Construção de um analisador léxico para reconhecimento dos tokens de uma linguagem hipotética

Eliton Traverssini[1], **Igor Beilner**[1]

[1]Universidade Federal da Fronteira Sul (UFFS)
Curso de Ciência da Computação – Chapecó – SC – Brasil

{eliton.traverssini, igor.beilner}@gmail.com

Resumo. *A análise léxica é uma tarefa de interesse tanto das linguagens naturais, como nas linguagens artificiais. Nas linguagens naturais, por exemplo, faz-se uso frequente de um analisador léxico aplicado em processadores de texto, já em linguagens artificiais, o interesse está em verificar se determinadas palavras pertencem à linguagem em questão. Este trabalho tem o objetivo de implementar um analisador léxico para reconhecer tokens de uma linguagem hipotética proposta ao longo deste artigo.*

1. Introdução

Um compilador está internamente divido em diversas etapas, como mostra a Figura 1, essa subdivisão de funcionalidades agrega a cada uma das fases a capacidade de modularização de cada tarefa, ou seja, a divisão das etapas de compilação faz com que se otimize ao máximo cada funcionalidade [2].

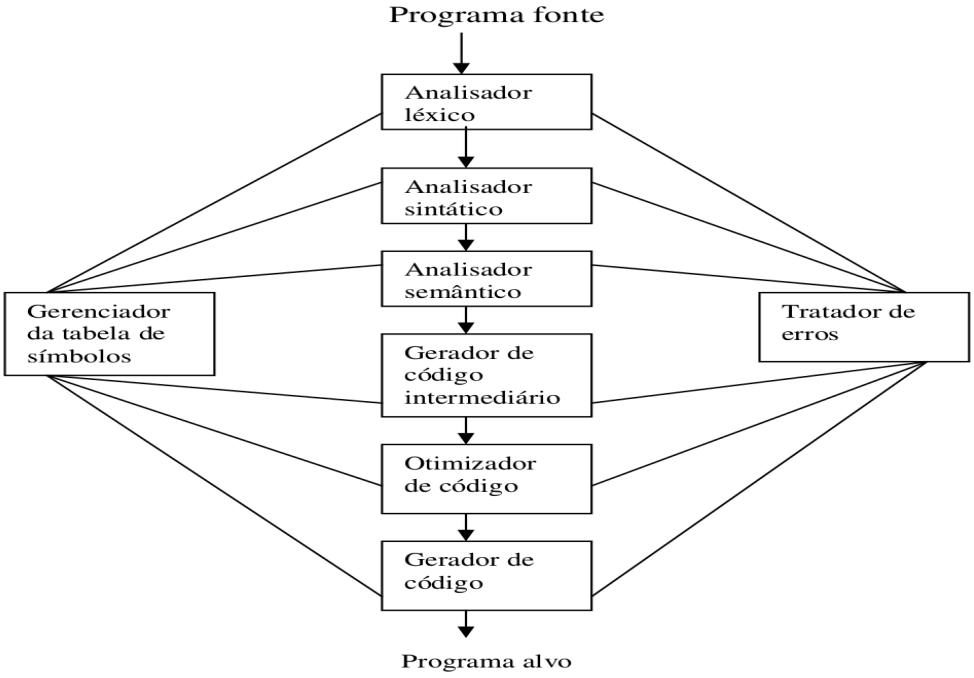

Figura 1. Etapas de um processo de compilação.

Conforme ilustra a Figura 1, a primeira fase de um processo de compilação é a análise léxica, esta etapa é responsável por analisar o código fonte e categorizar as

palvras reservadas, identificadores, marcadores e símbolos especiais em unidades lógicas denominadas *tokens*.

Neste trabalho foi construída uma *Máquina de Moore* para implementar o analisador léxico de um compilador, os detalhes da implementação e os conceitos pertinentes ao analisador léxico e à Máquina de Moore serão apresentados nas seções seguintes.

2. Fundamentação Teórica

A análise léxica é a primeira etapa que um compilador executa ao processar uma linguagem. Analisadores léxicos, são sistemas de estados finitos em que cada estado armazena o prefixo da palavra em análise [3]. Apesar da análise léxica ser a primeira etapa a ser executada pelo compilador, ela não é executada por completa para dai então iniciar a análise sintática. Como a análise sintática faz uso da fita de saída do analisador léxico, estas fases podem trabalhar em paralelo, assim, à medida que o analisador sintático necessita de tokens para serem analisados, ele solicita ao analisador léxico que verifique o código fonte e produza sua saída. Esta abordagem permite otimizar o desempenho do processo de compilação [2].

O analisador léxico pode trabalhar em duas etapas, *varredura* e *análise léxica*. A primeira etapa, é responsável por eliminar espaços em branco, tabulação e por desconsiderar comentários. A segunda etapa é que efetua de fato as tarefas mais complexas, como por exemplo, produção de *tokens* a partir de palavras reservadas.

A contrução de um analisador léxico aplica, geralmente, o conceito da máquina de Moore [1], que produz uma sequência de *Tokens* para uso do analisador sintático.

2.1. Máquina de Moore

Uma *Máquina de Moore* **M** é um Autômato Finito Determinístico (AFD) com saídas associadas aos estados. Sua definição formal é dada pela septupla :

$$\mathbf{M} = (\Sigma, Q, \delta, q_0, F, \Delta, \delta S)$$

onde:
Σ é o alfabeto de entrada.
Q é o conjunto de estados possíveis do autômato o qual é finito.
δ é a função de transição:

$$\delta\mathbf{:\ Q\ x}\ \Sigma \rightarrow \mathbf{Q}$$

$\mathbf{q}_0$ é o estado inicial, estando contido em Q.
F é o conjunto de estados finais, sendo este, um subconjunto de Q.
Δ é o alfabeto de saída.
δS é a função de saída δS**:** $\mathbf{Q} \rightarrow \Delta^*$ a qual é uma função total.

Uma Máquina de Moore aplicada a análise léxica possui um estado final associado a cada unidade léxica, onde cada estado final possui uma saída que descreve ou codifica a unidade léxica identificada e, para os demais estados a saída gerada é a palavra vazia.

3. Implementação e Resultados

O analisador léxico foi implementado com uma Máquina de Moore representada por um grafo em matriz de adjacências, onde os vértices do grafo representam os estados do autômato, as arestas são as transições entre os estados, e os pesos nas arestas são a

condição de transição. A linguagem implementada possui nove palavras reservadas, onze operadores e três separadores, o fragmento de código a seguir apresenta um exemplo de aplicação.

```
MAIN
    VAR i = 10;
    WHILE i > 0 DO
        i = i - 1;
        IF i = 2 THEN
            BREAK
        END
    END
    IF i ! 2 THEN
        i = 40;
    ELSE
        i = 20;
    END
RETURN 0;
```

Nota-se, pelo exemplo, que as palavras reservadas são case-sensitive, ou seja, são reconhecidas apenas em caixa alta, de modo contrário, os identificadores dever ser expressos em caixa baixa. Esta abordagem se deu por deixar o reconhecedor mais simples, reduzindo assim, o número de caracteres para a identificação de um *token*.

Para cada *token*, o analisador léxico imprime um símbolo na fita de saída da máquina de Moore, o qual representará a identificação do *token* encontrado antes de um símbolo separador (i.e. espaços em branco, tabulações, quebra de linha e operadores). Essa fita de saída é utilizada durante a etapa de análise sintática, a qual não é abordada neste trabalho. Na tabela 1 podemos observar os possíveis símbolos da fita de saída representando os respectivos *tokens*. Para simplificar a tabela, deve-se assumir que $[0 - 9]^+$ são constantes e $[a - z]^+$ são identificadores são quaisquer e, [erro] representa qualquer *token* não esperado pela linguagem.

Na seção seguinte é apresentado o formalismo da linguagem na qual é aplicado o analisador léxico.

3.1. Linguagem de Programação

A linguagem hipotética utilizada neste trabalho é uma Linguagem Livre do Contexto. Uma linguagem é dita livre do contexto se pode ser gerada a partir de uma Gramática Livre do Contexto (GLC) G, cuja formalidade é apresentada abaixo:

$$G = (N,T,P,S)$$

onde:
N é o conjunto dos símbolos não-terminais;
T é o conjunto dos símbolos terminais que correspondem ao alfabeto da linguagem;
P é o conjunto das regras de produção;
S denota o símbolo inicial das regras de produção.

Token	Símbolo	Token	Símbolo
MAIN	M	+	A
IF	I	-	S
THEN	T	*	X
ELSE	E	/	D
END	F	=	Q
WHILE	W	!	H
DO	D	¿	C
VAR	V	¡	N
BREAK	B	;	G
RETURN	R	[erro]	Z
(	L	)	P
$[0 - 9]^+$	K	$[a - z]^+$	J

Tabela 1. Mapeamento de tokens para um símbolo de saída.

Deste modo, a GLC para a linguagem hipotética para a aplicação do analisador léxico é descrita sob as seguintes regras:

N = {q0, q1, q2, q3, q4, q5, q6, q7}
T = [a z] + [0 9] + PR + SE
S = q0
P = {

q0 ::= MAIN q1 RETURN q2;

q1 ::= VAR q3; q1 | VAR q3=q4; q1 | q3=q4; q1

q1 ::= IF q5 THEN q6 END q1 | WHILE q5 DO q7 END q1 | ϵ

q2 ::= 0q2 | 1q2 | 2q2 | 3q2 | 4q2 | 5q2 | 6q2 | 7q2 | 8q2 | 9q2

q2 ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

q3 ::= aq3 | bq3 | cq3 | dq3 | eq3 | fq3 | gq3 | hq3 | iq3 | jq3 | kq3 | lq3 | mq3

q3 ::= vq3 | wq3 | xq3 | yq3 | zq3 | nq3 | oq3 | pq3 | qq3 | rq3 | sq3 | tq3 | uq3

q3 ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z

q4 ::= q4+q4 | q4-q4 | q4*q4 | q4/q4 | (q4) | q2 | q3

q5 ::= q4=q4 | q4!q4 | q4¡q4 | q4¿q4 | q4

q6 ::= q7 | ELSE q7

q7 ::= q1 | BREAK q1

}

4. Conclusão

Observando o comportamento do analisador léxico proposto neste trabalho, pode-se concluir que a implementação se deu de maneira correta, sendo aprovado em todos os casos de testes a que foi submetido. Em contrapartida, a construção do autômato é um ponto crítico da modelagem do reconhecedor, pois, como neste caso, houve um grande número de estados, deixando-o mais complexo para entendimento, ou até mesmo para possiveis correções ou atualizações no código fonte.

Referências

[1] Roberta C de Brito, Diogo M Martendal, and Henrique Eduardo M de Oliveira. Máquinas de estados finitos de mealy e moore. *UFSC*, 2003.

[2] Juliano Henrique Foleiss, Guilherme Puglia Assunção, EHM Cruz, RA Gonçalves, and Valéria Delisandra Feltrim. Scc: Um compilador c como ferramenta de ensino de compiladores. In *Anais do Workshop sobre Educação em Arquitetura de Computadores (WEAC 2009), São Paulo–SP*, pages 15–22, 2009.

[3] Paulo Blauth Menezes. *Linguagens formais e autômatos*. Sagra-Dcluzzato, 2000.