Construcción de Compiladores
Reglas Semánticas y Tipos
Luis Pedro Gonzalez
José Mariano Reyes

Consideraciones Semánticas y de Tipos

Se usará la gramática YAPL como base para las reglas semánticas y de tipos en la que nos basaremos en un futuro para poder realizar el compilador. Empezaremos con las reglas semánticas.

Reglas Semánticas:

Reglas semánticas para expresiones:

- Las expresiones aritméticas (+, -, *, /) sólo se permiten entre operadores de tipo Int.
- Las expresiones de comparación (<, <=, =) se permiten entre operandos de tipo Int o Bool.
- Las expresiones booleanas (and, or, not) sólo se permiten entre operandos Bool.
- El tipo estático de una expresión debe coincidir con el tipo declarado de la variable o atributo en una asignación.

Reglas semánticas para declaraciones:

- No se permite redeclarar una variable con el mismo nombre dentro del mismo scope o alcance.
- Las variables deben ser declaradas antes de usarse.
- Los atributos de una clase no se pueden redefinir en clases derivadas.

Reglas semánticas para métodos:

- El número y tipos de los parámetros actuales en una llamada a un método deben coincidir con la declaración del método.
- El tipo de retorno de un método debe coincidir con el tipo declarado en su definición.
- Si una clase derivada define un método con el mismo nombre que en su clase base, la firma (parámetros) debe ser la misma.

Reglas semánticas para clases:

- Las clases sólo se pueden heredar de una única clase base. No se permite herencia múltiple.
- No se permiten ciclos en la jerarquía de herencia de clases.
- Los métodos de la superclase son accesibles en la subclase, a menos que sean redefinidos.

Reglas semánticas para tipos:

- Los tipos de las expresiones en ambos lados de una asignación deben ser compatibles.
- Una variable sólo puede ser asignada si su tipo declarado es el mismo o subtipo del tipo de la expresión asignada.

Estas son las principales reglas semánticas que se pueden definir para el lenguaje YAPL a partir de la gramática YAPL usada para estos laboratorios.

Tipos de Datos

Aquí están los principales tipos de datos que se pueden definir para el lenguaje YAPL según la gramática:

Tipos Básicos:

- Int - Enteros con valores por default 0. No se puede heredar.
- Bool - Booleanos con valores true o false. Valor por default false. No se puede heredar.
- String - Cadenas de texto. Valor por default "". No se puede heredar.

Tipos de Clases:

- Las clases definidas por el usuario generan nuevos tipos.
- Object - Clase base de la jerarquía de herencia. Define métodos como abort(), type_name() y copy().
- IO - Define métodos para entrada/salida como out_string(), out_int(), in_string(), in_int(). No se puede redefinir.
- List - Clase definida por el usuario como ejemplo. Puede tener subclases.
- Cons - Clase derivada de List. Ilustra herencia.

Tipos Especiales:

- SELF_TYPE - Se usa para referir al tipo de la clase actual dentro de sus métodos. Reemplazado por el tipo de clase real en tiempo de compilación.

Reglas de Subtipado:

- Si B deriva de A, B es subtipo de A.
- Los objetos de la subclase pueden usarse donde se espera un objeto de la superclase.

Tabla de Símbolos

La tabla de símbolos almacena información sobre los identificadores utilizados en el programa como nombres de variables, métodos, clases, etc.

Cada entrada de la tabla de símbolos contendrá:

- Nombre del identificador
- Tipo del identificador
- Ámbito/Scope (global, local, parámetro, etc)
- Valor
- Tipo semántico
- Parámetros

Operaciones principales de la tabla de símbolos(Sujetas a cambios en código):

- Insertar(nombre, tipo, ámbito, posición) - Agrega una nueva entrada
- Buscar(nombre) - Busca una entrada por nombre y devuelve información
- IniciarAmbito() - Inicia un nuevo ámbito anidado
- FinalizarAmbito() - Finaliza el ámbito actual

Al insertar un nuevo identificador se verifica:

- Que no exista previamente en el ámbito actual
- Que los tipos de las expresiones sean compatibles si se redeclara

La tabla permite detectar errores como:

- Uso de variables no declaradas
- Redeclaración inválida

- Incompatibilidad de tipos

La tabla de símbolos se utiliza en cada fase del compilador para registrar la información de cada identificador y verificar la semántica del programa.