Analisadores léxicos

Professor: Luciano Ferreira Silva, Dr.

Analisadores léxicos

- Idéia: usar autômatos finitos para reconhecer se uma string esta associada à uma determina classe de tokens.

- Deste modo, deve-se traduzir a especificação de um autômato finito determinístico num programa de computador, para que a classificação possa ser feita de forma automática.

Analisadores léxicos

Descrição:

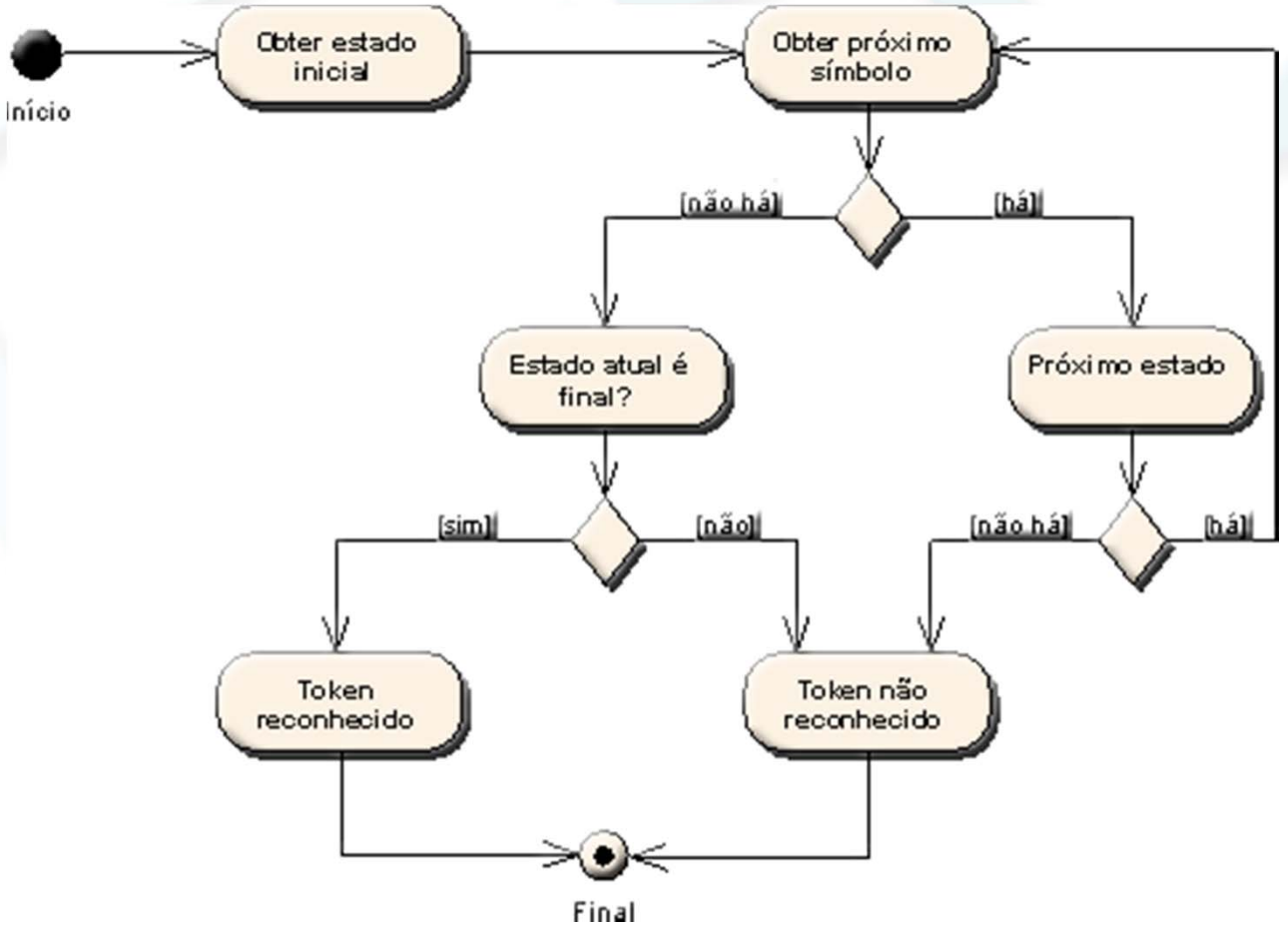
1. O analisador léxico recebe o nome de um arquivo que contém uma seqüência de caracteres.
2. Para cada lexema que é extraído desse arquivo, retorna a indicação de se o token é válido e qual é a sua classe.
3. O analisador encerra a execução normalmente após a análise do último token.
4. Caso algum token não seja reconhecido, o analisador indica a situação e passa ao próximo token, se possível; caso contrário, encerra a execução.

Analisadores léxicos

- **Um autômato finito e sua forma computacional :**
  - ✓ A forma mais simples de representar um autômato computacionalmente é por meio de uma tabela de transições. Ela oferecer:
    1. O estado inicial para o autômato;
    2. Dado um estado qualquer, indicar se este é um estado final (condição de aceitação);
    3. Dado um estado qualquer e um símbolo, a indicação de qual é o próximo estado.

Analisadores léxicos

- Diagrama de atividades para procedimento de reconhecimento de token:



Analisadores léxicos

- **Algoritmo para uma analisador léxico:**

```
RECONHECE(M, σ)
1  s ← ESTADO-INICIAL(M)
2  while TEM-SÍMBOLO(σ)
3  do c ← PRÓXIMO-SÍMBOLO(σ)
4        if EXISTE-PRÓXIMO-ESTADO (M, s, c)
5              then s ← PRÓXIMO-ESTADO (M, s, c)
6              else return false
7  if ESTADO-FINAL(M,s)
8  then return true
9  else return false
```

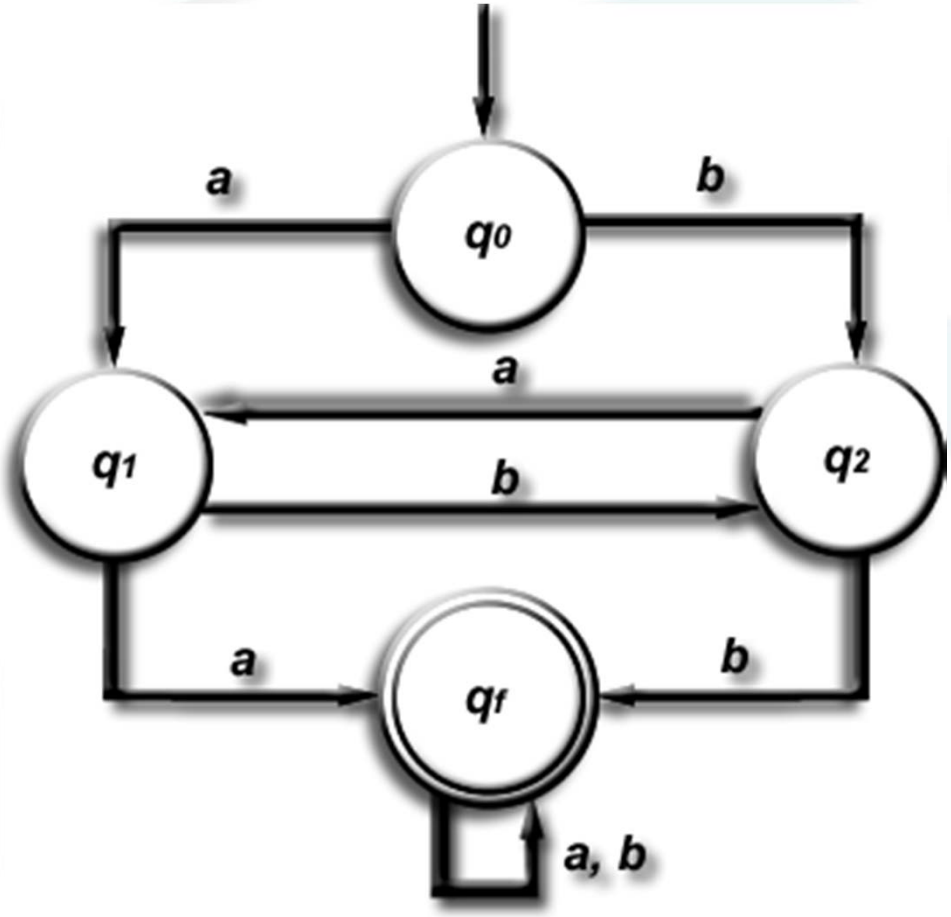
Exemplo de implementação

- Considere a seguinte linguagem sobre o alfabeto {a, b}: L={w | w possui aa ou bb como subpalavra}
- O autômato finito: M=({q0, q1, q2, qf}, {a, b}, δ1, q0, {qf}), onde δ é dado pela matriz abaixo, reconhece a linguagem L.

	q_0	q_1	q_2	q_f
a	q_1	q_f	q_1	q_f
b	q_2	q_2	q_f	q_f

Exemplo de implementação

- Diagrama do autômato:



Exemplo de implementação

- Considere a correspondência entre as matrizes abaixo:

	q_0	q_1	q_2	q_f
a	q_1	q_f	q_1	q_f
b	q_2	q_2	q_f	q_f

Corresponde

	0	1	2	3
0	1	3	1	3
1	2	2	3	3

- Obs.: a segunda matriz possui apenas valores inteiros e pode ser facilmente usada em qualquer linguagem de programação.

Exemplo de implementação

```
#include <cstdlib>
#include <iostream>

using namespace std;

int AutomatoM (char *str)
{
    int i;
    /* Declarando a matriz de estados do automato M */
    int M[2][4]={{1, 3, 1, 3},{2, 2, 3, 3}};

    /* Definindo o estado inicial */
    int e = 0;

    /* Recebera a indexacao referente ao caractere*/
    int c;
```

Exemplo de implementação

```
 /* Percorrendo toda a sentenca de avaliacao*/
    for(i=0;str[i]!='\0';i++)
    {
    switch(str[i])
    {
                    case 'a':
                         c=0;
                         break;
                    case 'b':
                         c=1;
                         break;
                    default:
/*Se encontrar um caractere fora do alfabeto a sentenca nao e
   reconhecida*/
                              return 0;
    }
```

Exemplo de implementação

```
/* Definindo o sistema de transicao*/
    e = M[c][e];
    }
/* Retornando o resultado de acordo com os estados finais do
   automato*/
if(e==3)
    return 1;
else
    return 0;
}
int main()
{
    char sentenca[10];

cout<<"|-----------------------------------------------
   ------------------------|\n";
 cout<<"|-Automato que reconhece a lingugem L = {w|w possui
   aa ou bb como subpalavra}-|\n";
```

Exemplo de implementação

```
cout<<"|-----------------------------------------------
   ------------------------|\n";
    cout<<"|----> Digite a sentenca para reconhecimento; ";
    cin>>sentenca;
    cout<<endl;
    cout<<endl;
    int res = AutomatoM(sentenca);
    if(res==1)
                cout<<"|----> O automato reconheceu a sentenca;
   "<<endl;
    else
                cout<<"|---->  O  automato  reconheceu  NAO  a
   sentenca; "<<endl;
    cout<<"|-----------------------------------------------
   ----------------------------|\n";
   system("PAUSE");
    return EXIT_SUCCESS;
}
```

3° Trabalho

1. Proponha 4 tipos de linguagens regulares não-ambíguas;
2. Determine os seus respectivos autômatos de reconhecimento e codifique-os em forma de funções ou métodos;
3. Construa um arquivo .txt contendo, mistamente, diversas sentenças de cada linguagem;
4. Use o seu varredor de lexamas para separar as sentenças, e as funções para classificá-las de acordo com cada linguagem.
5. Você deve armazenar cada lexama e sua respectiva classificação em uma estrutura de dados qualquer;
6. Ao final desta construção você terá implementado o seu primeiro analisador léxico!!!