UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
DISCIPLINA: Compiladores I
Trabalho Prático 3 – Compilador 2012-1 – Análise Léxica

Pedro Araujo Pires
Pedro Henrique Doffemond Costa

Introdução

A terceira fase da compilação, chamada análise semântica, é onde são verificados os aspectos semânticos (por exemplo a verificação de tipos, a verificação do fluxo de controle e a verificação da unicidade da declaração de variáveis) do código fonte, além de coletar informações necessárias para a próxima fase da compilação, que é a geração de código intermediário. Os erros detectados pela análise semântica não interferem com o andamento da compilação, sempre gerando um *warning*.

Objetivo

Implementar a análise semântica sobre o analisador sintático para a gramática da linguagem L2012-1, desenvolvido no trabalho anterior. O analisador semântico deve fazer a verificação de tipos em expressões e comandos.

Ferramentas

As ferramentas utilizadas neste trabalho foram o Lex para gerar o analisador léxico, e o YACC para gerar o analisador sintático,

Análise Semântica

A ferramente YACC, ao mesmo tempo em que gera um analisador sintático para uma determinada gramática, também provê formas de se fazer a análise semântica do código fonte. Sempre que há uma redução durante a análise sintática, o YACC permite a execução de ações, que são comandos em C. Além disso, também é permitido definir atributos para os símbolos não terminais da gramática, o que permite que sejá feita a verificação de tipos.

Nos trabalhos anteriores, o código da tabela de símbolos estava no arquivo de entrada para o Lex. Para uma melhor modularização do compilador, o código foi transferido para um arquivo separado (sym_table.c). Além disso, todo o código relacionado com a checagem de tipos está no arquivo type_checker.c.

O tipo das expressões foi determinado a partir do tipo de seus operandos. Quando dois operandos têm o mesmo tipo, o resultado da operação também possui esse tipo. Caso dois operandos não possuam o mesmo tipo, uma mensagem de aviso é impressa na tela. Quando há uma chamada de função, o tipo do(s) argumento(s) é checado. Caso seja uma função *built-in* da linguagem, o tipo é

checado de acordo com as convenções de tipos especificadas na documentação da linguagem Pascal. Se a função sendo chamada for um função declarada pelo programador, o tipo e a quantidade de parâmetros são checados. Os comandos de controle de fluxo `if` e `repeat` também são checados, pois ambos possuem expressões obrigatoriamente precisam ser boolean.

Modo de Utilização

O utilitário make foi utilizado para fazer a geração e a compilação dos analisadores léxico, sintático e semântico. Para gerar o executável basta executar o comando make no diretório que contém os códigos fonte. Isso irá gerar o executável `a.out`, que deve ser executado da seguinte forma:

```
./a.out < código_fonte
```

Testes

Para testar o analisador semântico foi utilizado um programa escrito na linguagem L2012-1 sem erros de sintaxe, mas com vários erros de tipo, que foram colocados para evidenciar o funcionamento do verificador de tipos. Este programa está relacionado abaixo:

```
 1  program test;
 2  i, j, k : integer;
 3  x, y, z : real;
 4
 5  integer PROCEDURE fatorial(value integer:n):
 6    x, result : integer;
 7  begin
 8    x := 1;
 9    result := 1;
10    repeat
11      result := result * x;
12      x := x + 1
13    until x <= n;
14    i := result
15  end
16
17  begin
18    i := fatorial(2, 'a');
19    x := fatorial(3);
20    if 2.0 then write(1) else write('b');
21    x := abs(12);
22    x := abs(12.0);
23    repeat write(1) until cos(false)
24  end
```

A saída do analisador está listada abaixo:

```
Warning (line 18): wrong number of parameters on function call: fatorial (have 1, supplied
2)
Warning (line 19): assignment type mismatch! -  real <- integer
Warning (line 20): 'if' statement requires a boolean condition (condition is 'real')
Warning (line 21): assignment type mismatch! -  real <- integer
Warning (line 23): 'cos' function requires an integer or real argument.
Warning (line 24): 'repeat' statement requires an expression that evaluates to a boolean
(expression is 'real')
```

O último *warning* acusou um erro no comando *repeat* na linha 24. No entanto, esse comando está na linha 23. Esse erro ocorreu pois, como é o último comando de um bloco, ele só foi reconhecido como comando após a leitura do end, o que incrementou o contador de linhas.

Conclusão

Este trabalho teve como objetivo construir mais uma parte do compilador, o analisador semântico. Para tal, foi necessário construir um verificador de tipos para a linguagem, cujas rotinas são chamadas durante a análise sintática. Esse fato proporcionou um melhor entendimento sobre como ocorre a verificação de tipos em um compilador, além de um melhor entendimento sbre o funcionamento da ferramenta YACC.

Bibliografia

http://pt.wikipedia.org/wiki/Análise_semântica
http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node71.html
http://dinosaur.compilertools.net/yacc/index.html
http://tldp.org/HOWTO/Lex-YACC-HOWTO-1.html