Universidad Nacional de Asunción
Facultad Politécnica

Alumno: Nelson Durañona - nelsonds.py@gmail.com
Materia: Diseño de Compiladores
Año: 2013

Descripción del trabajo:

La finalidad del programa es procesar una cadena de entrada , validar que esta sea una expresión regular válida con respecto a su estructura sintáctica y los símbolos disponibles (definidos por el usuario y/o predeterminados).
Obtener el NFA (autómata finito no determinístico) con el algoritmo de Thompson , utilizar esta salida como entrada para el siguiente paso que es obtener el DFA (autómata finito determinístico), renombrar los conjuntos de estados a estados simples para facilitar su lectura y minimizar los estados.
El resultado de cada paso es incluido en un reporte (formato html), incluido el gráfico del DFA mínimo (formato png).
Por ultimo con el conjunto de arcos del DFA mínimo, se genera código fuente en python que valida una cadena recibida como parámetro contra la expresión regular procesada.

Gramática

Se utilizo la siguiente gramática
expr → *concat '|' concat*
expr → *concat*
concat → *rep '.' rep*
concat → *rep*
rep → *atom '*' | atom '+' | atom '?'*
rep → *atom*
atom → *'(' expr ')'*
atom → *char*
char → **simbolo_def_usuario**

La gramática tiene las siguientes características que permitieron la implementación del análisis sintáctico descendente recursivo :
- No tiene recursiones por la izquierda
- Es posible elegir sólo una producción para un símbolo de pre-análisis.
- Cumple con las reglas de las expresiones regulares con respecto a la precedencia y asociatividad de operadores.

Inconvenientes con la gramática

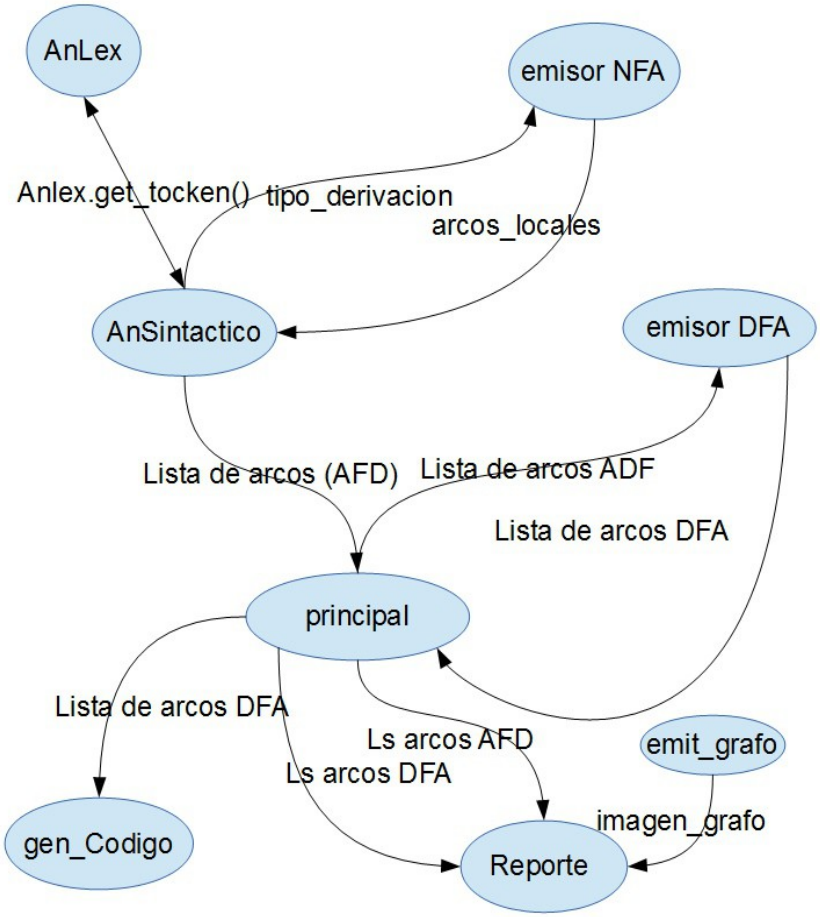
La gramática no reconoce este tipo de cadenas a*b.c , no obstante la cadenas a*.b.c si es reconocida. Explicando informalmente esto se debe a que la gramática *concatena repeticiones,* para resolver el problema de la precedencia y asociatividad por tanto requiere explícitamente el símbolo de concatenación. Sin embargo no es un problema mayor, de hecho Lex trabaja de la misma manera con las expresiones regulares .

Elección del lenguaje

Lenguaje a utilizado : Python
Los algoritmos a implementar requieren operaciones intensivas con conjuntos y listas de elementos, se requiere un lenguaje donde estos tipos de datos y sus operaciones sean built-in.
El interprete interactivo permite probar código antes de incorporarlo en unidades de programa

mayores y facilitar la detección temprana de errores.
Con esos argumentos se eligió el lenguaje.

Diagrama – Funcionamiento general del programa



Estructura de datos principal

La estructura de datos más importante es el arco definido como una tupla de 3 elementos
`x = ( <entero1> ,<entero2> , <caracter> )`
donde :`entero1` representa al estado de origen, `entero2` representa al estado de destino, y `caracter` es el símbolo de transición del estado de origen al estado de destino .
Entonces tanto el DFA como el NFA se definen en términos de listas de este tipo de dato.
`dfa_cualquiera = [ x1, x2, x3..xn]`
`nfa_cualquiera = [ y2, y2, y3..yn]`
Donde `x1,x2,x3` y `y2,y3,y4` son del tipo `( <entero1>,<entero2>,<caracter> )`