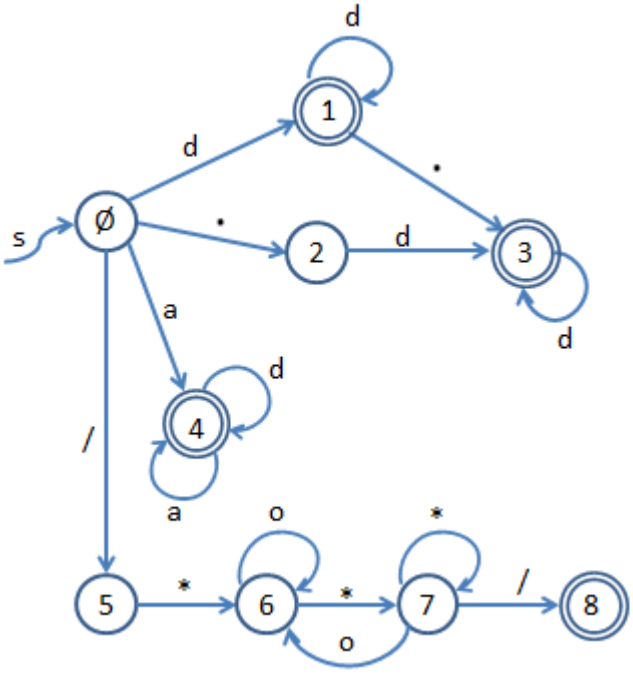
Algoritmos para o funcionamento de um AEF

O autômato de Estados Finitos que utilizaremos para apresentação da estrutura de dados e algoritmos segue:



Significado de "o" (others):

Seja A o conjunto de todos os símbolos que rotulam as arestas que tem origem num determinado estado.

"Others" denota qualquer símbolo em $\sum$ - A.

Veja que o elemento "others" representa diferentes conjuntos porque depende dos símbolos que rotulam as arestas que tem origem num determinado estado.

1. Estrutura de Matriz Quadrada e Vetor auxiliar para estados finais

Vetor Q indica se o estado é final. Q[q] $\in$ {0,1}.

Matriz M que representa as transições numa estrutura bidimensional.

A matriz M representa a função

$M: Q \times \Sigma \rightarrow Q \cup \{-1\}$

A função não está definida para todos os estados e símbolos do alfabeto, ou seja, existe (q, a) tal que f(q,a) = -1. Esta condição indica rejeição da cadeia. Supomos que, do ponto de vista de algoritmos, é possível que as colunas da matriz sejam indexadas pelos símbolos do alfabeto ou pelo valor ASCII dos caracteres. Deste modo, Σ = { 0, 1, 2, 3, ..., m} representa os símbolos do alfabeto e Q = {0, 1, 2, 3, ...n} é o conjunto de estados do AEF.

Índice	Est. Final
0	0
1	1
2	0
3	1
4	1
5	0
6	0
7	0
8	1

Estados \ Símbolos do Alfabeto	a	d	.	/	*	o
0	4	1	2	5		
1		1	3			
2		3				
3		3				
4	4	4				
5					6	
6					7	6
7				8	7	6
8						

A Matriz M representa a função $M: Q \times \Sigma \rightarrow Q \cup \{-1\}$.
A função pode não estar definida para todos os pares (q,a), $q \in Q$, $a \in \Sigma$.
Na prática, preenchemos com valor especial M(q,a) = -1, indicando que não existe transição para (q,a).

Neste caso, para cada símbolo, o acesso é direto na matriz M, de modo que a complexidade deste algoritmo tem k = n (k é o número de elementos da cadeia).

```
q ← q0;
c ← le_simbolo();
enquanto (não acabou a cadeia de entrada) faça
   se (M[q,c] não está definida) então
      rejeita();
   senão
      q ← M [q][c];
      c ← le_simbolo();
se (Q[q] é FINAL)
   aceita();
senão
   rejeita();
```

2. Utilizando uma estrutura de vetor de registros com vetor auxiliar para os estados

Vetor de Estados Q

	Estado Final	Prim_Trans
0	0	0
1	1	4
2	0	6
3	1	7
4	1	8
5	0	10
6	0	11
7	0	13
8	1	NULL

Neste exemplo, não existe transição que sai do estado 8.

Vetor de Structs M

	Estado	Símbolo	Próx. Est.
0	0	d	1
1	0	.	2
2	0	a	4
3	0	/	5
4	1	d	1
5	1	.	3
6	2	d	3
7	3	d	3
8	4	d	4
9	4	a	4
10	5	*	6
11	6	o	6
12	6	*	7
13	7	*	7
14	7	o	6
15	7	/	8
16	-1	-1	-1

A tabela M é construída de modo que as transições de cada estado ficam agrupadas começando pelo estado inicial zero. A última célula da matriz contem uma transição inválida.

As colunas Estado, Simbolo e Prox. Est. representam as transições do AEF.

A tabela Q tem o número do estado como índice. A coluna Prim_Trans indica o índice de M na qual está a primeira transição do respectivo estado. O campo Estado Final vale 1 se o respectivo estado é final e, em caso contrário, vale zero.

```
M: Vetor de Structs representando as Transições de estado.
Q: Vetor de Estados (final ou não e índice da primeira transição)
q: Estado do AEF durante o processo de reconhecimento
c: simbolo corrente da cadeia de entrada
i: Índice de percurso da tabela de Transições M.

q ← estado_inicial;  // Estado corrente
c ← le_simbolo();    // Símbolo corrente
i ← Q[q].prim_trans; // Índice da primeira transição do estado q0.

enquanto (não acabou a entrada) faça
   enquanto ((M[i].estado == q) && (M[i].simbolo != c)) faça
      i++;
   se (M[i].estado == q) então
      q ← M[i].prox_est;
      i ← Q[q].prim_trans;
      c ← le_simbolo();
   senão
      rejeita();   // Informa a rejeição e interrompe a execução
se (Q[q].estado_final) então
   aceita();
senão
   rejeita();
```

Erro na anotação de aula

O percurso sequencial da Matriz M ocorre no máximo o número de transições que cada estado possui. Este percurso é constante independentemente do comprimento da cadeia que deve ser analisada. Portanto, supondo que o AEF tem no máximo k arestas que saem de um determinado estado, a complexidade é de (k · n). O vetor auxiliar fornece de forma rápida o subconjunto dos estados que são finais e de forma direta a primeira transição de cada estado.

3. Autômato de Estados Finitos com transições em código.

Observação:

i) A função next_char() é responsável por entregar o próximo símbolo da cadeia de entrada e fazer o mapeamento do símbolo para os simbólicos LETRA, DIGITO, BARRA, ASTERISCO, PONTO e EOF repassando-os ao programa que trata o AEF

ii) Significado das constantes simbólicas:
LETRA: a..z, A..Z e (underscore)
DIGITO: 0 a 9.
ASTERISCO: *
PONTO: .
EOF: Quando acabar a cadeia de entrada, a função next_char(*) devolve o inteiro representado por EOF.

iii) A função erro(), entre outras coisas, altera a variável ERRO para verdadeiro.

iv) A função aceita(), entre outras coisas, altera a variável ACEITA para verdadeiro.

```
int ACEITA = 0;
int ERRO = 0;
char cadeia[100];
int estado = estado_inicial;
char simbolo;

simbolo = next_char();

  while ((!ERRO) && (!ACEITA)) {
    switch (estado) {
      case 0: if (simbolo == DIGITO) estado = 1;
              else if (simbolo == PONTO) estado = 2;
              else if (simbolo == LETRA) estado = 4;
              else if (simbolo == BARRA) estado = 5;
              else erro();
              break;
      case 1: if (simbolo == EOFF) aceita();
              else if (simbolo == PONTO) estado = 3;
              else if (simbolo != DIGITO) erro();
              break;
      case 2: if (simbolo == DIGITO) estado = 3;
              else erro();
              break;
      case 3: if (simbolo == EOFF) aceita();
              else if (simbolo != DIGITO) erro();
              break;
      case 4: if (simbolo == EOFF) aceita();
              else if ((simbolo != LETRA) &&
                       (simbolo != DIGITO)) erro();
              break;
      case 5: if (simbolo == ASTERISCO) estado = 6;
              else erro();
              break;
      case 6: if (simbolo == ASTERISCO) estado = 7;
              break;
      case 7: if (simbolo == BARRA) estado = 8;
              else if (simbolo != ASTERISCO) estado = 6;
              break;
      case 8: if (simbolo == EOFF) aceita();
              else erro();
    } // termino do switch

    simbolo = next_char();
  } // termino do while

if (ACEITA)
   printf("Cadeia aceita: %s\n\n", cadeia);
else
   printf("Cadeia rejeitada: %s\n\n", cadeia);
```

Observações:

i) Se um estado é final, verificar a condição de fim de cadeia;
ii) Se existe transição do tipo “other”, não colocar teste de erro;
iii) Se existe aresta cíclica, não há necessidade de atualizar o estado.

Alternativamente, em cada estado, chama-se uma função que trata do procedimento local seguida do comando “break”.