Universidad Católica San Pablo

Compiladores

Práctica 5 – Analizadores sintáctico descendente recursivo predictivo (2da 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:

- Práctica 4 (sobre implementación de una gramática y una TAS)

Indicaciones:

1. **Modificar la clase Gramática para:**
   - Tener una referencia al nodo inicial.
   - Crear una constante llamada **dolar**, que indicará final de cadena.
   - Tener una función que me devuelva todas las producciones ***getProducciones.***

2. **Implementar las operaciones de getPrimero**

   *Los **primeros** serán los nodos terminales que aparecen en primer lugar al analizar la producción del lado derecho.*

   Crear una función que calcule los primeros:

   "El primer nodo terminal a la derecha de la producción"

   Por ejemplo:

   - El primero de Ep es [+, -, lambda]
   - El primero de E es lo mismo que el primero de T, el cual tiene como primero a F y este tiene como primero a [(,num,id]
   - El primero de Tp es [*, /, lambda]

```
def getPrimeros():
```

```
        primeros = {}
        for nodo in noterminales:
            primeros[nodo] = getPrimero(nodo)
        return primeros;

def getPrimero( nodo ):
   #Encontrar todas las producciones donde nodo esté a la izquierda.
   #Para todas ellas verificar el primer nodo de la derecha.
   #Si es nodo terminal, guardarlo como un primero
    #Si es nodo no-terminal, lanzar la llamada de getPrimero a tal nodo.
    # retornar todos los no-terminales primeros.
```

3. **Implementar getSiguiente**

   *Los **siguientes** serían los nodos terminales que se esperan, una vez reconocido el nodo analizado.*

   Crear una función que calcule los siguientes:

```
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:

- Para la primera parte, colocamos $ en el siguiente de E.
- En los nodos de la derecha, solo encontramos una aparición de E.
- El siguiente de E entonces sería el primero del nodo a la derecha de E. Es decir, “)”
- Al final E, tendría como siguientes a [dolar ,)]
- Por ejemplo Ep, no tiene nodo a la derecha. En ese caso, su nodo siguiente será el mismo del nodo E (el que lo originó).
- Entonces el siguiente de Ep sería [dolar,)]

Además:

- 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)*