Projeto de Compilador
E4 para **Análise Semântica**

Prof. Lucas Mello Schnorr
schnorr@inf.ufrgs.br

1 Introdução

A quarta etapa do trabalho de implementação de um compilador para a linguagem consiste em verificações semânticas. Elas fazem parte do sistema de tipos com regras simples. A verificação de tipos é feita em tempo de compilação. Todos os nós AST (E3) terão agora um campo que indica o tipo de dado. O tipo de dado de um determinado nó da AST é definido a partir da natureza do nó ou a partir dos tipos de dados dos nós filhos usando as regras de inferência da linguagem.

2 Funcionalidades Necessárias

2.1 Tabela de símbolos

A tabela de símbolos guarda os símbolos já declarados em um escopo. Cada entrada na tabela possui uma chave e um conteúdo. A chave única é o lexema do símbolo e o conteúdo está descrito a seguir. A implementação deve prever que várias tabelas podem coexistir ao mesmo tempo, uma para cada escopo.

1. Defina uma estrutura para ser o conteúdo de uma entrada da tabela. Essa estrutura possui os seguintes campos: localização (linha), natureza (identificador ou função), tipo do dado do símbolo, dados do valor do token pelo `yylval` (veja E3).
2. Defina uma estrutura para ser uma tabela de símbolos. Essa estrutura pode ser uma lista de estruturas de entrada, como definida no ponto anterior.

2.2 Escopo local e aninhado

O escopo pode local da função e local de um bloco, sendo que este pode ser recursivamente aninhado. Utilize uma tabela de símbolos por escopo, usando uma pilha de tabelas para manter o controle de qual tabela está ativa em um instante da análise sintática.

1. Defina uma pilha de tabelas de símbolos. Inicialize-a vazia pois no início da análise sintática não temos escopo definido. Ao entrar em um escopo (de função ou de bloco de comandos), inicialize uma nova tabela de símbolos e a empilhe. Ao sair de um escopo, desempilhe a tabela do topo da pilha. As verificações e inserções na tabela estão descritas a seguir.

2.3 Verificação de declarações

Todos os identificadores devem ter sido declarados no momento do seu uso, seja como variável, seja como função. Deve-se também verificar que não houve dupla declaração ou se o símbolo não foi declarado. Caso o identificador já tenha sido declarado, deve-se lançar o erro `ERR_DECLARED`. Caso o identificador não tenha sido declarado no seu uso, deve-se lançar o erro `ERR_UNDECLARED`.

1. Nas regras gramaticais que envolvem o uso de um identificador, implemente uma ação para verificar se esse identificador já foi declarado. Verifica-se primeiramente no escopo atual (topo da pilha) e enquanto não encontrar, deve-se descer na pilha (sem desempilhar) até chegar no escopo que encontra-se na base da pilha, sempre presente. Caso o identificador não seja encontrado após este procedimento, temos a evidência que ele não foi declarado e portanto emitimos um erro semântico (`ERR_UNDECLARED`).
2. Nas regras gramaticais que envolvem a declaração de identificadores (declaração de variáveis locais e definição de funções), implemente uma ação para verificar se esse identificador já foi declarado no escopo atual (aquele que está no topo da pilha). Caso encontrá-lo, emitimos um erro semântico (`ERR_DECLARED`). Caso não encontrado, inserimos o mesmo na tabela corrente (topo da pilha) com seu tipo de dado.

2.4 Uso correto de identificadores

O uso de identificadores deve ser compatível com sua declaração e com seu tipo. Variáveis somente podem ser usadas em expressões e funções apenas devem ser usadas como chamada de função, isto é, seguidas da lista de argumentos. Caso o identificador que é uma variável seja usado como uma função, deve-se lançar o erro `ERR_VARIABLE`. Enfim, caso o identificador que é uma função seja utilizado como variável, deve-se lançar o erro `ERR_FUNCTION`.

1. Nas regras gramaticais que envolvem o uso de um identificador, implemente uma ação para procurar a sua natureza na pilha de tabela de símbolos. Caso a sua natureza no contexto da regra gramatical seja diferente da natureza encontrada na tabela de símbolos, considere a natureza da tabela de símbolos para emitir o erro (`ERR_VARIABLE`) caso a natureza na declaração do identificador for uma veriável; ou (`ERR_FUNCTION`) caso a natureza na declaração do identificador for uma função.

2.5 Verificação de tipos na AST

Uma declaração de variável ou definição de função permite ao compilador definir o tipo de dado. Esse registro é feito nos nós da AST correspondentes. Um nó da AST deve ter portanto um novo campo que registra o seu tipo de dado. Faça a inferência de tipos de dados em expressões (aritméticas, lógicas e relacionais), desconsiderando chamadas de funções, e no comando de atri-

buição. As regras de inferência de tipos da linguagem são as seguintes. (`int`, `int`) → `int`; (`float`, `float`) → `float`; (`bool`, `bool`) → `bool`; (`float`, `int`) → `float`; (`bool`, `int`) → `int`; (`bool`, `float`) → `float`. O tipo de dado da atribuição é determinado pelo tipo de quem recebe o valor atribuído.

1. Nas regras gramaticais que envolvem expressões aritméticas, defina o tipo de dado da operação aritmética a partir dos seus operandos seguindo as regras de inferência. Caso um operando seja um identificador, obtenha seu tipo procurando-o na tabela de símbolos. Caso um operando seja outra expressão, obtenha seu tipo do campo específico do seu nó da AST.

2.6 Mensagens de erro

Mensagens de erro significativas devem ser fornecidas. Elas devem descrever em linguagem natural o erro semântico, as linhas envolvidas, os identificadores e a natureza destes de uma maneira que o usuário do seu compilador compreenda o erro semântico.

A Códigos de retorno

Os seguintes códigos de retorno devem ser utilizados quando o compilador encontrar erros semânticos. O programa deve chamar `exit` utilizando esses códigos imediamente após a impressão da linha que descreve o erro. Na ausência de qualquer erro, o programa deve retornar o valor zero.

```
#define ERR_UNDECLARED       10 //2.3
#define ERR_DECLARED         11 //2.3
#define ERR_VARIABLE         20 //2.4
#define ERR_FUNCTION         21 //2.4
```

Estes valores são utilizados na avaliação objetiva.