Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática

Compiladores - 9797

Desenvolvimento de um Compilador para a Linguagem TNG

Professor: Felippe Fernandes da Silva

Discente

RA	Nome
115892	Gabriel Lima Dias
115672	Vinícius Kenzo Fukace

Introdução

A atividade proposta consiste na implementação de um compilador para uma linguagem aberta. O compilador deve ser capaz de interpretar o código-fonte escrito na linguagem em questão e gerar um código executável para a plataforma alvo. O processo é dividido nas etapas de análise léxica, análise sintática, análise semântica e geração de código.

O presente relatório tem como objetivo descrever a implementação de um compilador para a linguagem TNG (This is Not a Grammar). Apresenta-se o processo de desenvolvimento do compilador, as ferramentas utilizadas, modo de execução, e as facilidades e dificuldades do processo.

Desenvolvimento

Esta seção descreve o processo de desenvolvimento do compilador para a linguagem TNG. A linguagem foi criada com base nas estruturas de C e utiliza elementos de Python. A construção das etapas de análise léxica, sintática e semântica e geração de código foram realizadas utilizando a linguagem Python com a ferramenta ANTLR. O compilador desenvolvido traduz a linguagem TNG para C++.

ANTLR

ANTLR (ANother Tool for Language Recognition) é uma ferramenta para geração de código de analisador léxico e sintático para várias linguagens de programação, incluindo Python. O ANTLR é usado para construir compiladores, interpretadores e outras ferramentas de processamento de linguagem.

A ferramenta funciona analisando um arquivo de gramática, que descreve a sintaxe de uma determinada linguagem. Para este trabalho, “tng.g4” é o nome do arquivo de gramática, este arquivo define as regras e a estrutura da linguagem, incluindo seus tokens e sintaxe. Com o arquivo de gramática definido, o ANTLR gera código Python para o lexer e parser com base na gramática.

A análise semântica é feita durante a análise da árvore sintática gerada pelo parser. A ferramenta gera o arquivo “tngVisitor”, que oferece funções para percorrer a árvore, e a análise semântica é implementada no arquivo “tngSemantic”, que herda as funções do visitor e valida a estrutura da árvore sintática. A geração de código é feita em conjunto com a análise semântica, visto que o percurso completo da árvore é feito durante ambas as etapas.

Definição da Linguagem

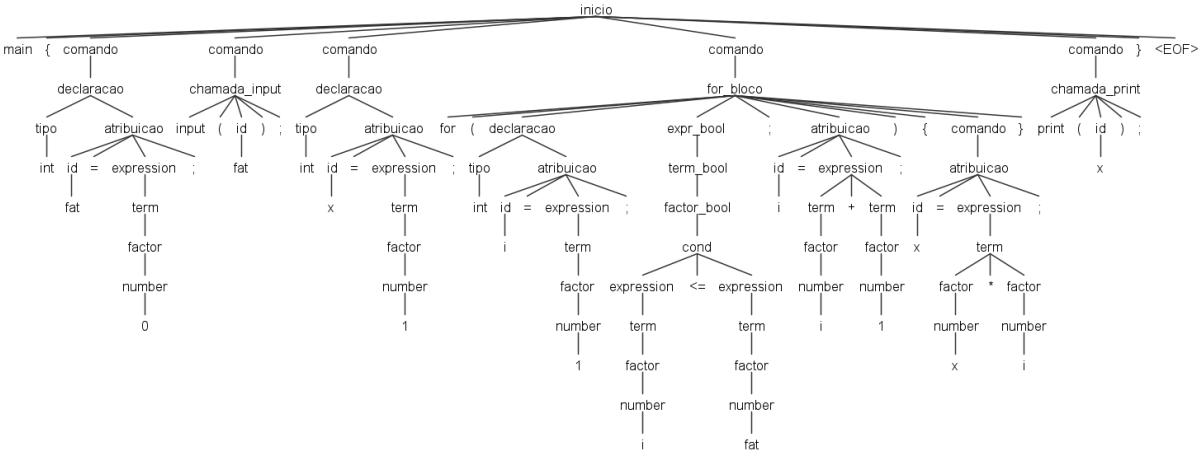
Escolheu-se basear a linguagem no conjunto de palavras-chave da linguagem C++ para as definições de tipos e estruturas de controle, assim como para as operações lógicas e aritméticas. Somente os tipos *int*, *float*, *string* e *bool* são considerados, junto às estruturas *if*, *else*, *for* e *while*. Em seguida, são apresentados tópicos sobre as decisões específicas de definição da linguagem que foram adotadas:

- O início do programa é marcado pela palavra-chave *main* seguida da abertura de chave (“{”), e o fim do programa é marcado pela fechadura da chave correspondente (“}”).
- As funções de entrada e saída de dados pelo terminal são definidas como *input* e *print*, respectivamente
- É realizado o casting de números *int* para *float* quando necessário, e vice-versa.
- Comentários seguem o mesmo padrão de C++, com a opção adicional de se usar “#” para se comentar uma linha.
- O suporte para definição de funções não foi implementado na definição da linguagem.
- Operações com *string* não foram adicionadas à linguagem.

Execução

Para executar o trabalho, recomenda-se clonar os arquivos do seguinte repositório: https://github.com/gabrielld06/compila

- Instalar todas as dependências com o comando "pip install -r requirements.txt"
  - Requer g++ para compilar o código gerado em C++
  - Foi utilizado Python na versão 3.10 para o desenvolvimento do trabalho
- Para gerar os arquivos de análise léxica e sintática, use o comando "antlr.bat compile"
- Para gerar a árvore sintática de um programa, use o comando "antlr.bat tree <nomeDoArquivo>"
  - Caso não seja fornecido um arquivo de entrada, "tests/test.tng" é utilizado por padrão



Exemplo de árvore sintática para o programa “fat.tng”

- Para executar o compilador com um arquivo de entrada, use "python3 tngCompiler.py <nomeDoArquivo>"

    - Caso não seja fornecido um arquivo de entrada, "tests/test.tng" é compilado por padrão
    - O arquivo C++ resultante é nomeado "output.cpp"
    - O arquivo executável resultante é nomeado "output.exe"
- Finalmente, executar o programa compilado com "./output"

Discussão

O processo de desenvolvimento do compilador foi facilitado pela ferramenta ANTLR, que gera automaticamente o código do lexer e do parser a partir da gramática especificada. A especificação da linguagem é feita de modo similar ao apresentado nas aulas, auxiliando o entendimento do código. A ferramenta também facilita os processos de análise semântica e geração de código através do uso das funções relacionadas à árvore sintática.

Dificuldades foram encontradas com a implementação da linguagem assembly, que foi considerada para ser a linguagem alvo do compilador. Após estas dificuldades, definiu-se C++ como a linguagem alvo, simplificando muito o processo de geração de código. Algumas dificuldades foram encontradas na implementação da análise semântica, especialmente no tratamento de atribuições e expressões booleanas, que exigiram soluções mais elaboradas e extensas, que podem ser consideradas como soluções não ideais.

Referências

https://www.antlr.org/
https://github.com/antlr/antlr4/blob/4.12.0/doc/index.md