1

Bom dia, meu nome é Rodrigo Vieira da Silva. Apresentarei o meu trabalho de conclusão de concurso que tem como tema o desenvolvimento de uma framework para construção de compiladores com conceitos fuzzy. Meu orientador é o professor Marcos Maurício Lombardi.

2

Essa é a agenda da apresentação, onde começarei falando um pouco dos objetivos do trabalho que está sendo desenvolvido, em seguida comentarei sobre as possíveis aplicações que a ferramenta proporciona, logo após comentarei sobre o status atual do projeto e os conceitos fuzzy que foram aplicados até o momento.

Posteriormente sigo demonstrando o protótipo e falando sobre os próximos passos que serão abordados no próximo semestre.

E finalmente comentarei sobre as contribuições que o projeto fornecerá para a comunidade de desenvolvedores.

3

Do título do meu trabalho extraio os principais conceitos que expõe os objetivos:

Framework: Conjuntos de bibliotecas e ferramentas que auxiliam no desenvolvimento de alguma aplicação.

Construção Compiladores: Um conjunto de ferramentas que auxiliam na definição e nas diversas etapas de um compilador, essas etapas comentarei mais para frente

Conceitos Fuzzy: Mas existem diversas ferramentas que auxiliam na construção de compiladores, o grande diferencial é abordagem de conceitos fuzzy, ou seja.

Uma ferramenta que auxilie a criação de compiladores tolerantes a erros

Mas para quê um compilador tolerante a erros?

4 Do mesmo modo que com um compilador convencional é possível o desenvolvimento de infinitos programas as aplicações de um compilador fuzzy são inumeráveis. Aqui temos alguns exemplos:

4.1– Linguagem de programação Iniciantes

É comum ouvir de professores e monitores que estudantes dos primeiros anos esquecem de muito de ponto e vírgula e correto fechamento de parênteses. Um compilador que seja tolerante a falhas e capaz de corrigir e mostrar esses tipos de erros, facilitaria o aprendizado da lógica em si.

4.2 – Consulta base de dados

Quando algum funcionário de uma empresa cliente de uma fábrica de software precisa de um relatório específico, dependendo do tamanho da empresa o processo é bem burocrático.

Uma ferramenta de processamento de linguagem natural inteligente, onde o próprio usuário poderia montar os seus relatórios, suas consultas seria um grande diferencial no mercado.

4.3 Automação doméstica

Uma outra possível aplicação, seria um assistente doméstico, capaz de interpretar comandos por voz dos residentes e aplicar nos eletrodomésticos que estão conectados.

Mas como o projeto facilita essas aplicações?

Para responder isso, comentarei um pouco sobre o status atual do projeto.

5

Nessa imagem temos as diversas etapas de um compilador,

Começa com o fluxo de caracteres de entrada ou código de origem, passando por diversos analisadores e geradores e finalmente chegando no código de destino.

O projeto abordará as seguintes etapas Análise Léxica, Análise Sintática e Gerador de Código

Análise léxica é a primeira etapa da compilação, ela responsável por quebrar o texto em tokens, ou seja, em um conjunto de chave e valor. Exemplo na frase:

Compiladores são práticos

Substantivo, verbo, adjetivo

Responsável por validar se as palavras fazem parte da linguagem e gerar uma lista desses tokens que são propagados para o próximo passo.

Análise sintática

Na mesma frase é preciso validar se os token estão na ordem correta, ou seja, nesse caso sujeito verbo predicado. Além de fazer isso o analisador gera uma árvore sintática que é propagada para as outras camadas.

Gerador de código é o responsável por criar o código de destino, no projeto será desenvolvido um conjunto de classes que facilitará esse processo, o usuário final terá apenas que informar o método de geração de código para cada regra. Explicarei com mais detalhes no protótipo.

Aplicando conceitos fuzzy nessas etapas temos:

Análise Léxica Fuzzy:

Enquanto o analisador convencional utiliza de um autômato o analisador léxico fuzzy precisa de um autômato fuzzy, como principal diferença vemos as pertinência em cada transição.

Aqui, de q0 para q1 com a letra ‘b’ vemos uma pertinência de 1, no entanto para qualquer outra letra do alfabeto temos uma pertinência 0.8. Esses conjuntos de transições caracterizam as 3 operações em cadeias: Remoção, Inserção e Substituição de caracteres.

Já o analisador sintático utiliza de uma gramática, o analisador fuzzy precisa de uma gramática fuzzy, aqui está a definição. Onde Vn é o conjunto de Variáveis

Vt é o conjunto de terminais

P conjunto de produções e S variável inicial.

A grande diferença está em cada regra, onde alpha é o valor de pertinência daquela regra que pode estar entre os valores de 0 e 1 .

Primeiramente vou mostrar o autômato fuzzy utilizado no analisador léxico.

Foi feito uma pequena aplicação para debugar e mostrar visualmente todos os recursos da framework. Vemos a pertinência de cada passo do algoritimo de validação do automato.

Agora vemos aqui um pequeno exemplo de validação sintática.

Aqui temos a árvore sintática.e mesmo errando os tokens ou sem um deles ainda ocorre a validação. Esse quadro é o grande diferencial, além do código destino temos o código que corrigido.

Finalmente mostrarei um exemplo completo de utilização um pouco mais complexo.

Como mencionei nas aplicações é gerador de relatórios onde digito a consulta em uma linguagem mais simples para usuários finais.

Esse é o que foi feito até agora, vamos voltar aos slides para ver os próximos passos,

Durante a pesquisa de gramáticas fuzzy acabamos esbarrando em outra, a gramática probabilística, é uma outra forma de abordar o problema de tolerância a falhas. Um dos algoritmos analisadores é o CYK, ao término do projeto será possível comparação entre os dois algoritmos e perceber qual teve melhor resultado.

Após isso então será desenvolvido diversos testes e por fim uma aplicação complexa, que demonstre todo o poder da framework.

Finalmente o projeto será opensource, tem como premissa a simplificação na utilização e ser completa. Embora existem alguns exemplos de conceitos fuzzy em compiladores, ainda não tem uma framework completa

Ao término , quando for disponibilizado na comunidade trará essa série de benefícios;

Muito obrigado.