Traducción – Conceptos teóricos

A grandes rasgos, un traductor es un software que lee un programa escrito en un lenguaje (*lenguaje fuente*) y lo traduce a un programa equivalente en otro lenguaje (*lenguaje objeto*). Como parte importante de este proceso de traducción, el traductor informa a su usuario de la presencia de errores en el programa fuente.

Se usa la traducción cuando se cuenta con un procesador (ya sea hardware o un intérprete) para el lenguaje objeto pero no para el lenguaje fuente. Si la traducción se realiza correctamente, la ejecución del programa traducido dará exactamente los mismos resultados que habría dado la ejecución del programa fuente.

El proceso de traducción se divide en dos fases o etapas:

- Fase de análisis.- La parte del análisis divide al programa fuente en sus elementos componentes y crea una representación intermedia del programa fuente.
- Fase de síntesis.- La parte de la síntesis construye el programa objeto deseado a partir de la representación intermedia.

De las dos partes, la síntesis es la que requiere las técnicas más especializadas, y es la que trataremos en esta sección.

Fase de síntesis

Consiste en generar el código objeto equivalente al programa fuente. Sólo se genera código objeto cuando el programa fuente está libre de errores de análisis, lo cual no quiere decir que el programa se ejecute correctamente, ya que un programa puede tener errores de concepto o expresiones mal calculadas. Por lo general el código objeto es código de máquina relocalizable o código ensamblador. Las posiciones de memoria se seleccionan para cada una de las variables usadas por el programa. Después, cada una de las instrucciones intermedias se traduce a una secuencia de instrucciones de máquina que ejecuta la misma tarea. Un aspecto decisivo es la asignación de variables a registros.

Generación de código intermedio

Después de los análisis sintáctico y semántico, algunos compiladores generan una representación intermedia explícita del programa fuente. Se puede considerar esta representación intermedia como un programa para una máquina abstracta. Esta representación intermedia debe tener dos propiedades importantes; debe ser fácil de producir y fácil de traducir al programa objeto.

La representación intermedia puede tener diversas formas. Existe una forma intermedia llamada "código de tres direcciones" que es como el lenguaje ensamblador de una máquina en la que cada posición de memoria puede actuar como un registro. El código de tres direcciones consiste en una secuencia de instrucciones, cada una de las cuales tiene como máximo tres operandos.

Otra forma de generar código intermedio es mediante código P o código diseñado para una máquina con pila. Esta es la forma que utilizaremos en nuestra práctica.

De acuerdo con la clase de código intermedio (por ejemplo, código de tres direcciones o código P) y de las clases de optimizaciones realizadas, este código puede conservarse como un arreglo de cadenas de texto, un archivo de texto temporal o bien una lista de estructuras ligadas. En los compiladores que realizan optimizaciones complejas debe ponerse particular atención a la selección de representaciones que permitan una fácil reorganización.

Gramática de atributos

La traducción puede especificarse mediante una gramática de atributos, que indique cómo construir un atributo "código" cuyo valor sea el *programa* objeto. Será necesario utilizar la tabla de símbolos para obtener información adicional necesaria para la traducción.