Universidade Federal de Mato Grosso
Campus Universitário do Araguaia
ICET – Ciência da Computação

Linguagens Formais e Autômatos
Prof. Ivairton Santos

Trabalho 1 - Analisador Léxico

Introdução

Análise léxica é o processo de análise de uma determinada entrada de dados composta por caracteres e então produzir uma sequência de símbolos equivalentes (geralmente códigos numéricos) chamados tecnicamente de *tokens*. Os *tokens* são dados que podem ser manipulados mais facilmente por outras aplicações, como por exemplo por um *parser* (analisador sintático).

De maneira simplificada, pode-se dizer que uma análise léxica consiste em avaliar se um conjunto de caracteres apresenta, ou possui, padrões esperados, identificando esses padrões.

Por exemplo, a primeira fase de um compilador consiste exatamente em identificar se o código escrito possui comandos adequados da linguagem.

Objetivo do trabalho

(*i*) Desenvolver um autômato (DFA) que seja capaz de identificar comandos da linguagem especificada; (*ii*) implementar um analisador léxico simplificado (a partir do autômato).

Especificações

Inicialmente deve-se desenvolver o autômato (DFA) que seja capaz de reconhecer os comandos da linguagem especificada. Esse autômato deverá ser desenhado e entregue como parte da resolução do trabalho.

A partir do autômato elaborado, será implementado o analisador léxico, que deverá utilizar a tabela de transição do autômato (uma matriz de estados). A implementação **não** deve utilizar um `case`, ou um conjunto de `if's` aninhados para aplicar as transições entre estados.

O analisador léxico implementado deverá ser capaz de fazer a leitura de um arquivo de entrada (contendo o código), realizar a análise e deverá gerar um arquivo de saída com a relação dos *tokens* processados/encontrados.

Considere como delimitador das palavras o "espaço", ou uma quebra de linha.

O trabalho pode ser implementado na sua linguagem de preferência.

Especificação da linguagem a ser reconhecida (C simplificado)

Relação de palavras reservadas e seus respectivos códigos (*tokens*):

- *Variável/identificador*: Qualquer palavra que não seja reservada e que inicie com '_' ou uma letra (ex: _cont, Flag, contador). Código do **token = 1**;
- *Numeral*: Qualquer número real, positivo ou negativo (ex: 34; -14; 77.99; 1111). Código do **token = 2**;
- *Comentário*: Qualquer palavra/frase que esteja entre os símbolos '/*' e '*/' (ex: /* funcao de teste */). Código do **token = 3**;
- *Tipo de dado*: Utilizado na declaração de variáveis e funções, são eles: `char, double, float, int, void`. Código do **token = 4**;
- *Decisão*: Comandos de decisão, são eles: `if, else, case`. Código do **token = 5**;
- *Laço*: Comandos de laço de iteração, são eles: `for, while, do`. Código do **token = 6**;
- *Operadores*: Símbolos operadores, são eles: `+, -, *, /, <, >, =, ==, !=`. Código do **token = 7**;

- *Retorno*: Comando de retorno de função: `return`. Código do **token = 8**;
- *Início de escopo*: Comando de marcação de início de escopo: {. Código do **token = 9**;
- *Fim de escopo*: Comando de marcação de fim de escopo: }. Código do **token = 10**;
- *Início de equação*: Comando de marcação de início de equação: `(`. Código do **token = 11**;
- *Fim de equação*: Comando de marcação de fim de equação: `)`. Código do **token = 12**;

Exemplo:

Considere como exemplo a elaboração de parte deste trabalho, restrita ao reconhecimento dos comandos `if` e `int`. O DFA proposto, específico para reconhecimento destes comandos, é:

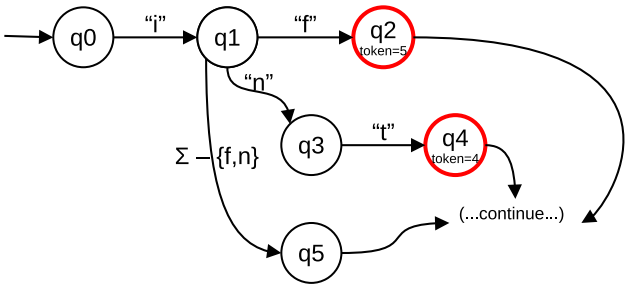

Figura 1: DFA elaborado para reconhecimento dos comandos "if"e "int".

Observe que o alfabeto é representado por Σ. Os estados q_2 e q_4 são finais e reconhecem, respectivamente, o comando "if"e "int". Portanto, no exemplo, ao alcançar o estado q_2 o sistema deve gerar o **token 5** e retornar posteriormente ao estado inicial (q_0).

Lembre-se que este DFA é apenas um pequeno recorte, ele deve ainda especificar a possibilidade de reconhecer indentificadores e demais comandos previstos da especificação.

Para este recorte do DFA, teremos a seguinte **tabela de transição** (parcial), mencionada na especificação do trabalho:

Σ	q_0	q_1	q_2	q_3	q_4
i	q_1				
f		q_2			
Término			*Tkn5*		*Tkn4*
n		q_3			
t				q_4	
(...Continue...)					

O termo "Término" representa o "espaço"ou fim de linha; O termo "Tkn" representa o código do token a ser gerado.

Seu algoritmo deverá se basear na sua tabela de transição, ou seja, à medida que os símbolos do alfabeto forem consumidos, você deverá consultar a tabela para verificar qual transição deverá ser executada. À medida que os estados finais forem alcançados, gere o token correspondente e continue a verificação de novas palavras.

Exemplo de arquivo de entrada:

Arquivo de entrada
if case contador int while /*teste*/ +

Respectivo arquivo de saída:

Arquivo de saída
5 5 1 4 6 3 7