Instituto Tecnológico de Costa Rica.

Unidad de Computación

Josué Esquivel V.

Katherine Salazar P.

Análisis Contextual Compilador Mini\_C#

Sede San Carlos.

12-octubre-2017.

Introducción

Un compilador es una herramienta fundamental en el área de programación que, a pesar de su importancia y uso frecuente, los estudiantes no conocen a profundidad los detalles del proceso que realiza. El compilador es un programa que se encarga de traducir un código escrito en un lenguaje de programación en alto nivel a un lenguaje de bajo nivel que la máquina pueda ejecutar las instrucciones.

En la etapa anterior se concluyó el análisis sintáctico, que se encarga de analizar la forma en la que el usuario escribe el programa, para esta segunda fase se validará el uso que el usuario le dé a los identificadores del programa. En síntesis, esta fase se debe de validar la existencia de los identificadores y el tipo de los mismo.

Análisis del Problema.

La fase del análisis contextual consiste en validar los identificadores mediante una tabla de símbolos, la cual contiene información pertinente como el tipo, el nombre, el contenido, el nivel donde se declaró y un apuntador. En la etapa anterior se utilizó la herramienta ANTLR4 para el reconocimiento de lenguajes, lo que facilita en este punto la inspección de los identificadores.

Cuando se analicen las expresiones se validará para los números enteros lo siguiente:

* Operadores de aritmética: +, -, \*, / y %.
* Operadores racionales: <, <=,>, >=, ==, !=.

Los operadores == y !=, and (&&) y or (||) se pueden utilizar para cadenas, ambos deben de dar como resultado “True” o “False” según la condición dada.

El programa debe de ser capaz de identificar los posibles errores en un programa, debe de identificarlos mediante la fila y columna. Debe de mostrar el listado con los errores encontrados.

Solución del Problema.

Para completar la tabla de símbolos se tiene que diferenciar cuando se encuentra la declaración de una clase en el código, de ser así se crea una segunda tabla de símbolos con los datos e información que tiene la clase.

Independientemente del caso, la tabla de símbolos se utiliza para identificar el id del dato mediante un método buscar. Con el método insertar se agrega la información obtenida a la tabla de símbolos, el método agrega el token (palabra reservada para este nuevo lenguaje), por ejemplo, para agregar un entero inserta la palabra “int”.

Con ayuda del árbol de sintaxis abstracta se recorre el camino a ejecutar por un conjunto de instrucciones.

Cada regla que ha sido agregada al archivo Parser se desarrollan las validaciones necesarias a cada una en el archivo VisitorAContextual. En este punto es donde se validan las reglas, mediante una secuencia de visitas a las reglas y validaciones de la misma. Las reglas se componen de otras reglas junto con la combinación de algunas de las palabras reservadas escritas en el archivo CScanner.

Si durante la ejecución del código ingresado por el usuario no se cumplen las reglas gramaticales establecidas se le mostrara al usuario en cual línea y columna se genera el error y de qué tipo de error se trata.

A continuación, se muestra el desarrollo de algunas de las reglas.

* methodDecl: tipoMet IDENTIFIER PIZQ (formPars)? PDER (varDecl)\* block

tipoMet: se obtiene el tipo de variable ingresada y se almacena.

IDENTIFIER: se obtiene y almacena el nombre a asignar.

PIZQ: (

(formPars)?: el símbolo ? dice que la regla formPars es opcional, si se llega a necesitar, se ejecuta dicha regla.

PDER: )

(varDecl)\*: el símbolo \* indica que la regla varDecl se puede ejecutar 0 o más veces.

Block: es una regla, es este caso se visita y ejecuta esta llamada.

* numStr: NUMBER | CHAR

Esta regla lo que hace es obtener un dato, un número o un texto. NUMBER y CHAR son palabras asignadas a número y texto respectivamente.

* asignation: ASIGN expr | PIZQ (actPars)? PDER | AUMENTO | DISMINUCION

Esta regla se divide en 4 partes pero solo una es la que se puede generar.

1. ASIGN expr: ASIGN es una palabra reservada que representa “=”.

expr es una regla, por lo tanto se debe de ejecutar esta regla.

1. PIZQ (actPars)? PDER: PIZQ es una palabra reservada que se representa con “(“

(actPars)?: el símbolo ? dice que la regla actPars es opcional, si se llega a necesitar, se ejecuta dicha regla.

PDER es una palabra reservada que se representa con “)”

1. AUMENTO es una palabra reservada que se representa con “++” y quiere decir que se tiene que hacer un aumento de unidad.
2. DISMINUCION es una palabra reservada que se representa con “--” y quiere decir que se tiene que hacer una disminución de unidad.

Análisis de resultados.

|  |  |  |
| --- | --- | --- |
| Tarea | Estado | Observaciones |
| Tabla de Símbolos | Completo |  |
| Control de Niveles de los identificadores | Completo |  |
| Control de re-declaraciones de identificadores | Completo |  |
| Control de existencia de métodos | Completo |  |
| Control de cantidad de parámetros | Completo |  |
| Control de nombres de campos en clases | Completo |  |
| Chequeo de tipos asignaciones | Completo |  |
| Chequeo de tipo en expresiones lógicas | Completo |  |
| Chequeo de tipos en métodos preestablecidos | Completo |  |
| Chequeo de tipos en Expresiones complejas | Incompleto | No se verifica el acceso a las listas |
| Chequeo de tipos en campos de clases | Completo |  |
| Chequeo de tipos en argumentos de métodos | Completo |  |
| Chequeo de tipos en los retornos de funciones | Completo |  |
| Reporte de Errores | Completo |  |

**Conclusiones y recomendaciones**

El proyecto le brinda al estudiante la oportunidad de analizar a fondo, en grandes rasgos, el proceso de un compilador, en específico la fase de análisis de la composición del código. Se logra analizar la validez de los datos ingresados por el usuario al editor de texto web. Esta fase del proyecto se completó el análisis contextual para un lenguaje de programación similar a C#.

La funcionalidad del editor se ve limitada, esto porque se utilizan solo algunas de las sentencias condicionales y funcionalidades del lenguaje de programación C#, aun así, el editor de texto es útil pero básico. El paso a paso de la creación del software queda abierto para el uso e investigación del estudiante, puede ver la esencia del mismo, además de agregar y modificar las funciones para investigación y creación de nuevos lenguajes de programación

Referencias

Para esta fase no fue necesario consultar fuentes externas.