Trabalho de Compiladores: Desenvolvimento de um analisador léxico

Obede J. Carvalho, Otavio Lara

[1] Departamento de Ciência da Computação - Universidade Federal de Lavras (UFLA)
Caixa Postal 3037 - 37200-000 - Lavras(MG) - Brasil

obedejc@computacao.ufla.br, otavioneveslara@gmail.com

Resumo. *Uma parte fundamental do processo de compilação é a análise léxica. O seguinte artigo descreve a construção de um análisador léxico para a linguagem J–.*

1. Introdução

Um compilador tem como função transformar um código fonte em um código objeto para ser executado por um computador ou por uma máquina virtual, esse processo não é simples, é necessário o uso de técnicas e a divisão da tarefa em partes distintas: análise léxica, análise sintática e análise semântica. O presente trabalho tem como objetivo o desenvolvimento de um analisador léxico para a linguagem *J–*, uma variação simplificada da linguagem Java.

2. Referêncial Teórico

Um analisador léxico ao análisar um código fonte deve classificar cada sequência de caracteres, chamados de lexemas, em suas repectivas classes e então gerar uma estrutura chamada *token*. Classes representam um tipo especial de lexemas relacionados, elas são descritas utilizando um padrão que pode ser uma expressão regular. Tokens são uma estrutura composta por um tipo (classe) e um valor (lexema). As classes encontradas na linguagem *J–* são operadores, separadores, palavras reservadas, identificadores, inteiros, caracteres e cadeias de caracteres. Os identificadores refênciam o mesmo elemento e precisam estar relacionados entre si, uma forma de otimizar a busca por essa relação é através da tabela de símbolos que relaciona os identificadores com uma posição na tabela, e então toda vez que um identificador aparece no código fonte um token com a posição da tabela de símbolos é relacionado ao identificador. [Aho et al. 1986]

3. Descrição do Trabalho

O trabalho foi desenvolvido utilizando a linguagem de programação Java. O programa lê a entrada de um arquivo e analisa caractere por caractere. Para analisar os caracteres da entrada, foram usados uma série de instruções *if* e *while* de acordo com o automato que foi defindo para cada classe de lexema.Alguns automatos podem ser vistos a baixo. O automato principal na figura 1, foi abstraído, cada estado representa um automato específico, por exemplo o estado ¡identifier¿ representa o automato da figura 2 que reconhece os identificadores.

O programa possui a função *getNextToken()* que retorna o próximo *token* do código fonte. Os token podem ser de três tipos: *Token*, *TokenErro*, *TokenIdentificador*.

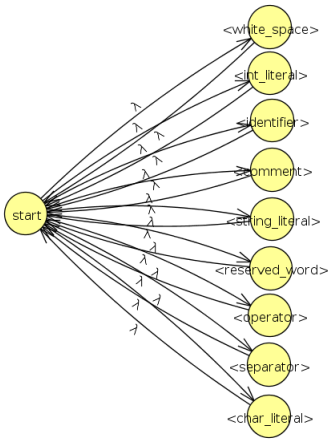

Figura 1. Automato geral

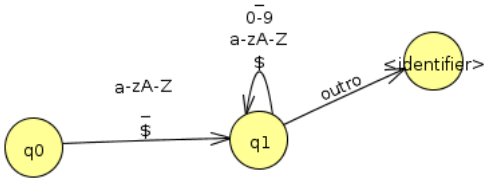

Figura 2. Automato que reconhece um identificador

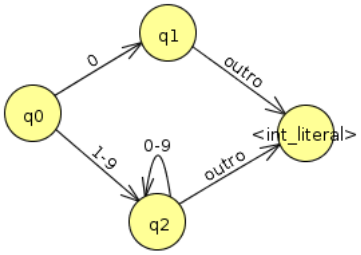

Figura 3. Automato que reconhece um inteiro

O *Token* armazena os lexemas relacionados a Palavras reservadas, Operadores, Separadores, Inteiros, Caracteres e Cadeias de caracteres. *TokenIdentificador* armazena os lexemas relacionados aos identificadores, se diferem de token por possuir endereço na tabela de símbolos dos identificadores. *TokenErro* armazena lexemas relacionados a erros léxicos como símbolo que não pertence a linguagem encontrado no código, cadeia de caracteres sem fechamento e outros erros indefinidos. Em ambos o programa conseguiu classificar todos os *tokens* corretamente e identificou todos os erros. Exemplos de erros inseridos no código fonte foram Cadeia de caracteres sem símbolo de fechamento, operadores &, / e ¡ que não são definidos na linguagem.

4. Testes e resultados

Foram realizados dois testes, um com todos os lexemas corretos e com todas as classes possíveis dentro da linguagem disponível, o outro teste foram inseridos todos os erros

possíveis identificados pelo analisador léxico. Em ambos os teste, todos os *tokens* foram reconhecidos e foram mostrados pelo programa

5. Conclusão

Conclui-se que o presente trabalho foi uma ótima fonte de aprendizado a respeito da construção de análizadores léxicos e técnicas de construção. Além de uma chance de relembrar automatos finitos.

Referências

Aho, A. V., Sethi, R., and Ullman, J. D. (1986). *Compilers: Principles, Techniques, and Tools*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.