Expressões Regulares e Linguagens

Expressões Regulares (ER)

AF e ER

Equivalências entre AFD, AFND, AF-λ, ER

Expressões Regulares (ER)

Uma ER sobre um alfabeto Σ é definida como:

a) $\varnothing$ é uma ER e denota a linguagem vazia
b) λ é uma ER e denota a linguagem contendo a palavra vazia, ie $\{\lambda\}$
c) Qualquer símbolo $x \in \Sigma$ é uma ER e denota a linguagem $\{x\}$
d) Se r e s são ER denotando as linguagens R e S então:
   - (r+s) ou (r|s) é ER e denota a linguagem $R \cup S$
   - (rs) é ER e denota a linguagem $RS = \{w=uv \mid u \in R \text{ e } v \in S\}$
   - (r^*) é ER e denota a linguagem R^*

Exemplos

- 00 é uma ER denotando a linguagem {00}
- (0+1)* denota a linguagem formada por todas as cadeias de 0´s e 1´s
- (0+1)* 00 (0+1)* denota todas as cadeias de 0´s e 1´s com ao menos dois 0´s consecutivos
- a+b*c denota um único a ou zero ou mais vezes b seguido de c

- (0+1)* 001 denota todas as cadeias de 0´s e 1´s terminadas em 001

- 0*1*2* denota qualquer número de 0´s seguido por qualquer número de 1´s seguido por qualquer número de 2´s

- 01* + 10* denota a linguagem consistindo de todas as cadeias que são um único 0 seguido por qualquer número de 1´s OU um único 1 seguido por qualquer número de 0´s.

Omissão de parênteses

- Para omitir parênteses devemos respeitar:
  - O fecho (*) tem prioridade sobre a concatenação (rs), que tem prioridade sobre a união.
  - A concatenação e a união são associadas da esquerda para a direita.
  - Ex: 01* + 1 é agrupado como (0(1*)) + 1 => L = {1, 0, 01, 011,...}

- Usamos parênteses quando queremos alterar a prioridade:

- (01)* + 1 => $L = \{1 \cup (01)^n \mid n >= 0\} = \{1, \lambda, 01, 0101, ...\}$

- 0(1* + 1) => $L = \{w \in \{0,1\}^* \mid$ w começa com 0 seguido de $1^n \mid n>=0\}$ ➔ Lei distributiva à esq = 01* + 01 = {0,01,011,0111,...}

Escreva a ER equivalente a:

- O conjunto de cadeias sobre {0,1} que termine com três 1´s consecutivos.

- O conjunto de cadeias sobre {0,1} que tenha ao menos um 1.

- O conjunto de cadeias sobre {0,1} que tenha no máximo um 1.

Escreva a ER equivalente a:

- O conjunto de cadeias sobre {0,1} que termine com três 1´s consecutivos.

$$(0+1)^*111$$

- O conjunto de cadeias sobre {0,1} que tenha ao menos um 1.

$$(0+1)^*1(0+1)^*$$

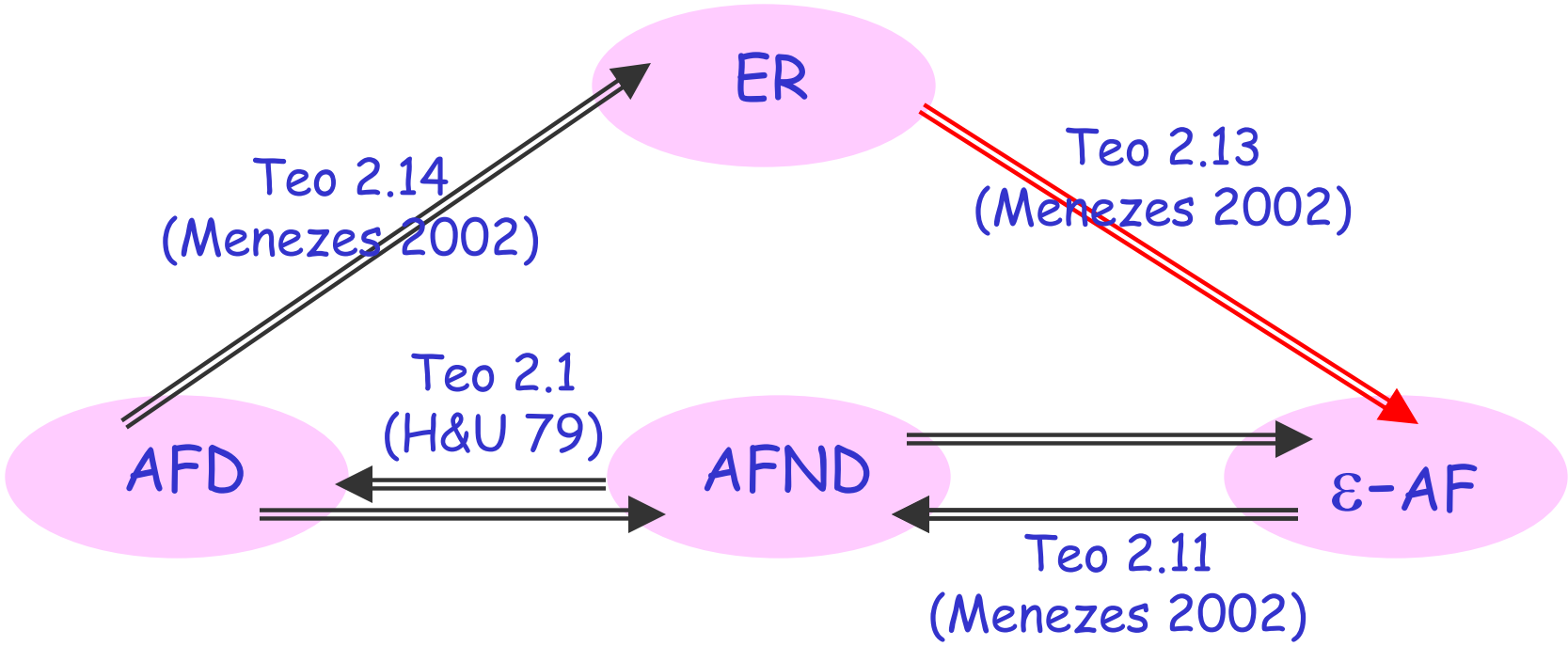
- O conjunto de cadeias sobre {0,1} que tenha no máximo um 1.

$$0^*(1+\lambda)0^*$$

AF e ER

- AF e ER representam exatamente o mesmo conjunto de linguagens, as Linguagens Regulares. Para mostrar isso, deve-se mostrar que:
  - toda linguagem definida por um AFD ou AFND é definida por uma ER. (por expressões dos caminhos ou por redução de estados)
  - toda linguagem definida por uma ER é definida por um AFD ou AFND. (na verdade, mostra-se que existe um ε-AFND que aceita a mesma linguagem)

Equivalências entre AFD, AFND, ε-AF, ER



Teorema: Toda linguagem definida por uma ER também é definida por um AF

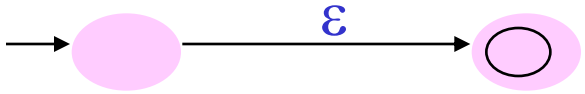
- Prova: Suponha que L=L(R) para uma ER R.

  Mostraremos que L=L(E) para algum ε-AFND E com:
  - exatamente um estado de aceitação
  - nenhum arco chega no estado inicial
  - nenhum arco sai do estado de aceitação.

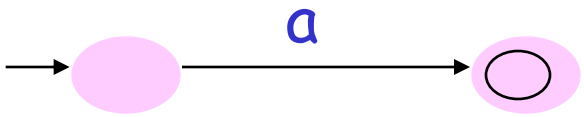
  A prova é por indução estrutural sobre R, seguindo a definição de ER:

  BASE: AF para as partes a, b, c da definição de ER.

(a)

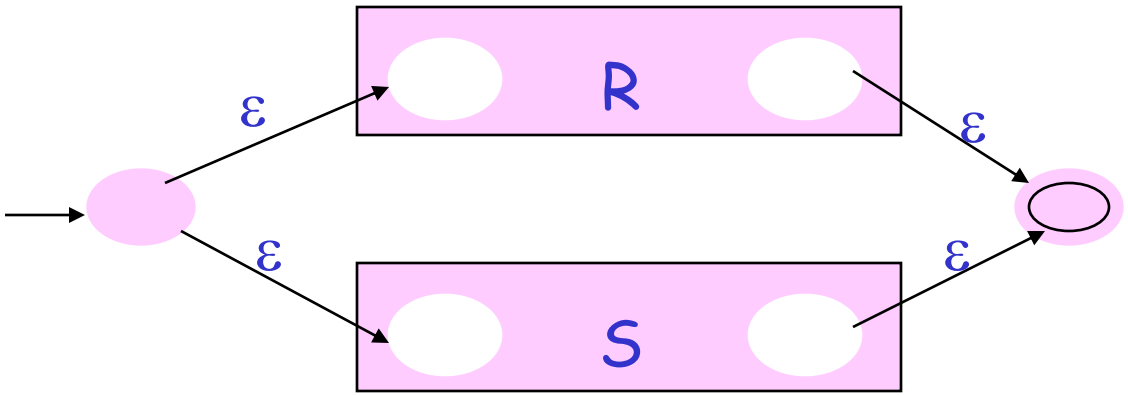


(b)

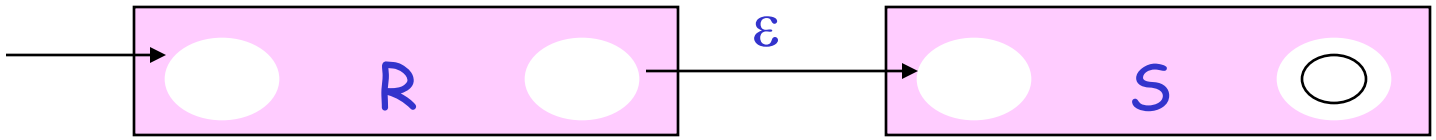


(c)

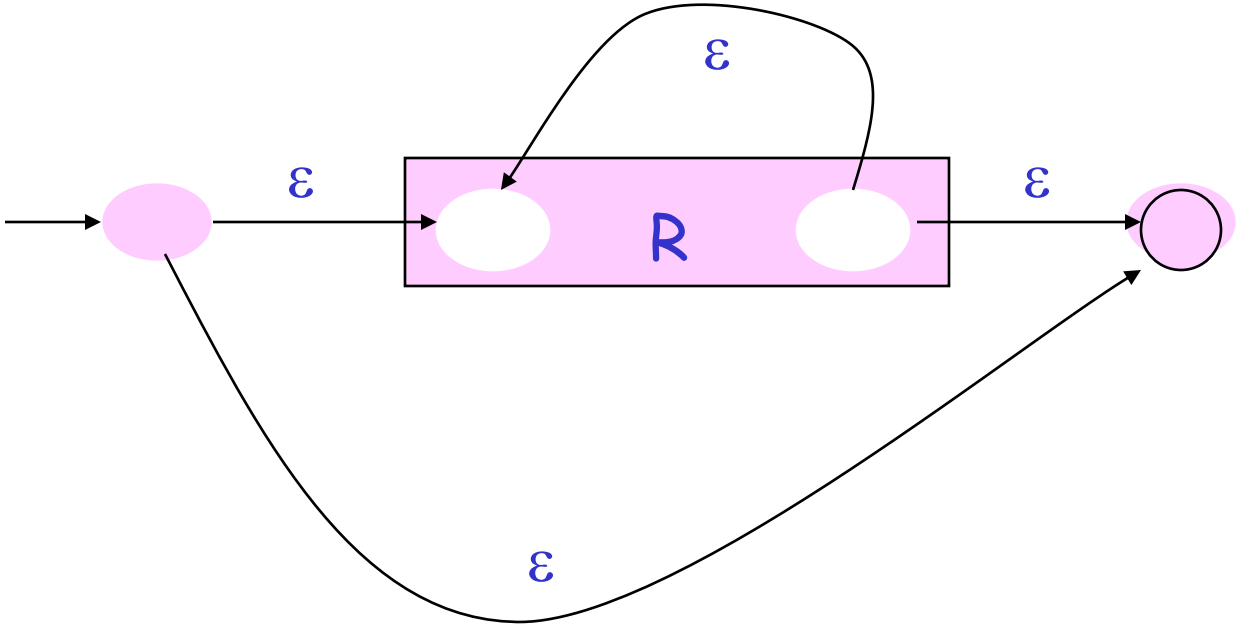
INDUÇÃO: AF para a parte (d) da definição de ER. Se R e S são AF para as respectivas ER R e S, então:



L(R) U L(S)



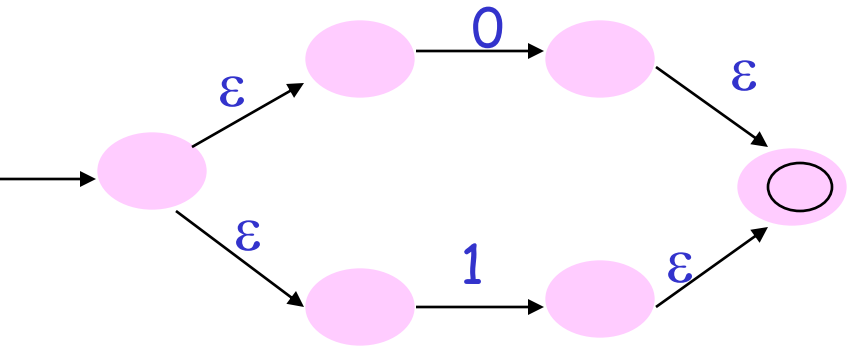
L(R)L(S)



L(R*)

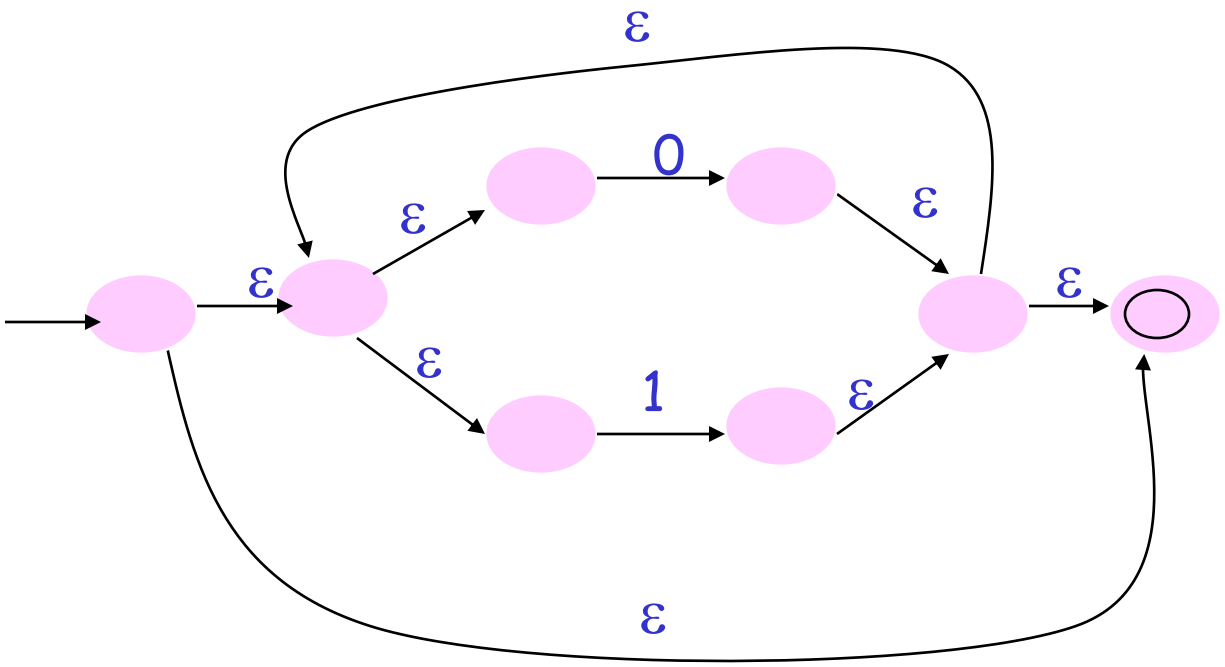
Exemplo

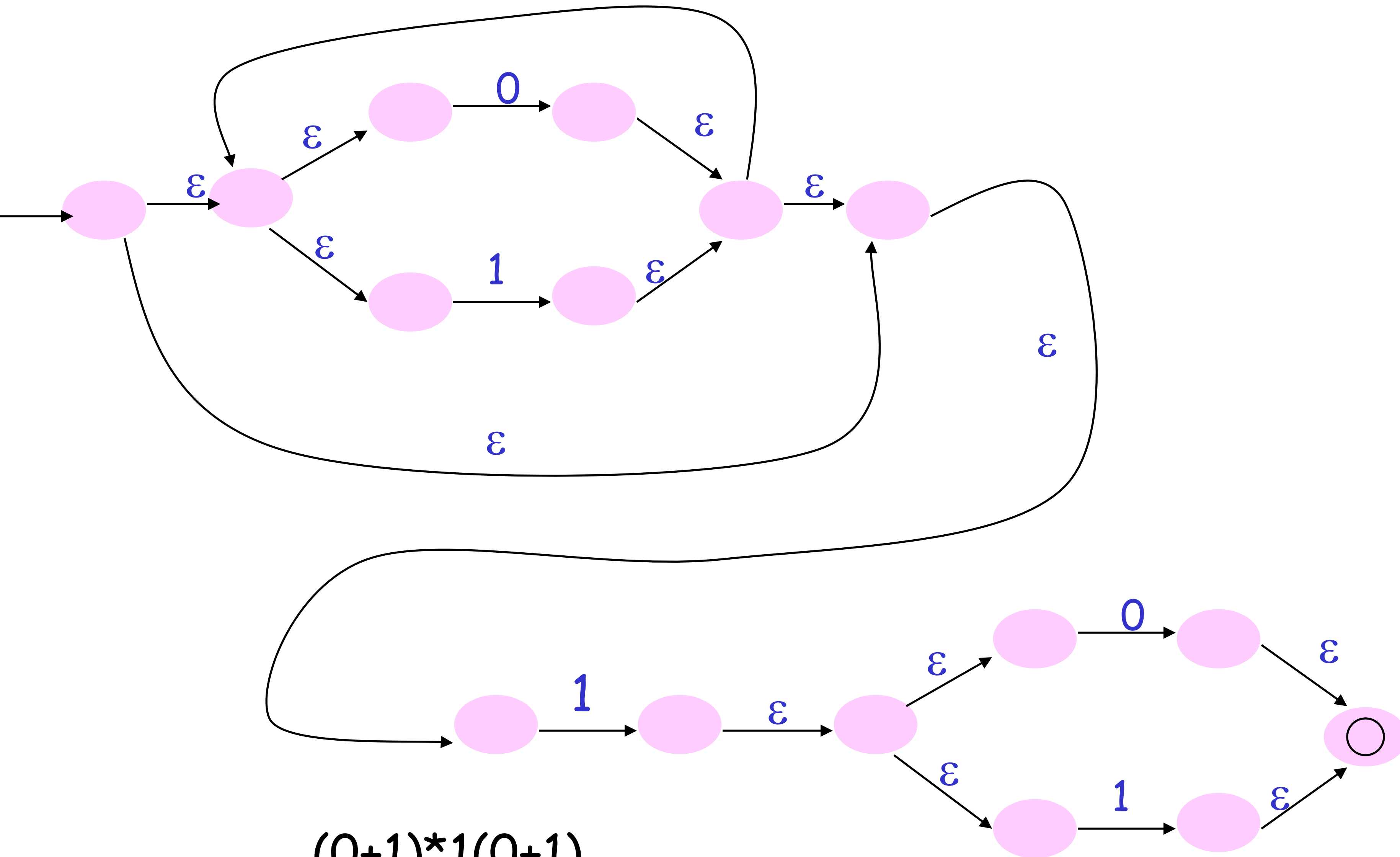
Seja a ER: (0+1)*1(0+1) - cadeias sobre {0,1} cujo penúltimo símbolo é 1.



0+1

(0+1)*





(0+1)*1(0+1)

Propriedades algébricas das ER

- $L + M = M + L$ (união é comutativa)
- $(L + M) + N = L + (M + N)$ (união é associativa)
- $(LM)N = L(MN)$ (concatenação é associativa)
- A concatenação é comutativa???

- $\emptyset + L = L + \emptyset = L$ ($\emptyset$ é o elemento nulo para união)
- $\lambda L = L\lambda = L$ (λ é o elemento nulo para concatenação)
- $\emptyset L = L\emptyset = \emptyset$

- $L(M + N) = LM + LN$ (lei distributiva à esq)
- $(M + N)L = ML + NL$ (lei distributiva à dir)

- $L + L = L$

- $(L^*)^* = L^*$
- $\emptyset^* = \lambda$
- $\lambda^* = \lambda$

- Algumas extensões de ER usadas em utilitários do UNIX

- $L^+ = LL^*$
- $L? = (L + \lambda)$ (usado no Lex para indicar opcional)

Exercícios

Escreva ERs para as linguagens dos:

- identificadores
- números reais
- inteiros
- cadeias de caracteres
- e comentários do Pascal.

Pascal, com L = {a..z,A..Z}; D = {0..9}

- ID: (L|_)(L|D|_)*

- Reais: (+|-|λ) (D⁺ . D⁺ (E (+|-|λ) D⁺ | λ) |
  D⁺ (. D⁺ | λ) E (+|-|λ) D⁺)

Observem que acima exigimos que o real tenha uma parte com ponto fixo ou com ponto flutuante, mas a linguagem pode não exigir e o seu real mínimo seria um inteiro:

[+|-] D⁺ [.D⁺] [E [+|-] D⁺] [x] = (x| λ)

- Inteiros: (+|-|λ) D⁺ = [+|-] D⁺
- Cadeias: ´ C* ´ onde C é ASCII menos ´

(com essa limitação não tratamos os acentos para não perder expressividade)

Comentários em Pascal

{ C* }

onde C é ASCII menos }

Aplicações: Analisadores Léxicos

- Analisadores léxicos (AL) de compiladores. O AL é a interface entre o programa fonte e o resto do compilador. Ele é responsável principalmente por:
  - "empacotar" os caracteres do programa e lhes dar um rótulo que será usado pelo analisador sintático montar a árvore sintática.
  - Os rótulos são:
    - identificadores,
    - os nomes dos símbolos simples (<, =, [,), etc.),
    - os nomes dos símbolos compostos (:= , <>, <=, .., etc.),
    - constantes inteiras,
    - constantes reais,
    - constantes literais (cadeias e caracteres)
    - constantes lógicas (true/false),
    - o nome das palavras-chaves.

AL são modelados por AF para depois serem programados em uma linguagem de programação.

Outra opção é utilizar um gerador de analisadores léxicos como o Lex, Flex.

A entrada para esses geradores é uma expressão regular e a saída é um programa que gerencia uma enorme tabela de transição de estados.