Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
SCC0605 – Teoria da Computação e Compiladores
Docente: Prof. Dr. Thiago A. S. Pardo

Analisador Semântico e Gerador de Código

Leonardo Alves Paiva 10276911
Rafael Pastre 9783675

São Carlos, 9 de julho de 2020

Sumário

Comentários sobre a gramática

Durante o desenvolvimento do trabalho, foram notadas duas especificidades sobre a gramática da linguagem P--, que serão brevemente descritas pois elas podem induzir erros durante escrita de um programa, caso o programador não esteja atento a elas.

A primeira é que a condição utilizada no comando “while” deve ser colocada entre parênteses, enquanto a utilizada no comando “if” não. E a segunda é que o comando “else” não pode ser precedido pelo caractere “;”.

Decisões de projeto

O analisador semântico foi implementado em C++, tendo como base os analisadores léxico e sintático desenvolvidos anteriormente. A implementação foi feita de maneira *ad hoc*, sendo a análise semântica controlada pelo analisador sintático.

Para a análise semântica, foi utilizada uma tabela de símbolos, implementada através de uma tabela hash. Cada elemento da tabela armazena: a cadeia, a categoria do token, o tipo, o endereço e o valor.

A cadeia armazenada é uma concatenação da cadeia lida com seu escopo, ou seja, é uma string na forma “escopo_identificador”. A categoria do token indica se este é uma variável, constante, nome do programa, nome do procedimento ou parâmetro de procedimento. O tipo indica se o elemento é real ou inteiro. O endereço, para variáveis e constantes, indica o endereço na pilha de dados em que estes são armazenados, e para os procedimentos, o endereço indica o valor da linha de código em que o procedimento foi gerado, e que será utilizado em sua chamada pelo comando CHPR.

Devido a solução ad hoc, os elementos da tabela são criados e inseridos na tabela, conforme o andamento da análise sintática, e, ao final da análise, se nenhum erro foi encontrado o código-alvo (em P-Código) é gerado. Além disso, essa abordagem também influencia em outros detalhes do projeto, principalmente no tratamento do escopo. Para os procedimentos, foi considerado que seus comandos só podem utilizar as variáveis internas e os parâmetros do procedimento, dessa forma, o tratamento de erros de escopo pode ser incorporado ao erro de variáveis não declaradas.

Outro detalhe importante, dessa vez quanto a geração de código, é que o endereço de variáveis definidas em uma mesma linha, por exemplo “var x, y, z : real;” será contabilizado na ordem inversa que as variáveis aparecem, ou seja, na pilha de execução a variável z terá endereço 0, a variável y terá endereço 1 e a variável x terá endereço 2.

Além disso, a saída do compilador também foi alterada. Nos trabalhos anteriores não havia geração de código, portanto os erros encontrados eram impressos em um arquivo de texto. Entretanto, com a adição da geração de código, os erros encontrados são impressos agora diretamente no terminal, e o código-alvo gerado é impresso em um arquivo de texto. Há ainda a possibilidade de imprimir a tabela de símbolos no terminal, que pode ser habilitado ou desabilitado ao inserir ou excluir o trecho de código “#define IMPR_TABELA”, presente na linha 19 do arquivo “sintatico.cpp”

Instruções de compilação e execução

Os códigos desenvolvidos foram implementados em C++, e a compilação dos códigos pode ser feita, utilizando o compilador G++, com o seguinte comando:

g++ lexico.cpp semantico.cpp sintatico.cpp -o compilador.exe

A execução do programa depende dos arquivos “reservados.txt” e “transicoes.txt”, portanto é necessário manter estes arquivos no mesmo diretório do compilador, visto que estes arquivos serão carregados no início da execução do compilador.

Além disso, o compilador tem como parâmetros de execução dois arquivos de texto, sendo que o primeiro deve conter o código-fonte em P--, que será lido pelo compilador, e o segundo deve ser o arquivo onde será escrito o código-alvo gerado pelo compilador.

O comando de execução pode variar de acordo com a plataforma em que o compilador está sendo testado. No Windows a execução é feita pelo comando:

compilador.exe <arquivo_fonte_pascal> <arquivo_saida>

Enquanto em um sistema Linux a execução é feita por:

./compilador.exe <arquivo_fonte_pascal> <arquivo_saida>

O programa foi testado em três plataformas: Um Windows 10 x64, um Windows 7 Home Basic SP1 x64 e um Ubuntu. Além disso, os códigos completos podem ser encontrados no repositório do GitHub: https://github.com/rafael-pastre/SCC0605-2020-Trabalho_03

Testes

Dois códigos em P-- foram feitos para testar o compilador. O primeiro consiste de um programa completamente correto, que utiliza todas as funcionalidades da linguagem como declaração de constantes, variáveis e procedimentos, utilização dos comandos de read, write, while, if, atribuição, for, chamadas de procedimento, e blocos de códigos. Este programa se encontra em anexo no arquivo “p.txt”.

O segundo arquivo, que se encontra no arquivo “p_com_erros.txt”, se consiste de uma variação do primeiro onde foram inseridos todos os tipos de erros semânticos reconhecidos pelo compilador: Variáveis e procedimentos não declarados, declaração repetida, incompatibilidade de parâmetros dos procedimentos, variáveis de escopo inadequado, e comandos inadequados com reais e inteiros. Dessa forma, a geração de códigos foi testada através do programa “p.txt”, e a detecção de erros semânticos através do programa “p_com_erros.txt”.

Erros encontrados em “p_com_erros.txt”

Erro na linha 12 : w. Identificador declarado novamente

Erro na linha 20 : s. Identificador nao declarado

Erro na linha 21 : w. Identificador nao declarado

Erro na linha 22 : procB. Identificador nao declarado

Erro na linha 27 : x. Identificador nao declarado

Erro na linha 31 : procB. Procedimento declarado novamente

Erro na linha 32 : d. Identificador declarado novamente

Erro na linha 43 : w. Divisao entre nao inteiros

Erro na linha 43 : x. Divisao entre nao inteiros

Erro na linha 44 : w. Atribuicao de real a inteiro

Erro na linha 52 : x. Atribuicao de real a inteiro

Erro na linha 66 : c. Read/Write com tipos diferentes

Erro na linha 67 : a. Numero excessivo de argumentos

Erro na linha 68 : a. Numero excessivo de argumentos

Erro na linha 68 : simb_v. Esperado : ')'

Erro na linha 68 : b. Esperado : ';'

Erro na linha 68 : simb_fpar. Esperado : ';'

Erro na linha 68 : simb_fpar. Esperado : end