Análise Sintática da Linguagem TPP

Guilherme B. Del Rio[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR) – Campo Mourão, PR – BRASIL

guilhermerio@alunos.utfpr.edu.br

Abstract. *This document describes the parsing part of the TPP language compiler, using Yacc tools through the PLY library, an automaton is created according to an input code and grammar specification, which uses the BNF standard, and then it is created an image with the tree resulting from the automaton.*

Resumo. *Este documento descreve a parte de análise sintática do compilador da linguagem TPP, utilizando as ferramentas do Yacc através da biblioteca PLY, é criado um autômato de acordo com um código de entrada e da especificação da gramática, que utiliza o padrão BNF, e então é criado uma imagem com a árvore resultante do autômato.*

1. Descrição da Gramática no padrão BNF

O padrão BNF ou *Backus-Naur Form*, é um formalismo para expressar gramáticas livres de contexto, ou seja, um modo formal de escrever linguagens formais. É composto pelas seguintes regras de derivação: **símbolo ::= expressão com símbolos**, onde o símbolo é um não terminal e a expressão é formada por sequências de símbolos, caso haja mais de uma opção para a expressão, essas devem ser separadas pela barra ”|”.

Para a linguagem TPP, as expressões BNF são (Tabela 1):

Símbolo	Expressão
programa ::=	lista_declaracoes
lista_declaracoes ::=	lista_declaracoes declaracao | declaracao
declaracao ::=	declaracao_variaveis | inicializacao_variaveis | declaracao_funcao
declaracao_variaveis ::=	tipo ”:”lista_variaveis
inicializacao_variaveis ::=	atribuicao
lista_variaveis ::=	lista_variaveis ”,”var | var
var ::=	ID | ID indice
indice ::=	indice ”[”expressao ”]” | ”[”expressao ”]”
tipo ::=	INTEIRO | FLUTUANTE

Símbolo	Expressão
declaracao_funcao ::=	tipo cabecalho | cabecalho
cabecalho ::=	ID ”(”lista_parametros ”)”corpo FIM
lista_parametros ::=	lista_parametros ”,”parametro | parametro | vazio
parametro ::=	tipo ”:”ID | parametro ”[]”
corpo ::=	corpo acao | vazio
acao ::=	expressao | declaracao_variaveis | se | repita | leia | escreva | retorna | erro
se ::=	SE expressao ENTAO corpo FIM | SE expressao ENTAO corpo SENAO corpo FIM
repita ::=	REPITA corpo ATE expressao
atribuicao ::=	var ”:=”expressao
leia ::=	LEIA ”(”var ”)”
escreva ::=	ESCREVA ”(”expressao ”)”
retorna ::=	RETORNA ”(”expressao ”)”
expressao ::=	expressao_logica | atribuicao
expressao_logica ::=	expressao_simples | expressao_logica operador_logico expressao_simples
expressao_simples ::=	expressao_aditiva | expressao_simples operador_relacional expressao_aditiva
expressao_aditiva ::=	expressao_multiplicativa | expressao_aditiva operador_soma expressao_multiplicativa
expressao_multiplicativa ::=	expressao_unaria | expressao_multiplicativa operador_multiplicacao expressao_unaria
expressao_unaria ::=	fator | operador_soma fator | operador_negacao fator
operador_relacional ::=	”<” | ”>” | ”=” | ”<>” | ”<=” | ”>=”

Símbolo	Expressão
operador_soma ::=	”+” | -”
operador_logico ::=	”&&” | —”
operador_negacao ::=	”!”
operador_multiplicacao ::=	”*” | ”/”
fator ::=	”(”expressao ”)” | var | chamada_funcao | numero
numero ::=	NUM_INTEIRO | NUM_PONTO_FLUTUANTE | NUM_NOTACAO_CIENTIFICA
chamada_funcao ::=	ID ”(”lista_argumentos ”)”
lista_argumentos ::=	lista_argumentos ”,”expressao | expressao | vazio

Tabela 1. Expressões no formato BNF

2. Formato da Análise Sintática realizado pela ferramenta

Existem quatro formatos possíveis a serem utilizados, LL(1), LR(1), LALR(1), SLR(1), mas para este trabalho foi utilizado apenas LALR(1). O formato LR define como será percorrido a árvore gerada pelo autômato, sendo L ”left”e R ”right”, da direita para a esquerda.

Diferente do LR(0), o LR(1) através dos símbolos após o ”.”, podemos verificar um símbolo seguinte ao estado em que nos encontramos, permitindo a localização dentro do autômato e identificando o caminho anteriormente seguido e o caminho que iremos seguir e então é gerada a tabela dos estados.

O LALR(1) é utilizado para otimizar o LR(1), ou seja, é analisado a tabela e identificado se há linhas de saída iguais, resultando na exclusão de linhas repetidas. LALR(1) é reconhecido pelo PLY, sendo assim uma ferramenta de melhor desempenho do que LR(1).

3. implementação e utilização da ferramenta Yacc

Para a implementação da parte sintática do compilador é utilizada a ferramenta Yacc, que implementa o componente de análise sintática do PLY.

Primeiramente é necessário fazer a construção do parser através do Yacc (Figura 1), onde especificamos a o formato da análise como sendo LALR, e então é aberto o arquivo que desejamos analisar e então o enviamos para o parser anteriormente criado. Para cada regra gramatical foi criada uma função com sua devida especificações, e um função inicial chamada **p_programa** (Figura 2), que indica o início da árvore do programa.

```
925   parser = yacc.yacc(method="LALR", optimize=True, start='programa', debug=True,
926                      debuglog=log, write_tables=False, tabmodule='tpp_parser_tab')
```

Figura 1. Construção do parser através do Yacc

```
30   def p_programa(p):
31       """programa : lista_declaracoes"""
32
33       global root
34
35       programa = MyNode(name='programa', type='PROGRAMA')
36
37       root = programa
38       p[0] = programa
39       p[1].parent = programa
```

Figura 2. Função p_programa

Observa-se a utilização do parâmetro *p*, este parâmetro nos traz uma sequencia contendo os valores de cada símbolo da gramática na regra correspondente, como por exemplo na regra de atribuição de um valor (Figura 3): *atribuicao : var ATRIBUICAO expressao*, temos o parâmetro *p* inicial, e então são criados nós para a mapeação de seus devidos símbolos da gramática, assim como a criação da árvore através da criação da ligação entre estes nós.

```
485   def p_atribuicao(p):
486       """atribuicao : var ATRIBUICAO expressao"""
487
488       pai = MyNode(name='atribuicao', type='ATRIBUICAO')
489       p[0] = pai
490
491       p[1].parent = pai
492
493       filho2 = MyNode(name='ATRIBUICAO', type='ATRIBUICAO', parent=pai)
494       filho_sym2 = MyNode(name=':=', type='SIMBOLO', parent=filho2)
495       p[2] = filho2
496
497       p[3].parent = pai
```

Figura 3. Função p_atribuicao

4. Exemplos de entrada e saída

Na entrada é dado um arquivo (Figura 4) com a extensão .tpp, que contém um algoritmo utilizando a linguagem referente a mesma, através da execução do parser, é gerado tanto um gráfico em linha de comando (Figura 5) quanto dois arquivos nos mostrando a árvore sintática do código (Figura 6).

Através das imagens geradas, podemos observar todo o caminho necessário para a criação da árvore sintática, desde a criação do programa, até a chegada aos estados terminais, como por exemplo o número inteiro 10 (dez) em uma das folhas (nós) da árvore.

```
1  { programa principal }
2  inteiro principal()
3  | tam := 10
4  | retorna(0)
5  fim
```

Figura 4. Exemplo de programa

```
guilhermerio@guilhermerio:~/Documentos/GitHub/Compiladores/Análise Sintática/codigo-start-parser$ python3 tppparser.py ../sintatica-teste
s/atribuicao.tpp
Generating LALR tables
WARNING: 47 shift/reduce conflicts
Generating Syntax Tree Graph...
programa
+-- lista_declaracoes
    +-- declaracao
        +-- declaracao_funcao
            |-- tipo
            |   +-- INTEIRO
            |       +-- inteiro
            +-- cabecalho
                |-- ID
                |   +-- principal
                |-- ABRE_PARENTESE
                |   +-- (
                |-- lista_parametros
                |   +-- vazio
                |-- FECHA_PARENTESE
                |   +-- )
                |-- corpo
                |   |-- corpo
                |   |   |-- corpo
                |   |   |   +-- vazio
                |   |   +-- acao
                |   |       +-- expressao
                |   |           +-- atribuicao
                |   |               |-- var
                |   |               |   +-- ID
                |   |               |       +-- tam
                |   |               |-- ATRIBUICAO
                |   |               |   +-- :=
                |   |               +-- expressao
                |   |                   +-- expressao_logica
                |   |                       +-- expressao_simples
                |   |                           +-- expressao_aditiva
                |   |                               +-- expressao_multiplicativa
                |   |                                   +-- expressao_unaria
                |   |                                       +-- fator
                |   |                                           +-- numero
                |   |                                               +-- NUM_INTEIRO
                |   |                                                   +-- 10
                |   +-- acao
                |       +-- retorna
                |           |-- RETORNA
                |           |   +-- retorna
                |           |-- ABRE_PARENTESE
                |           |   +-- (
                |           |-- expressao
                |           |   +-- expressao_logica
                |           |       +-- expressao_simples
                |           |           +-- expressao_aditiva
                |           |               +-- expressao_multiplicativa
                |           |                   +-- expressao_unaria
                |           |                       +-- fator
                |           |                           +-- numero
                |           |                               +-- NUM_INTEIRO
                |           |                                   +-- 0
                |           +-- FECHA_PARENTESE
                |               +-- )
                +-- FIM
                    +-- fim
Graph was generated.
Output file: ../sintatica-testes/atribuicao.tpp.ast.png
```

Figura 5. Exemplo de saída no terminal e comando utilizado

Referências

Gonçalves, P. R. A. (2021a). Aula 013 – análise sintática. https://moodle.utfpr.edu.br/pluginfile.php/1101423/mod_resource/content/2/aula-13-analise-sintatica-yacc.md.slides.pdf.

Gonçalves, P. R. A. (2021b). Aula 014 – análise semântica. https://moodle.utfpr.edu.br/pluginfile.php/261753/mod_resource/content/5/aula-14-analise-semantica-gramaticas-de-atributos-e-algoritmos-para-c
md.notes.pdf.

Gonçalves, P. R. A. (2021c). Aula 015 – análise semântica. https://moodle.utfpr.edu.br/

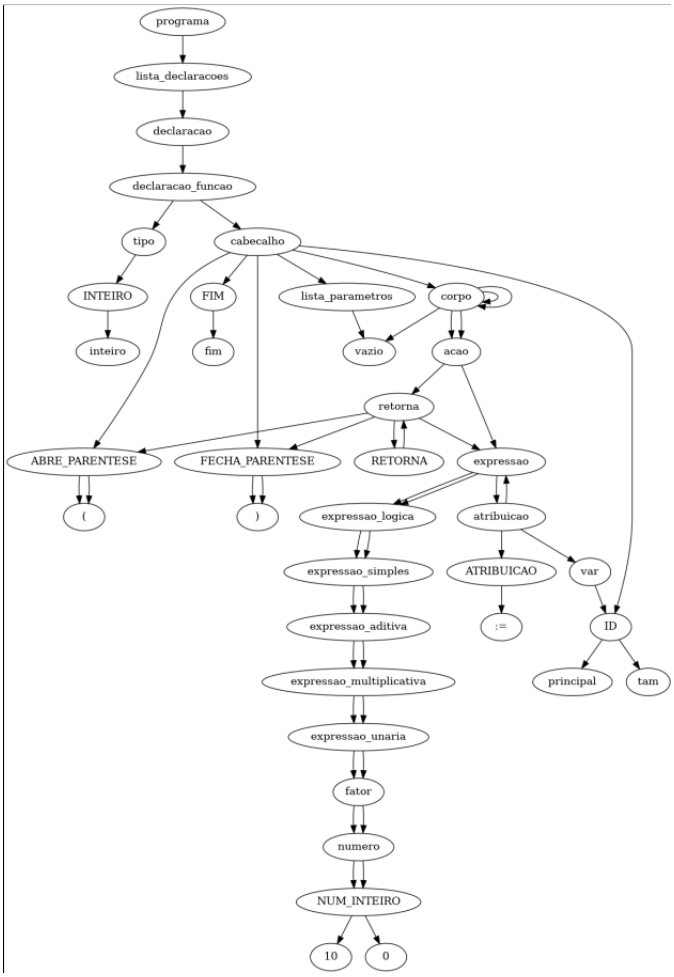

Figura 6. Árvore gerada através do código exemplo

pluginfile.php/1106666/mod_resource/content/0/
aula-15-analise-semantica-tabela-de-simbolos-tipos-de-dados-verificac
md.notes.pdf.

Gonçalves, P. R. A. (2021d). Vídeo aula 12: Análise sintática ascendente lr e lalr.