Projeto de compiladores : Análise léxica

Edson Lemes da Silva[1]

[1]Ciência da Computação – Universidade Federal da Fronteira Sul (UFFS)
Caixa Postal 181 – 89.802-112 – Chapecó – SC – Brasil

edson_lemes@live.com

Abstract. *This paper describes some theoretical aspects for lexical analysis in the compilation process. Conversely, it presents the definition of a hypothetical language and lexical analysis for the recognition of their tokens. In this way, the development details and their particularities are presented.*

Resumo. *Este trabalho descreve alguns aspectos teóricos para a análise léxica no processo de compilação. Por outro lado, apresenta a definição de uma linguagem hipotética e a análise léxica para o reconhecimento de seus tokens. Desta forma, são apresentados os detalhes de desenvolvimento e suas particularidades.*

1. Introdução

A análise léxica é a primeira etapa do processo de construção de compiladores. Nesta fase, são verificados se os símbolos lidos de um arquivo fonte pertencem ao alfabeto da linguagem, e também se eles formam tokens. Esse processo utiliza autômatos finitos determinísticos (AFD) como parâmetro para o processo de análise léxica, assim os tokens são verificados através das transições do autômato finito.

A aplicação descrita aqui atende a leitura de tokens e o arquivo fonte, posteriormente é feito a análise léxica construindo a fita de saída. A linguagem de programação usada é o Python, juntamente com algumas classes que já estão implementadas.

2. Autômatos finitos

Os autômatos finitos (AF) podem ser construídos sobre uma gramática regular (GR) ou uma sequência de símbolos que representam tokens. De acordo com [Menezes 2000, p. 69], um autômato finito é descrito por um sistema de estados finitos, com aplicações em modelos de estudos, tais como compiladores, linguagens formais e semântica formal.

Analisando de forma detalhada, os AFs podem apresentar duas características estruturais. Deste modo, um autômato finito pode ser determinístico (AFN) ou não-determinístico (AFND). A diferença recorrente entre eles, descreve a forma como ocorrem as transições de um estado para outro conforme o símbolo de entrada. Desta maneira, em um AFN há exatamente um estado possível em cada transição, enquanto no AFND podem haver múltiplos estados conforme os símbolos da linguagem.

Um autômato finito não-determinístico pode ser transformado em um determinístico, já que existe uma relação de equivalência entre ambos. Este processo é chamado de determinização de autômatos finitos. Esta rotina permite tornar os estados bem definidos e sequenciais, facilitando principalmente a implementação de analisadores léxicos. A determinização ocorre analisando e criando novos estados para eliminar os

indeterminismos, uma vez que estas situações dificultam o tratamento e verificação das possíveis palavras que pertencem à linguagem descrita pela gramática.

O outro processo a ser descrito é a minimização de autômatos finitos determinísticos. De acordo com o [Menezes 2000, p. 122], este processo sobre um AFD não influencia no tempo de processamento do autômato em si, mas sim no números de estados, ou seja, na complexidade de espaço. Em outras palavras, este processo busca gerar um AFD mínimo e equivalente. O algoritmo de minimização é dividido em três partes: eliminação de estados inalcançáveis, mortos e aplicação de classes de equivalência.

Os autômatos finitos neste trabalho serão usados como requisito para a análise léxica, onde é feito o reconhecimento dos tokens de um arquivo fonte.

3. Análise léxica

Na maioria das linguagens de programação, palavras reservados, operadores ou identificadores são considerados como tokens. Assim, estes tokens são utilizados durante o processo de compilação. A análise léxica é a primeira fase de um compilador. Seu objetivo é ler o texto-fonte e identificar os tokens que depois serão utilizados na análise sintática.

O analisador léxico lê um arquivo fonte e realiza o reconhecimento dos tokens e também algumas funções secundárias, tais como, remover os comentários e os espaços em branco (espaços ou tabulações) [Aho et al. 2008, p. 38].

A análise léxica utiliza um AFD para o reconhecimento de padrões. A partir da leitura de cada símbolo é verificado através das transições do autômato a ocorrência dos símbolos até o reconhecimento do token (quando alcança um estado final de aceitação).

Esse processo de reconhecimento de padrões precisa de uma estrutura que armazene informações sobre os tokens lidos, essa estrutura é chamada de tabela de símbolos (TS). Na etapa de análise léxica, na TS devem ser armazenados no mínimo informações sobre a definição de cada token, e a diferenciação de identificadores, que serão utilizados posteriormente na etapa de análise sintática.

Assim como foi descrito anteriormente, a análise léxica elimina as informações que são irrelevantes para a compilação, por exemplo os comentários do código-fonte. Deste modo, a saída do analisador léxico é uma sequência de tokens, que será armazenado em uma fita e repassada para a próxima etapa de compilação.

4. A aplicação

Este projeto implementa a análise léxica de uma linguagem hipotética a partir de um conjunto de tokens definidos. Neste trabalho, é feito o uso do algoritmo de determinização implementado na disciplina de linguagens formais e autômatos.

A aplicação lê um arquivo de entrada que contém os tokens definidos, e um arquivo fonte da linguagem hipotética. Após o carregamento dos tokens e a construção do AFD, é realizado a análise léxica através da leitura dos tokens do arquivo de entrada seguindo as transições definidas pelo autômato finito.

A fita de saída da análise léxica é a mesma dos dados da primeira instância da tabela de símbolos.

5. A implementação

Inicialmente, é preciso descrever o ambiente de programação. A linguagem utilizada para a aplicação, foi o Python; utilizando-se de alguns módulos (classes) já implementados para esta linguagem.

Como forma de compatibilidade, esta aplicação foi desenvolvida sob a versão 2.7.12 do python. Os módulos necessários para a execução são: `collections`, `copy`, `itertools` e `terminalTables` (pode ser instalado via pip, utilizando o comando "`pip install terminaltables`"). Além destes, também é necessário importar o módulo que implementa a determinização : `detAfnd`.

Na implementação, a análise léxica esta descrita por uma classe, chamada de `lexAlg`. Uma instância desta classe, recebe um parâmetro do tipo `String`, que representa o nome e o caminho do arquivo de leitura (onde estão descritos os tokens e/ou as gramáticas regulares).

A tabela de símbolos é descrita por uma lista, onde nesta fase apresenta três atributos: número do estado que reconhece o token, rótulo e o número da linha da ocorrência do token no código-fonte. Desta forma, as três informações sobre cada token são armazenadas na lista que representa a TS.

Tabela 1. Exemplo da tabela de símbolos com a estrutura do laço de repetição.

Número do estado	Rótulo	Número da linha
18	begin	1
58	while	2
53	true	2
61	a	3
60	=	3
12	1	3
38	end	5

A função principal que executa a análise léxica é chamada de *lexicalAnalysis*. Ela recebe como parâmetro um arquivo fonte que representa a descrição de um código em cima da linguagem hipotética. Assim, a partir do autômato finito definido, o processo de análise pode começar.

Primeiramente, é feito a leitura linha por linha do arquivo de entrada, desconsiderando os espaços em brancos que pode conter em cada linha. Caso há algum token na linha lida, começa-se a análise léxica dos elementos que estão descritos nesta linha.

A função *lexicalAnalysis* separa em uma lista temporária todos os possíveis tokens que estão contidos na linha lida, desta forma assume-se que o separados é representado por um ou mais espaços em branco entre todos os tokens.

Para cada token é lido caracter por caracter, que representam os símbolos. Assim, para cada símbolo é verificado se ele faz parte do alfabeto da linguagem; se um símbolo não pertence ao alfabeto, o token sendo lido já é desconsiderado e é gerado um erro léxico.

Durante a análise dos tokens, e mais especificamente os símbolos destes tokens,

é percorrido a tabela de transição a partir do estado inicial. Deste modo, caso o token seja totalmente percorrido e a análise no AFD alcançou um estado final, então o token é reconhecido.

As informações são acrescentadas na TS após a análise de cada token, sendo ele reconhecido ou não. Também durante esta etapa, é criado a fita de saída que será utilizada juntamente com a tabela de símbolos na fase de análise sintática.

A estrutura da fita de saída é a mesma da primeira instância da TS, as informações descritas nela são: o número do estado que reconhece o token e o rótulo dele, seguindo o formato: *nro:rot*. Deste modo, para simplificação não é criado um arquivo externo para a fita, mas sim utilizado a primeira entrada da TS.

Como forma de interação, caso ocorrer erros léxicos, eles seram informações através de mensagens contendo a especificação do token e a linha onde ocorreu o erro.

5.1. Execução do algoritmo

Os tokens estão descritos no arquivo *tokens.txt* e um exemplo de código-fonte esta escrito no arquivo *test.txt*.

A execução do algoritmo é feita a partir do interpretador do Python. Primeiramente, deve-se importar o arquivos que contém as classes implementadas, que estão nos arquivos "detAfnd.py" e no "lex.py". O próximo passo é criar uma instância para a classe *lexAlg*, passando como parâmetro o nome do arquivo de leitura.

O construtor da classe se encarrega de instanciar e executar o algoritmo de determinização. Assim, após a instanciação, deve-se chamar o método (função) `lexicalAnalysis`, passando como parâmetro o nome e o caminho do arquivo fonte da linguagem.

A saída do programa apresenta eventuais mensagens de erro. Alternativamente, há um arquivo teste para a execução do algoritmo, chamado de "test.py".

```
1 from detAfnd import *
  from lex import *
3
  test = lexAlg('tokens.txt')
5 test.lexicalAnalysis('test.txt')
```

Exemplo 1. Execução do algoritmo.

6. Conclusão

No trabalho foi demonstrado uma implementação de um analisador léxico conforme um conjunto de tokens que representam identificadores, delimitadores ou palavras reservadas de uma linguagem hipotética.

A aplicação desenvolvida utiliza o trabalho de determinização implementado na disciplina de linguagens formais e autômatos. Assim, a análise léxico usa o AFD gerado como forma de verificar os tokens. Toda a implementação foi desenvolvida na linguagem Python.

A construção da aplicação em classes permite um melhor gerenciamento de cada parte. Já que este projeto será utilizado e incrementado na próxima etapa, que é a análise

sintática. A documentação sobre a implementação descreveu as funções principais de forma detalhada, assim como no código-fonte, onde estão descritas de forma resumida as funcionalidades de cada uma delas.

Referências

Aho, A., Sethi, R., and Lam, S. (2008). *Compiladores: princípios, técnicas e ferramentas*. Addison-Wesley.

Menezes, P. F. B. (2000). *Linguagens formais e autômatos*. Editora Sagra Luzzatto, 3rd edition.