Universidad Austral de Chile

Conocimiento y Naturaleza

APD PLUS ULTRA

Profesor a cargo: Maria Eliana de la Maza W.

Integrantes: Edgar Alarcón
Javier Rojas
Diego Vera

Asignatura: Teoría de autómatas

INTRODUCCIÓN

Un autómata de pila o Push-Down determinista es un autómata que cuenta con un mecanismo que permita almacenamiento ilimitado y opera como una pila. un APD permite aceptar algunos lenguajes que no son regulares (lenguajes que no existe autómata finito determinista que lo acepte).

En este trabajo se realizó un software para windows, el cual tiene como objetivo simular cualquier autómata Push-Down determinista, facilitando la verificación de palabras que pertenecen o no a un lenguaje.

Para operar el software es necesario ingresar cada una de las transiciones correspondientes al APD, el estado inicial, si acepta por stack vacío o estado final (en caso de aceptar por estado final se debe especificar) y las palabras que se desean verificar.

DESARROLLO

Estructuras de datos

Para almacenar las transiciones ocuparemos una lista enlazada como se muestra a continuación:

```
typedef struct nodo{
    string estado;
    string simbolo;
    string simPila;
    string estSiguiente;
    string agregarpila;
    struct nodo *next;
}nodo;
```

donde cada campo representa :

$$\delta(estado, símbolo, simPila) = (estSiguiente, agregarpila)$$

A priori no se conoce cuantas transiciones se requieren agregar para cada APD, por lo cual una lista enlazada es la mejor opción debido a que es dinámica.

Se definió una estructura llamada “reg” la cual sirve para tener un registro del estado actual en el que se encuentra el APD después de leer un símbolo, consta de 2 campos:

“estadoActual” y “simbolo_leer” . El primer campo guarda el estado actual del autómata, mientras que el segundo campo guarda el próximo símbolo que se debe leer(si la palabra ingresada es “ab”, este campo guardará una “a” inicialmente, después de ser leída, almacenará una “b”).

```
typedef struct reg{

    string estadoActual;
    string simbolo_leer;
}reg;
```

La palabra a leer, el estado inicial y estado final se guardan en variables de tipo string, ya que determinamos que no es necesaria una estructura de datos para almacenarlos

Finalmente, utilizamos una pila de C++ std::stack , esta estructura tiene la finalidad de almacenar los valores de la pila del APD. Esta estructura es dinámica y cuenta con funciones para acceder al tope de la pila, agregar elementos y eliminar elementos, lo cual produce una clara ventaja al programar.

Métodos y algoritmos

crear_nodo(): Metodo encargado de asignar espacio de memoria a cada nodo que posee la lista enlazada.

agregarlist() : Este método se encarga de agregar las transiciones a la lista enlazada, en caso de ser la primera transición, se creará la lista.

actualizar_estado(): Este método se encarga de actualizar el registro, compara “estado Actual” , “símbolo_leer” y el tope de la pila con “estado”, “símbolo”, “simPila” de cada una de las transiciones almacenadas en la lista enlazada. Cuando se encuentra la transición que coincide con estos tres parámetros, estado actual se actualiza con el valor “estSiguiente” de la transición, a la pila se le agrega lo que hay en “agregarpila” de la transición, o bien se elimina el tope si “agregarpila” es una “e”, finalmente retorna un 0. En el caso de que no se encuentre transición que satisfaga las 3 comparaciones, automáticamente retorna un 1, esto significa que el autómata no acepta la palabra, por ende, no es necesario seguir recorriendo la palabra, haciendo eficiente al algoritmo.

vaciar_stack() : Este método simplemente elimina todo lo que se encuentre en el stack. Cada vez que se verifica que si una palabra pertenece o no al lenguaje, es necesario hacer un reset del stack, ya que este puede quedar con símbolos no deseados para la siguiente palabra a leer.

Presentación del software

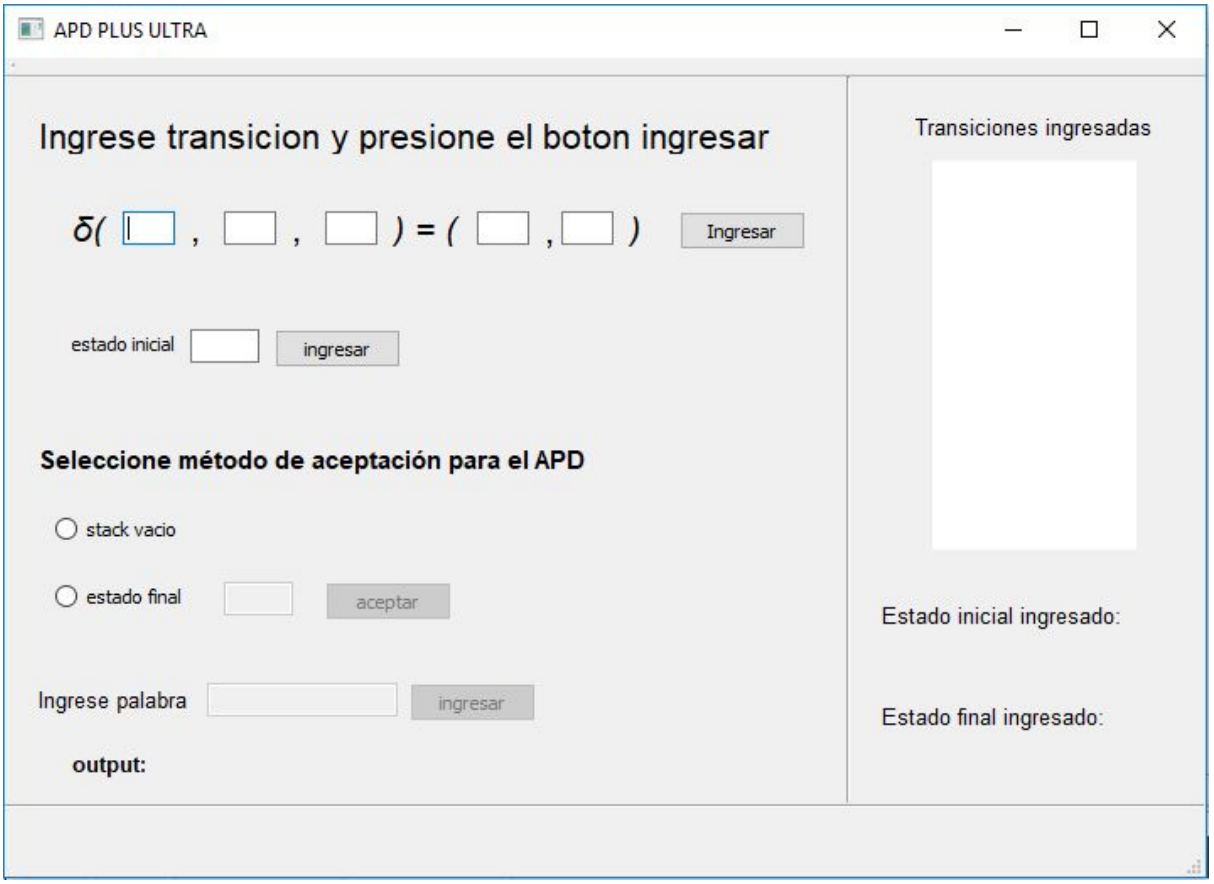
Nuestro software llamado “APD PLUS ULTRA” fue programado en el lenguaje C++, posee una interfaz gráfica realizada con QT creator. Los requisitos mínimos de funcionamiento son:

Software : Sistema operativo Windows 7 o superior de 32 o 64 bits

Hardware: 512 MB de memoria ram
Procesador : Intel Celeron N3350 para intel y AMD A6 para AMD
Espacio disponible: 100 MB

Manual de uso

Diríjase a la carpeta que contiene el software, dentro de esta se pueden visualizar varios elementos, busque el elemento llamado “APD PLUS ULTRA.exe” y haga doble clic sobre él. Luego de unos segundos aparecerá una ventana como la que se presenta a continuación.



Primer paso: Se debe rellenar los campos correspondientes a una transición, puede ir cambiando de un recuadro a otro con la tecla tab, una vez llenado los campos, presione “ingresar”, notará que en la sección derecha, la transición aparecerá escrita en el recuadro blanco para llevar un registro de las transiciones que ingrese. Para ingresar otra transición, basta con repetir el primer paso nuevamente las veces que sea necesario.

Segundo paso: Luego de ingresar todas las transiciones, llene el campo de estado inicial y presione el botón “ingresar”, también puede hacer uso de la tecla “enter” . Notará que en la sección de la derecha aparecerá el estado inicial que ingresó, si usted erró al ingresar el estado, no se preocupe, siempre podrá ingresarlo nuevamente.

Tercer paso: Seleccione si desea que el autómata acepta palabras por stack vacío o por estado final. Si elige este último, deberá rellenar el campo con el estado final y presionar el botón “aceptar”, también puede hacer uso de la tecla “enter”. En caso de que ingrese por estado final, en la sección de la derecha podrá visualizar el estado final ingresado, si usted erró al ingresar el estado, no se preocupe, siempre podrá ingresarlo nuevamente.

Cuarto paso: Escriba la palabra que desea verificar si pertenece o no al lenguaje y luego presione el botón “ingresar”, también puede hacer uso de la tecla “enter”. Notará que al lado de “output” aparecerá un mensaje diciendo si la palabra pertenece o no al lenguaje denotado por el APD ingresado por usted, si desea ingresar otra palabra, basta con repetir el cuarto paso las veces que se requiera.

Ejemplo de funcionamiento

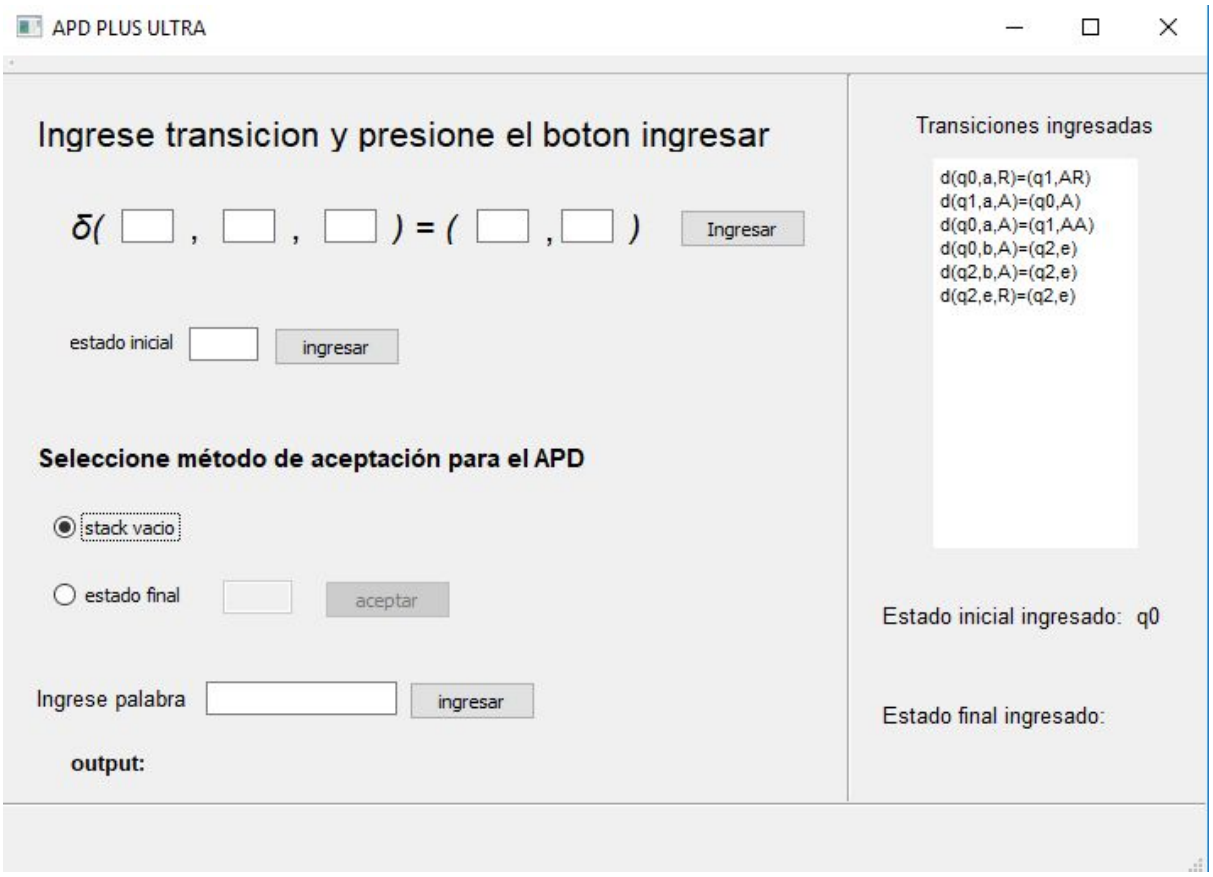
$sea\ L = \{a^{2n}b^n / \ n \geq 1\}$

cuyas transiciones son las siguientes:

$\delta(q_0, a, R) = (q_1, AR)$
$\delta(q_1, a, A) = (q_0, A)$
$\delta(q_0, a, A) = (q_1, AA)$
$\delta(q_0, b, A) = (q_2, e)$
$\delta(q_2, b, A) = (q_2, e)$
$\delta(q_2, e, R) = (q_2, e)$

el autómata acepta palabras por stack vacío y el estado inicial es q0.

Ingresando la información en el software, obtendremos lo siguiente:



para el ejemplo, usaremos las siguientes palabras:

-aab
-aaaabb
-ab

al ingresar las palabras, obtenemos el output del software correspondiente a cada palabra:

output: palabra 'aab' pertenece al lenguaje

output: palabra 'aaaabb' pertenece al lenguaje

output: palabra 'ab' NO pertenece al lenguaje

Conclusiones

El software APD PLUS ULTRA, cumple con el objetivo de simular cualquier autómata Push-Down, verificando si las palabras ingresadas pertenecen al lenguaje, lo realiza de forma eficiente gracias al uso de algoritmos eficientes. Posee una agradable interfaz para el usuario siendo además un software auto explicativo.

Gracias a las personas que probaron el software, se percató de algunos aspectos a corregir en el futuro. Uno de estos es no poder editar ni eliminar transiciones en caso de que el usuario se equivoque.

Dentro de las limitaciones se encuentra el tamaño del software, siendo aproximadamente de 90mb, esto debido a las librerías que necesita el software para ejecutarse en cualquier máquina.

Bibliografía

http://doc.qt.io/qtcreator/
http://www.cplusplus.com/reference/