UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

SISTEMAS DE INFORMAÇÃO

FUNDAMENTOS DE COMPILADORES

COMPILADOR Cmm

Trabalho apresentado como parte das avaliações parciais da disciplina Fundamentos de Compiladores do curso de Sistemas de Informação do Campus I da UNEB.

FABIO RAMOS DA SILVA CONCEIÇÃO

TÁRCIO ALMEIDA DOS SANTOS

2017.2

1. Introdução

O Projeto do Compilador Cmm consiste na construção de um compilador, seguindo as fases de planejamento, aquisição de material teórico, testes e, finalmente, a implementação. Cada parte do compilador (Léxico, Sintático, Semântico e Geração de Código) passou por essas fases. A nossa dupla seguiu essas fases para a construção do compilador e dedicou por volta de 2 semanas para construção de cada parte do compilador (com exceção da última, que dedicamos 2 semanas para semântico e geração de código).

2. Análise Léxica - Concluída

O Analisador Léxico é o ponto de partida de um projeto de compiladores. É ele quem tem contato com o arquivo fonte da linguagem. Conforme vai lendo este arquivo, através de chamadas feitas pelo analisador sintático, ele vai criando os tokens. Um token é uma unidade lógica dotada de significado de uma determinada linguagem. Cada elemento que faz parte da linguagem é identificado, classificado e organizado nessa etapa e o papel do analisador léxico é validar se esses elementos pertencem a gramática. Os autômatos são as estruturas utilizadas para definir se as cadeias pertencentes à uma linguagem, eles apenas dizem “sim” ou “não” ao fato de uma cadeia pertencer à uma linguagem. Caso pertença, então o token será montado.

O primeiro desafio nessa etapa foi imaginar uma estrutura de comandos que desse suporte a implementação de um Autômato Finito na linguagem C. Após reflexões, foram utilizadas estruturas facilitadoras dessa implementação. No quesito estrutura do dado foi escolhido o registro para materializar a existência de um token e seus campos necessários. Já no quesito comandos, foram escolhidos os laços de repetição “while” em paralelo a utilização do “switch”, que possibilita “caminharmos” entre os diversos estados de um autômato finito. Depois dessas escolhas, ficou muito claro o papel que tínhamos que fazer e ele foi concluído de forma clara.

Por fim, todos os requisitos solicitados nessa etapa foram cumpridos.

3. Análise Sintática - Concluída

O Analisador Sintático se refere a parte de checagem de estruturas. Conforme definido na gramática da linguagem e suas regras de produção, são feitas checagens. Essas checagens são feitas concomitantemente ao processo de leitura de tokens, pelo Analisador Léxico. Ou seja, a cada leitura de token, o analisador sintático vai seguir um caminho esperado e definido nas regras de produção, de acordo com que aquele token signifique para a linguagem. Caso o caminho não esteja dentro das regras da linguagem, é dado erro sintático.

Na implementação deste Analisador Sintático foram encontrados alguns desafios. O primeiro foi abstrair como o analisador sintático iria trabalhar com o analisador semântico. Teve que ser feita uma adaptação para que, a cada chamada da função “analex()”, o Analisador Léxico caminhasse

“um token” no arquivo fonte. Após esse entendimento, ficou menos complicado desenhar o caminho que seria seguido de acordo com o token lido. Outro desafio foi a existência de caminhos parecidos: existiam, de acordo com as regras de produção da gramática, caminhos que, para se diferenciarem, era necessária a leitura não só do token atual, mas também do próximo token para determinar o caminho que iria seguir. Quando entendemos esta parte, mais uma barreira de desafio foi rompida e conseguimos caminhar com mais facildade no projeto. O último desafio encontrado foi perceber que várias possibilidades poderiam ocorrer dentro de apenas um não-terminal. Ou seja, a exemplo do comando “para”, tudo que estava dentro do parêntesis poderia ocorrer ou não, sendo possível apenas a utilização do comando da seguinte forma “para(;;)”.

Por fim, vencidos os desafios, conseguimos concluir a codificação do analisador semântico respeitando todas as regras de produção associadas ao projeto da Linguagem CMM.

4. Análise Semântica – Parcialmente Concluída

A Análise Semântica consiste em verificar se as regras de consistência da linguagem estão sendo cumpridas. Ela irá verificar se o código está respeitando as regras de visibilidade e consistência dos identificadores. Também é esperado que seja verificada que cada expressão tenha um tipo adequado as regras da linguagem. Além disso, os escopos são consagrados, havendo uma clara separação entre qual parte é GLOBAL e qual parte é LOCAL. Uma importante estrutura utilizada é a Tabela de Símbolos, que funciona como uma pilha. Em tempo de compilação essa pilha vai crescendo e diminuindo conforme o analisador sintático vá caminhando nos tokens. Essa tabela armazena identificadores de funções, variáveis e parâmetros.

Alguns desafios foram encontrados na hora da implementação da Análise Semântica. Podemos dizer que o mais marcante foi abstrair a utilização do DDS na checagem de tipos até a função de atribuição. Foi necessário um tempo muito grande para entender como a recursão funcionava, montando árvores que eram produzidas de acordo com as regras de produção da gramática. Além disso, entender como fazer os atributos necessários chegassem ao topo da árvore também foi um empecilho. Outro desafio foi entender a importância da Tabela de Símbolos e o escopo nessa etapa do processo. Todas as regras de checagem de declaração de atributo, função; se o parâmetro estava no lugar correto; se o retorno está correto; etc. se aproveitavam da estrutura da Tabela de Símbolos e do seu funcionamento de pilha para acontecer de forma correta. Após ter esse entendimento, o processo de forma menos dificultosa.

Por fim, vencidos os desafios e feitas as abstrações, as regras semânticas estabelecidas foram implementadas na sua maioria. Ficamos em dúvida sobre como implementar a checagem de parâmetros do tipo “tipos_p_opc” pois esse pode não haver a inserção de ids na tabela, apenas os tipos. Dado esse desafio, ficamos presos nessa etapa, sem sucesso após várias tentativas de correção.

5. Gerenciador de Tabela de Símbolos e Gerenciador de Erros - Concluídos

O gerenciador de Tabela de Símbolos é a interface entre os Analisadores e a Tabela de Símbolos. A estrutura de dados escolhida foi a indicada pelo professor, é uma Tabela, porém se comporta, na maioria das vezes, como uma pilha. Há operações de empilhar, desempilhar e consultas. É essa estrutura, a Tabela de Símbolos, que irá armazenar os identificadores em tempo de compilação. Só percebemos a importância fatídica desta na implementação da Análise Semântica, onde, por muitas vezes, devido a sua estrutura de dados ser em formato de pilha, possibilitar muitas facilidades.

O Gerenciador de Erro é apenas uma função que recebe uma cadeia de caracteres. Basicamente, ele indica qual é o tipo do erro (Léxico, Sintático ou Semântico) e em seguida informa a linha que ocorreu o erro e imprime a cadeia de caracteres recebida.

6. Gerador de Código da MP – Não implementado

Não conseguimos implementar a geração do código da máquina de pilha pois nos confundimos nos conceitos e, após várias tentativas e insucessos, desistimos.

7. Conclusão

Participar de um projeto desse porte nos fez ter uma visão mais madura do que é o nosso curso. Ficamos conscientes da importância de se ter um planejamento, da aquisição de conhecimento teórico para melhor utilizar as ferramentas da linguagem e o que a literatura diz a respeito da temática do projeto (que é um compilador). Foi enriquecedor no sentido de que materializou algo que, sinceramente, não fazíamos ideias de como tornar real no início da matéria. Achávamos que era algo muito além e que não teríamos capacidade de trabalhar. Apesar de bastante complexo, o processo que foi feito para organizar esse trabalho facilitou muito a compreensão e abstração do funcionamento de um compilador. Como feedback, trazemos que a carga de conteúdo e a quantidade de etapas necessárias para a construção de um compilador seria melhor aproveitado caso a disciplina fosse divida em duas, como é feito em Redes, Banco de Dados e Etc. Na nossa opinião seria mais proveitoso, pelo menos no ponto de vista de alunos. Outra sugestão seria, também, trazer, pelo menos a nível de citação, outras possíveis aplicações dos conceitos utilizados nessa matéria, caso existam.