Instituto Tecnológico de Costa Rica

Departamento de Computación

Sede Alajuela

Curso:

Compiladores e intérpretes

Profesor:

Aurelio Sanabria

Estudiantes:

Carlos Adán Arguello Calderón - 201173805

Jefri Cárdenas Villatoro - 2013101392

Tipo de evaluación:

Laboratorio 2: Parser y AST

Justificación

La justificación de este proyecto es el poder procesar direcciones a la tica a través de mecanismos que permitan verificar si la estructura de una dirección brindada esta correcta con respecto a una gramática establecida. Esto para poder en esta segunda etapa elaborar herramientas que logren identificar errores de Sintaxis que muestren información relevante en caso de existir. Para esto en una primera etapa de Análisis Léxico se obtuvieron Tokens de acuerdo a una gramática, estos se recolectaron de oraciones con una estructura de direcciones a la tica, los Tokens se usarán en esta etapa para corroborar si la dirección y los Tokens están colocados en un orden correcto.

En el Parser se tendrá que verificar si los Tokens recibidos de entrada cumplen con la estructura establecida, para esto se debe comprender como está definida sino de otra manera el Parser no podrá funcionar de la manera esperada. Esta etapa tiene el papel de generar un árbol de sintaxis abstracta que es de utilidad para la etapa final del proyecto en el cual una vez esté completo se utilizará este mismo árbol y se deberá de decorar con detalles que faciliten su manipulación.

Implementación del parser

El parser está implementado de manera que en cada método de la clase *parser* obtiene el valor del siguiente token, verifica que efectivamente corresponda al token esperado, si es así este se encarga de ir agregando los elementos al árbol. Seguido llama al siguiente método a verificar y vuelve a solicitar el siguiente token para comparar, se repite hasta que el programa no tenga más tokens a obtener o que suceda un error en la ejecución que sería que el token recibido no corresponda al token esperado, en ese caso se termina la ejecución, se imprime un mensaje de el error correspondiente.

En caso de finalizar correcta o incorrectamente el árbol será mostrado con la excepción que si falló este estará incompleto.

El siguiente código muestra cómo está implementado el parser y su diseño.

```
def direccionTica(self):
    while(self.nexte() and self.FALLO):
        self.nextt()
        self.ASTLIST = []
        self.TOKENLIST = [""]
        self.ASTLIST.append("Direccion")
        self.inicio()
        if(self.FALLO):
            self.AST.agregar(self.ASTLIST, self.TOKENLIST)
    self.AST.ver()


def inicio(self):
    if(self.TOKEN == "Inicio"):
        self.ASTLIST.append("Inicio")
        self.TOKENLIST.append(self.TOKENVALUE)
        self.nextt()
        self.lugar()
    else:
        self.error("Se esperaba recibir el Token 'Inicio',", self.TOKEN)

def lugar(self):
    if(self.TOKEN == "Lugar"):
        self.ASTLIST.append("Lugar")
        self.TOKENLIST.append(self.TOKENVALUE)
        self.nextt()
        self.detalles()
    else:
        self.error("Se esperaba recibir el Token 'Lugar',", self.TOKEN)

def detalles(self):
    if(self.TOKEN == "Detalles"):
        self.ASTLIST.append("Detalles")
        self.TOKENLIST.append(self.TOKENVALUE)
        self.nextt()
        self.distancia()
    else:
        self.error("Se esperaba recibir el Token 'Detalles',", self.TOKEN)
```

```
def distancia(self):
    if(self.TOKEN == "Distancia"):
        self.ASTLIST.append("Distancia")
        self.TOKENLIST.append(self.TOKENVALUE)
        self.nextt()
        self.medida()
    else:
        self.error("Se esperaba recibir el Token 'Distancia',", self.TOKEN)

def medida(self):
    if(self.TOKEN == "Medida"):
        self.ASTLIST.append("Medida")
        self.TOKENLIST.append(self.TOKENVALUE)
        self.nextt()
        self.conectores()
    else:
        self.error("Se esperaba recibir el Token 'Medida',", self.TOKEN)

def conectores(self):
    if(self.TOKEN == "Conectores"):
        self.ASTLIST.append("Conectores")
        self.TOKENLIST.append(self.TOKENVALUE)
        self.nextt()
        self.cardinales()
    else:
        self.error("Se esperaba recibir el Token 'Conectores',", self.TOKEN)

def cardinales(self):
    if(self.TOKEN == "Cardinales"):
        self.ASTLIST.append("Cardinales")
        self.TOKENLIST.append(self.TOKENVALUE)
        self.nextt()
        self.fin()
    else:
        self.error("Se esperaba recibir el Token 'Cardinales',", self.TOKEN)



def fin(self):
    if(self.TOKEN == "Fin"):
        self.ASTLIST.append("Fin")
        self.TOKENLIST.append(self.TOKENVALUE)
    else:
        self.error("Se esperaba recibir el Token 'Fin',", self.TOKEN)
```

Observaciones

En esta sección se colocarán aspectos relevantes de la elaboración de este laboratorio como lo que son los aspectos positivos y negativos.

Aspectos Positivos

- El parser tiene una forma que permite ir por los métodos verificando los Tokens de una manera más cómoda.
- Tiene una manera fácil de verificar cuál Token hace falta.
- La definición del Parser es sencilla y clara.

Aspectos Negativos

- Al aplicar el parser en los métodos la creación de un objeto de tipo árbol durante la ejecución se hace complicada.
- La estructura del árbol no es la idónea para la implementación de un parser de otro tipo.
- El parser se pudo haber simplificado reutilizando código y evitar duplicar.