Construção de Compiladores
Daniel Lucrédio, Helena Caseli, Mário César San Felice e Murilo Naldi
Tópico 07 - Análise Semântica - Lista de Exercícios Resolvida
(Última revisão: fev/2020)

1) Diga quais são as três principais operações na Tabela de Símbolos, explique o que vem a ser cada uma delas e dê exemplos de momentos nos quais elas ocorrem.

R.
- **Inserção**: armazena informações fornecidas pelas declarações. Ocorre principalmente no momento de declaração de elementos, mas também pode-se considerar a atualização de valores para um elemento já inserido como parte da "inserção" e, assim, em um comando de atribuição, por exemplo, o valor da variável à esquerda de ":=" é inserido na Tabela de Símbolos na linha desse elemento e coluna "valor".
- **Busca**: recupera informações associadas a um elemento declarado no programa quando esse elemento é utilizado. Ocorre antes da inserção de um elemento na Tabela de Símbolos para verificar se o mesmo já foi declarado previamente e toda vez que um elemento é acessado, seja para verificar seu escopo, seu tipo ou outra informação relevante para a computação em questão.
- **Remoção**: remove (ou torna inacessível) a informação a respeito de um elemento declarado quando esse não é mais necessário. Ocorre ao final da execução de um procedimento para a remoção de suas declarações locais (procedimentos, variáveis) uma vez que essas não serão mais necessárias.

2) Quais são as duas principais regras que envolvem a definição de escopo estático em uma linguagem de programação?

R.
- **Declaração antes do uso**: as variáveis devem ser declaradas, para poderem ser utilizadas.
- **Aninhamento mais próximo**: quando existem dois ou mais escopos sobrepostos, aquele com maior nível de aninhamento tem preferência na definição dos nomes.

3) Quais são as duas principais opções para fazer a análise semântica de diferentes escopos utilizando tabela de símbolos? Quais as vantagens e desvantagens de cada uma?

R.
- Para todos os escopos, existe uma única tabela, onde cada entrada é uma pilha de símbolos. Símbolos no topo dessa pilha são aqueles que estão ativos em um determinado escopo. Assim que um escopo se encerra, os símbolos associados a esse escopo são removidos de suas respectivas pilhas. Nessa opção, a busca é mais fácil, pois as variáveis encontram-se todas em uma única pilha. No entanto, a inserção e remoção exigem uma varredura completa.
- Existe uma pilha de tabelas, e cada escopo gera uma nova tabela. A tabela no topo da pilha representa o escopo mais próximo. Nessa opção, a inserção e remoção são mais fáceis, no entanto, a busca exige navegar em várias tabelas.

4) Quais são as diferenças entre realizar análise semântica por meio de ações semânticas inseridas na própria gramática e o uso de um visitante? Quais as vantagens e desvantagens de cada abordagem?

R:
- Ao inserir ações semânticas na própria gramática, um gerador automático pode produzir um parser que realiza as ações durante a própria análise sintática. Ou seja, o analisador vai analisando a semântica AO MESMO TEMPO em que analisa a sintaxe e cria a árvore. Em contrapartida, um visitante atua sobre uma árvore já pronta, APÓS a análise sintática ter sido concluída.
- As vantagens de realizar análise semântica na própria gramática são: a) abordagem mais simples e direta. b) as ações ficam visíveis dentro da gramática, o que pode facilitar seu entendimento em casos mais simples, ajudando na sua criação/manutenção
- As desvantagens de realizar análise semântica na própria gramática são: a) as ações são executadas durante a análise sintática, portanto é necessário posicioná-las em local correto para que sejam executadas após o código necessário ter sido analisado. b) pelo mesmo motivo acima, é impossível, a partir de uma ação semântica, acessar trechos de código que só serão analisados mais para a frente (ex: um método que só foi declarado no final do arquivo não estará na tabela de símbolos no começo do arquivo). c) caso haja muitas ações, a gramática poderá se tornar ilegível, dificultando o trabalho. d) fica difícil realizar duas tarefas distintas sobre a mesma gramática, pois o código estará misturado (ex: durante a declaração de variáveis, é preciso manipular a tabela de símbolos e gerar código, ambas as ações ficarão misturadas na gramática).
- As vantagens de realizar análise semântica em um visitante são: a) preserva a legibilidade da gramática, mantendo-a em um arquivo separado. b) possibilita o acesso a qualquer parte do programa a qualquer momento, já que a árvore já está toda pronta. c) possibilita a criação de múltiplos visitantes, cada um com uma função diferente (ex: um visitante para verificar tipos, um visitante para analisar consistências diversas, um visitante para gerar código, etc). o que deixa o código mais modularizado e fácil de se manter.
- As desvantagens de realizar análise semântica em um visitante são: a) a necessidade de realizar duas (ou mais) análises distintas em momentos separados, o que pode deixar a compilação menos eficiente. b) pode causar duplicação de código (ex: tanto a análise de tipos como a geração de código dependem da tabela de símbolos, portanto pode haver código de manipulação da tabela de símbolos duplicado em dois visitantes). É possível usar herança e modularização para resolver os principais problemas, no entanto, mas é preciso cuidado. c) a associação entre as ações e a gramática é menos visível, visto que as regras sintáticas e as ações semânticas estão em arquivos separados.

5) Dada a seguinte gramática para reconhecer expressões aritméticas segundo o formato ANTLR

```
programa: expressao;
expressao: termo (op1 termo)*;
termo: fator (op2 fator)*;
fator: '(' expressao ')' | NUM;
op1: '+' | '-';
op2: '*' | '/';
NUM: '0'..'9'+;
WS: ( ' ' | '\n' | '\r' | '\t' ) -> skip;
```

Adicione ações semânticas de forma a contar quantos operadores aparecem em uma expressão.

R. (em vermelho abaixo)

```
Programa: expressao
{ System.out.println("Número de operadores:"+$expressao.cont); };
```

```
expressao returns [ int cont ]:
    t1=termo {$cont = $t1.cont;}
    (op1 t2=termo
        {$cont += $t2.cont + 1;}
    )*;
termo returns [ int cont ]:
    f1=fator
    {$cont = $f1.cont;}
    (op2 f2=fator
        {$cont += $f2.cont + 1;}
    )*;
fator returns [ int cont ]:
    '(' expressao ')' {$cont=$expressao.cont;} | NUM {$cont = 0;};
op1: '+' | '-';
op2: '*' | '/';
NUM: '0'..'9'+;
WS: ( ' ' | '\n' | '\r' | '\t' ) -> skip;
```

6) Dada a seguinte gramática para reconhecer expressões aritméticas segundo o formato ANTLR:

```
programa: expressao;
expressao: termo (op1 termo)*;
termo: fator (op2 fator)*;
fator: '(' expressao ')' | NUMINT | NUMREAL;
op1: '+' | '-';
op2: '*' | '/';
NUMINT: '0'..'9'+;
NUMREAL: '0'..'9'+ '.' '0'..'9'+;
WS: ( ' ' | '\n' | '\r' | '\t' ) -> skip;
```

Adicione ações semânticas de forma a determinar o tipo de uma expressão, conforme as seguintes regras:

1. operações de soma, subtração e multiplicação entre dois inteiros resultam em uma expressão inteira
2. operações de divisão envolvendo dois inteiros resultam em uma expressão real
3. qualquer operação que envolva um real (de qualquer um de dois lados) resulta em uma expressão real

R. (em vermelho abaixo)

```
grammar Expressoes;

programa returns [ String tipo ]:
    expressao
    { $tipo = $expressao.tipo; }
;
expressao returns [ String tipo ]:
    t1=termo
    { $tipo = $t1.tipo; }
    (
        op1 t2=termo
```

```
        { if($t2.tipo.equals("REAL")) $tipo = "REAL"; }
    )*
;
termo returns [ String tipo ]:
    f1=fator
    { $tipo = $f1.tipo; }
    (
        op2 f2=fator
           { if($f2.tipo.equals("REAL") || $op2.text.equals("/")) $tipo =
"REAL"; }
    )*
;
fator returns [ String tipo ]:
    '(' expressao ')' { $tipo = $expressao.tipo; } |
    NUMINT { $tipo = "INTEIRO"; } |
    NUMREAL { $tipo = "REAL"; }
;
op1: '+' | '-';
op2: '*' | '/';
NUMINT: '0'..'9'+;
NUMREAL: '0'..'9'+ '.' '0'..'9'+;
WS: ( ' ' | '\n' | '\r' | '\t' ) -> skip;
```