



Unidad V Análisis Semántico

M.C. Juan Carlos Olivares Rojas

Agenda

- 5.1 Analizador semántico
- 5.2 Verificación de tipos en expresiones.
- 5.3 Conversión de tipos.
- 5.4 Acciones agregadas en un analizador sintáctico descendente (top-down).
- 5.5 Pila semántica en un analizador sintáctico ascendente (bottom-up).
- 5.6 Administración de la tabla de símbolos.
- 5.7 Manejo de errores semánticos.



5.1 Analizador semántico

- Ajuste significativo
- Comprobación de tipos: operandos-operadores
- Comprobación del flujo de control:

```
for(;;)  
{  
    ...  
    break;  
    w= a+2;  
}
```



Analizador semántico

- Comprobación de unicidad
- `int a;`
- `char a; //una sola vez`
- Comprobación relacionadas con nombres



Analizador semántico

- Tabla de símbolos:
 - Estructura en memoria
 - Almacena información sobre los tipos
- Sistemas de tipo:
- Tipo básico: entero, carácter, real, lógico
- Nombres de tipo



Analizador semántico

- Constructores de tipo: estructuras, uniones, objetos
- Apuntadores: referencias a tipos
- Funciones `a=suma()`;

- Sistema de tipos: conjunto de reglas que determinan el criterio para asignar expresiones de tipo a las diferentes partes del código fuente



Analizador semántico

- Cada analizador semántico implementa un sistema de tipos
- Comprobación dinámica y estática
- Estática: compilación
- Dinámica: Ejecución

```
char a[5]; strcpy(a, "abcdefghijkl");
```



Analizador semántico

- Fuertemente tipificado
- Débilmente tipificado

Símbolo

```
{  
  nombre;  
  tipo;  
  ámbito;  
}
```



5.2 Verificación de tipos en expresiones

- La verificación de los tipos de datos se hace asignando el valor de tipo de cada una de los componentes léxicos.
- Estos valores se comparan para verificar que los tipos de datos coincidan y sean congruentes, de lo contrario no se pueden realizar los cálculos.



5.3 Conversión de tipos

- Hay situaciones en las cuales se tiene un valor de un tipo dado y se desea almacenar ese valor en una variable de un tipo diferente.
- En algunos tipos es posible almacenar simplemente el valor sin una conversión de tipos; lo que se denomina conversión automática.



Conversión de tipos

- Esto sólo es posible en algún lenguaje de programación, si el compilador reconoce que la variable destino tiene la suficiente precisión para contener el valor origen.
- En Java se puede almacenar un valor byte en una variable int, dado que este tipo de datos es de mayor precisión que el primero.



Conversión de tipos

- A esto se le llama ensanchamiento o promoción, dado que el tipo más pequeño se ensancha o promociona al tipo compatible más grande. Si por el contrario, se desea asignar un valor de variable int a una variable byte se necesita realizar una conversión de tipos explícita.
- En algunos casos se puede realizar la conversión pero se pueden perder datos, como por ejemplo al pasar un valor flotante a un entero.



Conversión de tipos

- A esto se le llama estrechamiento, dado que se estrecha explícitamente el valor para que quepa en el destino.
- La conversión de un tipo se realiza poniendo delante un nombre de tipo entre paréntesis, por ejemplo, (tipo) valor.
- `byte a; int b; a=(byte) b;`



Comprobación de tipos

- Existen dos tipos de comprobación: estática y dinámica.
- La comprobación ayuda a evitar la mayoría de los errores de programación. Ejemplos:
- Comprobación de tipos. Para saber si el operador aplicado a los operadores es correcto



Comprobación de tipos

- Comprobación de flujo de control. Se debe verificar que las instrucciones que cambia el flujo de un programa sean válidos. Ejemplo: break, goto.
- Comprobación de unicidad: definir un objeto una sola vez.
- Comprobación relacionadas con nombres. El mismo nombre debe aparecer dos veces. Variables que se declaran pero no utilizan



Comprobación de tipos

- La comprobación de tipos es la más complicada. Las demás comprobaciones son rutinarias.
- El operador % ocupa que los dos operandos sean enteros.
- + es una función suma(a,b) que está sobrecargada para distintos tipos de datos



Comprobación de tipos

- Siempre se diseñan reglas de tipos como los valores numéricos se convierten al de mayor jerarquía o el tipo de datos punteros sólo apunta al tipo de datos declarado.
- Algunos lenguajes revisan el tamaño de los arreglos (Java) de manera estática otros lo hacen de manera dinámica (en tiempo de ejecución).



Comprobación de tipos

- Diferenciar el uso de +, * enteros que con punteros (aritmética de punteros)
- Al conjunto de reglas que se definen para la comprobación de los tipos de datos se denomina sistema de tipos
- La mayoría de veces la recuperación de errores se suele omitir ya que el programa no finaliza pero tal vez no obtenga los valores deseados



Comprobación de tipos

- Generalmente en la etapa de análisis sintáctico se suelen agregar los tipos a la tabla de símbolos.
- Se revisa el árbol sintáctico para comprobar los tipos asignados.



Comprobación de tipos

- Existen conversiones explícitas en las cuales el usuario indica el tipo de datos
- $a = (\text{int})(23.3/18.2);$
- Las conversiones implícitas requieren de mayor tiempo de ejecución.
- Un ciclo de 1 a N tardó 5.4 y 48.4 nanosegundos utilizando conversiones implícitas.



Comprobación de tipos

- Polimorfismo: una función puede tener el mismo nombre con diferentes elementos. El tipo de datos debe ser diferente.
- Un ejemplo de polimorfismo son las plantillas en algún lenguaje de programación.
- Se debe considerar el ámbito de las variables (locales y globales).



5.4 Acciones agregadas en un analizador sintáctico descendente (top-down)

- Muchas de las actividades que realiza un analizador semántico no son estándares, dependerán del objetivo del lenguaje de programación; por ejemplo, en algunas aplicaciones es interesante conocer que los datos estén en algún rango válido o que ciertos valores se utilicen para uso reservado



Acciones agregadas a un analizador semántico

- En algunas ocasiones nos interesa conocer el significado de las palabras de algún lenguaje dependiendo del contexto (gramáticas de tipo 1) para diferenciar palabras polisemánticas.
- La Web es una base de datos en la mayoría de los casos sin sentidos por lo que la tercera generación de la Web será la llamada Web semántica.



5.5 Pila semántica en un analizador sintáctico ascendente (bottom-up).

- El diseño ascendente se refiere a la identificación de aquellos procesos que necesitan computarizarse con forme vayan apareciendo, su análisis como sistema y su codificación, o bien, la adquisición de paquetes de software para satisfacer el problema inmediato.



Pila semántica

- Los problemas de integración entre los subsistemas son sumamente costosos y muchos de ellos no se solucionan hasta que la programación alcanza la fecha límite para la integración total del sistema.
- Se necesita una memoria auxiliar que nos permita guardar los datos intermedios para poder hacer la comparación.



5.6 Administración de la tabla de símbolos

- La tabla de símbolos también recibe el nombre de ambiente. Un ambiente contiene un conjunto de parámetros que sólo son visibles en ese ambiente.
- La tabla de símbolos se mantiene durante todo el proceso de traducción agregando elementos específicos en cada paso.



Operaciones sobre la tabla de símbolos

- Inserta(símbolo)
- Existe(nombre)
- Tipo(nombre)

- Declaración → TIPO {tipo=obtengo(yytext());}
listavar PYC
- Listavar → var {inserta(símbolo);} | var {inserta(simbolo);}
- Var → ID {simbolo=yytext; símbolo.tipo=tipo; simbolo.amb=ambito;}



Operaciones sobre la tabla de símbolos

- Exprlog → PI exprlog {A=A;} PD
- |NOT exprlog {A=A;}|
- |exprlog {A1=A;} OPLOG exprlog {A2=A
- If(A1==INT && A2==INT)
- A=INT;
- Else
- A=ERROR_TIPO;}



5.7 Manejo de errores semánticos

- Los errores semánticos son pocos y los que existen no se pueden detectar tan fácilmente.
- Hasta esta etapa los errores son mostrados a los usuarios. Los demás errores ya son muy difíciles de detectar y generalmente se dan en tiempo de ejecución



Manejo de errores semánticos

- Algunos problemas se presentan durante la fase de gestión de memoria al pasar argumentos o al crear la pila semántica.
- Muchos errores se generan durante la etapa del enlazador, al tratar de obtener código existente de algunas funciones/métodos ya implementadas en bibliotecas/APIs



¿Preguntas?

