##### TCC 2015 – Engenharia da Computação

**1º CAPÍTULO TEÓRICO**

##### IDENTIFICAÇÃO

|  |  |
| --- | --- |
| **NO** | **NOME** |
| **111693** | **Rodrigo Vieira da Silva** |

|  |  |
| --- | --- |
| **e-mails** | **Fone / Cel.** |
| **FACENS: 111693@li.facens.br** | **15 3213-2014** |
| **particular: rodvieirasilva@gmail.com** | **15 9 9777-1897** |

**TÍTULO:** Framework para construção de compiladores com conceitos Fuzzy

**ORIENTADOR:** Marcos Maurício Lombardi Pellini Fernandes

Data da Entrega: / /2015

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Visto do Orientador Profª. Andréa

(Verificado em \_\_\_/\_\_\_/\_\_\_\_\_ )

**SUMÁRIO**

[1 INTRODUÇÃO 14](#_Toc429760134)

[2 PRINCÍPIOS DE COMPILADORES 15](#_Toc429760135)

[2.1 Conceitos 15](#_Toc429760136)

[2.1.1 Análise Léxica 16](#_Toc429760137)

[2.1.1.1 Linguagens Regulares 17](#_Toc429760138)

[2.1.1.2 Autômatos Finitos Determinísticos 17](#_Toc429760139)

[2.1.1.3 Expressões Regulares 18](#_Toc429760140)

[2.1.2 Análise Sintática 20](#_Toc429760141)

[2.1.2.1 Gramática Livres de Contexto 21](#_Toc429760142)

[2.1.3 Outras Análises 21](#_Toc429760143)

[2.2 Ferramentas 22](#_Toc429760144)

[3 LÓGICA NEBULOSA (FUZZY) 23](#_Toc429760145)

[3.1 Conceitos 23](#_Toc429760146)

# INTRODUÇÃO

# PRINCÍPIOS DE COMPILADORES

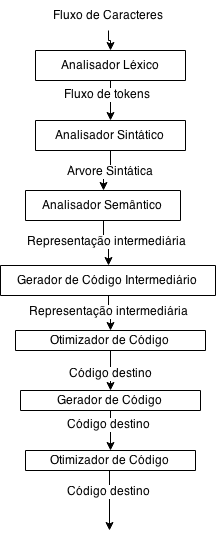
De forma abrangente um compilador converte um código fonte em um código destino. Os princípios e técnicas de construção de compiladores são utilizadas em diversas áreas de aplicação do conhecimento de um profissional envolvido com as disciplinas de computação. Com o aprendizado de técnicas básicas é possível abordar uma grande variedade de problemas quanto a tradutores de linguagens e máquinas. AHO, LAN E ULLMAN (1995)

Este capítulo abordará algumas dessa técnicas e conceitos que foram utilizados para o desenvolvimento desse projeto.

## Conceitos

Um compilador é essencial para maior eficiência na construção de um software, por possuir etapas bem definidas é possível a sua modularização e generalização em diversos itens como demonstra a figura 2.1. (AHO, LAN E ULLMAN,1995)

Figura 2.1 – Etapas de um compilador



Fonte: AHO, LAN E ULLMAN (1995)

### Análise Léxica

Análise Léxica, análise linear ou ainda scanning (esquadrinhamento) em um compilador é responsável por ler o código fonte e converter em um fluxo de tokens (palavras e tipos que compõe o texto) que será propagado para a próxima etapa, ou seja, recebe uma sequência de caracteres e gera uma lista sequencial de palavras chaves, pontuação e nomes ignorando comentários de código e espaços em brancos. (AHO, LAN E ULLMAN,1995), (RIGO ,2015).

Esse processamento é semelhante ao reconhecimento de cada palavra e sua classe gramatical – verbos, adijetivos e substantivos – de uma linguagem natural. Para realizar esse procedimento normalmente são utilizados expressões regulares e autômatos finitos. (RICARTE, 2008).

* + - 1. Linguagens Regulares

Para o correto entendimento de uma linguagem é necessário compreender alguns conceitos.

O primeiro conceito importante é o alfabeto, que se trata de um conjunto finito de símbolos, em segundo, uma cadeia que é uma sequência de símbolos que pertencem a um alfabeto, por exemplo “aoia” é uma cadeia válida sobre o alfabeto das vogais (a, e, i, o, u). Finalmente uma linguagem corresponde a um conjunto de cadeias sobre o alfabeto. Neste trabalho será apenas abordado a linguagem regulares que tem como característica possuir um autômato finito equivalente. (MACIEL, 2006).

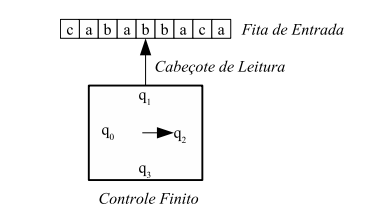
Para uma linguagem ser regular deve possuir as seguintes propriedades (MACIEL, 2006):

* Se a linguagem é igual a um conjunto vazio ou a linguagem é igual ao conjunto de qualquer símbolo pertencente ao alfabeto.
* L1 e L2 são linguagens regulares se uma linguagem regular é obtida a partir da união entre as duas linguagens.
* L1 e L2 são linguagens regulares se uma linguagem regular é obtida pela concatenação entre as duas linguagens.
* Se L1 é regular então a linguagem obtida a partir da concatenação de zero ou mais cadeias da linguagem L1 também é regular.
  + - 1. Autômatos Finitos Determinísticos

O autômato finito determinístico (AFD) é o modelo computacional existente menos complexo, são capazes de processar informações, recebendo uma entrada e exibindo uma saída, não possuem uma memória auxiliar e apesar disso ele é totalmente adequado à função de reconhecimento de cadeias de texto. (MARCIEL, 2006).

A Figura 2.2 demonstra a composição de um AFD, fita de entrada, cabeçote de leitura e um controle finito que é um conjunto de estados distintos.

Figura 2.2. – Autômato Finito Determinístico



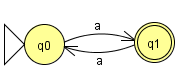
Fonte: MACIEL, 2006.

Um AFD é formado por uma quíntupla com a seguinte definição (AHO, LAN E ULLMAN ,1995):

1. Conjunto finito de estados.
2. Conjunto de símbolos de entrada.
3. Função de transição, que consiste na escolha de um próximo estado a partir de um símbolo e de um estado atual.
4. Estado inicial
5. Conjunto de estados finais

Pode-se representar um AFD através de um grafo de transição, como demonstra a figura 2.3.

Figura 2.3 – Grafo de transição AFD



* + - 1. Expressões Regulares

Entendido os conceitos de linguagens regulares e de um AFD é possível aplica-los para definir uma expressão regular e realizar a sua validação computacionalmente.

Uma expressão regular representa um padrão de cadeia de caracteres, ela é definida pelo conjunto de cadeias que “valida”. (LOUDEN, 2004).

Existem três principais operações em expressões regulares (LOUDEN, 2004):

1. Operação “OU”, normalmente representada pelo caractere ”|” (barra vertical);
2. Concatenação ou sequência;
3. Repetição, que normalmente é representada pelo caractere “\*”;

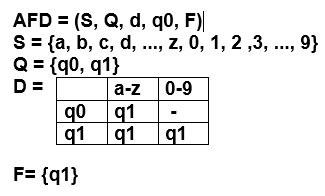
A seguir um exemplo onde são demonstrados uma expressão regular, sua linguagem regular correspondente e o grafo do autômato determinístico gerado a partir da expressão.

Figura 2.4 – Exemplo de expressão regular



Na figura 2.4 é possível observar uma expressão regular que valida uma cadeia que comece com um caractere que esteja entre o intervalo de “a”-z” e que termine com zero ou mais caracteres entre “a”-z” ou “0-9”.

Figura 2.5 – Definição Exemplo 1



A figura 2.5 monta a quíntupla que define um autômato finito determinístico onde:

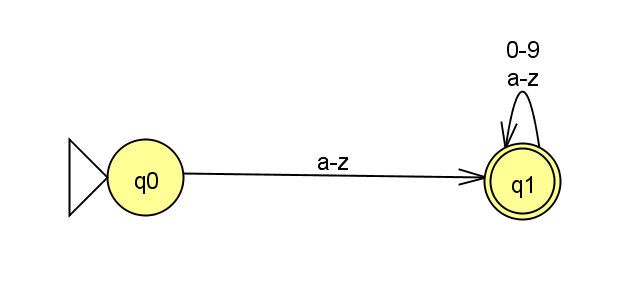
“S” é o alfabeto de símbolos que contempla nos intervalos de ‘a’ a ‘z’ e de ‘0’ a ‘9’.

“Q” é o conjunto de estados que formam o autômato.

“D” é a função de transição, que pode ser definida em formato de tabela, onde as linhas representam os estados de origem que ao consumir um símbolo da coluna da linha superior passa para um novo estado da célula correspondente.

Finalmente ”F” é o conjunto de estados finais representado pelo estado q1.

Figura 2.6 – Grafo Transição Exemplo 1



Na figura 2.6 é possível observar o grafo de transição do AFD correspondente a expressão regular do exemplo 1.

No exemplo 1 podemos validar um tipo de token de identificador (como nomes de variáveis, métodos), muito utilizado na maioria das linguagens de programação. (LOUDEN, 2004).

Existem diversas técnicas e algoritmos para minimização e simplificação de autômatos finitos determinísticos que não serão apresentados nesse trabalho, já no capítulo 4 será descrito as diversas alterações realizadas nos algoritmos de processam­ento e conversão de uma linguagem regular para um autômato.

­

### Análise Sintática

O analisador sintático é o responsável por gerar a árvore sintática a partir do fluxo de tokens que o analisador léxico montou, emitindo qualquer erro sintático encontrado durante o processamento. Para realizar o seu papel o analisador necessita de uma gramática representativa. (AHO, LAN E ULLMAN ,1995).

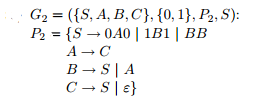
Existem diversas classificações de gramáticas segundo a classificação de Chomsky, neste trabalho será abordada as gramáticas livres de contexto.

2.1.2.1 Gramática Livres de Contexto

Uma gramática livre de contexto é formada por uma quádrupla com a seguinte definição (AHO, LAN E ULLMAN ,1995):

1. Conjuntos de variáveis ou não-terminais
2. Conjunto de símbolos terminais
3. Conjunto de produções, onde uma produção contém um lado esquerdo, composto por uma variável e um lado direito, que pode conter um conjunto de variáveis e terminais.
4. Uma varável inicial

Figura 2.7 – Exemplo gramática



Fonte: SAKATA, 2015.

A figura 2.7 demonstra a formalização de uma gramática, onde o conjunto de variáveis é composto por ‘S’, ‘A’, ‘B’, ‘C’, o conjunto de símbolos terminais possui os números 0 e 1, o conjunto P2 representa as diferentes regras de derivações e finalmente o S é a variável inicial.

As GLCs (Gramáticas Livre de Contexto) são capazes de representar a sintaxe de uma linguagem de programação e, assim, nessa etapa é possível a validação e geração da árvore de derivação. AHO, LAN E ULLMAN (1995).

### Outras Análises

Conforme demonstrado na figura 2.1, compiladores possuem diversas etapas bem definidas além dos módulos tratados nesse capítulo, porém sua generalização acaba se tornando muito mais complexa.

Neste trabalho na ferramenta disponibilizada será abordado apenas

## Ferramentas

Existem diversas ferramentas que auxiliam nas etapas bem definidas de um compilador, segue alguns exemplos:

*Flex (Faster Lex):* Ferramenta de geração de analisadores léxicos e sintáticos.

*Antlr (ANother Tool for Language Recognition):* Gerador de análises léxica, sintática e análise semântica (esse último bem limitado).

*Yacc (Yet Another Compiler-Compiler*): Programa criador de compiladores

*Xtext*: *Framework* baseada em *JAVA* para desenvolvimento de compiladores

**REFERÊNCIAS**

AHO, A. V.; Lam, M. S.; Sethi R.; Ullman, J. D. **Compiladores, Princípios, técnicas e ferramentas**. 2.ed. São Paulo: Pearson Education do Brasil, 2008. 633 p.

RIGO, S. Análise Léxica. Disponível em: <http://www.ic.unicamp.br/~sandro/cursos/mc910/slides/cap2-lex.pdf>. Acesso em: 05 setembro 2015.

RICARTE, I. **Introdução a Compilação.** 1.ed. Rio de Janeiro: Elsevier, 2008. 258 p.

MARCIEL, A. **Aplicação de autômatos finitos nebulosos no reconhecimento aproximado de cadeias**. 2006. 63 f. Dissertação (Mestrado em Sistemas Digitais) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2006.