SIMULACIÓN SOFTWARE: MÁQUINA DE TURING - INFORME FINAL

TEORIA DE AUTOMATAS

Integrantes: Leandro Calogueraea

Diego Rojas

Fabián Sánchez

Docente: María Eliana de la Maza

VALDIVIA – CHILE *15/06/2017*

INTRODUCCIÓN

Como apoyo práctico a la asignatura "Teoría de autómatas", se nos solicita confeccionar una aplicación informática capaz de replicar el funcionamiento de una *Máquina de Turing* (MT). Para el desarrollo de esta aplicación se da la libertad de escoger el lenguaje que estimemos pertinente según un previo estudio de sus fortalezas y limitantes, las cuales serán profundizadas en el transcurso del documento.

El software deberá ser capaz de construir internamente una estructura de MT, donde el usuario ingresará las transiciones, estados, símbolos y alfabeto que la determine como tal, así como ser capaz de evaluar una palabra y determinar si esta pertenece o no al lenguaje que describe nuestra máquina.

ÍNDICE:

ALAN TURING Y SU MÁQUINA

ALAN MATHISON TURING (Londres, 1912- Wilmslow, Reino Unido, 1954) es considerado uno de los padres de la computación, además de contribuir decisivamente en campos como la informática teórica y la criptografía.

Entre sus más destacables hitos científicos encontramos: la función calculable, la máquina de Turing, el pre-desarrollo de la computadora Colossus (Ordenador diseñado para romper el código Enigma en la segunda guerra mundial), la desencriptadora Bombe, la prueba sobre inteligencia artificial, entre otros aportes conceptuales y técnicos que fueron fundamentales para el desarrollo de la ciencia.

MÁQUINA DE TURING: La máquina de Turing (MT) es un dispositivo de reconocimiento de lenguaje. Es más general que cualquier autómata finito y que cualquier autómata de pila, debido a que ellas pueden reconocer tanto lenguajes regulares, como lenguajes de libre contexto.

La máquina de Turing tiene un control finito de estados, un Cabezal lector y una cinta donde eventualmente irá la palabra de entrada. La cinta es de longitud infinita hacia la derecha, llenándose de caracteres "blancos" y de longitud finita hacia la izquierda, teniendo así un estado inicial. El cabezal cumple la función de lectura y escritura, por lo que la cinta puede ser modificada en su ejecución. Este cabezal es fijo, no así la cinta que tiene capacidad de movimiento bidireccional, por lo que puede pasar varias veces por un mismo carácter.

Esta máquina opera en torno a un determinado alfabeto más un identificador comúnmente llamado "blanco", un conjunto de estados finitos y un conjunto de transiciones entre dichos estados. Su funcionamiento se basa en una función de transición, que recibe un estado inicial y una cadena de caracteres (la cinta, la cual es finita por la izquierda) pertenecientes al alfabeto de entrada. Luego va leyendo una celda de la cinta, borrando el símbolo, escribir el nuevo símbolo perteneciente al alfabeto de salida y finalmente avanza a la izquierda o a la derecha (solo una celda a la vez), repitiendo esto según se indique en la función de transición, para finalmente detenerse en un estado final o de aceptación, representando así la salida.

DESARROLLO

Para llevar a cabo esta tarea escogimos el lenguaje de programación computacional Python, basándonos en la versatilidad de este lenguaje del cual podemos rescatar 2 grandes razones que avalan nuestra elección:

LENGUAJE INTERPRETADO:

Al ser Python un lenguaje interpretado, facilita y optimiza de manera sustancial la programación de alguna aplicación, ya que es capaz de ejecutar instrucciones por parte sin necesidad de que el programa esté completo. Esta cualidad permite ir verificando la correcta implementación de funciones en el momento y así sortear errores rápidamente a diferencia de otros lenguajes que requieren compilar la totalidad de la aplicación.

FÁCIL DE USAR:

Este lenguaje posee la particularidad de ser absolutamente orientado a objetos. Esto hace que el uso de funciones y métodos sean fáciles de implementar y heredar.

También utilizamos el control de versiones Git para trabajar a distancia de forma segura y llevar un recuento de los cambios hechos en cada momento a nuestro programa y usar un repositorio único.

Basándonos en las fortalezas de Python optamos por definir las siguientes **estructuras de datos**:

TRANSICIONES:

- **(Variable tipo String de transiciones)** Cada transición será una 5-tupla ingresada por el usuario, y constará de: ="ESTADO_ACTUAL,SIMBOLO_ACTUAL_DEL_CABEZAL,ESTADO_SIGUIENTE,SIMBOLO_A_ESCRIBIR,DIRECCION_DE_AVANCE".

  Separando cada elemento de esta por ",” y cada tupla del String separada por ";”.

- **(Arreglo Bidimensional de transiciones:[1..m][1..n] de tipo String)** Cada elemento del arreglo será una transición obtenida del String de transiciones, usando solo una 3-tupla para albergar la transición, separadas por comas.

- **[m]**: tipo entero, almacena la cantidad de estados.

- **[n]**: tipo entero, almacena la cantidad de letras del alfabeto, incluido el "blanco" (Ambos obtenidos a partir del análisis de las 5 tuplas ingresadas).

Análogamente para el arreglo bidimensional de transiciones, se usará un:

- **Arreglo unidimensional** de estados:[1..m] de tipo String.
- **Arreglo unidimensional** de letras del alfabeto:[1..n] tipo String (Ambos obtenidos de las transiciones ingresadas por el usuario).

PALABRA DE ENTRADA:

- Almacenado en un arreglo de tipo String.

ESTADO INICIAL:

- Almacenado en una variable de tipo String.

ESTADO FINAL:

- Almacenado en una variable de tipo String.

Para desarrollar esta aplicación nos apoyamos del modelo de desarrollo Vista–Controlador para asociar entornos gráficos a Python, usando Qt como diseñador de la interfaz y PySide como intérprete entre ambas plataformas.

Nuestro esquema de desarrollo cuenta con 4 archivos:

- MainWindow.py: Archivo que genera Pyside con el comando pyside-uic, usando como esqueleto gráfico la UI creada con Qt Designer.
- Controller.py: el cual interpreta las funciones gráficas con los métodos asociados.
- CreaMT.py: El cual consta de funciones primarias de cálculos lógicos a ser usados en la instanciación de la Máquina.
- MTuring1Cinta.py: El cual contiene la estructura principal de la Máquina y métodos de validaciones necesarias.

MÉTODOS CLAVE :

- obtEstadosAlfabeto(entrada): Filtra la entrada de transiciones obteniendo un arreglo de estados y de letras del alfabeto, pertenecientes a la máquina con la que se quiere trabajar
- llenarTransiciones(estados, alfabeto, entrada): Usando el arreglo de estados y el de la letras del alfabeto obtenido anteriormente, se crea y llena la matriz bidimensional ubicando correctamente las transiciones con sus respectivos estados actuales y símbolos de lectura, el núcleo de la máquina de turing actual.
- aceptaPalabra(palabra): En él se encuentra el algoritmo de trabajo para aceptar una palabra que pertenezca al lenguaje definido por la máquina, usando las transiciones, estados y el alfabeto para lograrlo.

PROGRAMA

NOMBRE: Simulador Máquina de Turing 1-Cinta by Caloguerea-Rojas-Sanchez

LENGUAJE:

- Python 2.7.6 - compilador Versión GCC 4.8.2.
- Librerías PySide basado en la tecnología QT para entorno gráfico.

  Adicionalmente:

  - IDE: Ninja – IDE 2.3
  - SublimeText3
  - Plataforma de desarrollo: Linux distribución Ubuntu/Kubuntu
  - Control de versiones: GitHub

REQUISITOS MÍNIMOS:

Software:

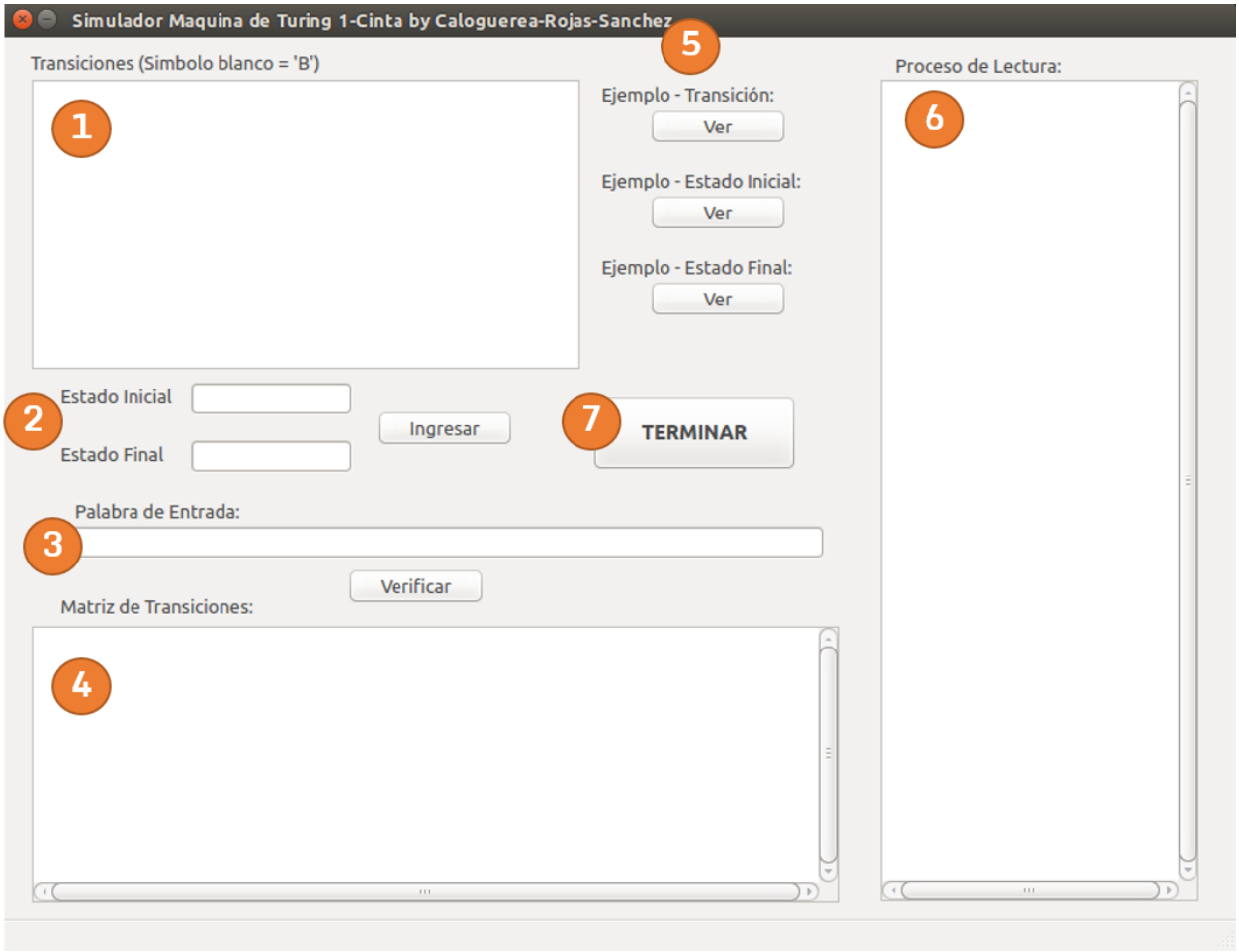
- Sistema operativo Linux con paquetería "Wine" instalada
- SIstema operativo Windows 7+ , arquitectura 32 y 64 bits.
- Nota: no se necesita ningún otro tipo programa previo para poder ejecutar esta aplicación.

Hardware:

- 1GB de memoria RAM
- Procesador de 1GHz
- Almacenamiento 7mb. mín y 27mb. como máx, dependerá del sistema de archivos.
- Periféricos: Monitor, dispositivo señalador (Mouse) y teclado para ingreso de parámetros

CÓMO USAR:

Para ejecutar la aplicación es necesario dirigirse al directorio del programa, y ejecutar el archivo "MTLauncher.exe". Una vez ejecutado se lanzará una interfaz similar a la siguiente imagen:



En la imagen podemos apreciar los siguientes aspectos:

1 – Corresponde a un recuadro para ingresar las respectivas tuplas ("TextBox"). En este apartado se debe ingresar la cadena completa que contendrá cada tupla y a su vez los estados y transiciones correspondientes. Al ingresarlas el programa verificará que siga la correcta sintaxis de separación de elementos por comas y tuplas por punto y coma. En caso de error se indicará mediante una ventana emergente solicitando el ingreso nuevamente.

2 – Tenemos 2 recuadros, ambos con sus respectivas etiquetas explicativas de qué es lo que solicitan (Estado inicial y Estado final).

3 – Recuadro que permite al usuario ingresar la palabra a evaluar bajo la máquina previamente ingresada. De no haberse realizado el paso anterior, el software notificará de esto solicitando la corrección.

4- De ser validado los pasos anteriores se estructurará una matriz de transiciones, la cual se desplegará en una tabla "Matriz de Transiciones"

5 – En caso de que el usuario tenga alguna duda de cómo interactuar con este programa, hemos dejado a disposición en el apartado el segmento de ayuda, el cual le proporcionará la información requerida de cómo se debe ingresar el dato solicitado.

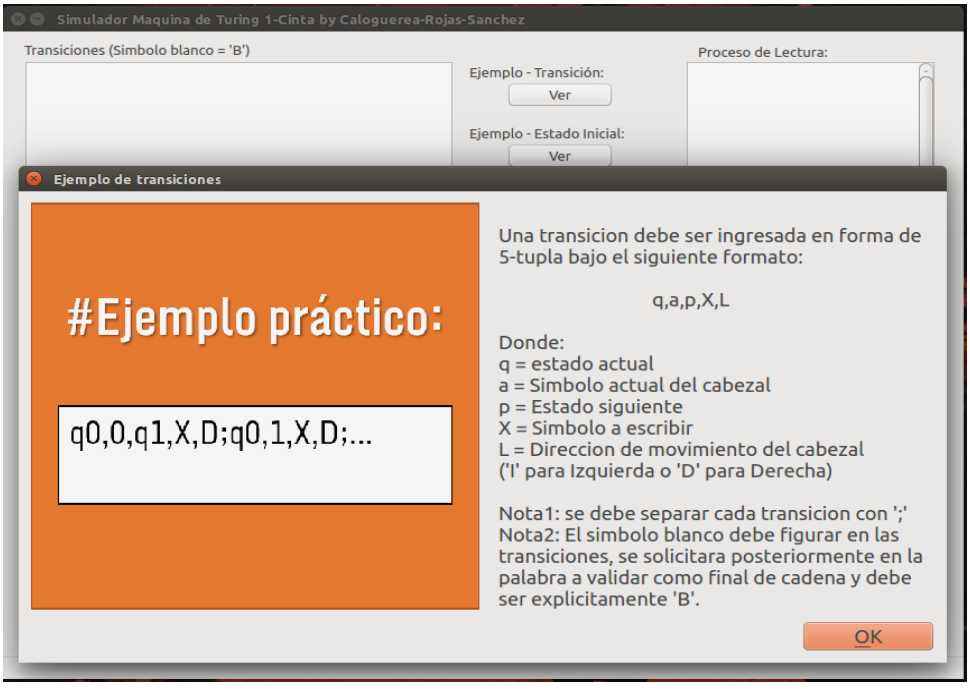
6 - Este textbox vertical mostrará al usuario cada paso del recorrido de la palabra y sus transformaciones por estados, mejorando la comprensión sobre el funcionamiento de este software.

7- Botón "terminar" realiza la función de cerrar el programa una vez terminado su uso (La X en la parte superior de la ventana realiza la misma acción).

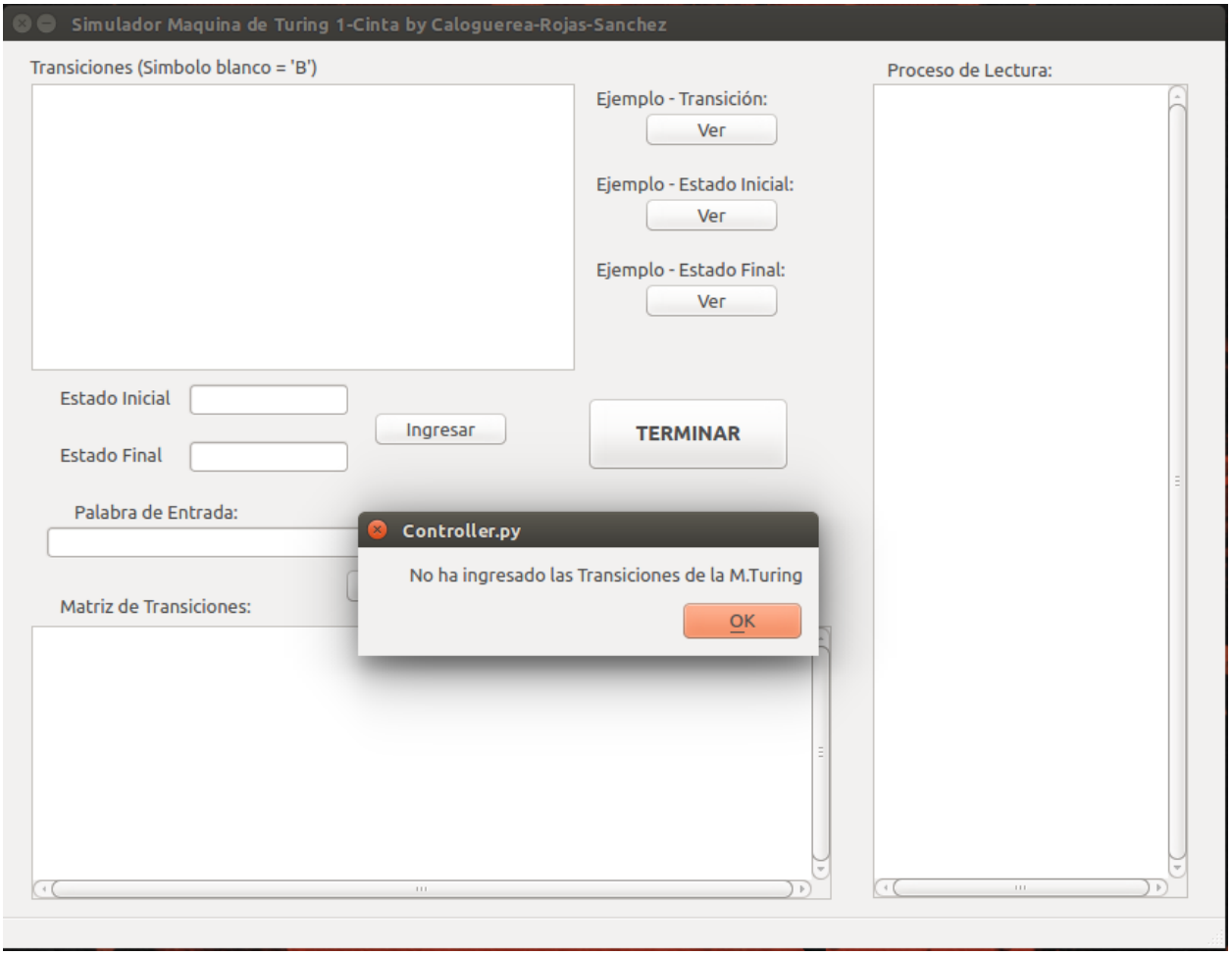
EJEMPLO DE FUNCIONAMIENTO:

Ya hemos revisado la interfaz y su distribución en el segmento anterior. Daremos inicio a la demostración de funcionamiento con las vistas que entrega la aplicación al consultar los botones de ayuda.

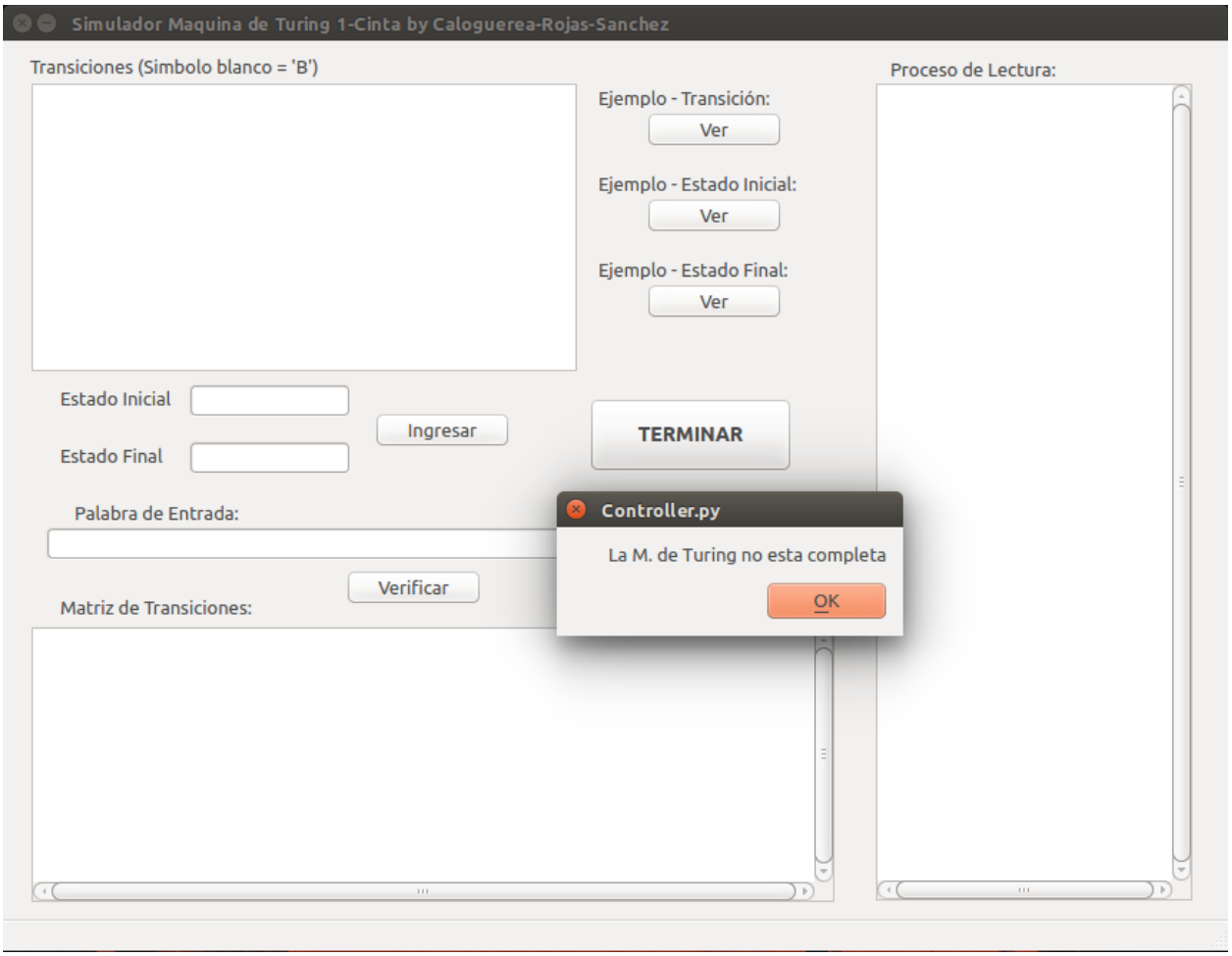
(Ventana emergente al presionar ver en "Ejemplo - Transición". Otros mensajes similares serán desplegados para los otros casos de ayuda).



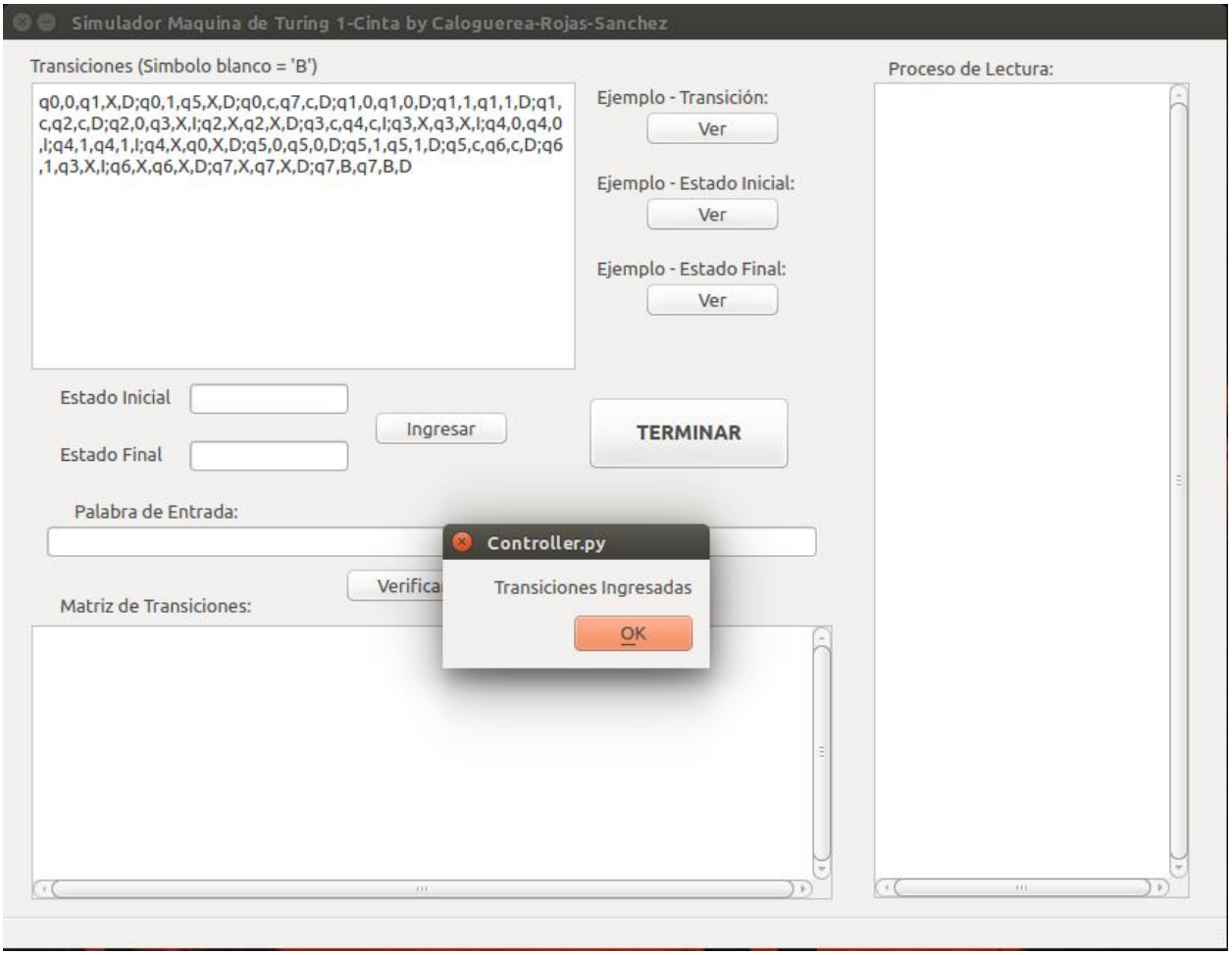
El siguiente ejemplo corresponde al comportamiento del programa en el caso de que se presione el botón "ingresar" sin contenido alguno, y su mensaje de retroalimentación.



Análogamente informará un mensaje similar al intentar "Verificar" una palabra sin que se haya ingresado y creado internamente una MT (para esto se deben validar transiciones, estado inicial y final previamente).

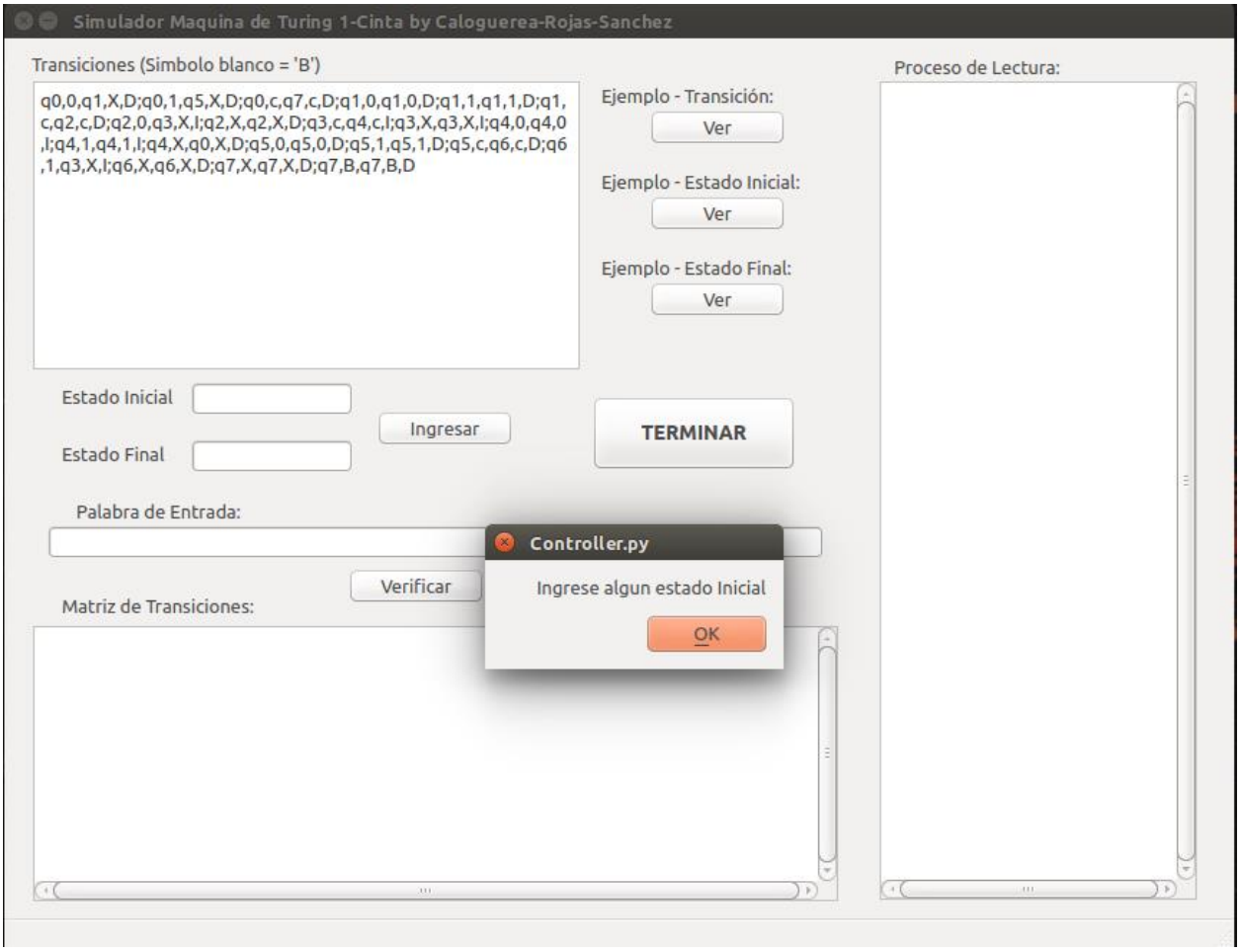


En este ejemplo podemos ver cómo muestra las transiciones ingresadas, y una ventana emergente informando del ejercicio realizado.



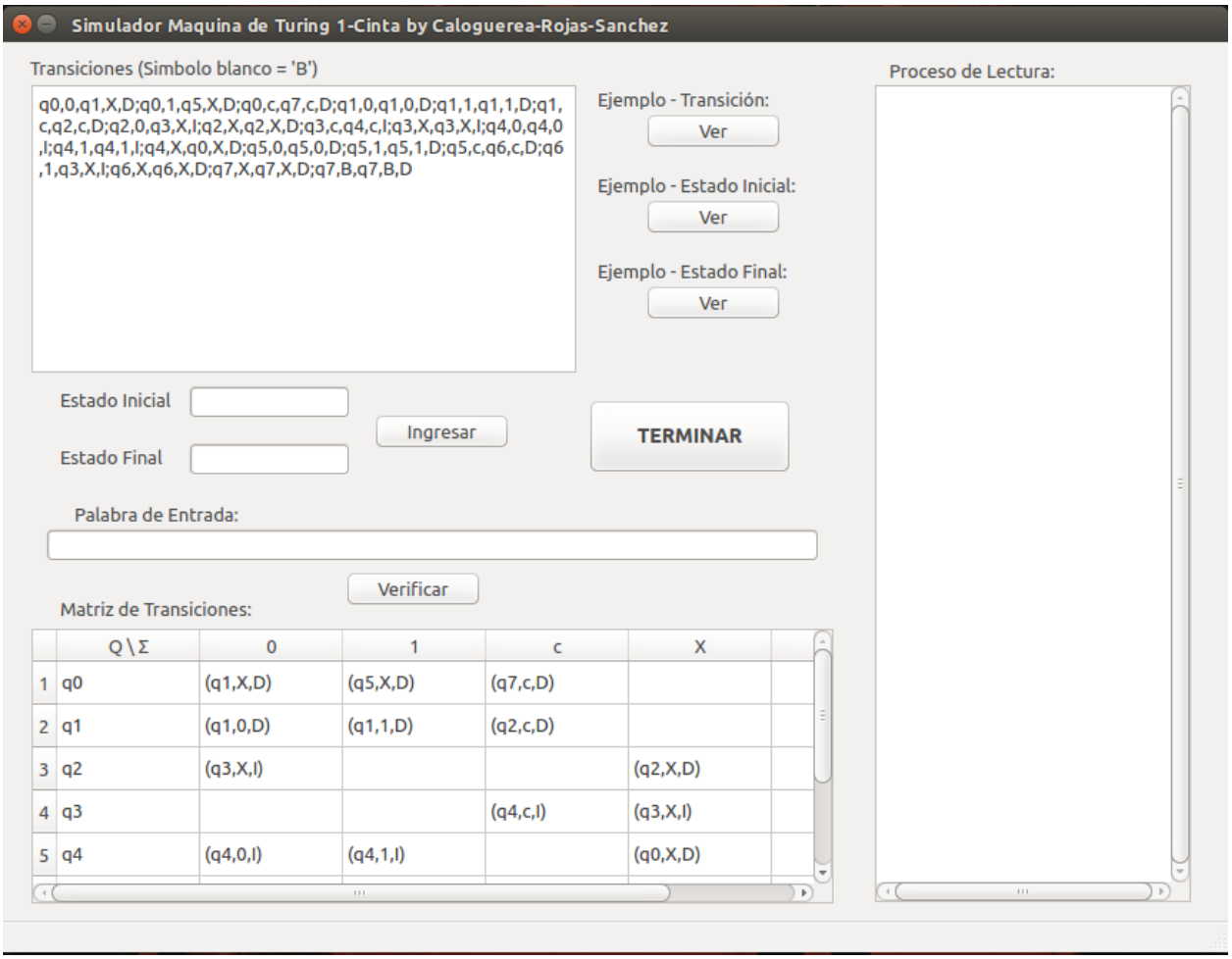
Seguido de la validación de las transiciones correctamente ingresadas, el programa solicitará los datos necesarios respectivamente (estado inicial y estado final en caso de que falte alguno de estos).

Durante el ingreso de las transiciones, si no están indicados los estados Inicial y Final, informará para completar su función de ingreso.



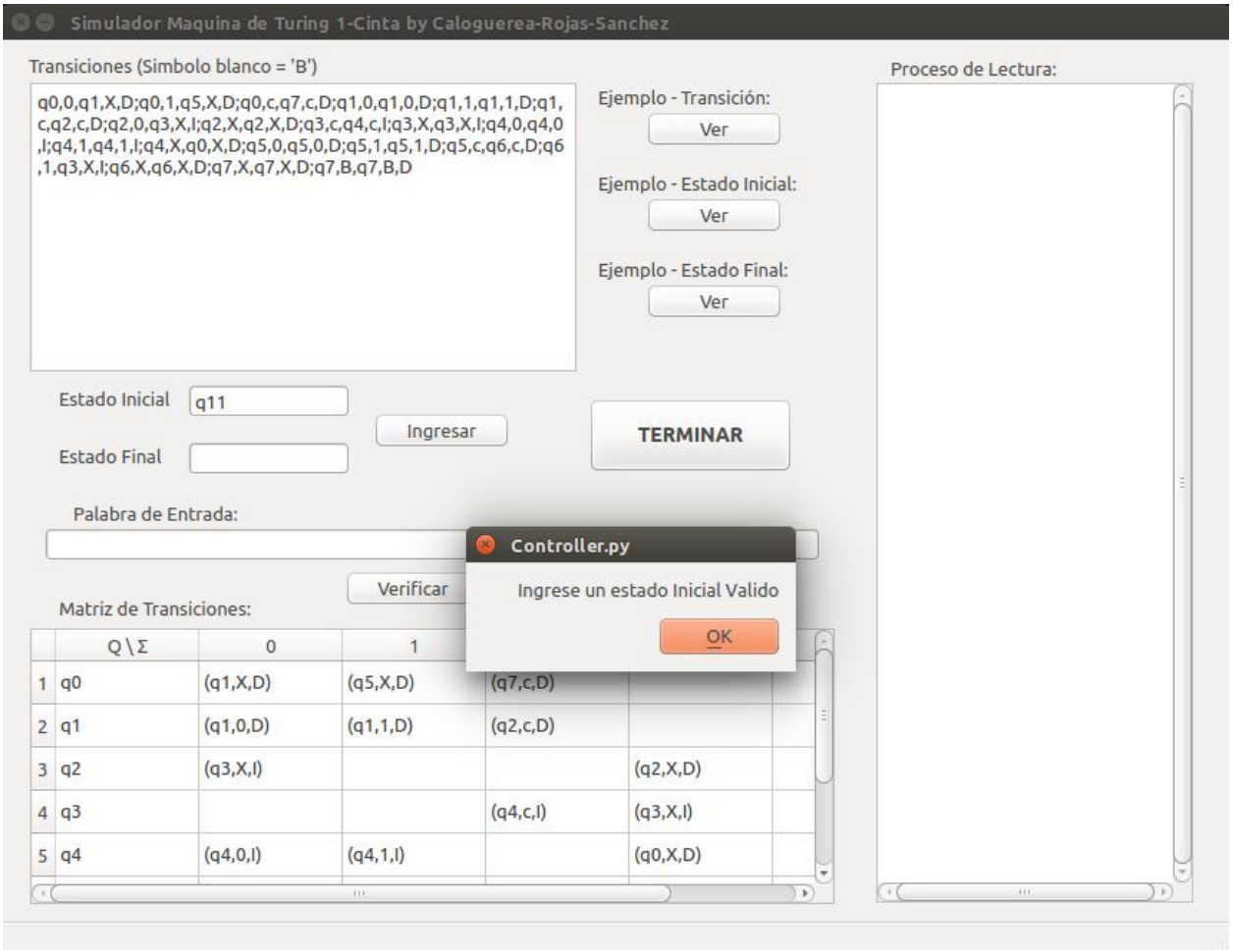
Posteriormente, una vez ingresados los campos necesarios para insertar la MT, se generará una tabla con las transiciones que realiza en su proceso de lectura.

Independiente de si se ingresan o no los estados; inicial y final, si se validaron propiamente, se generarán sobre la matriz de transiciones como podemos apreciar en la siguiente imagen.



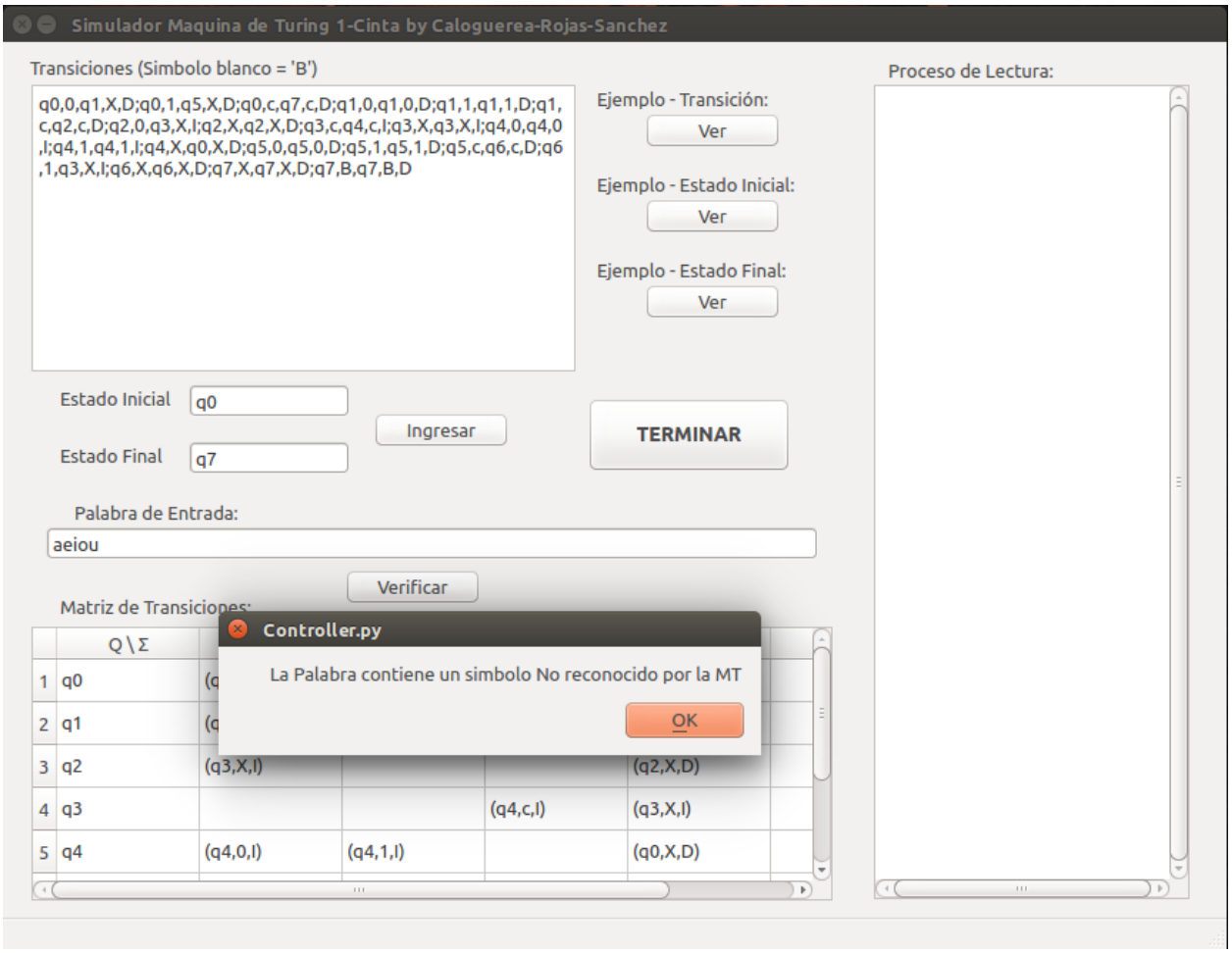
En caso de ingresar un estado que no pertenece a las transacciones ingresadas previamente se informará como muestra la siguiente imagen.

Desde lo mencionado, la matriz de transiciones se generará a partir de las "transiciones" ingresadas, independiente de si hay o no entrada de "Estado Inicial" o "Estado Final".



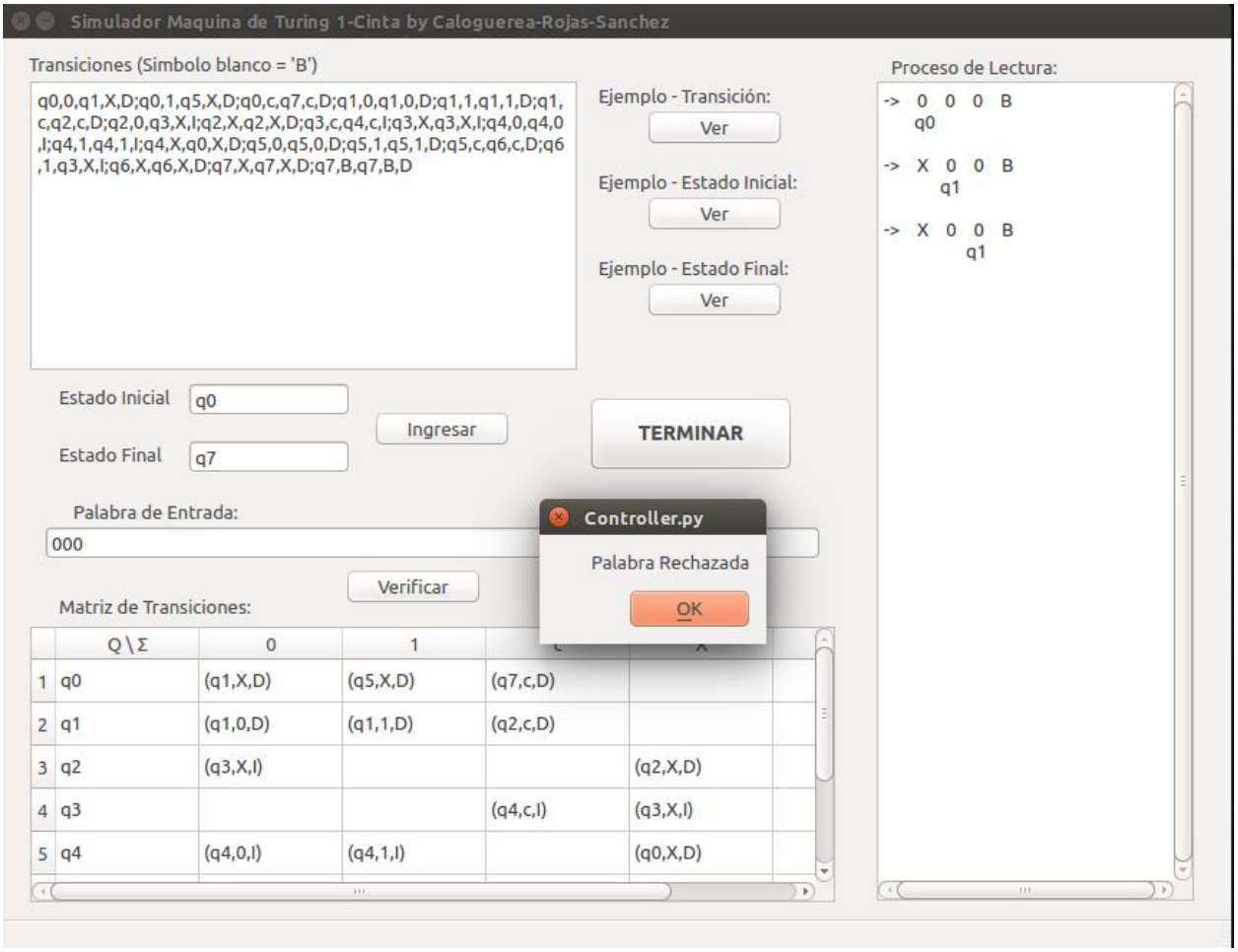
Cabe mencionar que la matriz de transiciones, no es requisito para el funcionamiento, es solo una muestra esquemática de su funcionar.

Ejemplo de verificación 1: A continuación podemos apreciar cómo el programa identifica la palabra ingresada. En este caso nos alerta de que la palabra contiene símbolos que no pertenecen al esquema definido por las transiciones.



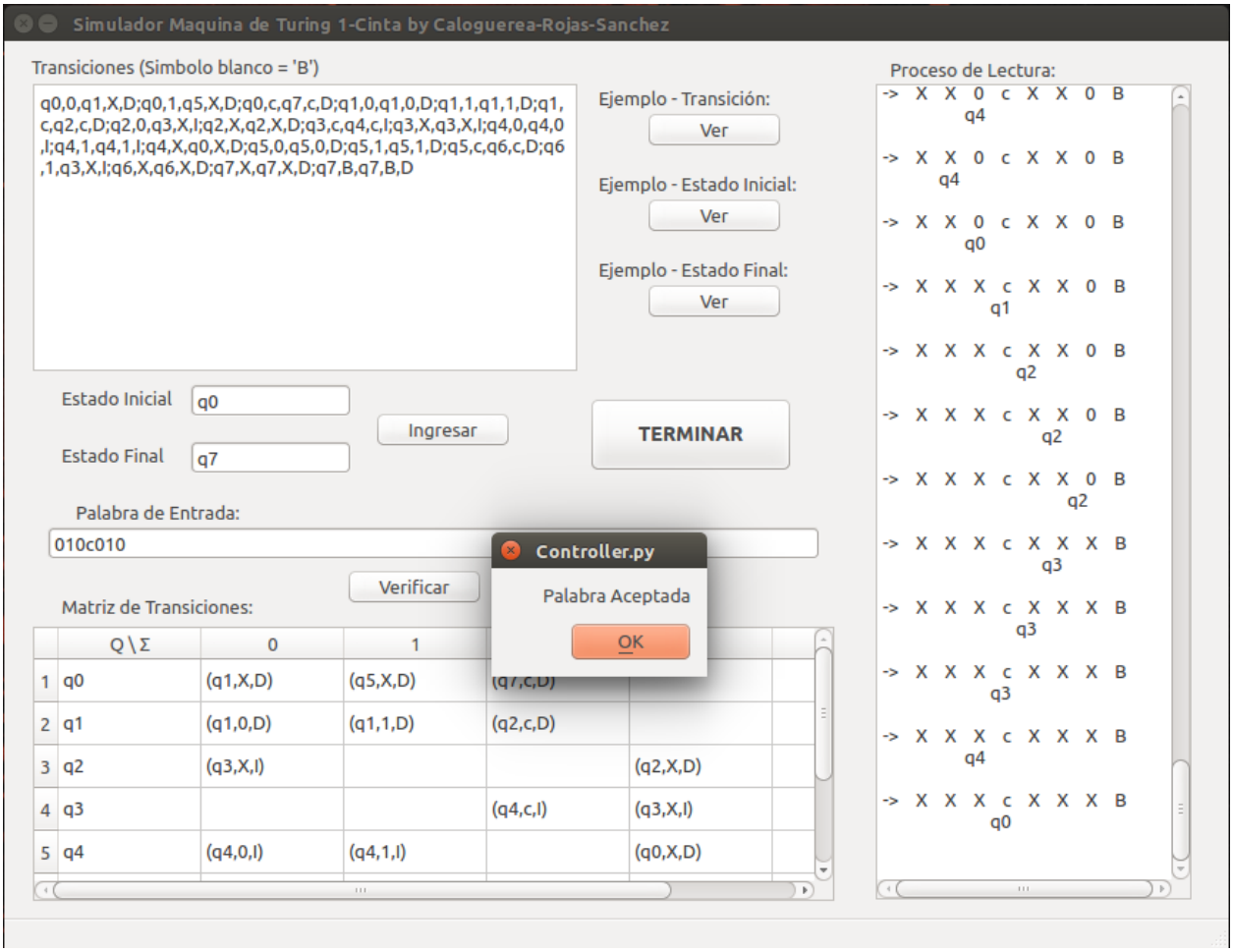
Lo que significa que si al momento de validar, ingresamos caracteres no válidos para su comprobación, y que además no tengan relación con el alfabeto de las transiciones ingresadas, el programa identificará si hay símbolos no válidos y el programa quedará a la espera de una comprobación válida.

Ejemplo de verificación 2: Al probar con una palabra que sí pertenece al alfabeto descrito por la máquina pero que no es aceptado por ella, nos aparecerá el siguiente mensaje.



Indicamos que al momento de ingresar la palabra a revisar por la MT, en el recuadro derecho mostrará el seguimiento de lectura paso a paso según los estados de la palabra y sus modificaciones, entregando finalmente el resultado en un cuadro de diálogo.

Ejemplo de verificación 3: Al ingresar una palabra perteneciente al alfabeto y que es aceptada por la MT nos mostrará el siguiente mensaje (Podemos apreciar que como es válida en el alfabeto, se ejecuta la verificación de palabra y se muestra dicho proceso en "Proceso de lectura").



BIBLIOGRAFÍA

http://www.dma.eui.upm.es/historia_informatica/Doc/Personajes/AlanTuring.htm

http://maquinaturing.blogspot.com/p/funcionamiento-de-la-maquina-turing.html

CONCLUSIÓN

Podemos concluir que la teoría de una (MT) creada hace décadas es totalmente compatible con los esquemas de la informática actual y por ende totalmente representable mediante herramientas modernas. Esto demuestra que la ciencia de la computación actual se basa en los principios lógicos establecidos por estos padres de la informática quienes visualizaron cintas y cabezales que hoy llamamos arreglos, listas, índices o punteros.

MEJORAS:

- Se añadió una interfaz gráfica que facilita el uso y comprensión de la ejecución de nuestro programa
- Validaciones de Transiciones con cantidad de elementos mínimos requeridos y combinacion "estado_actual,simbolo_actual_del_cabezal" única
- Validación de palabra de entrada
- Validación estado inicial y estado final
- Matriz de Transiciones
- Seguimiento del proceso de lectura.
- Ayuda interactiva, mejor explicado el programa.

LIMITACIONES:

- Se trató de cambiar icono al ejecutable MTLauncher, sin resultado.
- No logramos incorporar al programa la propiedad de esconder la ventana Shell del ejecutable.
- No se pudo hacer implementación decorativa al programa.