FACULDADE DE ENGENHARIA DE SOROCABA
CAMPUS ALEXANDRE BELDI NETTO
Reconhecida pela Portaria Ministerial n° 367 de 03/06/1980
MANTIDA PELA
Associação Cultural de Renovação Tecnológica Sorocabana
Declarada de Utilidade Pública Federal - Decreto n° 86.431 de 02/10/1981

TCC 2015 – Engenharia da Computação

DOCUMENTO DE PROJETO

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: 08 / 06 /2015

_____

Visto do Orientador

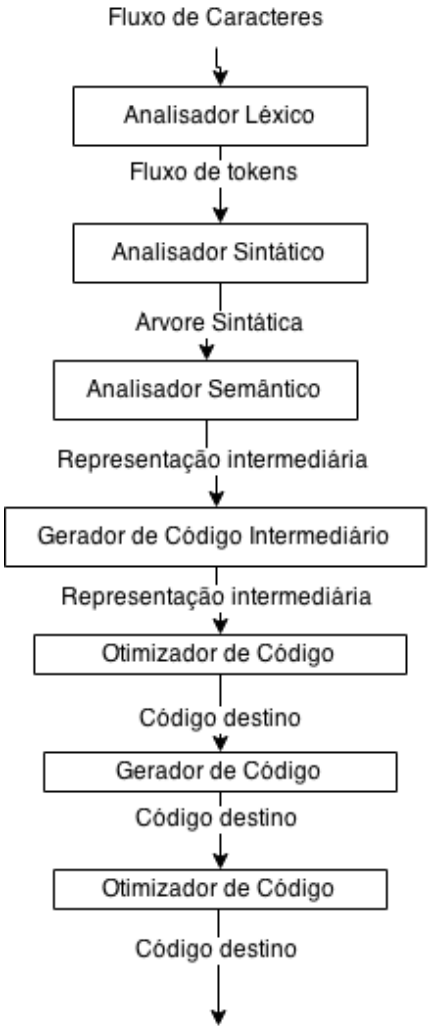
Sumário

1. CENÁRIO

1.1 Compilador

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 1.

Figura 1 – Diferentes etapas de um compilador

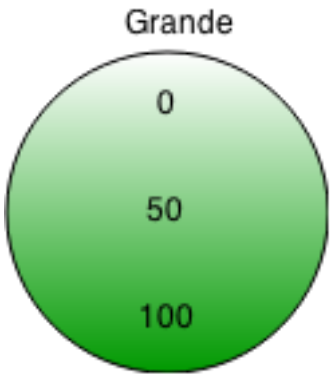


1.2 Lógica *Fuzzy*

A Lógica *fuzzy* ou lógica nebulosa, diferente da lógica clássica que possui valores lógicos 0 e 1 (pertence ou não pertence), admite valores intermediários entre 0 e 1, assim sendo um grau de pertinência de determinado valor a um determinado conjunto.

A figura 2 demonstra as diferentes pertinências dos valores 0, 50 e 100 no conjunto “Grande”, onde branco representa pertinência 0 (mínima) e verde pertinência 1 (máxima).

Figura 2 – Pertinências de 0, 50 e 100 no conjunto Nebuloso “Grande”.



1.3 Conceitos *Fuzzy* aplicado a compiladores

Esse trabalho possui como objetivo unir os dois conceitos apresentados aplicando a lógica *fuzzy* nas seguintes etapas de compilação:

1.3.1 Análise Léxica

É a primeira fase de um compilador, responsável por ler o fluxo de caracteres que compõe o arquivo de entrada e os agrupar em sequências significativas em que cada elemento da sequência possui um *token* (Nome e valor).

Ao aplicar o conceito *fuzzy* será substituído o autômato convencional utilizado para um autômato *fuzzy*, com essa alteração o analisador será capaz de aceitar erros léxicos, identificando o *token* que mais se aproxima com a palavra errada.

1.3.2 Análise Sintática

É a segunda fase do compilador, a partir dos *tokens* gerados da análise léxica essa etapa é responsável por gerar uma representação intermediária em formato de árvore.

Com os conceitos *fuzzy* aplicados o algoritmo de validação sintática será trocado por um algoritmo capaz de aceitar e se recuperar de erros sintáticos, para isso ao invés de uma gramática livre de contexto convencional será necessário a definição de uma gramática livre de contexto *fuzzy*.

1.3.3 Gerador de Código

A partir de uma representação intermediária essa camada realiza a conversão de um código origem em um código destino.

Será disponibilizado uma interface simples para que a geração de código seja facilitada para os usuários da *framework*.

2. PESQUISA DE MERCADO

Embora existam conceitos de aplicação de lógica *fuzzy* em compiladores, nenhuma obteve tanta visibilidade quanto as ferramentas de auxílio a compilação tradicionais. Segue alguns exemplos:

2.1 Ferramentas tradicionais

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

Antlr (ANother Tool for Language Recognition): Gerador de análises léxica, sintática e semântica.

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

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

2.2 Conceitos Fuzzy

Flex (Fuzzy Lexical Analyser): Conceito proposto por Alexandru Mateescu, Arto Salomaa, Kai Salomma e Sheng Yu publicado em 14/05/1995 no *JUCS ("Journal of Universal Computer Science").*

Representação Gramatical Nebulosa Livre de Contexto: Conceito que envolve o reconhecimento *fuzzy* de gramáticas, proposto por diversos autores nos últimos anos.

3. TESTE DE HIPÓTESE

Existem diversas plataforma livres por que a escolha de ambiente *.NET* e linguagem de programação C#?

A *Microsoft* surpreendeu a todos ao abrir os fontes da *framework .NET,* porém a plataforma ainda continua proprietária. Esse ambiente foi escolhido devido à ausência de ferramentas para a mesma, a facilidade no desenvolvimento e o conhecimento prévio adquirido. Todavia o mais importante da biblioteca será os algoritmos desenvolvidos que poderão ser facilmente migrados para outras plataformas.

Qual a aplicação prática da ferramenta e dos conceitos *fuzzy* abordados nos algoritmos?

Com os conceitos *fuzzy* em uma framework será possível gerar um compilador suscetível a erros e com isso existem diversas possíveis aplicações práticas:

- Após o *OCR* (*Optical Character Recognition*) – Posteriormente ao reconhecimento de caracteres será possível passar por um compilador para validar e corrigir a sentença lida, conforme os parâmetros esperados
- Linguagem de programação para iniciantes – Uma linguagem simplificada que tolera e corrige erros de programadores iniciantes auxiliando-os e ensinando-os.
- Acesso a informações em banco de dados – Uma ferramenta que interprete uma linguagem natural e assim qualquer usuário final conseguirá consultar,

sem a necessidade de conhecer a ferramenta em que os dados estão guardados

- Automação doméstica – Reconhecedor de comandos de linguagens naturais para ser interpretados por um *hardware* que controle os componentes eletrônicos domésticos.

Enfim a ferramenta poderá ser abordada nas mais diversas aplicações e quando a linguagem de entrada é uma linguagem natural os resultados se tornam ainda mais interessantes.

4. O QUE SERÁ FEITO

Na primeira apresentação será demonstrado a implementação de automatos *Fuzzy*, o reconhecimento parcial de cadeias de textos e por fim um protótipo da *framework* com uma aplicação de testes, com as seguintes etapas parcialmente desenvolvidas:

- Análise Léxica *Fuzzy*
- Análise Sintática LR(1) *Fuzzy* Modificada
- *Parser* Sintático para código destino

Ao término do projeto será disponibilizado um exemplo completo de utilização da *Framework* e diversos testes de gramáticas para demonstrar o correto funcionamento da biblioteca.

5. O QUE NÃO SERÁ FEITO

Nesse trabalho não será abordado um estudo sobre performance e otimização de algoritmos nas diferentes análises.

6. BENEFÍCIOS

Embora existam diversas ferramentas que auxiliam na criação de compiladores a sua configuração e utilização complexa acaba desmotivando os desenvolvedores a criar um compilador (seja para estudo ou para aplicações do mercado), uma vez a ferramenta

disponibilizada e tendo como premissa a simplificação os desenvolvedores se sentirão mais confortáveis quando for necessário a construção de um compilador.

Ainda como benefício está a disponibilização de uma ferramenta com conceitos *fuzzy* completa, as aplicações disponíveis que possuem o conceito não abordam todo o processo de compilação.

7. METAS GLOBAIS

As principais metas desse trabalho é o desenvolvimento de uma ferramenta de fácil extensão e utilização para a construção de compiladores e o estudo dos conceitos *fuzzy* aplicado a diferentes partes do processo de compilação.

Demonstrar a utilização da *framework* e disponibilizar como software livre na comunidade.

8. METAS INTERMEDIÁRIAS

Comparação entre as diferentes ferramentas do mercado que auxiliam a criação de compiladores.

Comparação entre os algoritmos que abordam análises sintática e léxica com conceitos *fuzzy*.

9. RECURSOS UTILIZADOS

Durante o desenvolvimento do projeto será utilizado a plataforma .NET, *IDE (Integrated Development Environment*) *Visual Studio 2013*, C# como linguagem de programação, ambiente *desktop* (Área de trabalho Windows) para demonstração de exemplos e aplicações.

Ainda será utilizado *XML(eXtensible Markup Language)* como linguagem para configuração e definição da gramática e demais parâmetros da ferramenta.

10. BIBLIOGRAFIA

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.

BEDREGAL, B. C.; CORREA, C. λ-ALN: autômatos lineares não-determinísticos com λ-transições. In: XXXIII CNMAC, 33. 2010, Águas de Lindóia, SP. **Anais.**; São Carlos, SP; 2011.

FUINI, M. G. **Sistema de Recuperação de Imagens Baseado na Teoria Computacional das Percepções e em Linguagens Formais Fuzzy.** 2006. 106 f. Dissertação (Mestrado em Engenharia de Computação) – Universidade Estadual de Campinas, Campinas, 2006.

JIN, J.; LI, Q.; LI, C. On Intuitionistic Fuzzy Context-Free Languages. **Hindawi Publishing Corporation Journal of Applied Mathematics** New York, NY, v. 2013 p. 1-16, dez. 2012.

LOUDEN, k. L. **Compiladores, princípios e práticas**. 1.ed. São Paulo: Cenage Learning, 2004, 569 p.

RIGNEL, D. G. de S.; CHENCI, G. P.; LUCAS, C. A. Uma Introdução a lógica Fuzzy. **Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica**, v. 01, n. 1 p. 1-28, dez. 2011.

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.

SAKATA, T. C. Análise Sintática Descendente. Disponível em: <http://www.li.facens.br/~tiemi/Tc1/analise-desc.pdf>. Acesso em: 02 fevereiro 2015.

SAKATA, T. C. Análise Sintática LR(1) e LALR. Disponível em: <http://www.li.facens.br/~tiemi/Tc1/analise-lr1.pdf>. Acesso em: 03 fevereiro 2015.

SANDRI, S.; Correa, C. Lógica Nebulosa. In: V Escola de Redes Neurais, 5. 1999, São José dos Campos, SP. **Anais.**; São José dos Campos, SP; ITA, 1999.