Centro Universitário de Belo Horizonte – UniBh
Departamento de Ciências Exatas e Tecnologia
Ciência da Computação
Teoria da Computação e Linguagens Formais
Professor Moisés Henrique Ramos Pereira

TPI – Implementação de um AFD
Analise Léxica

Rodrigo Reis – RA 406204771

Outubro – 2012

I – Introdução

O Trabalho proposto pelo professor Moisés Ramos, consite em implementar um analisador léxico (que é um AFD) que percorra um arquivo fonte e encontre os tokens presentes no arquivo. Os tokens que devem ser reconhecidos com seus respectivos padrões de formação de lexemas[1], são dados abaixo:

Token	Padrão de Formatação
ID	*letra(letra+digito)**
nreal	$digito^{+}.digito^{*}$
nint	$digito^{+}$
nstring	*"(letra+digito+simbolo)*"*
op	*'+' + '-' + '*' + '/' + '='*

O programa principal deverá chamar lexan. A execução do programa deve ser:

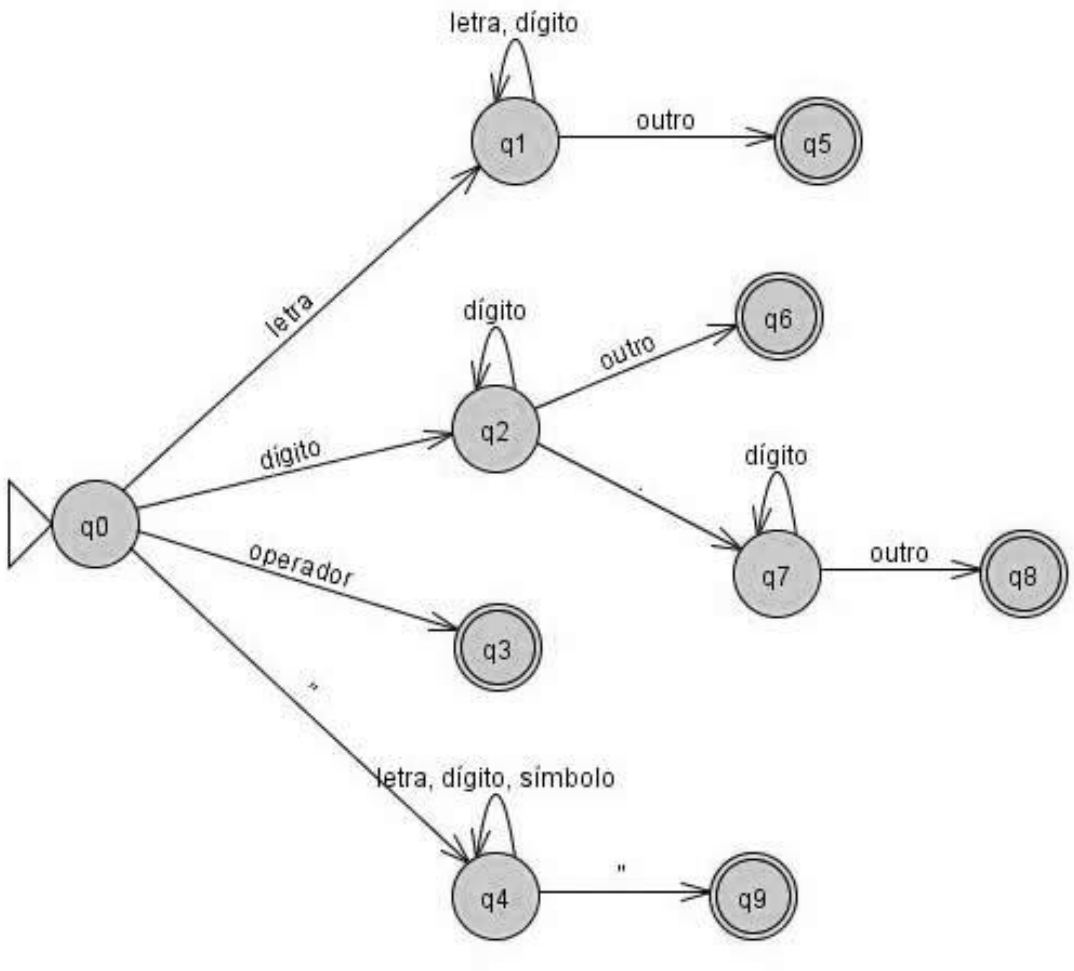
lexan <fonte>

Sendo <fonte> o endereço do arquivo a ser analisado. A saída do programa deve ser um segundo arquivo com os tokens reconhecidos.

II – Implementação

[1] Todos os nomes que atribuímos às variáveis são lexemas do token identificador, os padrões de formação dos lexemas de cada token são expressões regulares que determinam como avaliar se um lexema é de um determinado token.

Para chegar a uma solução de implementação, o primeiro passo tomando foi o criar o desenho do AFD em si,



após este, analisando o mesmo, foram criadas 3 estruturas no paradigma orientado à objetos, duas classes (uma para representar o AFD e outra para representar Alfabetos) e um enumerador (que representa os possíves estados do AFD) conforme seguem abaixo:

Estado.java

```
public enum Estado {

    Inicial,
    ID,
    nint,
    nreal,
    op,
    nstring
}
```

Alfabeto.java

```
public class Alfabeto extends ArrayList<Character> {
    public Alfabeto(String simbolos) {
        for (int i = 0; i < simbolos.length(); ++i) {
            if (!this.contains(simbolos.charAt(i))) {
                this.add(simbolos.charAt(i));
            }
        }
    }
}
```

Afd.java

```
public class Afd {
    public Alfabeto letra;
    public Alfabeto digito;
    public Alfabeto simbolo;
    public Alfabeto operador;
    public Estado estado;
    public String texto;
    Path caminho;

    public Afd(String arquivo) {
        this.caminho = Paths.get(arquivo);
        this.texto = Arquivo.texto(this.caminho.toString());
        this.estado = Estado.Inicial;
        this.letra = new
Alfabeto("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz")
;
        this.digito = new Alfabeto("0123456789");
        this.simbolo = new Alfabeto("'!@#$%¨&(),:>< ");
        this.operador = new Alfabeto("+-*/=");
    }

    public boolean letra(char caractere) {
        return letra.contains(caractere);
    }

    public boolean digito(char caractere) {
        return digito.contains(caractere);
    }

    public boolean operador(char caractere) {
        return operador.contains(caractere);
    }

    public boolean simbolo(char caractere) {
        return simbolo.contains(caractere);
    }

    public boolean letraDigito(char caractere) {
        return letra(caractere) || digito(caractere);
    }

    public boolean letraDigitoSimbolo(char caractere) {
        return letra(caractere) || digito(caractere) ||
```

```
simbolo(caractere);
    }

    public boolean aspas(char caractere) {
        return (caractere == '"');
    }

    public boolean ponto(char caractere) {
        return (caractere == '.');
    }

    public void analisar() {

        String conteudo = "\n";
        String token = "";
        String saida = "";

        for (int i = 0; i < this.texto.length(); ++i) {

            char caractere = this.texto.charAt(i);

            switch (this.estado) {
                case Inicial:

                    if (simbolo(caractere)) {
                        token = token + caractere;
                        this.estado = Estado.Inicial;

                    } else if (letra(caractere)) {
                        if (token.length() > 0) {
                            conteudo = conteudo + "UNKNOW[" +
token + "]\n";
                            token = "";
                        }
                        token = token + caractere;
                        this.estado = Estado.ID;

                    } else if (digito(caractere)) {
                        if (token.length() > 0) {
                            conteudo = conteudo + "UNKNOW[" +
token + "]\n";
                            token = "";
                        }
                        token = token + caractere;
                        this.estado = Estado.nint;

                    } else if (operador(caractere)) {
                        if (token.length() > 0) {
                            conteudo = conteudo + "UNKNOW[" +
token + "]\n";
                            token = "";
                        }
                        token = token + caractere;
                        this.estado = Estado.op;

                    } else if (aspas(caractere)) {
```

```
                        if (token.length() > 0) {
                            conteudo = conteudo + "UNKNOW[" +
token + "]\n";
                            token = "";
                        }
                        token = token + caractere;
                        this.estado = Estado.nstring;
                    }
                    break;

                // ID
                case ID:
                    if (letraDigito(caractere)) {
                        token = token + caractere;
                        this.estado = Estado.ID;
                    } else {
                        if (!simbolo(caractere)) {
                            conteudo = conteudo + "ID[" + token
+ "]\n";
                            saida = saida + "ID ";
                            token = "";
                        }
                        --i;
                        this.estado = Estado.Inicial;
                    }
                    break;

                // nint
                case nint:
                    if (digito(caractere)) {
                        token = token + caractere;
                        this.estado = Estado.nint;
                    } else if (ponto(caractere)) {
                        token = token + caractere;
                        this.estado = Estado.nreal;
                    } else {
                        if (!simbolo(caractere)) {
                            conteudo = conteudo + "nint[" +
token + "]\n";
                            saida = saida + "nint ";
                            token = "";
                        }
                        --i;
                        this.estado = Estado.Inicial;
                    }
                    break;

                // nreal
                case nreal:
                    if (digito(caractere)) {
                        token = token + caractere;
                        this.estado = Estado.nreal;
                    } else {
                        if (!simbolo(caractere)) {
                            conteudo = conteudo + "nreal[" +
token + "]\n";
```

```
                            saida = saida + "nreal ";
                            token = "";
                        }
                        --i;
                        this.estado = Estado.Inicial;
                    }
                    break;

                // op
                case op:
                    conteudo = conteudo + "op[" + token + "]\n";
                    saida = saida + "op ";
                    token = "";
                    --i;
                    this.estado = Estado.Inicial;
                    break;

                // nstring
                case nstring:
                    if (letraDigitoSimbolo(caractere)) {
                        token = token + caractere;
                        this.estado = Estado.nstring;
                    } else if (caractere == '"') {
                        token = token + caractere;
                        conteudo = conteudo + "nstring[" + token
+ "]\n";
                        saida = saida + "nstring ";
                        token = "";
                        //--i;
                        this.estado = Estado.Inicial;
                    }
                    break;
            }
        }

        if (token.length() > 0) {
            conteudo = conteudo + "UNKNOW[" + token + "]\n";
        }

        conteudo = conteudo.trim();
        saida = saida.trim();
        System.out.printf("OK..Resultado:\n\n%s\n", conteudo);

        if (this.caminho.getParent() == null) {
            try {
                String path = new
java.io.File(".").getCanonicalPath();
                Arquivo.salvar(Paths.get(path + "\\analise_"
                        + this.caminho.getFileName()), saida);
            } catch (IOException ex) {
            }
        } else {
            Arquivo.salvar(Paths.get(this.caminho.getParent()
                    + "\\analise_" +
this.caminho.getFileName()), saida);
        }
```

```
    }
}
```

Após este passo foi criado um classe principal que trata os parâmetros de linha de comando e faz a chamada para o AFD.

Lexan.java

```
public class Lexan {
    public static void main(String[] args) {

        if (args.length > 0){
            Afd automato = new Afd(args[0]);
            automato.analisar();
        }else {
            System.out.println("A execucao do programa deve
ser: "
                    + "lexan <fonte>. Onde <fonte> e o caminho
do "
                    + "arquivo a ser analisado.");
        }
    }
}
```

Além desta foi criada também uma classe de manipulação de arquivos de texto para facilitar a analise do mesmo pelo AFD.

Arquivo.java

```
public class Arquivo {

    static final int TAB = 9;
    static final int NOVA_LINHA = 10;
    static final int QUEBRA_LINHA = 13;
    static final int ESPACO_BRANCO = 32;

    public static String texto(String nomeArquivo) {
        if (nomeArquivo != null || !nomeArquivo.equals("")) {
            try {
                FileReader in = new FileReader(nomeArquivo);
                BufferedReader reader = new BufferedReader(in);
                StringBuilder buffer = new StringBuilder();

                int caractere;
                while ((caractere = reader.read()) != -1) {
                    if (caractere != TAB && caractere !=
NOVA_LINHA
                            && caractere != QUEBRA_LINHA &&
caractere != ESPACO_BRANCO) {
                        buffer.append((char) caractere);
```

```
                    }
                }
                return buffer.toString();

            } catch (Exception ex) {
                System.out.printf("Error: %s",
ex.getMessage());
                return null;
            }
        } else {
            System.out.println("Arquivo invalido.");
            return null;
        }
    }

    public static void salvar(Path arquivo, String conteudo) {
        try {
            Files.deleteIfExists(arquivo);
            Files.createFile(arquivo);
            Files.write(arquivo, conteudo.getBytes());

            System.out.println();
            System.out.printf("O Arquivo %s foi salvo com
sucesso.", arquivo.toString());
            System.out.println();
        } catch (IOException ex) {
            System.err.format("IOException: %s%n", ex);
        }
    }
}
```

Feita a implementação foram criados 3 arquivos que foram utilizados como teste, abaixo seguem os mesmo apresentando as suas respectivas saídas.

teste1.txt

```
asdas232##valor = 55555.6       + 324 + "Oi Mundo"
     valor = 55555.6       + 324 + "Oi Mundo"
```

analise_teste1.txt

```
ID op nreal op nint op nstring ID op nreal op nint op nstring
```

teste2.text

```
"tclf analise lexica" 458 +-maira12 +caros12312@#$@
```

analise_teste2.txt

```
nstring nint op op ID op
```

teste3.txt

```
"automato finito, deterministico" e uma maquina de 1.00+
estados finitos, auto$% 15 1.00e uma" 1
```

analise_teste3.txt

```
nstring ID nint op nreal ID
```

III – Código-Fonte

Foi criado um repositório para o projeto no google code no seguinte URL https://code.google.com/p/tclf-lexan/ e todo o código-fonte do mesmo pode ser baixado através de svn-checkout em http://tclf-lexan.googlecode.com/svn/trunk/

IV – Referências Bibliográficas

AHO, Alfred V. ; SETHI, Ravi ; ULLMAN, Jeffrey D.; LAM, Monica S. Compiladores: Princípios, Técnicas e Ferramentas . 2a ed. São Paulo: Pearson, 2007.

LOUDEN, Kenneth C. Compiladores: princípios e práticas . Thomson, 2004.

VIEIRA, Newton José (2004). Linguagens e Máquinas: Introdução aos Fundamentos da Computação.
WIKIPEDIA, http://en.wikipedia.org/wiki/Deterministic_finite_automaton
Deterministic Finite Automaton