Universidad Católica San Pablo

Compiladores

Práctica 3c – Analizadores sintáctico descendente recursivo predictivo (3ra parte)

Objetivo:

- Construir un analizador sintáctico descendente predictivo.
- Escribir una tabla de análisis sintáctico
- Escribir el algoritmo de reconocimiento de una cadena utilizando el analizador.

Requisito:

- Tener la práctica 3b (sobre getPrimero y getSiguiente)

Indicaciones:

1. **Modificar getSiguiente**

   El método de siguiente es necesario para determinar si una gramática es LL(1).

```
def getSiguientes():
        siguientes = {}
        siguientes[nodoInicial] = [dolar]

        for  nodos in getProducciones():
         # Si es que no existiera un nodo a la derecha, su siguiente
         # será el siguiente del lado izquierdo de su producción original.

         # El siguiente de un nodo, es el primero del nodo de su derecha
           # Si entre los elementos primero del nodo de la derecha,
             hubiera el elemento lambda (vacío).
           # Se trata como el caso anterior entonces el siguiente será
             el siguiente del lado izquierdo de su producción original
        return  siguientes;
```

Por ejemplo:

- El siguiente de T sería los primeros de Ep.

- Además de adicionar los primeros de Ep, nos damos cuenta que Ep tiene al elemento *lambda* como primero.
- Entonces, debemos adicionar a los siguientes de T, los siguientes de Ep.
- Así, los siguientes de T serían: {+,-}*(primeros de Ep)* unión {$,)} *(siguientes de Ep)*

2. Crear la tabla de análisis sintáctico

Hacemos una evaluación NoTerminal vs Terminales y buscamos en la gramática, para cada NoTerminal, qué producción originó el elemento no terminal.

```
def crearTabla(self, … ):
    tas =  # AQUI UTILIZAR SU ESTRUCTURA CREADA

    for nodoNt in NoTerminal:
        for nodoT in getPrimero(nodoNt):
            if nodoT   !=  lambda
                 tas[ nodoNt ][ nodoT ] =
                 buscarProduccion( nodoNt, nodoT )
            else:
                for nodoT2 in getSiguiente(nodoNt):
                    tas[nodoNt][nodoT2]  = lambda

            # tas[?][?] = ?   depende de su implementación
```

La función **buscarProducción** ubica la producción desde la cual se originó el elemento nodoT.

Verificar si la tabla creada automáticamente es la misma creada de forma estática en la práctica 3a

3. Implementar algoritmo de validación de cadena

- Para validar una entrada, esta debe ser tokenizada y colocada dentro de una cola. Al final se le debe de adicionar el símbolo de dolar.
- Tener una pila para ver los nodos que se van expandiendo en la validación.
- La idea es ir haciendo match entre los elementos que apuntan la pila y la cola en conformidad con la tabla de análisis sintáctico.

A seguir, implementar el siguiente algoritmo de validación:

```
analiza( cadena ) {
    entrada <- cola(cadena)
    pila <- iniciar una pila vacía

    pila.push( DOLAR )
    pila.push( gramatica->estadoInicial )
    entrada.push( DOLAR )

    mientras( !vacia(entrada) Y !vacia(pila) ) {
        si( entrada.top() = pila.top() ) {
            entrada.pop();
            pila.pop();
        } sino {
            tmp <- pila.pop();
            para  x  en tas[tmp][entrada.top()].reverse()
                si x != "lambda"
                    pila.push(x)
        }
    }

    retorna vacia(entrada) Y vacia(pila)
}
```

4. **Verifique las entradas para:**

num + num + num + num

(num + num) + (num + num)

num * (num * num)

(num *) num