UNIVERSIDADE ESTADUAL PAULISTA
“JÚLIO DE MESQUITA FILHO”

Analisador Sintático

Beatriz Ferreira de Lima
Douglas Brandão dos Santos
Instituto de Biociências, Letras e Ciências Exatas — IBILCE, Universidade Estadual Paulista "Júlio de Mesquita Filho" — UNESP, Rua Cristóvão Colombo 2265, Jd.Nazareth, 15054-000, São José do Rio Preto – SP, Brasil. E-mail: beatriz.f.lima@unesp.br, douglas.brandao@unesp.br

25 de Outubro de 2019

Resumo

Neste documento é especificado o processo de projetar e implementar um analisador sintático, construído com base em um analisador léxico desenvolvido anteriormente. Para isso, foram utilizadas bibliotecas como GNU Bison e GNU Flex. Além disso, será formalizada a definição da gramática livre de contexto utilizada no projeto, além de comentários a respeito das adaptações necessárias no analisador léxico.

1. Introdução

Levando em consideração as etapas de tradução executadas por um compilador em um código, tem-se: a análise léxica, sintática e semântica. Este trabalho visa realizar estudos mais aprofundados acerca da análise sintática.

A sintaxe de uma linguagem de programação descreve a forma apropriada dos programas a serem escritos. Para a especificação de uma sintaxe, é utilizada uma notação amplamente reconhecida, chamada gramática livre de contexto. (AHO et. al., 1986)

Dessa forma, com o intuito de exercitar a etapa de análise sintática do processo de tradução exercido pelos compiladores, este documento tem a finalidade de voltar a abordar a linguagem desenvolvida pelos autores, descrevendo a mesma por meio de seu alfabeto bem como por sua gramática livre de contexto, sendo esta utilizada para a confecção do analisador sintático, por meio da biblioteca GNU Bison.

Além disso, foram descritas todas as mudanças necessárias no analisador léxico, para que a integração com a biblioteca Bison fosse feita de forma correta.

Por fim, anexado a este documento encontra-se o manual do usuário, para que este seja capaz de realizar a instalação do ambiente e executar o analisador sintático desenvolvido.

2. Linguagem

2.1 Alfabeto da linguagem

O alfabeto da linguagem é o conjunto de todos os símbolos que podem compor determinada cadeia de caracteres pertencentes a linguagem em questão. Tal conjunto é composto pelos seguintes elementos:

$$\sum = \{a,...,z,0,...,9,,,;,(,), [,],\{,\},/,*,-,+,\&,|,!,=,>,< \}$$

2.2 Gramática

A gramática é responsável por definir a forma correta das cadeias da linguagem. Ou seja, pode-se dizer que a gramática nada mais é do que o conjunto de regras de formação para as cadeias pertencentes a determinada linguagem. Portanto, caso uma cadeia não se encaixe em nenhuma dessas regras, esta não pertence a linguagem, caracterizando assim, um erro sintático.

A gramática utilizada para a confecção do analisador sintático é do tipo livre de contexto, estas são amplamente utilizadas para especificações de linguagens de programação. A gramática empregregada na fase de análise sintática do compilador da linguagem desenvolvida pelos autores é ilustrada a seguir:

G =(V, {a-z,0-9,-,+,/,*,},{,},(,),[,],=,&,|,!,.,,,;,:,%,',}, P, S)

V = {S,BODY, COMANDO, DECLARATION, ATTRIBUTION, INITIALIZER, EXPR, ERRO, EXPR_STMT,TYPE, VARIABLE, ISYMBOL, IVAR, INT, FLOAT, STRING, OPERATOR}

P = {

S ⇒ begin BODY end | epsylon

BODY ⇒ COMANDO; BODY S| DECLARATION; BODY S| ATTRIBUTION; BODY S |INITIALIZER BODY S| EXPR; BODY S|S| ERRO,

COMANDO ⇒ if(EXPR){EXPR_STMT}|if (EXPR){EXPR_STMT} else{EXPR_STMT}|for(EXPR_STMT EXPR_STMT EXPR){EXPR_STMT} |while(EXPR){EXPR_STMT}| print(INITIALIZER),

DECLARATION ⇒ TYPE VARIABLE| TYPE ATTRIBUTION,

ATTRIBUTION ⇒ VARIABLE = INITIALIZER | VARIABLE = EXPR,

INITIALIZER ⇒ VARIABLE| INT| FLOAT| STRING,

EXPR ⇒ (EXPR)| VARIABLE = INITIALIZER| VARIABLE OPERATOR INITIALIZER| VARIABLE OPERATOR| OPERATOR VARIABLE,

EXPR_STMT ⇒ EXPR;,

ERRO ⇒ ISYMBOL| IVAR,

VARIABLE ⇒ [A-Za-z][A-Za-z0-9]*,

OPERATOR ⇒ + | - | * | / | != | ++ | -- | == | >= | > | <= | < |%| && | || | ! ,

TYPE ⇒ int| float| string,
INT ⇒ [0-9]+,
FLOAT ⇒ [0-9]+ . [0-9]+,
STRING ⇒ (\"(\\.|[^"\\])*\"),
ISYMBOL ⇒ "@"|"#"|"$"|"?"|"÷"|"π"|"√"|"¶"|"∆"|"£"|"~",
IVAR ⇒ ([0-9]|ISYMBOL)[A-Za-z][A-Za-z0-9]*

3. Adaptações feitas no analisador léxico

Foram necessárias adaptações no analisador léxico desenvolvido anteriormente, para que este integre-se perfeitamente a GNU Bison. Primeiramente, foi necessária a quebra de determinadas expressões regulares, de forma que fosse possível obter caracteres isolados. Tal ação foi tomada principalmente para caracteres que necessitam ser abertos e em seguida fechados, tais como, (,), [,], {, }, “,”. Além disso, foram adicionadas as marcações “begin” e “end”, que indicam respectivamente o início e término do software em execução.

Ademais, também foram separadas as expressões para os comandos pertencentes a linguagem, tais como: if, else, for, while, e print.

Por fim, enquanto que no projeto original do analisador léxico, a cada token recebido este era impresso na tela, após a adaptação, cada token recebido é retornado para a fase de análise sintática, para que este seja agregado as cadeias da linguagem, podendo ou não estar de acordo com as regras da gramática.