simple/), ente muchos otros.

Información adicional sobre los archivos JSON y otras bibliotecas para manejar dichos archivos se encuentran en https://www.json.org/json-en.html

4 Ejemplo

Supongamos que se quiere simular una máquina de Turing M definida como

$$M = (\{q_0, q_1, q_f\}, \{0,1\}, \{0,1,\sqcup\}, \delta, q_0, \sqcup, \{q_f\})$$

cuya función de transición está dada por:

$$\delta(q_0, 0) = (q_1, 1, R); \quad \delta(q_1, 1) = (q_0, 0, R); \quad \delta(q_1, \sqcup) = (q_f, \sqcup, R).$$

El archivo `M.json` con la descripción de la máquina M correspondería al siguiente:

```
{
    "Estados" :["q0", "q1", "qf"],
    "Entrada" :[ "0", "1" ],
    "Cinta" :[ "0" ,"1", "_"],
    "Inicial" :"q0",
    "Blanco" :"_",
    "Finales" :["qf"],
    "Transiciones" :[
        ["q0", "0" ,"q1", "1" ,"R"],
        ["q1", "1" ,"q0", "0" ,"R"],
        ["q1", "_", "qf", "_", "R"]
    ]
}
```

El programa de este ejemplo está escrito en Go (ustedes pueden usar el lenguaje de su preferencia), y fue compilado a un binario. Su ejecución sería la siguiente:

```
$ ./bin/main
Archivo con la descripcion de la MT : M.json
Inserte la cadena de entrada : 010
```

Posterior a ello, imprimiría la salida:

```
Las configuraciones son :
|q0|010
1|q1|10
10|q0|0
101|q1|_
101_|qf|_
La cadena fue aceptada
```

Como se puede apreciar en el ejemplo, mediante barras verticales separamos al estado actual del resto de la cadena en la cinta. Otras formas de separar el estado actual de la cadena son aceptables, en el README indiquen la forma en la que realizan dicha separación.

5 Consideraciones

Algunas consideraciones **importantes** para la implementación del programa y del manejo de los archivos JSON son las siguientes:

- Tanto el formato del archivo JSON como las palabras utilizadas para las llaves ("Estados", "Entrada", "Cinta", "Inicial", "Blanco", "Finales" "Transiciones") deben de ser iguales a los utilizados en el ejemplo.
- Todos los elementos (estados, símbolos de la entrada y símbolos de la cinta), deben de estar definidos como cadenas y no como otro tipo de datos de JSON.
- Es recomendable (mas no obligatorio) pausar la ejecución una breve cantidad de tiempo entre cada impresión de una configuración para evitar la saturación de la terminal en caso de que la máquina se cicle, además de que les puede ayudar a depurar su programa.
- La máquina de Turing universal que implementarán debe ser capaz de recibir en su especificación los movimientos de la cabeza lectora izquierdo ("L") y derecho ("R"), además un movimiento neutro ("N") en el cual la cabeza lectora se mantendrá en la misma posición después de una transición. Pueden usar tal movimiento si lo desean en la solución de ejercicios.

6 Desarrollo

1. (8 pts.) Programar el algoritmo universal que lea la descripción de una máquina de Turing en JSON, reciba la cadena de inicio en la cinta e imprima las configuraciones que la máquina produce para dicha entrada.
2. (2 pts.) Diseñe una máquina de Turing M que acepte el lenguaje de las cadenas $w \in \{0,1,2\}^*$ de la forma $0^n1^n2^n$ tal que $n \geq 1$.

   Algunos ejemplos de posibles cadenas de entrada son:

   - $000111222 \in L(M)$.
   - $00011222 \notin L(M)$.
   - $00001111 \notin L(M)$.
   - $012 \in L(M)$.
   - $\varepsilon \notin L(M)$.

7 Extra

Para obtener hasta puntos extra realice lo que se pide en los dos siguientes puntos:

3. (2 pts.) Agregue el manejo de dos pistas a su implementación del algoritmo universal obtenida en el punto 1 de la sección anterior.
4. (1 pt.) Ocupando esas dos pistas proporcione una máquina de Turing para

$$\{xx \mid x \in \{0,1\}^*\}.$$

   Para este fin, en el archivo JSON se podrán definir los símbolos en Σ y Γ como arreglos, aunque su programa seguirá recibiendo como segundo argumento una secuencia w de caracteres a los que dará el formato requerido para su implementación particular de múltiples pistas; sin embargo, los arreglos, representando símbolos, sí pueden estar presentes al imprimir las configuraciones.