UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

CAMPUS CHAPECÓ

CURSO DE CIÊNCIA DA COMPUTAÇÃO

Ana Clara Brusamarello Barbosa, Richard Facin Souza

CONSTRUÇÃO DO ANALISADOR LÉXICO

CHAPECÓ

2024

ANA CLARA BRUSAMARELLO BARBOSA, RICHARD FACIN SOUZA

CONSTRUÇÃO DO ANALISADOR LÉXICO

Trabalho apresentado ao Curso de Ciência da Computação, da Universidade Federal da Fronteira Sul (UFFS) como requisito parcial para aprovação na disciplina de Construção de Compiladores.

Professor: Braulio Mello

CHAPECÓ

2024

1. RESUMO

Este projeto apresenta o desenvolvimento de um reconhecedor léxico utilizando técnicas de Linguagens Formais e Autômatos. Inicialmente, são definidos os tokens da linguagem, incluindo palavras reservadas, identificadores, símbolos especiais e constantes, os quais são fornecidos via arquivo de entrada conforme o projeto prático em LFA (Linguagens Formais e Autômatos). Em seguida, é explorado o uso da aplicação para gerar um AFD (Autômato Finito Determinístico).

É detalhada a implementação do algoritmo de mapeamento do AF para reconhecimento léxico, demonstrando como o autômato é utilizado para identificar tokens na entrada fornecida.

Por fim, são apresentadas a geração da fita de saída e a Tabela de Símbolos, contendo informações essenciais das cadeias ou sentenças processadas, tais como linha, identificador e rótulo.

2. INTRODUÇÃO

Os reconhecedores léxicos desempenham um papel fundamental no processo de compilação, sendo responsáveis pela análise da estrutura léxica de um programa, identificando e classificando os diferentes elementos que o compõem, como palavras reservadas, identificadores, símbolos especiais e constantes. Essa fase inicial é crucial para garantir a correta interpretação do código fonte e sua posterior transformação em código executável.

O problema abordado neste trabalho é a implementação de um reconhecedor léxico utilizando técnicas de Linguagens Formais e Autômatos, com ênfase na geração de um AFD a partir dos tokens da linguagem fornecidos. Nosso objetivo principal é desenvolver uma solução eficiente e precisa para essa etapa crucial do processo de compilação, visando facilitar a análise subsequente do código fonte e contribuir para a construção de sistemas robustos e confiáveis.

Ao longo deste trabalho, exploraremos em detalhes a abordagem adotada, os desafios enfrentados e os resultados obtidos, demonstrando a importância e a viabilidade prática dos reconhecedores léxicos no desenvolvimento de software.

3. REFERENCIAL TEÓRICO

Inicialmente, é importante compreender os conceitos básicos de LFA, que incluem linguagens regulares, livres de contexto e sensíveis ao contexto.

No contexto de Autômatos, destacam-se os Autômatos Finitos Determinísticos (AFD) e Não-Determinísticos (AFND), bem como suas propriedades e operações, como união, interseção e complemento. O algoritmo de minimização de AFD também é relevante para otimizar a representação dos tokens da linguagem.

Outro aspecto crucial é o algoritmo de mapeamento do AFD para reconhecimento léxico, que envolve a definição de estados de aceitação e a transição entre os estados do autômato de acordo com os tokens da linguagem.

Por fim, é relevante considerar a geração da fita de saída e a construção da Tabela de Símbolos, que requerem uma estrutura de dados eficiente para armazenar informações como linha, identificador e rótulo das cadeias processadas.

Ao aplicar e compreender esses fundamentos teóricos, é possível desenvolver um reconhecedor léxico robusto e eficiente, capaz de analisar e identificar os tokens da linguagem de forma precisa e automatizada.

4. IMPLEMENTAÇÃO E RESULTADOS

A parte de análise léxica no código é responsável por identificar e validar os tokens presentes em um programa de entrada. Primeiramente, o programa é lido linha por linha de um arquivo chamado "programa.txt". Em seguida, cada linha é dividida em *tokens*, que são as unidades básicas de significado do código, como palavras-chave, identificadores e operadores.

A função “verificarToken()” é utilizada para verificar cada token em relação a um AFD fornecido como argumento. O AFD representa um conjunto de regras que descrevem como os tokens devem ser reconhecidos. Para cada token, a função atualiza o estado atual do AFD de acordo com cada símbolo do token.

Ao final da verificação de um token, é determinado se ele é válido ou não, com base no estado final alcançado pelo AFD. Se o estado final for um estado de aceitação, o token é considerado válido. Caso contrário, é marcado como inválido.

Os resultados da análise léxica são registrados em uma tabela de símbolos, que contém informações sobre cada token, como o número da linha em que foi encontrado, o próprio token e o estado final alcançado pelo AFD.

A função “fazerAnaliseLexica()” atua como o ponto central do processo de análise léxica, unindo todas as etapas desde a leitura do código fonte até a geração da tabela de símbolos e da fita de saída. Durante esse processo, os tokens são devidamente registrados na tabela de símbolos, enquanto aqueles que não são reconhecidos são identificados como 'x', indicando uma falha na análise léxica.

5. CONCLUSÃO

Destacamos o desenvolvimento de um analisador léxico em Python, utilizando autômatos finitos determinísticos para reconhecer tokens em programas de entrada. A implementação das diversas funções para manipulação e transformação de autômatos, juntamente com a geração de autômatos a partir de tokens e gramáticas regulares, permitiu uma análise eficiente e precisa do código fonte.

Durante o desenvolvimento da análise léxica, um desafio significativo foi a implementação no código para reconhecer os tokens em relação ao AFD. O principal obstáculo foi lidar com os diferentes estados do AFD e atualizar corretamente o estado atual com base nos símbolos dos tokens.

Para superar esse desafio, foram utilizadas técnicas de iteração sobre os tokens e de atualização dos estados do AFD de acordo com os símbolos encontrados. A cada símbolo processado, o estado do AFD era atualizado de acordo com as transições definidas no autômato.

Os resultados finais demonstram a viabilidade da implementação do analisador léxico, fornecendo, na teoria, uma ferramenta útil para validação de código fonte em linguagens de programação.

Para continuidade do trabalho, sugerimos a expansão do analisador para incluir outras fases do processo de compilação, como análise sintática e geração de código intermediário.