1.7 Manejo de errores semánticos

Es una de las misiones más importantes de un compilador, aunque, al mismo tiempo, es lo que más dificulta su realización. A veces unos errores ocultan otros.

A veces un error provoca una avalancha de muchos errores que se solucionan con el primero.

Es conveniente un buen manejo de errores, y que el compilador detecte todos los errores que tiene el programa y no se pare en el primero que encuentre. Hay, pues, dos criterios a seguir a la hora de manejar errores:

- Pararse al detectar el primer error.
- Detectar todos los errores de una pasada.

El análisis semántico es posterior al sintáctico y mucho más difícil de formalizar que éste. Se trata de determinar el tipo de los resultados intermedios, comprobar que los argumentos que tiene un operador pertenecen al conjunto de los operadores posibles, y si son compatibles entre sí, etc. En definitiva, comprobará que el significado de lo que se va leyendo es válido.

La salida "teórica" de la fase de análisis semántico sería un árbol semántico. Consiste en un árbol sintáctico en el que cada una de sus ramas ha adquirido el significado que debe tener. En el caso de los operadores polimórficos (un único símbolo con varios significados), el análisis semántico determina cuál es el aplicable.

Un error semántico se produce cuando la sintaxis del código es correcta, pero la semántica o significado no es el que se pretendía. La construcción obedece las reglas del lenguaje, y por ello el compilador o intérprete no detectan los errores semánticos. Los compiladores e intérpretes sólo se ocupan de la estructura del código que se escribe, y no de su significado. Un error semántico puede hacer que el programa termine de forma anormal, con o sin un mensaje de error. Hablando en términos coloquiales, puede hacer que el equipo se quede "colgado".

Hay que señalar que los posibles errores ya deben estar considerados al diseñar un lenguaje de programación. Por ejemplo, considerar si cada proposición del lenguaje de programación comienza con una palabra clave diferente (excepto la proposición de asignación, por supuesto). Sin embargo, es indispensable lo siguiente:

El compilador debe ser capaz de detectar errores en la entrada;

El compilador debe recuperarse de los errores sin perder demasiada información;

Y sobre todo, el compilador debe producir un mensaje de error que permita al programador encontrar y corregir fácilmente los elementos (sintácticamente) incorrectos de su programa.

Los mensajes de error de la forma:

*** Error 111 ***

*** Ocurrió un error ***

*** Falta declaración ***

*** Falta delimitador ***

No son útiles para el programador y no deben presentarse en un ambiente de compilación amigable y bien diseñado. Por ejemplo, el mensaje de error 'Falta declaración' podría reemplazarse por

*** No se ha declarado la variable Nombre ***

o en el caso del delimitador omitido se puede especificar cuál es el delimitador esperado. Además de estos mensajes de error informativos, es deseable que el compilador produzca una lista con el código fuente e indique en ese listado dónde han ocurrido los errores.

No obstante, antes de considerar el manejo de errores en el análisis léxico y sintáctico, hay que caracterizar y clasificar los errores posibles. Esta clasificación nos mostrará que un compilador no puede detectar todos los tipos de errores.

Clasificación de Errores

Durante un proceso de resolución de problemas existen varias formas en que pueden surgir errores, las cuales se reflejan en el código fuente del programa. Desde el punto de vista del compilador, los errores se pueden dividir en dos categorías:

Errores visibles y Errores invisibles

Los errores invisibles en un programa son aquellos que no puede detectar el compilador, ya que no son el resultado de un uso incorrecto del lenguaje de programación, sino de decisiones erróneas durante el proceso

de especificación o de la mala formulación de algoritmos. Por ejemplo, si se escribe

a : = b + c ; en lugar de a : = b * c ;

el error no podrá ser detectado por el compilador ni por el sistema de ejecución. Estos errores lógicos no afectan la validez del programa en cuanto a su corrección sintáctica. Son objeto de técnicas formales de verificación de programas que no se consideran aquí. Para conocer más sobre la verificación de programas, consulte, por ejemplo, [LOEC 87].

Los errores visibles, a diferencia de los errores lógico, pueden ser detectados por el compilador o al menos por el sistema de ejecución. Estos errores se pueden caracterizar de la siguiente manera:

Errores de ortografía

Errores que ocurren por omitir requisitos formales del lenguaje de programación.

Estos errores se presentarán porque los programadores no tienen el cuidado suficiente al programador. Los errores del segundo tipo también pueden ocurrir porque el programador no comprende a la perfección el lenguaje que se utiliza o porque suele escribir sus programas en otro lenguaje y, por tanto, emplea las construcciones de dicho lenguaje (estos problemas pueden presentarse al usar a la vez lenguajes de programación como PASCAL y MODULA-2, por ejemplo).