Analisador Sintático

Heitor de Lima Belém

Universidade de Brasília 160123950@aluno.unb.br

1 Motivação

Este trabalho tem como objetivo a apresentação das análises léxica e sintática, responsáveis por ler caractere por caractere de um programa, agrupar e formar *tokens* reconhecidos por uma linguagem e também por avaliar se a sequência de tokens produzidos obedecem a regras de uma linguagem (gramática). Para fixação do conteúdo obtido através do livro base da disciplina [ALSU07], foi proposto, inicialmente, o desenvolvimento dos analisadores léxico e sintático para a linguagem *C-IPL* [Nal], baseada nos princípios da linguagem *C*. O objetivo dessa linguagem é acrescentar uma estrutura de dados não existente em *C*, as listas.

Listas implementam uma coleção organizada de valores, assim como os *arrays*, entretanto, elas possuem operações especiais de acesso, adição e remoção de itens, que podem variar de linguagem para linguagem. Para a *C-IPL* [Nal], foram apresentadas operações de acesso aos elementos através dos operadores '?' e '!', operações de remoção utilizando o operador '%' e de atribuição por meio do ':'. Por fim, são descritas funções para operar sobre as listas, sendo elas: filter, representada pelo operador binário '<<' e map, representada por '>>'.

2 Descrição da análise léxica

Para implementar o analisador léxico da linguagem proposta, foi utilizado o *software FLEX* (*Fast Lexical Analyzer*), uma ferramenta geradora de programas que reconhecem padrões léxicos em texto [Est]. A estrutura de um arquivo reconhecido pelo *FLEX*, que possui a extensão .l, é dividida em três partes:

1. Definições
   Definições de funções, constantes, variáveis globais e inclusão de bibliotecas. Para este trabalho, foram criadas 3 variáveis globais: *errors_count*, *line_idx* e *column_idx*, que representam, respectivamente, a quantidade de erros obtidos durante a análise e o número da linha e coluna atual.
2. Regras
   Aqui são escritas as expressões regulares que vão procurar padrões no arquivo juntamente com as ações a serem tomadas ao encontrar tais padrões.
3. Código
   Nesta seção, encontra-se o código da função principal do arquivo com extensão *.l*, é aqui que será colocado o código gerado pelo *FLEX*.

3 Descrição da análise sintática

Para o analisador sintático deste trabalho, foi utilizado o *software Bison* [CS21], que consiste em um programa que gera, a partir das regras de uma gramática livre do contexto, um analisador sintático LR(1) canônico.

A estrutura básica de um arquivo reconhecido pelo *Bison*, que tem a extensão *.y* segue a mesma ideai do Flex, com as mesmas seções. Entretanto, na seção de regras do Bison é onde ficam as regras da gramática livre de contexto, que se assemelham à seguinte forma:

```
non-terminal
  : non-terminal terminal
  | terminal
;
```

3.1 Tabela de Símbolos

A tabela de símbolos é uma estrutura auxiliar, utilizada pelo compilador, para localizar variáveis ou funções (símbolos) utilizados durante a execução de um programa.

Para este trabalho, a implementação dessa estrutura foi realizada através da utilização de um *array* de *structs* do tipo *T_Symbol*, que armazena informações importantes para a próxima etapa do processo de compilação: a análise semântica.

As informações guardadas para cada símbolo são: conteúdo do símbolo, linha, coluna, escopo do identificador e flag para indicar se o símbolo é variável ou função.

SYMBOL TABLE

TYPE	IDENTIFIER	IS_VARIABLE	SCOPE	LINE_IDX	COLUMN_IDX
int	b	1	1	2	9
float list	c	1	1	3	16
int	funcA	0	2	1	5

Figura 1. Tabela de símbolos

3.2 Árvore sintática

Para a criaçãao da árvore sintática, foi utilizada uma estrutura de dados composta por nós não terminais e terminais. Cada nó da árvore possui informações

sobre a regra atual da gramática, campo para indicar se o nó é terminal ou não, caso não seja terminal, existe também um campo para conectar aos nós filhos.

Com essa estrutura, é possível percorrer a árvore utilizando um algoritmo de busca em profundidade, apresentando dados relevantes sobre cada nó visitado e, posteriormente, buscando informações necessárias para a análise semântica.

4 Descrição dos arquivos de teste

Os arquivos utilizados para verificar o funcionamento do analisador sintático desenvolvido no trabalho estão no subdiretório */tests*. Arquivos com o prefixo *success_*, representam os casos em que a análise sintática não identifica nenhum erro durante o processo. Já os arquivos com o prefixo *wrong_* englobam os casos em que o analisador identifica erros sintáticos.

Os erros apresentados para cada arquivo estão identificados abaixo:

- *wrong_ex1.c*

```
[PARSER] Line: 3 | Column: 9 => ERROR syntax error,
unexpected '=', expecting ';'
[PARSER] Line: 5 | Column: 5 => ERROR syntax error,
unexpected '='
```

- *wrong_ex2.c*

```
[PARSER] Line: 2 | Column: 9 => ERROR syntax error,
unexpected '=', expecting ';'
[PARSER] Line: 4 | Column: 9 => ERROR syntax error,
unexpected '=', expecting ';'
[PARSER] Line: 6 | Column: 5 => ERROR syntax error,
unexpected ';'
```

5 Compilação e execução do programa.

Requisitos para compilação: os programas *FLEX* e *BISON* (versão 2.6.4 e 3.7.6, respectivamente), o compilador *GCC* (versão 11.1.0), GNU Make (versão 4.3). Com isso devidamente instalado, é possível prosseguir para os seguintes passos.

- No diretório raiz do projeto, executar o comando:

```
$ make all
$ ./tradutor ./tests/<nome_do_arquivo>.c
```

- Se desejar executar o valgrind para verificar possíveis vazamentos de memória, basta executar os seguintes comandos:

```
$ make all
$ make valgrind ./tradutor ARGS="<caminho_arquivo_teste>"
```

Referências

[ALSU07] A.V. Aho, M.S. Lam, R. Sethi, and J.D. Ullman. *Compilers: Principles, Techniques, Tools*. Pearson/Addison Wesley, 2nd edition, 2007.

[CS21] Robert Corbett and Richard Stallman. Bison. https://www.gnu.org/software/bison/manual/bison.pdf, Online; acessado 08 de Agosto de 2021.

[Est] W. Estes. Flex: Fast lexical analyser generator. https://github.com/westes/flex. Online; acessado 08 de Agosto de 2021.

[Gup21] Ajay Gupta. The syntax of c in backus-naur form. https://tinyurl.com/max5eep, Online; acessado 08 de Agosto de 2021.

[Nal] Cláudia Nalon. Trabalho prático - descrição da linguagem. https://aprender3.unb.br/mod/page/view.php?id=464034. Acessado pela última vez em 10/08/2021.