Analisador Léxico Com Base Em Autômatos Finitos Determinísticos Gerados A Partir De Gramáticas E Tokens Variados

Mateus Azor Frutuoso

Universidade Federal da Fronteira Sul (UFFS)

Resumo

O texto apresenta o desenvolvimento de um analisador léxico, etapa essencial na construção de compiladores, responsável por identificar tokens em cadeias de caracteres com base em uma linguagem formal. Os conceitos abordados incluem linguagens formais, expressões regulares, autômatos finitos (AFND e AFD) e o papel da análise léxica. A implementação ocorreu em seis etapas, incluindo a criação e determinização do AFND, conversão para uma estrutura utilizável e o reconhecimento de tokens.

Introdução

Os reconhecedores léxicos constituem uma etapa fundamental na construção de compiladores, sendo sua principal função identificar sequências de caracteres e agrupá-las em tokens, como palavras chaves, operadores, variáveis, dentre outros. Essa etapa é essencial para transformar o código em linguagem formal em uma representação compreensível para os próximos estágios da compilação, como a análise sintática e semântica.

Este trabalho tem como objetivo o desenvolvimento de um analisador léxico capaz de reconhecer sentenças pertencentes a uma linguagem formal específica, com base em gramáticas e tokens previamente definidos em um arquivo de entrada, mapeando as transições do autômato para identificar corretamente os tokens em uma cadeia, gerando assim uma fita de saída com cada elemento reconhecido.

Referencial Teórico

Uma linguagem formal é definida como um conjunto de cadeias construídas a partir de um alfabeto finito. Essas cadeias são classificadas conforme sua complexidade, (regular, livre de contexto, sensível ao contexto e recursivamente enumerável). No reconhecimento léxico as linguagens regulares são as mais relevantes, pois permitem representar padrões simples e frequentes como identificadores, números e operadores. As expressões regulares são uma notação formal para descrever linguagens regulares. Por exemplo, uma expressão regular como [a-zA-Z_][a-zA-Z0-9_]* pode ser utilizada para representar identificadores válidos em muitas linguagens de programação.

Toda expressão regular pode ser convertida em um Autômato Finito Não Determinístico (AFND), contudo, AFNDs não são eficientes para implementação, pois permitem múltiplas transições pelo mesmo símbolo no mesmo estado, por isso fazemos sua determinização.

Um autômato finito determinístico é uma máquina teórica usada para reconhecer linguagens regulares. Ele é composto por um conjunto finito de estados, um alfabeto de entrada, uma função de transição, um estado inicial e um conjunto de estados finais. O algoritmo percorre a cadeia de entrada símbolo por símbolo, realizando transições entre estados até atingir um estado final, o que indica o reconhecimento de um token. A principal característica do AFD é que, para cada par (estado atual, símbolo de entrada), existe apenas uma transição possível, por isso determinístico.

A análise léxica é a primeira etapa da compilação e consiste na conversão da cadeia de entrada em uma sequência de tokens. Cada token é uma estrutura composta por diversas informações variadas do reconhecimento, dependendo da implementação. O analisador léxico também pode construir uma tabela de símbolos, onde são armazenadas informações relevantes

sobre os identificadores encontrados. Essa tabela é útil para as etapas seguintes do compilador, como análise semântica e geração de código.

Implementação e Resultados

A implementação segue seis etapas principais, começando pela criação do cabeçalho do AFND que é representado por uma matriz, nesta etapa percorremos as gramáticas e tokens de entrada obtendo todos os símbolos válidos da linguagem, armazenando suas posições e adicionando-os ao cabeçalho. Seguimos para a próxima etapa onde o AFND será criado propriamente com a análise das gramáticas e tokens de entrada, adicionando as linhas na matriz que representam os estados e contém as transições por cada símbolo correspondente e criando também a lista de estados finais.

A próxima etapa é a determinização do AFND, transformando-o em um AFD, isso é feito ao percorrer a matriz criando novos estados para os indeterminismos e ajustando as transições e estados finais, no final da execução teremos uma nova matriz que compõe o AFD, essa matriz terá seu índice de estados, (linhas), atualizado para manter uma ordem coerente. Na etapa subsequente apenas tratamos as transições inválidas, adicionando um estado de final de erro e as direcionando para o mesmo.

Na penúltima etapa transformamos o AFD, até então uma matriz, em uma estrutura que possa ser utilizada mais facilmente na interpretação das cadeias de entrada, o analisador léxico, que nesse caso é composto por uma lista ordenada de dicionários, cada dicionário representa um estado e recebendo um símbolo retorna o estado correspondente, ou no caso, sua posição na lista. Para finalizar, na última etapa aplicamos a cadeia de entrada no AL e adicionamos os tokens reconhecidos na fita de saída.

Conclusões

O desenvolvimento deste analisador léxico proporcionou uma aplicação prática dos conceitos estudados em Construção de Compiladores, especialmente no uso de expressões regulares, autômatos finitos e algoritmos de determinização.

Durante o processo as dificuldades que se destacaram foram: o tratamento de múltiplas transições no AFND e a manutenção da consistência dos estados durante as etapas do processamento. Porém elas foram superadas com sucesso, resultando em uma ferramenta funcional. Já as melhorias mais relevantes seriam de otimização de código para melhor performance.