**ANALIZADOR LÉXICO,**

**PRIMERA ENTREGA COMPILADORES**

**Realizado por:**

**Luis Miguel Marulanda Torres**

**Angel David Santa Giraldo**

**Juan Camilo Fernández**

**Presentado a:**

**Fernando Galindres**

**Compiladores**

**Ingeniería de sistemas y computación**

**2019-1**

**UTP**

**Analizador Sintáctico**

**Descripción del proyecto**

En este trabajo, se realizó la implantación de un traductor de ast con la capacidad de generar códigos utilizable en graphviz para el lenguaje Mini Java utilizando la librería ply.yacc en el lenguaje de programación Python. Adicionalmente, cabe mencionar que entre las herramientas utilizadas está el analizador léxico y sintactico implementado para la primera y segunda entrega del semestre. Para la realización del proyecto fue necesario proceder de la siguiente manera:

1. Analizar la herramienta Graphviz con el fin de comprender su utilización.
2. Determinar los cambios necesarios en el AST presentado inicialmente para comprender lo que se necesita implementar.
3. Crear un Método de traducción mediante atributos sintetizados.
4. Ejecutar el código con múltiples elementos de prueba.
5. Generar la traducción dirigida por la sintaxis.
6. Mostrar mediante Graphviz la traducción.
7. Personalizar los elementos de la traducción.

**Descripción del proceso del AST:**

Para la construcción del árbol hubo diferentes inconvenientes, en primer lugar, existía una confusión con el proceso de creación de clases, es decir, se creía que había que crear una clase diferente para proceder con cada grupo de elementos comunes en la gramática. Sin embargo, se logro llegar al resultado final de la siguiente manera:

Se identificó que todos los nodos tienen la posibilidad de tener hijos, los cuales podrán imprimirse por conveniencia para mostrar el recorrido deseado. La clase “Node” la definimos con el fin de darle tratamiento similar a las reglas gramaticales, es decir, que se puedan imprimir o desplazar según el caso.

La clase nodo contiene entonces un constructor y un imprimir, en la cual, el constructor reúne los atributos (hijos y tipo) además, tiene un método imprimir el cual procede de la siguiente manera:

Sí el nodo es de clase padre, es decir, tiene hijos, entonces de manera recursiva el método se va llamando, hasta imprimir incluso los nodos que no tienen hijos (que serían hojas).

Se agrego la función traducir para la clase nodo, mediante el cual se puede generar el código para Graphviz, el método funciona mediante atributos sintetizados.

Finalmente, la salida del programa AST, es una cadena con la traducción.

**Retroalimentación**

Para la entrega, se necesitó de una semana de trabajo, en la cual el proceso más demorado fue el de corrección de errores para obtension del AST necesario. La parte del AST fue también compleja, pero más trabajable en el sentido de que se hizo una misma clase para todas las producciones y se tuvieron en cuenta las recomendaciones del profesor. La comprensión del AST fue difícil, y la parte más fácil fue la de poner los resultados en Graphviz.