UFRGS - INF01147 - Compiladores - 2013/1
Turma - Prof. Marcelo Johann

Trabalho Prático - Especificação da Etapa 4:
Verificação Semântica

Resumo:

Na quarta etapa do trabalho de implementação de um compilador para a linguagem **ling** é necessário fazer verificação semântica, ou seja, uma série de testes de coerência comportamental das construções sintáticas reconhecidas e representadas na AST.

Visão Geral das Verificações:

a. Declarações – Devem ser verificadas as declarações. Todos os identificadores devem ter sido declarados, seja como variável, ponteiro, vetor ou como função. Os símbolos definidos como identificadores na tabela de símbolos devem ter seu tipo trocado para um desses tipos conforme a declaração, verificando-se se não houve dupla declaração ou símbolo não declarado;
b. Uso correto – o uso dos identificadores deve ser compatível com sua declaração. Variáveis somente podem ser usadas sem indexação, vetores somente podem ser usados com indexação, e funções apenas devem ser usadas com chamada, isto é, seguidas da lista de argumentos entre parênteses;
c. Tipos de dados – As declarações também devem registrar os tipos de dados, em um novo campo, dataType, na tabela de símbolos. Com o auxílio dessa informação, quando necessário, os tipos de dados corretos devem ser verificados onde forem usados, em expressõs aritméticas, relacionais, lógicas, ou para índices de vetores;
d. Argumentos e parâmetros – A lista de argumentos deve ser verificada contra a lista de parâmetros formais na declaração da função. Cada chamada de função deve prover um argumento para cada parâmetro, e ter o tipo compatível;

Definições Semânticas:

- Há quatro tipos diferentes de identificadores:
  escalares, vetores, ponteiros, funções
- Há três tipos de dados para declarações:
  word, byte, bool
- Há quatro tipos de literais:

inteiros, booleanos, caracteres, strings

- Literais string só devem ser usados no comando output.
- Literais inteiros e caracteres são intercambiáveis, e podem aparecer em quaisquer expressões aritméticas, e serem atribuídos a dados numéricos (word, byte). Não é necessário verificar o tamanho (word ou byte) nas expressões e atribuições - isso não deve gerar nem erro nem warning.
- Ponteiros podem aparecer em expressões, em operações como "pt+1, por exemlpo. Neste caso, pt deve ser um ponteiro e o resultado somente pode ser atribuído a um ponteiro, ou usado como ponteiro (de-referenciação). Há, portanto, dois tipos de verificações a serem feitas em expressões: booleanos versus aritméticos, e aritméticos versus ponteiros. Em ambos os casos, existe a possibilidade da identificação ser feita pelo tipo do nodo filho da árvore (tipo do operador) ou pelo tipo de dado (dataType) do identificador, se o nodo filho é um identificador (AST_SYMBOL). A diferença está no fato de que as operações booleanas podem ser verificadas somente no nodo local da árvore, em relação aos seus filhos. Já para verificar os ponteiros, é necessário que a árvore seja previamente percorrida das folhas até a raiz, anotando o tipo de dado correto nos nodos. Isso porque um operador + pode resultar tanto em um escalar inteiro como em um ponteiro, eisso só é possível descobrir e verificar recursivamente à partir das folhas. Por exemplo, seja: word a; word $pt; Então a = 1 + 2 + 3 está correto, enquanto a = 1 + 2 + pt está errado..

Perguntas e Respostas

- Devemos verificar o tipo dos ponteiros? Não, por simplicidade
  * Se o programador quiser, pode fazer wordpt = bytept;
  ** Mas quando o programador não faz nada errado, o compilador nao pode fazer nada errado. Portanto, devemos manter o dataType correto do ponteiro;
- Posso somar ponteiro com ponteiro? Não! Soma pode ser de dois aritméticos ou de um aritmético e um ponteiro;
- Posso subtrair um ponteiro? Não, por simplicidade;
- Posso multiplicar um ponteiro? Não, por simplicidade;
- Posso dividir um ponteiro? Não, por simplicidade;
- Posso fazer comparações (relacionais) de ponteiros? Não;

Lista de Verificações

- variáveis redeclaradas
- anotar tipo (natureza) nas tabela hash
- anotar tipo de dado (dataType) na tabela hash
- variáveis não-declaradas
- verificar natureza, se:
  - escalares são usados como escalares
  - vetores são usados como vetores
  - funções são usadas como funções
  - [ponteiros são usados como ponteiros]

    - não esqueça de verificar no lado direito (expressões) e no lado esquerdo (atribuições)
- verificar tipos de dados nas expressões
- verificar argumentos de chamada de função versus parâmetros:
    - não pode haver menos argumentos
    - não pode haver mais argumentos
    - os tipos devem ser compatíveis (não iguais, lembre-se)
- verificar tipo do valor de retorno da função
- verificar índices dos vetores (não pode ser bool ou ponteiro), tanto na expressão quanto na atribuição

Controle e organização do seu código fonte

Você deve seguir as mesmas regras das etapas anteriores para organizar o código, permitir compilação com **make**, permitir que o código seja rodado com **./etapa4**, e esteja disponível como **etapa4.tgz**.

Porto Alegre, 16 de Maio de 2013