Reconhecimento Léxico

Luiz Paulo Grafetti Terres (2121101010)[1]

[1]Universidade Federal da Fronteira Sul

Abril 2024

1 Resumo

Este trabalho apresenta detalhes teóricos da etapa de compilação responsável pelo reconhecimento léxico, e discute a implementação de um reconhecedor léxico, tarefa requisitada no componente curricular de Construção de Compiladores, ministrada pelo Professor Dr.Braulio Adriano de Mello. A implementação discutida é do autor, e está disponível em: `https://github.com/luizpgt/Lexical_analysis.git`.

2 Introdução

Em Construção de Compiladores, é um elemento comum para algumas etapas de compilação, um "arquivo de programa", que possui palavras reservadas, identificadores e outros símbolos especiais. Neste trabalho, trataremos por arquivo de programa, um arquivo qualquer com entradas que busquem estar em conformidade com uma linguagem regular definida.

A etapa de Reconhecimento Léxico, discutida e implementada neste trabalho, é responsável pela identificação e classificação do conteúdo presente no arquivo de programa. Essa etapa verifica se existem erros lexicais presentes no arquivo, o que impossibilitaria as próximas etapas de compilação (como reconhecimento sintático) de produzir os resultados esperados.

O Reconhecedor Léxico deste trabalho busca classificar arquivos de programa de linguagens regulares genéricas (ou, à escolha do usuário), necessitando-se assim ter acesso ao Autômato Finito Determinístico, que definirá as variáveis e palavras reservadas possíveis no arquivo de programa.

2.1 Problema

Necessidade de classificação de palavras reservadas, identificadores e outros símbolos especiais de arquivo de programa de linguagem regular genérica, e disposição dos resultados encontrados para as próximas etapas de compilação.

2.2 Objetivo

Implementar um Reconhecedor Léxico, capaz de, utilizando a linguagem reconhecida em um autômato finito determinístico (projeto implementado como trabalho final da disciplina de Linguagens Formais e Autômatos), classificar o conteúdo de um arquivo de programa genérico e disponibilizar os resultados encontrados em uma tabela de símbolos e fita de saída.

3 Referencial Teórico

Este trabalho trata da análise léxica de um arquivo de programa, e trata-se de uma etapa de compilação. Um analisador léxico, como o implementado, é um programa responsável por ler o arquivo de programa de entrada, e classificar de acordo com as especificações da linguagem, a qual classe de elementos pertence cada palavra contida no arquivo. As palavras encontradas dentro do arquivo de programa podem ser classificadas entre palavras reservadas, identificadores, símbolos especiais ou constantes. Uma palavra qualquer pode pertencer a uma e somente uma dessas classes.

A etapa de análise léxica depende diretamente do uso de um autômato finito determinístico. Esse autômato, justamente, foi implementado como trabalho final da disciplina de Linguagens Formais e Autômatos, e essa implementação é utilizada neste trabalho.

O resultado da etapa de análise léxica produz informações que são relevantes para a próxima etapa de compilação, que envolve análise sintática. Na análise sintática, o arquivo de programa é avaliado em relação à ordem dos elementos presentes nele. São produzidos como saída da análise léxica uma fita de saída e uma tabela de símbolos. A fita de saída consiste em uma lista de estados finais (do autômato finito determinístico) que foram reconhecidos na leitura do arquivo de programa. Na Tabela de Símbolos, são salvos para cada elemento encontrado no arquivo, seu estado final no AFD, o seu label ("valor de fato" do elemento, encontrado no arquivo de programa) e linha onde está localizado.

4 Implementação e Resultados

Para melhor entendimento dessa seção, recomenda-se que o leitor tenha acesso à implementação do autor. Com ela, poderá testá-la e perceber mais detalhes de implementação.

4.1 Definição da Linguagem

As palavras que serão aceitas no arquivo de programa, precisam encontrar-se no arquivo `\finite\_automata\_input.txt"`. Neste arquivo, deverão ser definidos os tokens da linguagem, e suas gramáticas regulares. Com base no trabalho desenvolvido na disciplina de Linguagens Formais e Autômatos, as

entradas nesse "arquivo da linguagem" serão tratadas, e serão definidas em um único Autômato Finito Determinístico.

Esta parte do projeto está coberta no Submódulo `Deterministic_finite_automaton`, discutido na disciplina de Linguagens Formais e Compiladores, e não será detalhado nesse trabalho. Exemplo de valores no arquivo da linguagem:

```
¨ This is a comment !
¨ This files contains regular grammars and tokens.
¨ Note that E should be substituted by epsilon
se
entao
senao

<S> ::= a<A> | e<A> | i<A> | o<A> | u<A>
<A> ::= a<A> | e<A> | i<A> | o<A> | u<A> | E
```

4.2 Análise léxica

A etapa do analisador léxico, tem como entrada o arquivo de programa, `\example_program_file.txt"`. Nesse arquivo estarão presentes as palavras que serão classificadas, e ao final da etapa de análise léxica, estará produzida uma Fita de Saída, e uma Tabela de Símbolos.

É importante notar que a implementação da fase de análise léxica se baseia em um modelo. Esse modelo é uma classe chamada `Lexical_Analyzer`, e possui como atributos mais importantes a tabela determinística de transições (do autômato finito determinístico), uma **fita de saída** e uma **tabela de símbolos**. No modelo também são definidas questões úteis menores, como o **símbolo final de sentença**, e os **estados inicial** e de **erro**.

Primeiramente, na leitura do arquivo de programa, as palavras são capturadas em uma lista de palavras. A captura pode ser encontrada no arquivo `input_/lexical_scanner.py` é da forma:

```
Data: file: the input text file
Result: sentences: a list of tuples containing sentences and their
        respective line numbers
chars_sep_que_nao_aparecem ← \n, “ ”
chars_separadores_que_aparecem ← +, -, *, /
chars_separadores ← chars_separadores_que_aparecem +
 chars_sep_que_nao_aparecem
sentence ← [] ;  // list of characters of a word:  "se" = [s e]
sentences ← [] ;    // list of tuples:  (sentence, line number)
line_number ← 0 ;            // initialize line number counter
for char in file do
    if char in chars_separadores then
        if sentence is not empty then
            add tuple (sentence, line_number) to sentences;
            clear sentence;
        end
        if char in chars_separadores_que_aparecem then
            add tuple ([char], line_number) to sentences;
        end
        if char is \n then
            line_number ← line_number + 1
        end
    end
end
```

Algorithm 1: Leitura dos elementos do arquivo de programa.

E com um arquivo de programa da forma:

```
se + entao
aeiou
```

Pode-se esperar “sentences” como uma lista contendo:

[([s e], 1), ([+], 1), ([e n t a o], 1), ([a e i o u], 2)]

A partir dessa lista de sentenças, são produzidas as fita de saída e a tabela de símbolos, por meio da função `models.lexical_analyzer.analyze_sentences()`, que é descrita por:

```
Data: sentences: lista de tuplas (list_of_chars, line_num) capturadas na
      função de leitura do arquivo de programa
Result: Fita e Tabela de Símbolos
foreach sentence, line_file in sentences do
    estado ← inicial;
    foreach char in sentence do
        col ← column_of_char_in_afd();
        if col equals error_state then
            estado ← erro;
            break;
        end
        row ← row_of_state_in_afd();
        estado ← afd[row][col];
        if estado equals error_state then
            break;
        end
    end
    if estado equals error_state then
        break;
    end
    capturar_sentence_em_string ← string(sentence);
    // verifica se o estado é final, e o adiciona à Fita e TS
    valor_reconhecido ←
     capturar_valor_reconhecido_neste_estado(estado);
    if valor_reconhecido is not available then
        estado ← erro;
        valor_reconhecido ←
         capturar_valor_reconhecido_neste_estado(estado);
    end
    adicionar_a_tabela(estado, valor_reconhecido, linha_arquivo,
     sentence);
    adicionar_a_fita(estado);
end
```

Algorithm 2: Análise léxica dos elementos lidos no arquivo de programa.

O resultado da chamada dessa função, gera um objeto do modelo `Lexical_Analyzer`, com uma tabela de símbolos e fita de saída.

5 Conclusões

Em síntese do que foi discorrido neste trabalho, pode-se notar que o objetivo deste trabalho foi alcançado. A implementação prática de um Reconhecedor Léxico sobretudo agrega ao aluno um entendimento mais completo da teoria apresentada em sala de aula. Há espaço para melhorias na implementação do autor, que tem objetivos didáticos e deixa a desejar em alguns aspectos como otimização do tempo de execução.