Universidad Católica del Norte
Facultad de Ingeniería y Ciencias Geológicas
Departamento de Ingeniería de Sistemas y Computación

Proyecto de Compiladores

Semantic
CC-A71

PROFESOR: Brian Keith N.
AYUDANTES: Juan Chimaja O. – Fernando Hunt T.
FECHA: Segundo Semestre 2015

Objetivo

Realizar la implementación de un analizador semántico para el lenguaje K*, con la finalidad de verificar la correspondencia de tipos en el programa y verificar errores semánticos simples. No es necesario implementar políticas de recuperación de errores, pero debe encontrar la mayor cantidad de errores posibles, se considerará bueno si se aborta al primer error, pero no se optará a nota máxima en dicho caso (máximo 6). La entrada del programa será el árbol de análisis sintáctico abstracto generado por el *parser*, el cual se examinará y se le agregarán valores correspondientes a la información de tipo de cada nodo. La salida del programa será un árbol sintáctico abstracto aumentado con información de tipos.

Entregables

Se deben entregar los siguientes elementos a más tardar el día 27 de noviembre a las 23.59.

1. **Analizador Semántico.**
   - **a.** Código fuente en lenguaje Java en la IDE NetBeans (SymbolTable, ScopeAnalysisVisitor, TypeCheckVisitor, ExtendedGrapherVisitor, etc).
   - **b.** Siete programas de ejemplo en K* (tres ejemplos buenos y cuatro malos).
   - **c.** Documento .dot con el código generado para la aplicación **graphviz**, al recorrer el árbol de análisis sintáctico abstracto con la información de tipo para cada nodo.
2. **Informe en LaTeX**, utilizar https://www.sharelatex.com/ como herramienta.
   - **a.** Resumen.
   - **b.** Introducción.
   - **c.** Trabajo realizado.
   - **d.** Resultados.
   - **e.** Discusión de los resultados.
   - **f.** Conclusiones.
   - **g.** Referencias.

Evaluación

- Esta fase del taller pesa un 35% de la nota final de taller.
- El informe vale 30% de la nota y el código 70% de la nota.
- Las soluciones se evaluarán en términos de correctitud y claridad.
- Se evaluará el diseño y documentación del proyecto, además.
- Se evaluará la claridad, contenido y estética del informe.
- Si la solución entregada es mucho más complicada que lo mínimo necesario se descontará puntaje, pudiendo incluso llegar a no recibir puntos.
- Soluciones parciales de los ejercicios conseguirán puntajes parciales.
- Se descuenta 0.001 puntos de la nota final por cada minuto de atraso a partir de la hora de entrega.

Análisis Semántico

El análisis semántico se llevará a cabo en dos recorridos (o pasadas) del AST, explicados en detalle a continuación.

Proceso de declaraciones de variables y funciones

Tiene como objetivo construir la tabla de símbolos, que contiene la información asociada a los identificadores usados en el programa, los que pueden ser variables, funciones y parámetros formales de los funciones. La tabla de símbolos debe ser un árbol, que refleja la estructura de bloques anidados del programa, para representar las reglas de visibilidad de los nombres. Por ejemplo, si una función define la variable x y un bloque dentro de la misma función también define x, entonces la x más anidada es visible y oculta la x previa. En cada punto donde se pueden definir variables se crea un contexto que contiene los nombres definidos ahí. Por ejemplo al procesar un programa se crea un contexto que contiene todos los nombres de función y variables globales que se definen en el programa. Al procesar un función se crea un contexto que contiene los parámetros formales del función y que esta enlazado al contexto del programa. Se crean contextos en los siguientes puntos:

1. Al procesar el programa se crea el contexto raíz, que contendrá las variables globales y los funciones.
2. Al procesar una función se crea un contexto que contiene los parámetros formales de la función y las variables locales declaradas en la función.
3. Al procesar un bloque dentro de una función, se crea un contexto que contiene todas las variables locales declaradas en el bloque.

El resultado de este proceso es una tabla de símbolos que contiene todos los nombres usados en el programa y la información de tipos necesaria.

Los errores que se detectan en esta pasada son (puede que alguno no se aplique al lenguaje K*):

1. Tipo de variable no definido.
2. Identificador de variable duplicado.
3. Nombre de función duplicado (el lenguaje K* soporta sobrecarga, utilizando los parámetros de entrada para diferenciar las funciones).
4. Tipo de retorno de la función no declarado.
5. Tipo de parámetro no declarado.

Verificación de consistencia de tipos
En esta se determina el tipo de cada nodo del árbol AST, de acuerdo a las reglas definidas en el **Anexo A del libro "Construcción de compiladores" (Kenneth C. Louden)**.

Al procesar cada nodo del árbol AST, debe determinar los tipos de cada uno de sus nodos dependientes recursivamente, determinando si los tipos de los nodos dependientes son una combinación legal y en caso de serlo determinar el tipo del nodo.

Los errores semánticos de esta pasada son (nuevamente, puede que alguno no se aplique a K*):

1. Invocación de función no definida.
2. Operadores aritméticos requieren operando **int**.
3. Comparación entre tipos no disponibles.
4. No se puede determinar el tipo del lado izquierdo de la asignación.
5. No se puede determinar el tipo del lado derecho de la asignación.
6. Tipo incompatible en asignación.
7. Variable no inicializada.
8. La condición de la iteración debe ser de tipo entero.
9. La condición de la selección debe ser de tipo entero.
10. Invocación de la función incompleta.
11. Tipo de retorno no corresponde al declarado en la función.

Claramente pueden buscar y reportar más errores que los enumerados (dependiendo de la complejidad del error encontrado y la documentación en el código e informe).

Considere además incluir las funciones de entrada o salida consideradas como predefinidas en el ambiente global.

- int input (void) {…}
- void output (int x){…}

La función input no tiene parámetros y devuelve un valor entero desde el dispositivo de entrada estándar. La función output toma un parámetro entero, cuyo valor imprime a la salida estándar unto con un retorno de línea.

Estructura del Proyecto.
La estructura del proyecto debe separar claramente las etapas anteriores del análisis semántico, además se debe distinguir claramente entre las diferentes partes de este último (análisis de alcance y chequeo de tipos), también es necesario considerar aparte todas las estructuras de tablas necesarias (tabla de símbolos y elementos almacenados en ella). El análisis semántico debe implementarlo usando el patrón de diseño *visitor*. Finalmente, considere explicar la estructura de su proyecto en el informe.

Representación Gráfica del AST
Existe una gran cantidad de documentación referente a graphviz, considere investigar el desarrollo de este elemento. Considere realizar pruebas de sus resultados en http://www.webgraphviz.com/, http://sandbox.kidstrythisathome.com/erdos/ o de cualquier otra forma que estime conveniente.

Puntaje Adicional
Se otorgará puntaje adicional (en una cantidad similar) si continúa con la implementación de los elementos adicionales solicitados en el taller anterior de parsing.