DÍA MES AÑO NOTA /

Escuela Politécnica Nacional

Nombre: Fernando Elicer Ibarra Villagómez

El análisis semantico es la etapa del compilador encargada de garantizar que las estructuras del programa no solo sean sintácticamente validas si no semánticamente correctas. Esto incluye comprobación de tipos, verificación de uso de identificadores y control de ámbitos, entre otros.

Propósito del analisis semantico.

Despues del análisis lexico y sintactico el compilador debe asegurarse de que el programa tenga sentido en su contexto.

- Que una variable esté declarada antes de su uso.
- Que una operación entre tipos sea valida
- Que las llamadas a subfunciones tengan el numero y tipo correcto de argumentos.

La verificación de tipos asegura que las operaciones se apliquen a datos compatibles. Es fundamental para detectar errores en tiempo de compilación, evitar comportamientos inesperados y garantizar seguridad en tiempo de ejecución.

- Sistemas de tipos estaticos: la verificación se hace en tiempo de compilación
- Sistemas de tipos dinámicos. los tipos se verifican.

Se estudian técnicas como el análisis hacia adelante y hacia atrás, que implican recorrer el grafo de flujo de control en diferentes direcciones dependiendo del tipo de información que se desea recolectar.
Se representan varios algoritmos para resolver estos problemas de flujo de datos mediante sistema de ecuaciones y técnicas iterativas, además de conceptos como conjuntos IN y OUT, que contienen la información que entra o sale de cada bloque.
Otro punto importante tratado es el papel del dominator tree, una estructura que permite determinar que bloques dominan a otros. Esta información es esencial para ciertas optimizaciones y transformaciones como la traducción de SSA (Static Single Assignment form)
Se destacan algunas optimizaciones específicas que el compilador puede aplicar para mejorar considerablemente el rendimiento del problema Una de ellas es la propagación de copias, que consiste en reemplazar el uso de una variable por su valor original si este no ha sido modificado, lo que puede simplificar expresiones y reducir la cantidad de accesos a memoria o registros. También se incluye la eliminación de código redundante, que detecta y elimina instrucciones que realiza calculos innecesarios o repiten resultados previamente obtenidos permitiendo así que el código sea mas compacto y eficiente. Otra técnica importante es la movilización de código fuera de los bloques, la cual traslada fuera del cuerpo de un bucle aquellas operaciones cuyo resultado no cambia en cada iteración.

Fernando Ibáñez