Analisador Léxico de Java para Ruby

Francisco Laurindo Costa Jr, David Victor Cavalcante

Instituto Federal do Ceará – Campus Maracanaú (IFCE)
Caixa Postal 61939-140 – Maracanaú – CE – Brasil

Eixo da Ciência da Computação

{laucostajr,david.v.c.2013dd}@gmail.com

Abstract. *The lexical analyzer reads the source program (character by character) and returns the tokens that form it; A token knowledge a pattern of characters having some meaning in the source program like identifiers, operators, keywords, delimiting numbers and etc.*

Resumo. *O analisador léxico lê o programa-fonte (caractere por caractere) e retorna os tokens formadores deste; Um token descreve um padrão de caracteres tendo algum significado no programa-fonte como os identificadores, operadores, palavras-chave, números delimitadores e etc.*

1. Introdução

A análise léxica é o processo de analisar a entrada de linhas de caracteres (tal como o código-fonte de um programa de computador) e produzir uma seqüência de símbolos chamados “símbolos léxicos” (lexical tokens), ou somente “símbolos” (tokens), que podem ser manipulados mais facilmente por um parser (leitor de saída). A Análise Léxica é a forma de verificar determinado alfabeto, quando analisamos uma palavra podemos definir através da análise léxica se existe ou não algum caracter que não faz parte do nosso alfabeto, ou um alfabeto inventado por nós. Responsável pela busca dos componentes lexicais ou palavras que compõem o programa fonte, de acordo com regras ou padrões.A entrada do analisador léxico pode ser definida como uma sequência de caracteres.

O objetivo principal da primeira fase, a análise léxica, é identificar as sequências de caracteres que constituem as unidades léxicas, denominadas tokens. O analisador léxico lê o programa fonte caracter a caracter, verificando se os caracteres lidos pertencem ao alfabeto da linguagem, identificando os tokens e desprezando comentários e espaços em branco desnecessários. Os tokens são classificados em categorias, que podem ser basicamente palavras reservadas, identificadores, símbolos especiais e constantes. Além da identificação de tokens, o analisador léxico, em geral, inicia a construção da tabela de símbolos e gera mensagens de erro quando identifica tokens não aceitos pela linguagem em questão. A saída do analisador léxico é uma lista de tokens enumerados e com uma descrição, que é passada para a próxima fase. Muitas vezes, o analisador léxico é implementado como uma subrotina que funciona sob o comando do analisador sintático. Para a construção do analisador léxico, poderão ser utilizados em sua especificação os seguintes formalismos: expressões regulares, definições regulares e autômatos finitos.

A principal função é ler os caracteres de entrada e produzir como saída uma sequência de componentes lexicais que o analisador usa para fazer a análise. Essa

interação é geralmente aplicada convertendo o analisador léxico em uma sub-rotina ou co-rotina do analisador.

No processo de compilação, o analisador léxico é responsável pela identificação dos tokens, ou seja, das menores unidades de informação que constituem a linguagem em questão. Assim, pode-se dizer que o analisador léxico é responsável pela leitura dos caracteres da entrada, agrupando-os em palavras, que são classificadas em categorias. Estas categorias podem ser, basicamente, as seguintes: • palavras reservadas: palavras que devem aparecer literalmente na linguagem, sem variações. • identificadores: palavras que seguem algumas regras de escrita, porém podem assumir diversos valores. São definidos de forma genérica. Geralmente, as regras de formação de identificadores são as mesmas utilizadas para a formação de palavras reservadas. Nesse caso, é necessário algum mecanismo para decidir quando um token forma um identificador ou uma palavra reservada. • símbolos especiais: sequências de um ou mais símbolos que não podem aparecer em identificadores nem palavras reservadas. São utilizados para composição de expressões aritméticas ou lógicas, comando de atribuição, etc. São exemplos de símbolos especiais: “;” (ponto-e-vírgula), “:” (dois pontos), “:=” (atribuição). • constantes: podem ser valores inteiros, valores reais, caracteres ou literais. • comentário: qualquer cadeia de caracteres iniciando com e terminando com símbolos delimitadores, utilizada na documentação do programa fonte.

Pode-se assumir que os analisadores léxicos para todas as linguagens são iguais, diferenciando apenas o conjunto de palavras reservadas e símbolos especiais, que deverá ser fornecido pelo usuário. Muitas destas ferramentas são criadas a partir de notações baseadas em expressões regulares, que servem para especificar os padrões dos tokens, onde um algoritmo é utilizado para compilar tais expressões regulares em programas reconhecedores de tokens, que validam se as cadeias de entrada fazem parte ou não de uma determinada linguagem.

2. Metodologia

O desenvolvimento do analisador em Java (JVM-Java Virtual Machine) possibilita o seu uso independente de plataforma, já que a JVM funciona em diversas plataformas. Assim para utilizar o analisador é apenas ter instalado a JVM em um computador, não precisando instalar um sistema operacional inteiro para o uso da aplicação, por exemplo. O analisador léxico irá reconhecer o alfabeto da linguagem de programação Ruby, que é uma linguagem de programação interpretada multiparadigma, de tipagem dinâmica e forte, com gerenciamento de memória automático, originalmente planejada e desenvolvida no Japão em 1995, por Yukihiro "Matz" Matsumoto, para ser usada como linguagem de script.

O programa do analisador léxico recebe o nome de um arquivo que contém os tokens que devem ser analisados. O analisador submete ao autômato cada token. O autômato retorna a indicação se o token pertence à linguagem por ele reconhecida. Caso algum token não seja reconhecido, o analisador indica qual foi o token e encerra a execução. O núcleo desse programa é a funcionalidade de reconhecimento de uma string (o token) pertencente a uma linguagem regular por um autômato, figura 1.0.

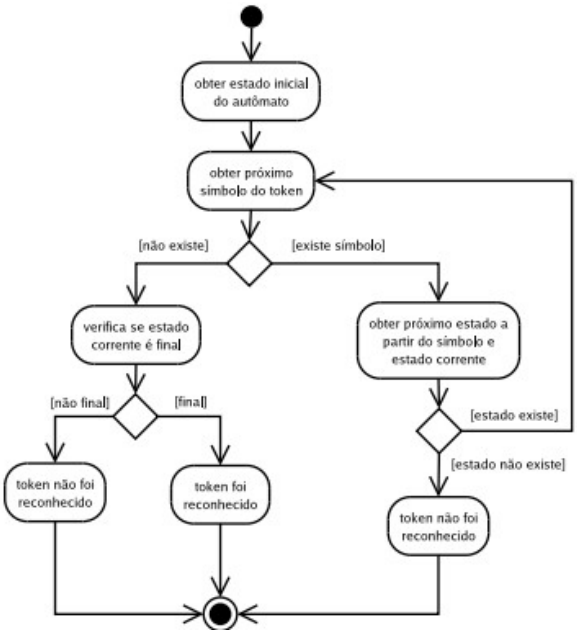

Figura 1.0 - Fluxograma do analisador léxico

A fundamentação por trás do processo de reconhecimento de tokens, pode-se definir o núcleo do analisador léxico como uma implementação de um autômato finito, que reconhece strings como símbolos válidos de uma linguagem ou dá uma indicação de que a string não pertence à linguagem.

Expressões regulares podem ser usadas para especificar exatamente quais valores são legais para os tokens assumirem. Alguns tokens são simplesmente palavras-chave, como if, else e for. Outros, como identificadores, podem ser qualquer sequência de letras e dígitos, desde que não correspondam a uma palavra-chave e não comecem com um dígito. Algumas ferramentas pré-processam e convertem em token os arquivos de origem e, em seguida, comparam os tokens lexicais com uma biblioteca de coletores.

A implementação desse analisador léxico requer uma descrição do autômato que reconhece as sentenças da gramática ou expressão regular de interesse. A forma computacional mais simples para representar o autômato é através de uma tabela de transições, a partir da qual deve ser possível obter as seguintes informações: 1. O estado inicial para o autômato; 2. Dado um estado qualquer, indicar se este é um estado final (condição de aceitação); e 3. Dado um estado qualquer e um símbolo, a indicação de qual é o próximo estado.

Além de realizar o processamento associado ao autômato, o analisador léxico tem a função de obter os símbolos a partir do arquivo com o código-fonte. Em geral, esses símbolos são oriundos de um arquivo com formato interno de texto (sequências de caracteres). O analisador léxico gerado por esse arquivo de especificação irá receber como entrada strings que eventualmente irão conter constantes inteiras. Se uma constante inteira for encontrada, ela será substituída na saída pela string correspondente especificada na ação, para cada um dos formatos de constantes reconhecidos.

O ponto de partida para a criação de um analisador léxico usando lex é criar o arquivo com a especificação das expressões regulares que descrevem os itens léxicos

que são aceitos. Este arquivo é composto por até três seções: definições, regras e código do usuário. Essas seções são separadas pelos símbolos %%. A seção de regras, onde são especificadas as expressões regulares válidas e as correspondentes ações do programa. Cada regra é expressa na forma de um par padrão-ação, pattern action. Para a descrição do padrão, jflex define uma linguagem para descrição de expressões regulares. Esta linguagem mantém a notação para expressões regulares, ou seja, a presença de um caráter indica a ocorrência daquele carácter; se é uma expressão regular, indica a ocorrência dessa expressão zero ou mais vezes; e se também é uma expressão regular, então é a concatenação dessas expressões que indica a ocorrência da expressão.

3. Resultados e Discussões

A análise léxica pode ser dividida em duas etapas, a primeira chamada de escandimento ou scanner que é uma simples varredura removendo comentários e espaços em branco, e a segunda etapa, a analise léxica propriamente dita onde o texto é quebrado em lexemas.

Três termos relacionados a implementação de um analisador léxico:

- Padrão: é a forma que os lexemas de uma cadeia de caracteres pode assumir. No caso de palavras reservadas é a sequência de caracteres que formam a palavra reservada, no caso de identificadores são os caracteres que formam os nomes das variáveis e funções.
- Lexema: é uma sequência de caracteres reconhecidos por um padrão.
- Token: é um par constituído de um nome é um valor de atributo esse último opcional. O nome de um token é um símbolo que representa a unidade léxica. Por exemplo: palavras reservadas; identificadores; números, etc.

O analisador léxico deve permitir identificar na linguagem repetições de subconjuntos permitindo que seja possível identificar e classificar esses subconjuntos, por exemplo o subconjunto de palavras reservadas. Nessa fase o processamento de uma linguagem pode ser feito por gramáticas regulares podendo ser formalmente descrito por expressões regulares. As rotinas que processam essa linguagem modelam algoritmos construídos a partir de autômatos finitos.

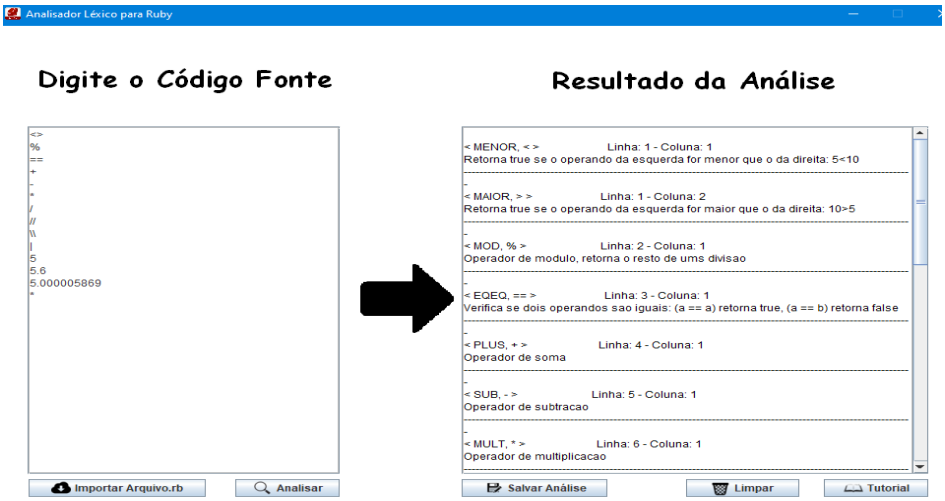

Figura 2.0 Amostra do funcionamento da aplicação

De acordo com os testes realizados o analisador léxico consegue identificar todos os tokens cadastrados durante o desenvolvimento, que consistir no alfabeto da linguagem Ruby. Todos os botões funcionaram como o desejado. Notou-se a cada analise que os resultados retornavam a representação de cada Token ou qualquer carácter que fosse inserido, retornando se o dado inserido era invalido ou fazia parte da linguagem mostrando se é uma variavel, condicional, palavra reservada entre outros possíveis resultados, como ilustra a figura 2.0 acima.

4. Referências

Aho, A. V., Sethi, R.; Ullman, J. D. Compiladores: Princípios, técnicas e ferramentas. São Paulo: Pearson, 2007. 344 p.

Louden, K. C. Compiladores: Princípios e práticas. 1ª ed. São Paulo: Thomson Pioneira, 2004. 351 p

Ricarte, I. O. (2004) “ Programação de Sistemas ”, UNICAMP, Brasil. Acesso: http://www.dca.fee.unicamp.br/~eleri/ea876/04/cap9.pdf

SETHI, Ravi; ULLMAN, Jeffrey D. Compiladores: princípios, técnicas e ferramentas. Rio de Janeiro: LTC, 1995. 344 p.

Ana Maria de; TOSCANI, Simão Sirineo. Implementação de linguagens de programação: compiladores. 2.ed. Porto Alegre: Sagra Luzzato, 2001. 194 p.

Link da Apresentação:

https://www.youtube.com/watch?v=Rjj_MuowAl8&feature=youtu.be

Link Executável/SourceCode:

https://github.com/S3V3NXD/Analisador_L-xico_Ruby

Link do Manual de Instalação e Uso:

https://drive.google.com/file/d/1pUUIrjuvI3WR5LkJdYhZTuV0eSXjouQW/view?usp=sharing