Universidad Austral de Chile
Escuela de Ingeniería Civil Informática

Simulador Autómata Pushdown
Teoria de Autómatas

Nicole Millalaf S.
Inti Romero M.
Alberto Lagos T.

Profesora:
Maria Eliana de la Maza W.

Valdivia - 8 de junio de 2010

Introducción

Un lenguaje es un conjunto de palabras (una palabra es una secuencia ordenada de símbolos de algún alfabeto; un alfabeto es un conjunto finito de símbolos). L= {hola} es un lenguaje (de una sola palabra) y L_1= {Hola, Como, Estas} es otro.
Los lenguajes se clasifican en conjuntos, de los cuales, algunos son subconjunto de otros (según lo visto en clase). Ejemplo de estos conjuntos serían: los Lenguajes de Libre Contexto, los Lenguajes regulares, los Lenguajes Recursivos Enumerables , etc . De los nombrados, los que interesan en esta ocasión, son los Lenguajes de Libre contexto.
Los Lenguajes Libres de Contexto (L.L.C.) forman una clase de lenguajes que incluye a la de los Lenguajes Regulares pero, de acuerdo con la Jerarquía de Chomsky, es más amplia que ellos.
Estos lenguajes son importantes tanto desde el punto de vista teórico, por relacionar las llamadas Gramáticas Libres de Contexto con los Autómatas Pushdown, como desde el punto de vista práctico, ya que casi todos los lenguajes de programación están basados en los L.L.C.

Para reconocer si un lenguaje es de Libre Contexto, se utiliza un Autómata Pushdown.

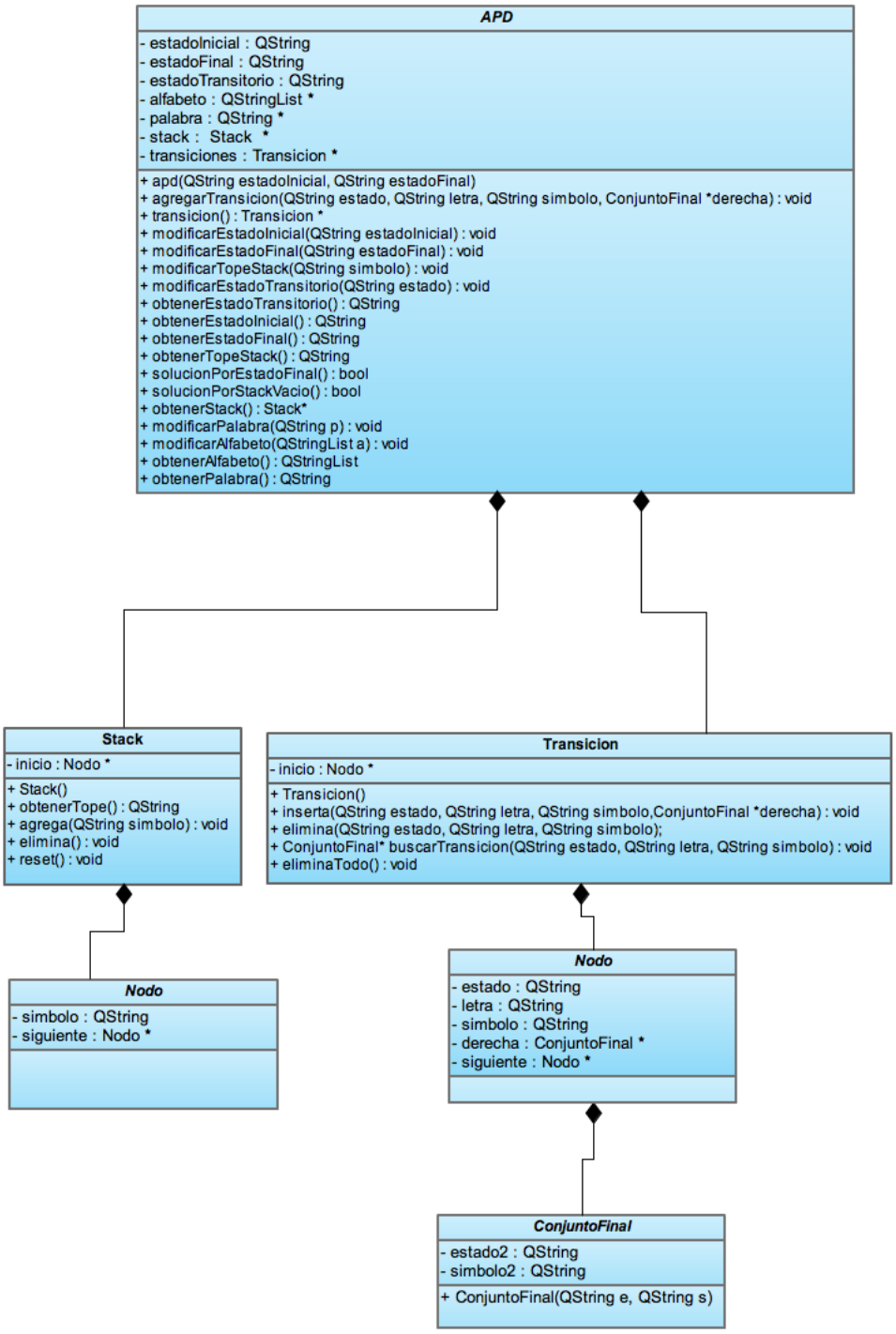
Un Autómata Pushdown es un modelo matemático de un sistema. Éste recibe una cadena constituida por símbolos de un alfabeto y determina si tal cadena pertenece al lenguaje que el autómata reconoce (éste lenguaje es obviamente un L.L.C.).

El objetivo de este trabajo es el de diseñar e implementar, mediante en un lenguaje de programación a elección, un Autómata Pushdown con dichas funciones.

Estructuras de Datos Utilizadas

Para implementar nuestro APD utilizamos el lenguaje C++ Nuestro programa consta en general (sin contar las clases de la interfaz) con 4 clases:

Diagrama de Clases



Como se aprecia en la figura, la clase APD tiene como atributos el estado inicial, estado final, un stack (de tipo Stack), y transiciones. El atributo transiciones es de tipo Transicion (el cual corresponde a una lista de objetos de la clase Transicion), que contiene los datos de las respectivas transiciones del APD. Cada objeto de la lista contiene los siguientes atributos:

Lado izquierdo de la transición:

- estado
- letra
- símbolo

Lado derecho de la transición:

- estado2
- simbolo2

El programa funciona así:

Se crea un objeto, llamado **autómata**, de la clase APD. El usuario ingresa el estado inicial y estado final, los cuales se guardan en los respectivos atributos de **autómata** e ingresa las transiciones del APD, las cuales se almacenan en el atributo transiciones. Cuando el usuario ingresa la palabra a evaluar y presiona "Verificar", el programa agrega el símbolo "R" al inicio del stack (atributo de **autómata**) crea un estado actual = **q0** (estado inicial) y verifica si existe alguna transición que en su lado izquierdo sea igual a:

δ(estado actual, primera letra de la palabra, tope del stack)

Si es que existe, se obtienen los datos de **derecha** (objeto de la clase ConjuntoFinal que corresponde al lado derecho de la transición) y se verifica si es que hay alguna transición que en su lado izquierdo sea igual a:

δ(**estado2**, segunda letra de la palabra, **primer símbolo de simbolo2**)

y así sucesivamente.

Si no existe, se rechaza la palabra.
Nuestros principales métodos para realizar las funciones de un APD son:

- **Insertar(QString estado, QString letra, QString simbolo, ConjuntoFinal derecha):** Método que inserta una transición a la lista de manera ordenada. Primero ordena por estado y luego por letra.
- **buscaTransición(estado, QString letra, QString simbolo):** Método que busca en la lista de transiciones el lado izquierdo de la transición que coincida con los datos recibidos, y entrega un objeto de la clase ConjuntoFinal que contiene la transición derecha respectiva.
- **Elimina(QString estado, QString letra, QString simbolo):** Método que elimina de la lista, la transición que, en su lado izquierdo, coincida con los datos recibidos. Este método se usa cuando el usuario desea eliminar una transición ingresada.

Para realizar la unión entre la interfaz y la estructura de un APD utilizamos los siguientes métodos (solo describimos los mas importantes)

- **ingresarNuevoAutomata():** Simplemente borra todos los datos anteriores para que se pueda ingresar nuevos datos. (estados, transiciones, palabras, etc).
- **editarTransicion():** Obtiene la transición editada (previamente seleccionada) y la guarda en la **lista enlazada.**
- **verificarPalabra():** Luego de ingresado todo lo necesario para ocupar el programa, se ejecuta esta función (cuando se hace click en el botón *Validar Palabra*)
- **eliminarTransicion():**Esta función elimina de la lista de la interfaz gráfica la transición seleccionada ademas de borrarla de la lista enlazada.

Cabe destacar que los métodos descritos anteriormente pertenecen a la clase **mainwindow**.
NOTA: Los metodos anteriores prefijo **on** y postfijo **clicked**

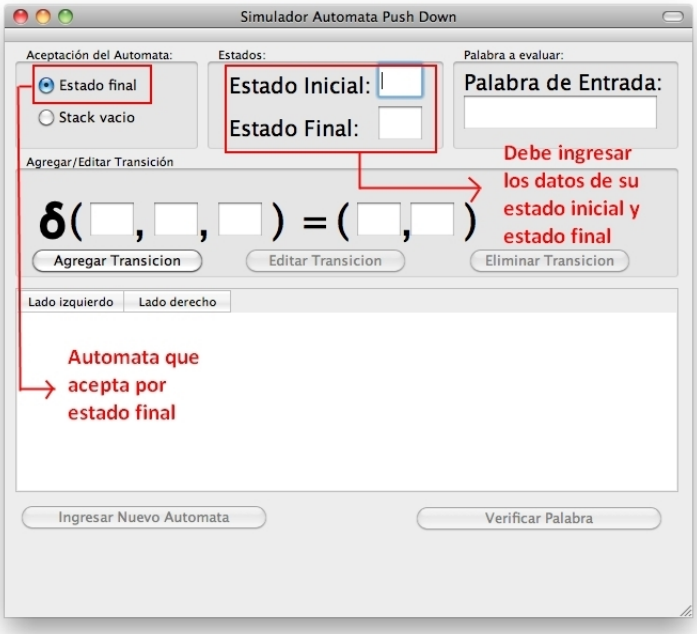
Especificación del programa

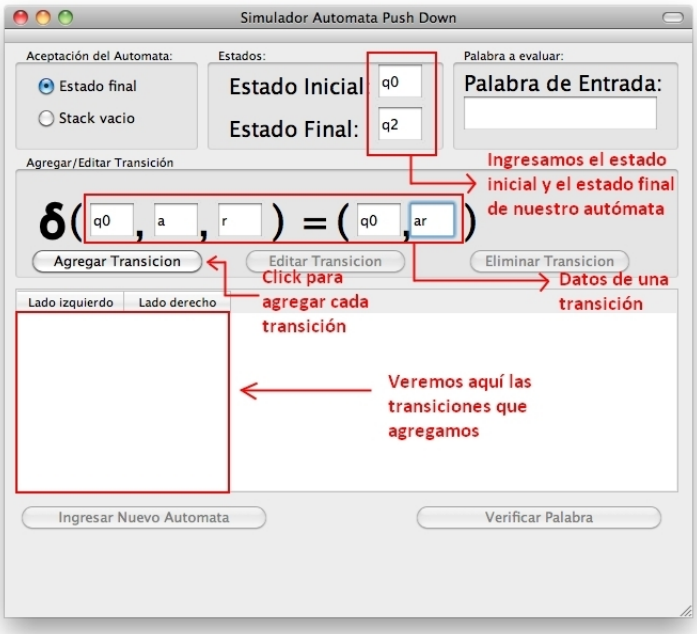
- **Nombre del programa:** Simulador Automata Push Down
- **Lenguaje de Implementación:** C++ y Qt
- **Requerimientos básicos de hardware:** El espacio necesario, en disco duro, que se requiere para utilizar este software es de 1 MB. Siendo suficiente los computadores actuales.
- **Requerimientos básicos de software:** El archivo ejecutable que generamos al utilizar Qt es soportado por diversas plataformas, por lo que no es un problema ocuparlo en diferentes sistemas operativos. Pero en este especifico caso es solo posible ejecutarlo en plataforma OS X (MAC)

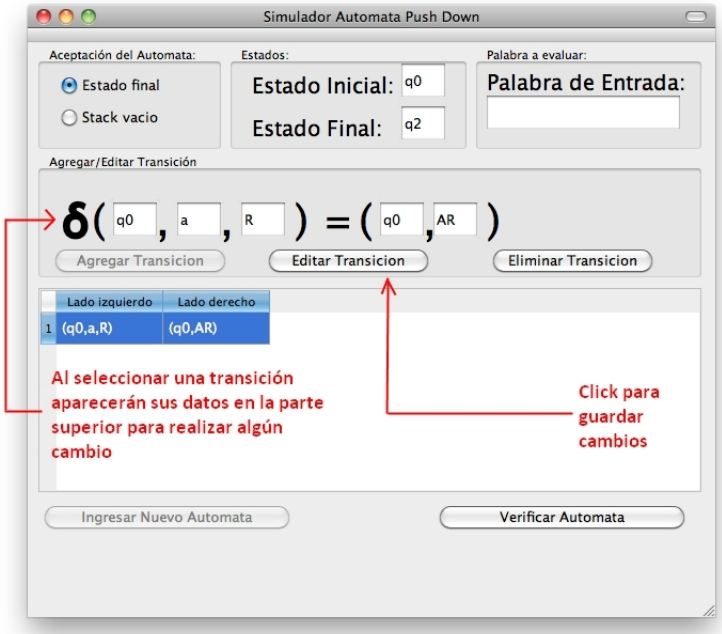
Manual de uso

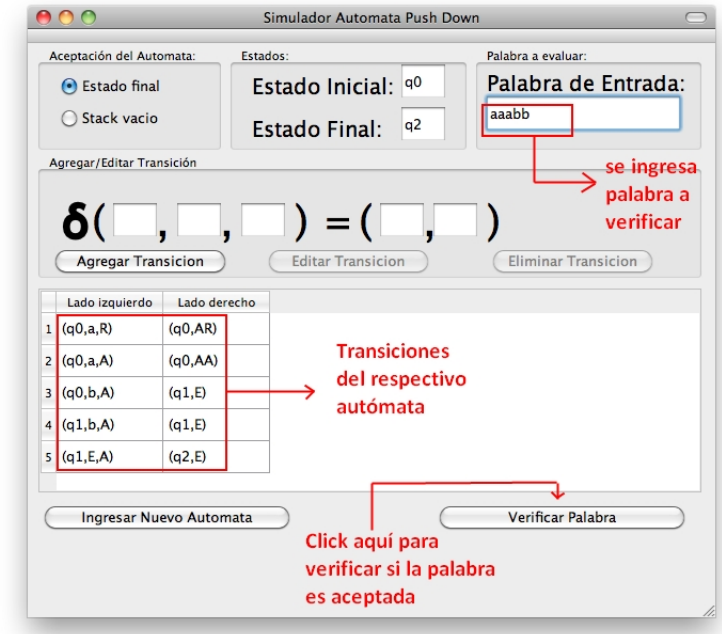
En la carpeta **debug** que se encuentra dentro de la carpeta del proyecto llamada **Simulador Automata Push Down**, una vez en la carpeta especificada veremos un archivo llamado **Simulador Automata Push Down.exe**, lo abrimos y ya estamos en nuestra aplicación. Los requerimientos fueron indicados anteriormente y para saber como utilizarlo o que poner en cada campo, solo basta dejar el mouse un momento sobre el lugar del que se desea saber información.

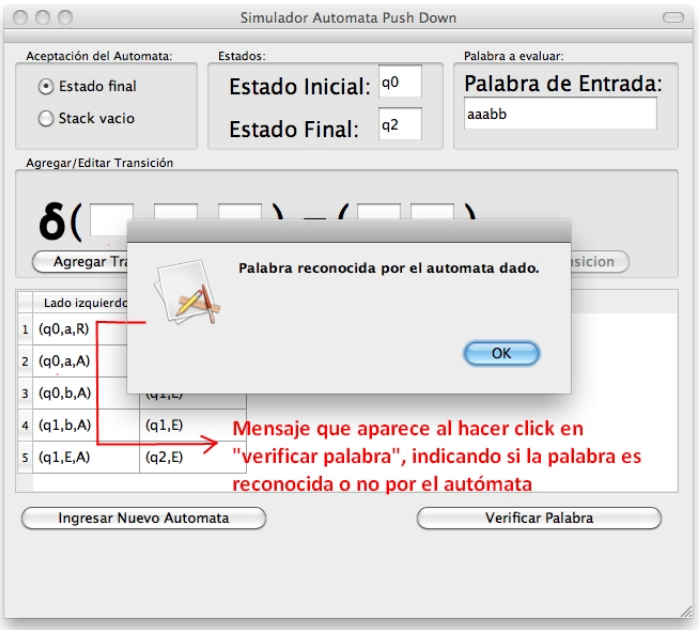
Un ejemplo de funcionamiento se verá mediante imágenes, las cuales se irán describiendo para visualizar con mayor facilidad su funcionalidad. Autómata que acepta por estado final:


Simulador Automata Push Down
Aceptación del Automata:
Estado final
Stack vacio
Estados:
Estado Inicial:
Estado Final:
Palabra a evaluar:
Palabra de Entrada:
Agregar/Editar Transición
δ(, ,) = (,)
Agregar Transicion
Editar Transicion
Eliminar Transicion
Debe ingresar los datos de su estado inicial y estado final
Lado izquierdo
Lado derecho
Automata que acepta por estado final
Ingresar Nuevo Automata
Verificar Palabra


Simulador Automata Push Down
Aceptación del Automata:
Estado final
Stack vacio
Estados:
Estado Inicial
Estado Final:
q0
q2
Palabra a evaluar:
Palabra de Entrada:
Agregar/Editar Transición
δ(q0 , a , r) = (q0 , ar)
Agregar Transicion
Editar Transicion
Eliminar Transicion
Ingresamos el estado inicial y el estado final de nuestro autómata
Click para agregar cada transición
Datos de una transición
Lado izquierdo
Lado derecho
Veremos aquí las transiciones que agregamos
Ingresar Nuevo Automata
Verificar Palabra


Simulador Automata Push Down
Aceptación del Automata:
Estado final
Stack vacio
Estados:
Estado Inicial: q0
Estado Final: q2
Palabra a evaluar:
Palabra de Entrada:
Agregar/Editar Transición
δ(q0 , a , R) = (q0 , AR)
Agregar Transicion
Editar Transicion
Eliminar Transicion
Lado izquierdo
Lado derecho
1 (q0,a,R) (q0,AR)
Al seleccionar una transición aparecerán sus datos en la parte superior para realizar algún cambio
Click para guardar cambios
Ingresar Nuevo Automata
Verificar Automata


Simulador Automata Push Down
Aceptación del Automata:
Estado final
Stack vacio
Estados:
Estado Inicial: q0
Estado Final: q2
Palabra a evaluar:
Palabra de Entrada:
aaabb
se ingresa palabra a verificar
Agregar/Editar Transición
δ(, ,) = (,)
Agregar Transicion
Editar Transicion
Eliminar Transicion
Lado izquierdo
Lado derecho
1 (q0,a,R) (q0,AR)
2 (q0,a,A) (q0,AA)
3 (q0,b,A) (q1,E)
4 (q1,b,A) (q1,E)
5 (q1,E,A) (q2,E)
Transiciones del respectivo autómata
Ingresar Nuevo Automata
Verificar Palabra
Click aquí para verificar si la palabra es aceptada



Autómata que acepta por stack vacío

Los siguientes pasos son los mismos que para el autómata que acepta por estado final.

Conclusión

Para nosotros conocer el comportamiento y funcionamiento de los APD es importante ya que estos son los que aceptan a los L.L.C. y la mayoría de los lenguajes de programación se basan en ellos; por lo que esta implementación reforzó bastante nuestros conocimientos sobre Autómatas Pushdown.

Nuestro programa puede mejorar, una mejora podría ser, que la lista de transiciones se reemplace por una tabla hash para que la búsqueda sea más rápida; nos ahorraríamos métodos de ordenamiento y aumentaría la velocidad de ejecución del programa.

Como limitación, destacamos el hecho de que se pueda ocupar sólo por alguien que ya conoce de autómatas pushdown; si bien, un alumno, para ciertos casos, puede usarlo para verificar si sus autómatas son correctos, pero no se garantiza veracidad en lo que concluya el simulador de APD ya que éste fue creado desde la base de que el autómata ingresado es correcto.