



# Expressões Regulares

Expressões Regulares (ER)

# Expressões Regulares (ER)

Uma ER sobre um alfabeto  $\Sigma$  é definida como:

- a)  $\emptyset$  é uma ER e denota a linguagem vazia
- b)  $\lambda$  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  $RS = \{w \mid u \in R \text{ e } v \in S\}$
  - $(r^*)$  é ER e denota a linguagem  $R^*$

# Exemplos

1.  $00$  é uma ER denotando a linguagem  $\{00\}$
2.  $(0+1)^*$  denota a linguagem formada por todas as cadeias de 0's e 1's =  $\{\lambda, 0, 1, 00, 01, 10, 11, 000, \dots\}$  Pode ser também  $(a^*b^*)^*$
3.  $(0+1)^* 00 (0+1)^*$  denota todas as cadeias de 0's e 1's com ao menos dois 0's consecutivos
4.  $a+b^*c$  denota um único  $a$  e todas as cadeias consistindo de zero ou mais vezes  $b$  seguido de  $c$ . A linguagem é formada por  $\{a, c, bc, bbc, bbbc, \dots\}$

- $(0+1)^* 001$  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 e um único 1 seguido por qualquer número de 0's =  $\{0,01,011,\dots,1,10,100,\dots\}$

## 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  $(0(1^*)) + 1 \Rightarrow L = \{1, 0, 01, 011, \dots\}$
- Usamos parênteses quando queremos alterar a prioridade:
- $(01)^* + 1 \Rightarrow L = \{1 \cup (01)^n \mid n \geq 0\} = \{1, \lambda, 01, 0101, \dots\}$
- $0(1^* + 1) \Rightarrow L = \{w \in \{0,1\}^* \mid w \text{ começa com } 0 \text{ seguido de } 1^n \mid n \geq 0\} \rightarrow \text{Lei distributiva à esq} = 01^* + 01 = \{0, 01, 011, 0111, \dots\}$

## Exercícios

1. O conjunto de cadeias sobre  $\{0,1\}$  que termine com três 1's consecutivos.
2. O conjunto de cadeias sobre  $\{0,1\}$  que tenha ao menos um 1.
3. O conjunto de cadeias sobre  $\{0,1\}$  que tenha no máximo um 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)
- Exercício: a **concatenação** é comutativa???
  
- $\emptyset + L = L + \emptyset = L$  ( $\emptyset$  é o elemento nulo para união)
- $\lambda L = L\lambda = L$  ( $\lambda$  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 LR usadas em utilitários do UNIX
- $L^+ = LL^*$
- $L? = (L + \lambda)$  (usado no Lex para indicar opcional)



# Exercícios

- Faça ER para:
- identificadores
- números reais com sinais
- Inteiros com sinais
- cadeias de caracteres (imprimíveis)
- e comentários do **Pascal** com { ....}
  
- reais do **Fortran** (.5 e 5. além dos padrões de reais de Pascal)

# Sobre Non-Printable Characters (1)

<http://www.regular-expressions.info/characters.html>

- You can use special character sequences to put non-printable characters in your regular expression.
- Use `\t` to match a tab character (ASCII 0x09), `\r` for carriage return (0x0D) and `\n` for line feed (0x0A).
- More exotic non-printables are `\a` (bell, 0x07), `\e` (escape, 0x1B), `\f` (form feed, 0x0C) and `\v` (vertical tab, 0x0B).
- Remember that Windows text files use `\r\n` to terminate lines, while UNIX text files use `\n`.

## Sobre Non-Printable Characters (2)

- You can include any character in your regular expression if you know its hexadecimal ASCII or ANSI code for the character set that you are working with.
- In the Latin-1 character set, the copyright symbol is character 0xA9.
- So to search for the copyright symbol, you can use `\xA9`.
- Another way to search for a tab is to use `\x09`. Note that the leading zero is required.

## Sobre Non-Printable Characters (3)

- Most regex flavors also support the tokens `\cA` through `\cZ` to insert ASCII control characters.
- The letter after the backslash is always a lowercase `c`.
- The second letter is an uppercase letter `A` through `Z`, to indicate `Control+A` through `Control+Z`.
- These are equivalent to `\x01` through `\x1A` (26 decimal).
- E.g. `\cM` matches a carriage return, just like `\r` and `\x0D`.
- In [XML Schema regular expressions](#), `\c` is a [shorthand character class](#) that matches any character allowed in an XML name.
- If your regular expression engine supports [Unicode](#), use `\uFFFF` rather than `\xFF` to insert a Unicode character.
- The euro currency sign occupies code point `0x20AC`. If you cannot type it on your keyboard, you can insert it into a regular expression with `\u20AC`.

- Pascal, com  $L = \{a..z, A..Z\}$ ;  $D = \{0..9\}$
  - ID:  $(L|_)(L|D|_)^*$
  - Reais:  $(+|-|\lambda) (D^+ . D^+ (E (+|-|\lambda) D^+ | \lambda) | D^+ (. D^+ | \lambda) E (+|-|\lambda) D^+ )$
  - 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:
  - $[+|-] D^+ [.D^+] [E [+|-] D^+]$
  - Inteiros:  $(+|-|\lambda) D^+ = [+|-] D^+$
  - Cadeias:  $' C^* '$  onde  $C$  é ASCII menos  $'$
- (com essa limitação não tratamos os acentos na moda antiga do Pascal para não perder expressividade)

# Comentários em Pascal

- { C\* }
- onde C é ASCII menos }

# Reais Fortran

- $(+|-|\lambda) (D^+ . | .D^+ | D^+ . D^+) (E (+|-|\lambda) D^+ | \lambda)$
- $O_u$
- $[+|-] (D^+ . | .D^+ | D^+ . D^+) [E [+|-] D^+]$