



LFTC - Aula 02

Linguagens regulares - introdução

Celso Olivete Júnior

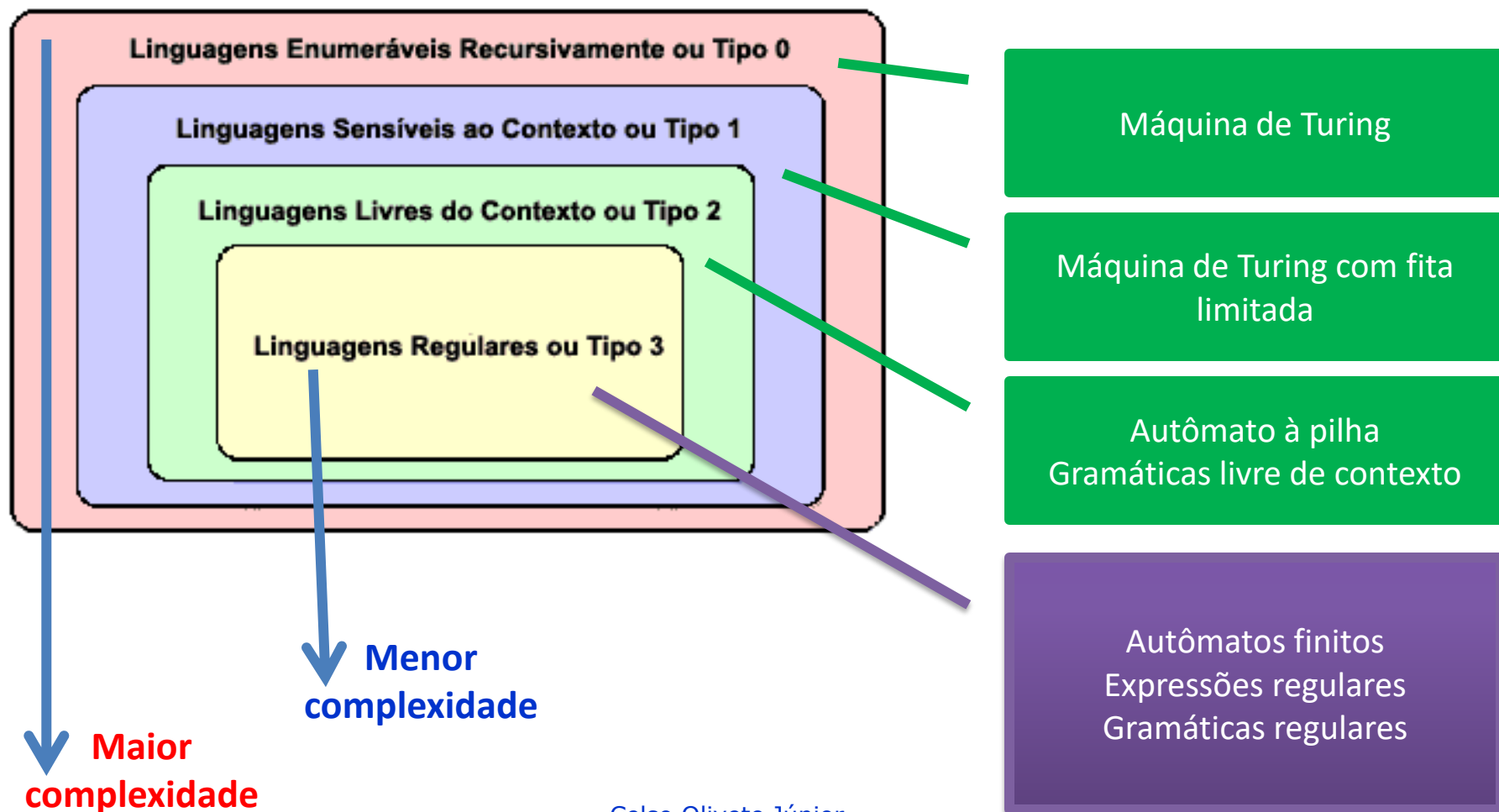
celso.olivete@unesp.br



Na aula passada...

- Visão geral
 - Linguagens regulares
 - expressões regulares
 - autômatos finitos
 - gramáticas regulares

Classificação das Linguagens segundo **Hierarquia de Chomsky** e seus reconhecedores





Na aula de hoje...

- Linguagens regulares: **Expressões regulares**
- Referência bibliográfica

HOPCROFT, J. E.; ULLMAN, J. D.; MOTWANI, R. *Introdução à Teoria de Autômatos, Linguagens e Computação*. Editora Campus, 2002 → Capítulos 1 e 3



Roteiro

- Definições prévias: alfabeto, strings e linguagem
- Expressões regulares



Roteiro

- Definições prévias: alfabeto, strings e linguagem
- Expressões regulares



Definições prévias

- **Alfabeto ou Vocabulário:** Conjunto finito não vazio de símbolos. Símbolo é um elemento qualquer de um alfabeto. Representado por Σ

Ex: $\Sigma = \{a,b\} \rightarrow$ alfabeto formado pelas letras a e b

$\Sigma = \{0,1,2,3,4,5,6,7,8,9\} \rightarrow$ alfabeto formado pelos dígitos de 0 a 9

- **Cadeia, String ou Palavra:** Concatenação finita de símbolos de um alfabeto. Define-se como cadeia vazia ou nula uma cadeia que não contém nenhum símbolo.

Ex: $aab \rightarrow$ string sobre o alfabeto $\Sigma = \{a,b\}$

$123094 \rightarrow$ string sobre o alfabeto $\Sigma = \{0,1,2,\dots,9\}$

ε ou λ representam uma cadeia vazia



Definições prévias

- **Comprimento de uma string:** Número de símbolos de uma cadeia.

Ex: $\Sigma = |\text{aab}| = 3$

$$\Sigma = |123094| = 6$$

$$\Sigma = |\varepsilon| = 0$$

- **Concatenação de string:** Define-se a concatenação z de uma cadeia x com uma cadeia y , como sendo a concatenação dos símbolos de ambas as cadeias, formando a cadeia xy . $|z| = |x| + |y|$

Ex: $x = \text{abaa}; \quad y = \text{ba} \quad \rightarrow \quad z = \text{abaaba}$

$x = \text{ba}; \quad y = \varepsilon \quad \rightarrow \quad z = \text{ba}$



Definições prévias

• **Produto de alfabetos:** É o produto cartesiano de alfabetos.

Ex: $V1 = \{a, b\}$ $V2 = \{1, 2, 3\}$ $\rightarrow V1.V2 = V1 \times V2 = \{a1, a2, a3, b1, b2, b3\}$

• Observe que $V1.V2 \neq V2.V1$

• **Exponenciação de alfabetos:** São todas as cadeias de comprimento n sobre V (V^n). $V^0 = \{\varepsilon\}$, $V^1 = V$, $V^n = V^{n-1}.V$

Ex: $V = \{0, 1\}$

• $V^3 = V^2.V = (V.V).V = \{00, 01, 10, 11\}.\{0, 1\} = \{000, 001, 010, 011, 100, 101, 110, 111\}$



Definições prévias

- **Fechamento (Clausura) de um Alfabeto:** Seja A um alfabeto, então o fechamento de A é definido como $A^* = A^0 \cup A^1 \cup A^2 \cup \dots \cup A^n \cup \dots$

Portanto A^* = conjunto das cadeias de qualquer comprimento sobre o alfabeto a (inclusive nenhum).

Ex: $A = \{1\}$

$$A^* = \{\epsilon, 1, 11, 111, \dots\}$$

- **Fechamento Positivo de A :** $A^+ = A^* - \{\epsilon\}$



Definições prévias

- **Prefixo** de uma palavra é qualquer sequência inicial de símbolos de uma palavra
- **Sufixo** é qualquer sequência final de símbolos de uma palavra
- Relativamente à palavra **abcb**, tem-se que:
 - ϵ , a, ab, abc, abcb são os prefixos
 - ϵ , b, cb, bcb, abcb são os respectivos sufixos



Definições prévias

- **Subpalavra** de uma palavra é qualquer sequência de símbolos contígua da palavra
- Qualquer prefixo ou sufixo de uma palavra é uma subpalavra



Definições prévias: relembrando...

- **Linguagem formal:** é uma coleção de cadeias de símbolos, de comprimento finito. Estas cadeias são denominadas sentenças da linguagem, e são formadas pela justaposição de elementos individuais, os símbolos ou átomos da linguagem.



Linguagem formal

•Ex: $\{ab, bc\}$

linguagem formada pelas cadeias ab ou bc)

$\{ab^n, a^n b; n \geq 0\}$

linguagem formada por todas as cadeias que começam com "a" seguido de um número qualquer de "b"'s (exemplo $a, ab, abb, aab, aaab, \dots$) ou começam com um número qualquer de "a"'s seguidos de um "b"



Problemas

- Na teoria de LFTC, um problema é a questão de **decidir** se determinado **string** é um elemento de uma linguagem
- Se Σ é um **alfabeto** e L é uma **linguagem** sobre Σ , então o problema é:
 - dado um **string** w em Σ^* , definir se w está ou não em L .



Problemas

- Exemplo:

dado um **string** w em Σ^* ,
definir se w está ou não
em L .

Linguagem L : consiste em um número
igual de 0's e 1's

string $w = 01110 \rightarrow w$ **não** é elemento de L

string $w = 0110 \rightarrow w$ **é** elemento de L



Formas de definir uma linguagem

1. Por meio da descrição do conjunto finito ou infinito de cadeias (Formalismo Descritivo)

Ex: Linguagem com quantidade par de elementos;

$$L = \{ 0^n 1^n \mid n \geq 0 \}$$

2. por meio de uma Gramática/Expressão Regular que a gere (Formalismo Gerativo)



Definindo uma linguagem

- É definida usando um "formador de conjuntos" ou

Formalismo Descritivo:

$\{ w \mid \text{algo sobre } w \}$

- Essa expressão é lida como "o conjunto de palavras w tais que (seja o que for dito sobre w à direita da barra vertical)". Exemplos:

$\{w \mid w \text{ consiste em um número igual de 0's e 1's}\}$

$\{w \mid w \text{ é um número inteiro binário primo}\}$

$\{w \mid w \text{ é um programa em } C \text{ sintaticamente correto}\}$



Definindo uma linguagem

- Também é comum substituir w por alguma expressão com parâmetros e descrever os strings na linguagem declarando condições sobre os parâmetros. Exemplo:

$$\{0^n 1^n \mid n \geq 1\}.$$

"o conjunto de 0 elevado a n 1 elevado a n tal que n maior ou igual a 1";

essa linguagem consiste nos strings $\{01, 0011, 000111, \dots\}$.



Definindo uma linguagem: outro exemplo

$$\{0^i 1^j \mid 0 \leq i \leq j\}.$$

Essa linguagem consiste em strings com alguns 0's (possivelmente nenhum) seguidos por um número igual ou superior de 1's.



Formas de definir uma linguagem

1. Por meio da descrição do conjunto finito ou infinito de cadeias (Formalismo Descritivo)

2. por meio de uma Gramática/Expressão Regular que a gere (Formalismo Gerativo)



Roteiro

- Definições prévias: alfabeto, strings e linguagem
- Expressões regulares e linguagens



Expressões regulares

- é uma das formas de **definir** uma linguagem regular
- podem ser consideradas uma "linguagem de programação"
 - aplicação de pesquisas em textos
 - descrever componentes de software
- estão relacionadas com as gramáticas regulares (**definição/geração de sentenças**) e com os autômatos finitos (**aceitação de sentenças**).



Expressões regulares

- servem como uma linguagem de entrada para muitos sistemas de processamento de strings. Exemplos:



Expressões regulares

- comando de pesquisa `grep` do unix
 - O comando `grep` vai procurar um arquivo, texto ou qualquer outra entrada e retornar quaisquer linhas que contenham a string especificada. Digamos que você precisa procurar dentro de um arquivo de log por todos os erros que tenham a ver com o `mysql`.

```
grep 'mysql' arquivo-de-log.log
```



Expressões regulares

- como parte de um componente do compilador

- especificar:

- forma de um identificador

$[a-zA-Z_][a-zA-Z_0-9]^*$

- de um número inteiro com ou sem sinal

- ...



Os operadores de expressões regulares (ER's)

- Como já foi dito, as ER's denotam linguagens regulares. Exemplo:

• $01^* \mid 10^*$: linguagem (L) que consiste em todas as strings que começam com um 0 seguido por qualquer quantidade de número 1 (inclusive nenhum) OU começam com um 1 seguido por qualquer quantidade de 0



Os operadores de expressões regulares

- Os tipos de operadores sobre as ER's são:
 - União
 - Concatenação
 - Fechamento



Os operadores de expressões regulares

• Operador união

- Denotado por $|$
- $L | M$ (Linguagem L união com a Linguagem M).

Corresponde a:

- Conjunto de strings que estão em L ou M , ou em AMBAS.

Exemplo: $L = \{001, 10, 111\}$ e $M = \{\epsilon, 001\}$.

$$L | M = \{\epsilon, 001, 10, 111\}$$



Os operadores de expressões regulares

- Operador concatenação

- Denotado por $.$ (ponto) ou sem nenhum operador
- $L.M$ ou LM (Linguagem L concatenada com Linguagem M). Corresponde a:
 - Conjunto de strings que podem ser formados tomando-se qualquer string em L e concatenando-se com qualquer string em M .



Os operadores de expressões regulares

- Operador concatenação

- Conjunto de strings que podem ser formados tomando-se qualquer string em L e concatenando-se com qualquer string em M .

Exemplo: $L = \{001, 10, 111\}$ e $M = \{\epsilon, 001\}$.

$LM = \{001\epsilon, 10\epsilon, 111\epsilon, 001001, 10001, 111001\}$.

Os 3 primeiros símbolos de LM é o resultado da concatenação de cada sequência de L com ϵ .



Os operadores de expressões regulares

- Operador **fechamento** (ou estrela, ou fechamento de Kleene)

- Denotado por $*$ (asterisco)
- L^* (Fechamento sobre a Linguagem L).

Corresponde a:

- Conjunto de todos strings que podem ser formados tomando-se qualquer número de strings de L (inclusive nenhum), possivelmente com repetições.



Os operadores de expressões regulares

- Operador fechamento (ou estrela, ou fechamento de Kleene)

Exemplo: $L = \{0, 11\}$

$L^* = \{011, 11110, \dots \epsilon\}$

Observe que L é infinito



Construindo expressões regulares

- É necessário algum método para agrupar os operadores com seus operandos, tais como parênteses.
- Por exemplo, a álgebra aritmética começa com constantes como inteiros e números reais, além de variáveis, e elabora expressões mais complexas com operadores aritméticos como $+$ e $*$



Construindo expressões regulares

- A álgebra de ER's segue esse padrão → usa constantes e variáveis que denotam linguagens e operadores para as 3 operações ($|$ união, $.$ concatenação e $*$ fechamento)
- A base para esta definição consiste em:
 1. as constantes ε e \emptyset (vazio) são ER's, denotando as linguagens $L(\varepsilon) = \{\varepsilon\}$ e $L(\emptyset) = \{\emptyset\}$
 2. se a é qualquer símbolo, então a é uma ER, denotando a linguagem $L(a) = \{a\}$
 3. L (letra maiúscula e itálico) representa qualquer linguagem



Construindo expressões regulares

1. Se E e F são ER's, então $E \mid F$ é uma ER denotando a **união** de $L(E)$ e $L(F)$

$$L(E \mid F) = L(E) \mid L(F)$$

2. Se E e F são ER's, então EF é uma ER denotando a **concatenação** de $L(E)$ e $L(F)$

$$L(EF) = L(E)L(F)$$



Construindo expressões regulares

3. Se E é uma ER, então E^* é uma ER denotando o fechamento de $L(E)$

$$L(E^*) = (L(E))^*$$

4. Se E é uma ER, então (E) é uma ER denotando a mesma linguagem que E

$$L((E)) = L(E)$$



Exemplos de expressões regulares

Exemplos

ER formada por 01	ER = 01
ER para strings que são formadas por zero ou mais ocorrências de 01	ER = (01)* é diferente de 01*
ER formada por 0's e 1's alternados	(01)* (10)* 0(10)* 1(01)* resultando, por exemplo, na sequência 010101 101010 01010 10101

Obs: cuidado com o uso dos parênteses!!



Exemplos de expressões regulares

Exemplos

ER para strings que são formadas por zero seguido por qualquer ocorrência de 1 (inclusive nenhuma)

ER = 01^*

ER formada por todas as palavras sobre (a,b) contendo **aa** como subpalavra

ER = $(a|b)^*aa(a|b)^*$



Expressões regulares: precedência de operadores → essencial na omissão de parênteses

Ordem de precedência em ER's



- | | |
|---|--|
| 1 | O operador fechamento (*) é o de precedência mais alta |
| 2 | Em seguida, o operador de concatenação (.) |
| 3 | Por último, todas as operações de união () |



Expressões regulares: precedência de operadores

• Exemplo: $ER = 01^* | 1$

• é agrupada como $(0(1^*)) | 1$.

Agrupando...	
1	O operador fechamento é realizado primeiro
2	Em seguida, o operador de concatenação 0 e (1^*) $\rightarrow 0(1^*)$
3	Por último, o operador de união ($ $)

• Linguagem gerada \rightarrow todos os strings formados por 0 seguido por qualquer número de 1's (inclusive nenhum) **OU** 1

$$L = \{1, 0, 01, 011, \dots\}$$



Expressões regulares: exemplo de aplicação

- como componente léxico
 - Identificadores da linguagem Pascal que são compostos por letras (a.-zA-Z) ou underline(_) seguido por qualquer combinação de letras, underline ou dígitos (0...9)
- Supondo que:
 - letras são representadas por L
 - underline por $_$
 - e os dígitos por D
- ER = $L | _ (L | _ | D)^*$



REGEXP - simulador de expressões regulares

http://tools.lymas.com.br/regexp_br.php#

Expressão

`^(a|b)*c$`

Texto 1

aaaaaaabac



Texto 2

c



Ignorar

Maiusc./minusc.

☐

Considerações

- ❖ ER deve ser escrita **iniciando** com **^**
- ❖ ER deve ser **finalizada** com **\$**

Manual



<http://www.gnu.org/s/libtool/manual/emacs/Regexps.html>

REGEXP - simulador de expressões regulares

ER que reconhece identificadores de uma linguagem de programação **deve ser escrita da seguinte forma**


$^[a-z_][0-9a-z_]*\$$

TESTE DE REGEXP (EXPRESSÕES REGULARES)
Sync, share and backup, online or in LAN - encrypted, direct and independent


 

Expressão

Texto 1



Texto 2





Exercícios (em dupla)

1. Descreva as linguagens denotadas pelas ER's abaixo sobre o alfabeto $\Sigma = \{0,1\}$.

a- $0 \mid 10^*$

b- $(0 \mid 1)0^*$

c- $(0011)^*$

d- $(0 \mid 1)^* 1(0 \mid 1)^*$

e- 0^*11^*0

f- $0(0 \mid 1)^*0$

g- $(\varepsilon + 0)(\varepsilon \mid 1)$

h- $(000^* \mid 1)^*$

i- $(0^* \mid 0^*11(1 \mid 00^*11)^*)(\varepsilon \mid 00^*)$



Exercícios

2. Sobre o $\Sigma=\{a,b\}$, defina expressões regulares que representam as linguagens cujas sentenças estão descritas a seguir:

- ☐ Possuem comprimento maior ou igual a 3;
- ☐ Possuem comprimento menor ou igual a 3;
- ☐ Possuem comprimento diferente de 3;
- ☐ Possuem comprimento par;
- ☐ Possuem comprimento ímpar;
- ☐ Possuem comprimento múltiplo de 4.

3. Fazer o conjunto de exercícios da seção 3.1 do livro do HOPCROFT, páginas 96 e 97.

4. Faça um simulador para Expressões Regulares (qualquer linguagem de programação)