Compiladores, Sem: 2020-1, 3CV1, Práctica 2, 1 de Noviembre del 2020

Práctica 2: Implementación del Algoritmo McNaughton-Yamada-Thompson

Ayala Segoviano Donaldo Horacio

Escuela Superior de Cómputo
Instituto Politécnico Nacional, México
ayala.segoviano.donaldo@gmail.com

Objetivo: Utilizando los principios de la programación Orientada a Objetos, diseñar y desarrollar la clase Thomsonconel método:

- convertir(expresión_regular:string): AFN

1. Introducción

El algoritmo AlgoritmoMcNaughton-Yamada-Thompson (también conocido como método de Thompson), sirve para obtener autómatas finitos no deterministas con transiciones vacías (AFND-ϵ) a partir de expresiones regulares (ER).

2. Implementación de la clase Thompson

2.1. Clase Nodo

2.1.1. Descripción

Para la implementación de la clase Thompson se implementó la clase Nodo, que permite generar arboles binarios de expresión y así simplificar la generación del AFN.

2.1.2. Atributos

La clase Nodo cuenta con los siguientes atributos:

- (-)**Nodo*** hijo_izquierdo : guarda el hijo izquierdo de dicho nodo del árbol.
- (-)**Nodo*** hijo_derecho : guarda el hijo derecho de dicho nodo del árbol.
- (-)**char** símbolo : guarda el símbolo en la expresión regular.

2.1.3. Métodos

La clase Nodo cuenta con los siguientes métodos:

- (+)**Nodo*** obtener_hijo_izquierdo(): regresa el hijo izquierdo del nodo.
- (+)**Nodo*** obtener_hijo_derecho(): regresa el hijo derecho del nodo.
- (+)**void** asignar_hijo_izquierdo (Nodo* nodo): asigna el parámetro nodo al hijo izquierdo.
- (+)**void** asignar_hijo_derecho (Nodo* nodo): asigna el parámetro nodo al hijo derecho.
- (+)**char** obtener_simbolo : regresa el símbolo que contiene el nodo
- (+)**void** asignar_simbolo(char sim) : asigna el símbolo sim al atributo símbolo.

2.2. Clase Thompson

2.2.1. Descripción

La clase Thompson implementa el algoritmo de McNaughton-Yamada-Thompson que genera un AFN a partir de una expresión regular.

2.2.2. Atributos

La clase Nodo cuenta con los siguientes atributos:

- (-)**Nodo*** arbol_de_expresion : es la raíz del árbol binario de expresión generado a partir de la expresión regular.
- (-)$map < pair < int, char >, unordered_set < int >>$ transiciones : guarda las transiciones del autómata. el formato es el siguiente map< (estado_actual, símbolo), estado_siguiente >.

2.2.3. Métodos

La clase Nodo cuenta con los siguientes métodos:

- (+)**AutomadaFinitoNoDeterminista** convertir(string expresión_regular): recibe una expresión regular y genera el AFN correspondiente mediante el método de Thompson.
- (-)**int** prioridad(char símbolo): regresa la prioridad según la jerarquía de operadores en expresiones regulares.
- (-)**Nodo*** generar_arbol_de_expresion(Nodo *arbol, string postfijo): genera y devuelve la raíz a un árbol de expresión binario.
- (-)**void** inorden_a_postorden(string infijo, string &postfijo): Convierte la expresión regular de su forma infija a su forma postfija.
- (-)**bool** es_caracter(char símbolo): Regresa verdadero si el carácter enviado es una letra de [a-z], o falso en caso contrario.
- (-)**string** preprocesar_infijo(string &infijo): añade el operador explícito '.' a la expresión regular.
- (-)**pair< pair<int,int>, vector<pair<int,char>>>**: generar_transiciones(Nodo* raiz, int &id): El método generar transiciones genera las transiciones del AFN a partir del árbol binario de de expresión de la expresión regular, regresa { (inicio, final), {transiciones_a_final}} de la presente transición generada.

3. Conclusiones

Al terminar la realización de la práctica se pudo llegar a la conclusión de que el algoritmo es de gran utilidad al momento de construir analizadores léxicos, ya que permite generar un autómata a partir de una expresión regular. En cuanto a la implementación, se llegó a la conclusión de que el tiempo de implementación sobrepasó el tiempo que se esperaba.