COMPARAÇÃO DE MÉTODOS DE ANÁLISE SINTÁTICA

THALES DE M. GONÇALVES[1], ALVARO C. NETO[2]

[1]Graduando em Tecnologia de Análise e Desenvolvimento de Sistemas, Bolsista PIBIFSP, IFSP, Câmpus Barretos, thales_de_mattos@hotmail.com
[2]Docente, IFSP, Câmpus Barretos, alvaro@ifsp.edu.br

Área de conhecimento (Tabela CNPq): 1.03.03.01-4 Linguagens de Programação

Apresentado no
8° Congresso de Inovação, Ciência e Tecnologia do IFSP
06 a 09 de novembro de 2017 - Cubatão-SP, Brasil

RESUMO: Esta pesquisa em andamento tem como propósito realizar uma comparação entre métodos de análise sintática, sendo estes LR(1) e Descendente Recursivo. Com esta finalidade, estão sendo empregadas as ferramentas FLEX e GNU Bison para auxiliar no desenvolvimento dos analisadores léxico e sintático LR(1) e um sistema gerador de códigos para teste também está sendo desenvolvido. Resultados preliminares indicam sucesso (a) na geração de códigos para teste e (b) na análise da implementação em LR(1).

PALAVRAS-CHAVE: analisador léxico-sintático; computação; compiladores; linguagens; programação.

COMPARISON OF SYNTACTIC ANALYSIS METHODS

ABSTRACT: This in-development research aims to compare between syntactic analysis methods, specifically LR(1) and Recursive Descent. FLEX and GNU Bison are being used to construct the lexical and LR(1) syntactical analyzers and a test code generator is also under development. Preliminary results indicate success in both (a) test code generation, and (b) LR(1) implementation analysis.

KEYWORDS: computing; compilers; languages; lexical-syntactical analyzer; programming.

INTRODUÇÃO

Os compiladores analisam o código-fonte garantindo que as regras gramaticais foram obedecidas. Para isso, são empregados três analisadores: léxico ou linear, sintático ou hierárquico e semântico (AHO; SETHI; ULLMAN, 1995).

O analisador léxico ou linear lê o código-fonte como uma sequência de caracteres e os categoriza em lexemas (*tokens*), que podem ser definidos por expressões regulares (AHO; SETHI; ULLMAN, 1995).

O analisador sintático recebe os *tokens* gerados pela análise léxica e verifica se as posições são válidas para alguma produção gramatical. As produções gramaticais podem conter símbolos terminais (*tokens* ou símbolos literais) e símbolos não-terminais que correspondem a outras produções gramaticais (AHO; SETHI; ULLMAN, 1995).

Existem diversos métodos para analisar as regras gramaticais de um certo código-fonte escrito em uma dada linguagem. Foram utilizados nesta pesquisa dois métodos para se analisar sintaticamente códigos. O primeiro deles é o método *left-to-right, rightmost derivation* (LR(1)) também conhecido como *bottom-up parsing*, que consiste em primeiramente ler os *tokens* para depois categorizá-los de acordo com alguma produção gramatical. O número entre parênteses indica a quantidade de lexemas que devem ser lidos à frente da produção atual para identificar seu final, também conhecido como *lookahead* (WIRTH, 2005).

O outro método a ser comparado é o Descendente Recursivo, que é estruturado em sub-rotinas recursivas, sendo uma para cada regra gramatical. Dessa forma, a montagem da árvore de derivação gramatical assemelha-se à pilha de chamada das sub-rotinas, criando uma representação direta das derivações. (AHO; SETHI; ULLMAN, 1995).

MATERIAL E MÉTODOS

Estão sendo utilizadas na pesquisa as ferramentas FLEX e GNU Bison, ambas sem restrição de uso para fins acadêmicos e um computador para desenvolver e testar os demais programas necessários para a conclusão da pesquisa.

A metodologia consiste em obter um analisador léxico (por meio da ferramenta FLEX), um analisador sintático LR(1) (por meio da ferramenta GNU Bison) e um analisador sintático Descendente Recursivo a ser desenvolvido manualmente e comparar os desempenhos de ambos os analisadores usando um código extenso gerado por outra ferramenta, também em desenvolvimento.

A listagem 1 demonstra as expressões regulares que definem respectivamente os *tokens* “ID” e “STRING” para a ferramenta FLEX:

Listagem 1. Expressões regulares que definem os *tokens* “ID” e “STRING”.

```
[_a-zA-Z][_a-zA-Z0-9]*  return ID;
[\'].*[\']              return ID;
```

Fonte: próprio autor.

A primeira linha determina que um “ID” é uma sequência de caracteres iniciada por um *underline* ou uma letra de a a z minúscula ou maiúscula, seguida por *n* caracteres que podem ser: *underline*, letra de a a z minúscula ou maiúscula ou um dígito numérico.

A segunda linha determina que um “ID” também pode ser um apóstrofo seguido de *n* caracteres quaisquer e um outro apóstrofo.

A listagem 2 demonstra a estrutura que define a produção gramatical de atribuição, neste caso “atrib”:

Listagem 2. Produção gramatical de atribuição baseada na linguagem C.

```
atrib :     ID '=' exp  {;}
|     ID'+''+'    {;}
|     ID'-''-'    {;}
;
```

Fonte: próprio autor.

A primeira linha define “atrib” como um *token* “ID” seguido de um símbolo de igualdade e um símbolo não-terminal “exp”.

A segunda linha define “atrib” como um *token* “ID” seguido de dois símbolos de adição.

A terceira linha define “atrib” como um *token* “ID” seguido de dois hífens.

A quarta linha é um ponto-e-vírgula, determinando o fim daquela regra de produção gramatical.

RESULTADOS E DISCUSSÃO

Este trabalho encontra-se em andamento e somente possui resultados preliminares. Assim, não é possível apresentar a comparação entre os métodos de análise sintática de forma definitiva. O analisador sintático LR(1) já foi testado parcialmente e é capaz de processar os arquivos de entrada fornecidos, sem travamentos. O analisador sintático Descendente Recursivo ainda não foi finalizado e portanto, não pôde ser testado.

A tabela 1 apresenta os resultados dos testes de desempenho do analisador LR(1) por meio de um código-fonte gerado pelo gerador de código desenvolvido.

Tabela 1. Tempo de execução do analisador léxico e sintático LR(1) sobre quantidades de caracteres

Quantidade de caracteres no código-fonte	Tempo consumido em processo de análise (s)
417000	~0,015
625500	~0,024
834000	~0,029

Fonte: próprio autor.

CONCLUSÕES

Até o momento, a implementação do analisador sintático LR(1) encontra-se finalizado e em estágio de testes, sem verificação definitiva de seu desempenho. O analisador Descendente Recursivo ainda encontra-se em desenvolvimento, uma vez que este trabalho encontra-se em andamento.

A ferramenta de geração de código de teste já está concluída e é capaz de gerar um arquivo de teste contendo qualquer quantidade pré-definida de construções sintáticas, baseadas na gramática da linguagem C.

Assim que ambas as implementações estiverem devidamente implementadas e testadas, outra ferramenta será criada para realizar os testes de desempenho de forma aleatória, visando averiguar qual das duas técnicas de análise é mais eficiente.

AGRADECIMENTOS

Agradecemos por todo o apoio que o Instituto Federal de São Paulo fornece à nossa pesquisa e aos autores cujas obras foram imprescindíveis ao nosso desenvolvimento.

REFERÊNCIAS

AHO, Alfred V.; SETHI, Ravi; ULLMAN, Jeffrey D.. Compiladores: Princípios, técnicas e ferramentas. [s.l.]: Addison-wesley, 1995. Disponível em: <http://comp.ist.utl.pt/aaa/Prog/Compiladores - Principios Tecnicas E Ferramentas (Pt Br).pdf>. Acesso em: 06 jul. 2017.

WIRTH, Niklaus. Compiler Construction. Zurique: Addison-wesley, 2005. Disponível em: <http://www.ethoberon.ethz.ch/WirthPubl/CBEAll.pdf>. Acesso em: 06 jul. 2017.