UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COLEGIADO DE CIÊNCIA DA COMPUTAÇÃO

Aspectos semânticos-Compiladores

Alunos: Vitor Gilnek / Gabriel Medina Data: 28/07/2022

Aspectos Semânticos da Linguagem Upper

Upper é uma linguagem fortemente tipada baseada na sintaxe da linguagem C, onde resumidamente podemos definir:

- Tipos de dados: A linguagem possui os tipos Inteiro, Ponto Flutuante e Booleano.
- Operadores: Possui operadores lógicos, aritméticos e relacionais.
- Estruturas: Dispõe das estruturas de repetição, como if e else, salto, como de exemplo o while e por último mas não menos importante a própria estrutura principal MAIN.
- Atribuições: Consiste de uma operação binária do tipo “a=b”, de forma com a variável da esquerda receba o valor da direita.

Com isso em mente é possível afirmar que a linguagem Upper permite atribuições e comparações desde que as variáveis sejam do mesmo tipo, a tornando uma linguagem fortemente tipada. Desta forma, temos que ela só aceitaria comparações como, inteiros com inteiros, booleanos com booleanos e assim por diante, de maneira que as variáveis declaradas no início continuem com o mesmo tipo durante a execução. Abaixo é possível ver diversos exemplos de atribuições válidas dentro de um programa em UPPER.

- BOOL x = True
- BOOL y = x
- INT i = 42
- INT j = 8 + i
- INT k = j
- FLOAT a = 3.141
- FLOAT b = 1.523 - a
- FLOAT c = b

Pensando nas comparações, esta regra se aplica também, tanto para o if, quanto para o while, onde neles será comparada apenas variáveis de mesmo tipo, como por exemplo:

- IF(j > i)
- WHILE (a != b)
- IF (c < a)

- IF (!x)

A linguagem UPPER permite estes tipos de comparações, entretanto deve-se atentar ao fato de que não pode haver variáveis que compartilham do mesmo nome; variáveis que não foram declaradas anteriormente e são utilizadas no programa resultarão em erro, já que só é possível utilizá-las depois de declarar elas.

As variáveis declaradas dentro da função poderão ser utilizadas apenas dentro da função onde houve esta declaração; já variáveis que tenham sido declaradas dentro do programa, mas que porventura não tenham sido utilizadas durante a execução, resultará em um warning que possibilitará o desenvolvedor se atentar a este detalhe.

Abaixo é possível ver um exemplo que seria validado pela análise semântica:

```
MAIN
{
            INT c;
            INT a = 10;
            INT b = 2 * 10;

            INT var = 3 + 1 / 2 + 1 * 2;
            INT var = 1;

            BOOL v = True;

            IF (var < 10) {
                IF (var == 5) {
                    v = False;
                }
                var = var + 1;
            }
        }
```

Como um exemplo de um programa que não seria validado pela análise semântica temos:

```
MAIN
{
            INT c;
            INT a = 10;
            FLOAT b = 1.15 * a;
}
```

Este programa exibiria um warning para “c” que não está sendo utilizado e uma mensagem de erro devido a operação entre a variável “a” e “b”.

Para a análise semântica são seguidos os seguintes passos:

- construção de tokens com profundidade (essa profundidade é usada para saber qual o contexto e onde esta variável seria utilizada)

- construção de uma tabela de variáveis
    - verifica se variáveis são únicas
- verificação de atribuições
- verificação de condições dentro dos if e while
- verificação de variáveis inúteis

Com uma GLC bem definida não há necessidade de se preocupar com erros léxicos e sintáticos, já que essas receberam um tratamento de erro prévio, assim cabendo a análise semântica fazer estas operações.