UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

CIÊNCIA DA COMPUTAÇÃO

LUCAS SMANIOTTO SCHUCH

VALTEMIR GOMES DA SILVA JUNIOR

COMPILADORES - PROJETO 1

ANALISADOR LÉXICO

ABRIL, 2024

CHAPECÓ - SC

1. Resumo:

Este trabalho apresenta o desenvolvimento de um analisador léxico visando a construção de um analisador capaz de identificar cadeias de um alfabeto. A partir de uma abordagem teórica e prática, serão discutidos os principais conceitos envolvidos na criação do analisador, incluindo a definição de tokens, geração de autômato finito determinístico (AFD) e a implementação de um algoritmo de mapeamento do autômato finito (AF), além da geração da fita de saída e da Tabela de Símbolos, com destaque para informações da cadeia/sentença.

2. Introdução:

Nos processos de compilação de software, analisadores léxicos desempenham o papel crucial, responsáveis por identificar e classificar tokens que compõem as sentenças de um código fonte e determinando se a cadeia faz parte de certa linguagem ou não. Os tokens podem incluir palavras reservadas, identificadores, símbolos especiais e constantes. São essenciais para a interpretação correta do programa pelo compilador.

O presente trabalho concentra-se na construção de um analisador léxico eficiente e preciso, destinado a realizar essa tarefa crítica de forma otimizada. Utilizando técnicas baseadas em autômatos finitos para o reconhecimento léxico, o objetivo é desenvolver uma ferramenta capaz de processar rapidamente o código-fonte e fornecer uma representação estruturada dos tokens identificados.

Ao longo deste artigo, exploraremos não apenas a teoria por trás dos analisadores léxicos e autômatos finitos, mas também sua aplicação prática na implementação de ferramentas de compilação.

3. Referencial teórico:

A fase de análise léxica é uma das etapas fundamentais no processo de compilação, precedendo a análise sintática e semântica. Para entender o desenvolvimento de um analisador léxico, é necessário revisar os conceitos teóricos subjacentes, incluindo a definição de tokens, as técnicas de construção de AFD's e os algoritmos utilizados para a geração da tabela de símbolos.

Tokens são os elementos básicos para as linguagens, constituindo unidades mínimas reconhecíveis por um compilador. Dentre eles se incluem palavras chaves básicas de qualquer linguagem de programação, identificadores de variáveis e funções, símbolos especiais e constantes, sendo a sua definição essencial para a correta interpretação do código fonte pelo compilador, visto que cada token possui um significado específico dentro de uma linguagem.

Os AFD's são modelos matemáticos utilizados para reconhecer padrões em sequência de caracteres. Dentro da análise léxica, um AFD pode ser construído para representar o reconhecimento dos tokens da linguagem. Ele é muito importante na função de garantir que todos tokens sejam identificados corretamente e sem ambiguidades.

A Tabela de Símbolos é uma estrutura de dados utilizada pelo compilador para armazenar informações dos tokens identificados durante sua análise. A sua geração envolve a coleta e organização das informações a respeito dos tokens presentes à medida que são reconhecidos, sendo que esses dados são manipulados por algoritmos eficientes para garantir que a tabela seja construída de forma precisa. Dessa forma, o compilador consegue acessar facilmente as informações necessárias durante todo processo subsequente da compilação.

Compreender esses conceitos é essencial para projetar e implementar um analisador que seja capaz de identificar e classificar corretamente os elementos do código fonte, contribuindo para o sucesso do processo de compilação. Uma boa forma de se iniciar a construção de um algoritmo capaz de realizar a análise léxica seria partir de uma descrição do que ele deverá executar.

O programa do analisador léxico deve receber o nome de um arquivo que contém os tokens que devem ser analisados. O analisador submete ao autômato cada token escrito. O autômato retorna a indicação se o token pertence à linguagem que ele reconhece. Caso algum token não seja reconhecido, o analisador indica qual foi o token e encerra a execução, ou analisa o próximo token inserido em um arquivo.

4. Implementação e resultados:

Considerando o que o programa deveria executar, foi desenvolvido um analisador léxico para o reconhecimento de tokens a partir da leitura de um arquivo de entrada. Obtemos o estado inicial do autômato e o primeiro símbolo do token. Dessa forma, foi verificado as situações caso seu símbolo existisse na linguagem ou não.

Caso o símbolo seja válido, obtemos o próximo estado a partir do símbolo e estado corrente. Se o próximo símbolo corresponde ao estado existente, refazemos o processo e caso não exista, afirmamos que o token não foi reconhecido. Caso o símbolo não exista na linguagem, verificamos se o estado corrente é final. Se o mesmo se apresenta como estado final, reconhecemos o token.

A implementação do analisador léxico inclui a definição de funções para reconhecimento de strings, verificação de tokens com base no AFD e geração da tabela de símbolos. O método “fazerAnaliseLexica” representa o núcleo da funcionalidade do analisador léxico, onde ocorre a análise propriamente dita do código fonte fornecido. Este método inicia chamando a função “reconhecerStrings”, que é responsável por ler o arquivo de entrada contendo o programa a ser analisado e segmentá-lo em tokens por linha. A partir dessa estrutura de dados, cada token é verificado utilizando o AFD fornecido como argumento para a função “verificarToken”.

Dentro de um loop principal, o método percorre cada linha do código fonte e, para cada token em uma linha, invoca a função “verificarToken”. Esta função realiza a análise léxica de cada token, verificando sua correspondência com as transições definidas no AFD. Se um token for reconhecido corretamente, ele é adicionado à tabela de símbolos. Caso contrário, é registrado na tabela de símbolos como um token não reconhecido.

Após a análise de todas as linhas do código fonte, finalizamos a geração da tabela de símbolos e geramos a fita de saída, fornecendo uma representação visual dos tokens reconhecidos e do processo de análise léxica.

Em conjunto, o método “fazerAnaliseLexica” integra todas as etapas do processo de análise léxica, desde a leitura do código fonte até a geração da tabela de símbolos e da fita de saída. Tokens são corretamente registrados na tabela de símbolos, enquanto tokens não reconhecidos são devidamente identificados como "x".

5. Conclusões:

Durante o desenvolvimento do analisador léxico, foram abordados os fundamentos teóricos necessários para sua criação e implementação prática em código.

Foi identificado dificuldades em reconhecer um token gerador por uma gramática, pois quando é avaliado o último símbolo a ser tratado, não estava claro o que devia ser feito. Após revisar os fundamentos da teoria, foi concluído que devia ser adicionado mais uma etapa de validação. Se o último símbolo fosse validado, e este estado possuísse uma epsilon transição, era realizado o reconhecimento token. Implementamos uma verificação para reconhecer se uma gramática chegou ao seu término, visto que todo token gerado por uma gramática possui um símbolo terminal representado por um epsilon.

Apesar do desafio exposto durante a implementação, o analisador foi capaz de identificar e classificar corretamente os tokens presentes no arquivo de entrada, garantindo a correta definição de tokens e resultados promissores expostos pela representação estruturada da Tabela de Símbolos e da Fita de saída.

Em resumo, o desenvolvimento deste analisador léxico proporcionou uma experiência valiosa no processo de compilação e destacou sua importância na interpretação correta de linguagens. Com base nos resultados, existe o potencial de uma expansão ao trabalho visando melhorias para incluírem a otimização do desempenho do analisador léxico, implementação de técnicas adicionais relacionadas ao reconhecimento de tokens e sua integração com ambientes de desenvolvimento de aplicação para depuração de código.