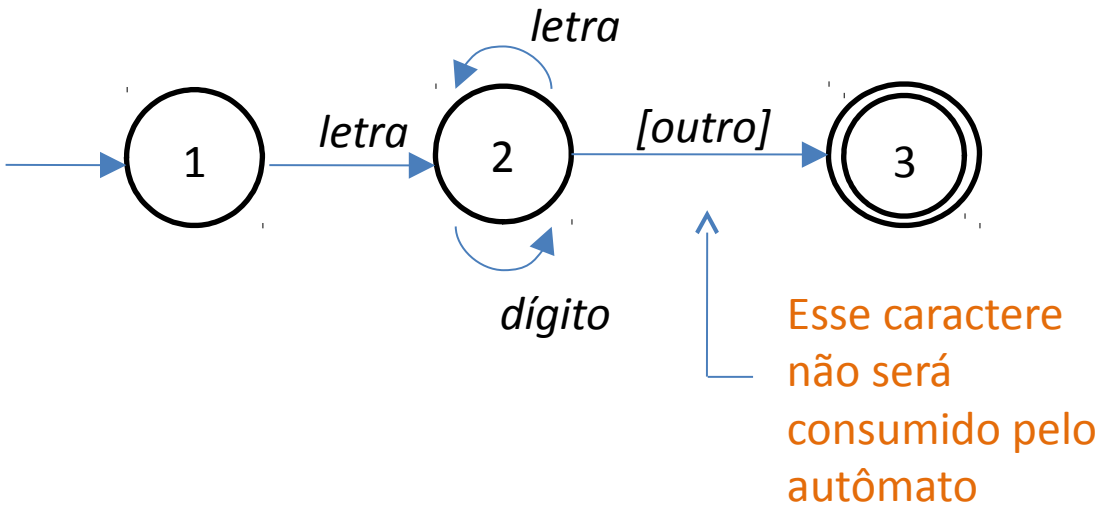
Compiladores
Análise Léxica
Implementação de AFD

Prof. Dr. Luiz Eduardo G. Martins
(adaptado por profa Dra Ana Carolina Lorena)
UNIFESP

Implementação de AFD

- Há muitas formas de traduzir um AFD em código
- Considere o seguinte AFD para reconhecimento de identificadores:

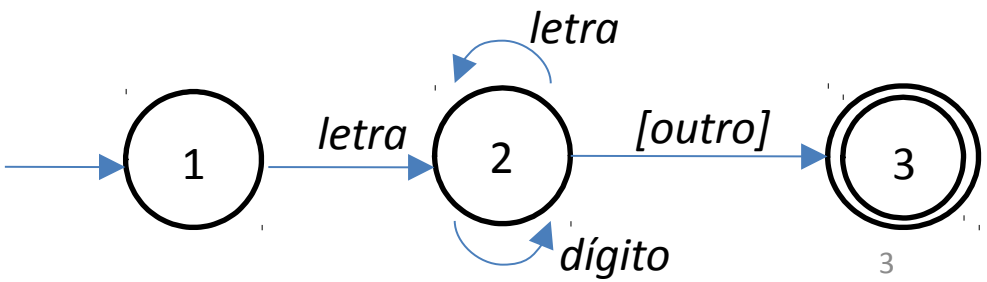


Implementação de AFD: código direto

- Podemos simular esse AFD escrevendo o código da seguinte forma:

Indicado apenas para sistemas de varredura muito simples

```
{início – estado 1}
if próximo caractere for letra then
    avance entrada;
    {estado 2}
    while próximo caractere for letra ou dígito
        avance entrada; {permanece no estado 2}
    end while
    {passa para o estado 3 sem avançar entrada}
    aceitação;
else
    {erro ou outros casos}
end if
```



Implementação de AFD: case

- Uma alternativa melhor de implementação, seria usar uma variável para manter o estado corrente e tratar as transições como declarações *case*

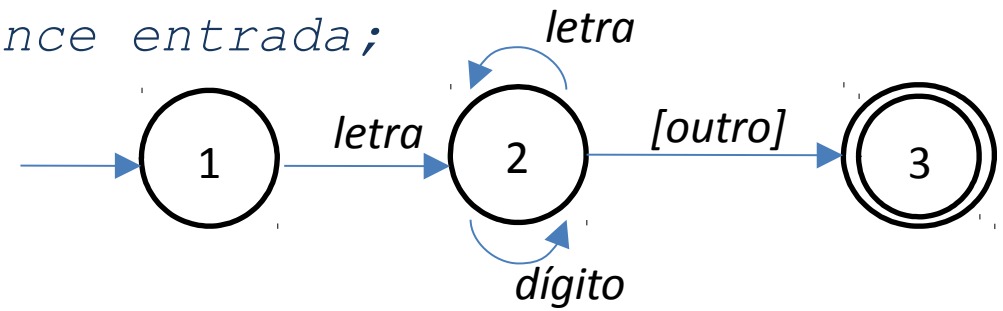
Exemplos:

x=0;

nota01;

1x;

```
estado :=1; {início}
while estado = 1 ou 2
   case estado of
   1: case caractere de entrada of
      letra:  avance entrada; estado := 2;
      default estado := 4 {erro ou ignore};
      end case;
   2: case caractere de entrada of
      letra, dígito: avance entrada;
      default estado:=3;
      end case;
   end case;
end while;
If  estado = 3 then  aceitação else erro ou ignore;
```



Implementação de AFD: tabelas

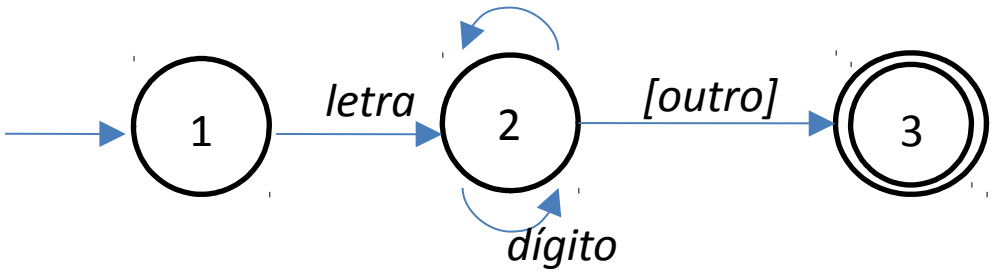
- Métodos algoritmos dirigidos por tabela proporcionam a implementação de um código genérico para AFD (máquina de estados)
- Considere a tabela a seguir para o AFD de reconhecimento de identificadores

caracteres de entrada

estados

	letra	dígito	outro	Aceitação
1	2			não
2	2	2	[3]	não
3				sim

Colchetes indica que não avança a entrada



Implementação de AFD: tabelas

- Um algoritmo dirigido por tabela para o AFD de reconhecimento de identificadores poderia utilizar as seguintes matrizes:
  - Uma matriz de inteiros *T[estado,ch]*
  - Uma matriz booleana *Avance[estado,ch]*
  - Uma matriz booleana *Aceita[estado]*

Exemplos:

x=0;

n1=2;

1x=3;

```
estado :=1;
erro:=F;
ch:=próximo caractere de entrada;
if ch not letra then erro:=V;
while not Aceita[estado] and not erro
    novoestado:= T[estado,ch];
    if Avance[estado,ch] then ch:=próximo caractere de entrada;
    estado:= novoestado;
end while;
if Aceita[estado] then aceitação
```

	letra	dígito	outro	Aceitação
1	2 V			F
2	2 V	2 V	3 F	F
3				V

Implementação de AFD: tabelas

- O algoritmo dirigido por tabela:
  - Tem código reduzido
  - É genérico (mesmo código para diferentes problemas)
  - É mais fácil de alterar e manter
  - Mas tabelas podem ficar muito grandes
  - E muitas posições da tabela são vazias (pode necessitar implementar com alguma técnica para matrizes esparsas)

Implementação de AFND

- No caso de implementar um AFND, é necessário armazenar as transições ainda não tentadas e retroceder a elas em caso de falha
- É mais comum então usar AFDs, pois é mais fácil de tratar
- É possível converter um AFND em um AFD equivalente

Analisador Léxico

- Cabe ao sistema de varredura (analisador léxico) ler os caracteres do código-fonte e organizá-los em unidades lógicas para as outras partes do compilador (como o analisador sintático)
- É necessário implementar, como parte do sistema de varredura, uma rotina que leia os caracteres do arquivo de entrada (programa fonte), e retorne o caractere lido

Implementação de Analisador Léxico

- Como alternativa, considere a função *proximoChar()* apresentada no código a seguir:

```
#include <fstream>
#include <iostream>

ifstream arq("teste.txt");

char proximoChar()
{
  char ch;
  arq.get(ch);  // lê um caractere do arquivo texto
  return(ch);
}

int main()
{
  char ch;
  while(!arq.eof())
  {
   ch = proximoChar();
   cout << ch;
  }
  return 0;
}
```

Bibliografia

- Bibliografia consultada

Capítulo 2 LOUDEN, K. C. **Compiladores: princípios e práticas.** São Paulo: Pioneira Thompson Learning, 2004

RICARTE, I. **Introdução à Compilação.** Rio de Janeiro: Editora Campus/Elsevier, 2008

AHO, A. V.; LAM, M. S.; SETHI, R. e ULLMAN, J. D. **Compiladores: princípios, técnicas e ferramentas.** 2ª edição – São Paulo: Pearson Addison-Wesley, 2008