Compiladores: Implementação e explicação sobre um compilador para a linguagem T++: Análise Semântica

Matheus Sapia Guerra

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Caixa Postal: 271 Campo Mourão - PR - Brasil

guerramatheus2@gmail.com

Abstract. *This article aims to describe details of the application of a semantic analogue, taking into consideration an analysis and a syntactic analysis lexic is already ready and working. For an implementation it was used in python language with the text editor Sublime text.*

Resumo. *Este aritgo tem por objetivo descrever detalhes da implementação de um análisador semântico, levando em consideração que a análise lélixa e a análise sintática já esteja pronta e funcionando. Para a implementação foi utilizado a linguagem python com o editor de textos Sublime text.*

1. Introdução

A análise semântica é talvez uma das fases mais complicadas de um compilador, pois pode ser considerado como um trabalho "artesanal", ou seja, por mais que a Ebnf, e as demais fases anteriores sejam iguais, a análise semântica é algo que varia muito de programador para programador. Neste artigo será explicado o que foi implementado na análise semântica da linguagem Tpp. Sendo assim cada análisador sintático tem uma caracteristica, tais como: O que aconteçe se declarar duas variaveis com o mesmo nome, utilizar uma variavel diferente, operações com tipos diferentes, entre outros.[johnidm 2017]

2. Fundamentação Teorica

Nesta seção será apresentado a fundamentação teorica de elementos e conceitos utilizados em uma análise semântica.

2.1. Análise Semântica

A análise semântica tem por finalidade pegar os erros e validação de regras que não eram possíveis de serem realizadas na análise léxica e a na análise sitntática. Como descrito anteriormente não existe uma "receita de bolo"que dita como fazer uma análise semântica, sendo assim a análise está sujeita ao conhecimento do programador.

Um dos principais erros semânticos é o escopo dos identificadores, ou seja, o compilador deve garantir que variáveis e funções estejam declaradas em locais que podem ser acessados onde esses identificadores estão sendo utilizados. Outro erro que se resolve na análise semântica é a compatibilidade de tipos, por exemplo, o que aconteceria se uma variavel do tipo inteiro recebece uma string? Isso é resolvido nesta etapa do projeto.

3. Materiais

Foram utilizados como materiais o *Sublime text* com a função de editor de textos e Python como linguagem de programação.

4. Implementação

Nesta seção será abordado assuntos referente a implementação da análise sintática, considerando que a análise léxica e a análise sintáticajá esteja implementada.

De inicio foi criado estrutras para função e simbolos, a estrutura da função conta com os atributos, tipoRetorno, nome, parametros[] e uma flag utilizado. A estrutura simbolo contém os atributos tipo, escopo, nome, dimensão e uma flag utilizado. Os nomes são compreensiveis, mas talvez o termo dimensão seja um pouco diferente, sendo assim, o termo dimensão é um inteiro que pode ir de 0..n, uma variável comum tem valor 0 um vetor unidimensional tem tamanho 1 e assim por diante. Segue trecho do código que descreve essas classes.

Código 1. class Funcao

```
class Funcao():
  """docstring for Funcao"""
  def __init__(self, tipoRetorno, nome, parametros):
    self.tipoRetorno = tipoRetorno
    self.nome = nome
    self.parametros = parametros
    self.utilizado = 0

  def __str__(self):
    return str(self.tipoRetorno) + ' ' + self.nome + ' ' +
       str(self.parametros)
```

Código 2. class Simbolo

```
class Simbolo():
  def __init__(self, tipo, escopo, nome, dimensao):
    self.tipo = tipo
    self.escopo = escopo
    self.nome = nome
    self.dimensao = dimensao
    self.utilizado = 0

  def __str__(self):
    return self.escopo + ' ' + self.tipo + ' ' + self.nome + '
       ' + str(self.dimensao) + ' ' + str(self.utilizado)
```

A classe semântica é onde tudo acontece, por ser uma classe muito grande com muitos métodos e chamados de função será mostrada só o começo da mesma neste artigo.

Código 3. class Semantica

```
class Semantica():
  def __init__(self, codigo):
    self.listaDeSimbolos = []
    self.listaDeFuncoes = []
    self.codigo = codigo
    self.arvore = Parser(codigo).ast
```

```
7       self.addVarFunc()
8       self.montarTabelaSimbolos(self.arvore)
9       self.verificaPrincipal()
10      self.verificaVarUtil()
```

A semântica consiste em percorrer a árvore sintática e definir regras e válidações que garantem o funcionamento correto do compilador, é está parte que cada programador faz de um jeito diferente.

5. Resultados

Como resultado obtemos os famosos Warnings, que diz que algo está incorreto. Segue abaixo um exemplo de warning gerado pelo analisador semântico.

Código 4. class Teste-0001

```
1 inteiro: a
2 flutuante: b
```

Note que não foi declarado a função principal, sendo assim a mensagem de saida é:

Código 5. class Saida terminal

```
1 $ ERRO: Funcao principal nao declarada!
```

6. References

Referências

johnidm (2017). Compiladores para humanos. internet. https://johnidm.gitbooks.io/compiladores-para-humanos/content/part1/semantic-analysis.html Acessado em 16/11/2017.