UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

COMPILADORES

Expressões Regulares

Docentes: Ruben Moisés Manhiça
Cristiliano Maculuve

Maputo, 5 de abril de 2023

Conteúdo da Aula

1. Linguagens regulares;
2. Introdução as Expressões Regulares;
3. Aplicação de Expressões Regulares;
4. Uso de Expressões Regulares em Java

Linguagem Regular

Teorema: Uma linguagem é regular se e somente se algum AF que a descreve, como é possível transformar uma ER em Autômato finito, podemos facilmente melhorar a afirmação

Esse teorema tem duas direções. Enunciamos e provamos cada uma das direções como um lema separado.

- Lema 1: Se uma linguagem é descrita por uma expressão regular então ela é regular.
- Lema 2: Se uma linguagem é regular então ela é descrita por uma expressão regular.

Expressões Regulares

- Uma Expressão Regular (ER), também chamada de REGEX (Do inglês Regular Expression), é um método formal de se especificar um padrão de texto.
- É uma composição de símbolos, caracteres com funções especiais, chamados “metacaracteres” que, agrupados formam uma sequencia, ou *expressão regular*.

Expressões Regulares

- Uma expressão regular é testada em textos e retorna sucesso caso este texto obedeça exatamente a todas as suas condições.
- Neste caso dizemos que o texto “casa” com a expressão regular.

- As ERs servem para se dizer algo abrangente de forma mais rigorosa.
- Definido o padrão, tem-se uma lista (finita ou não) de possibilidades de casamento.
- Exemplo:

  [rgp]ato

  pode casar com

  “rato”, “gato” e “pato”

Expressões Regulares

- Expressões regulares são ferramentas úteis no desenho de compiladores para linguagens de programação.

- ***tokens***, tais como os nomes de variáveis e constantes podem ser descritos com expressões regulares.

Utilidade das Expressões Regulares

- Data
- Horário
- Endereço de e-mail
- Endereço IP
- URL
- Número de telefone, Números de Bilhete de identidade, cartão de crédito
- Etc;

Testes com ERs

- Vários editores de texto e linguagens de programação têm suporte às ERs.
- Ex: Java, C/C++, C#, CMD, Terminal, etc

Metacaracteres

Lista []

- Simboliza todos os caracteres que podem aparecer numa determinada posição.
- Exemplo:

```
grep ‘^[cC]arlos’ /etc/passwd
```

Ponto .

- Simboliza “qualquer” caracteres numa determinada posição
- Exemplo:

```
grep ‘^.[aeiou]’ /etc/passwd
```

- Exemplo:

```
grep ‘^.....’ /etc/passwd
```

Metacaracteres

Chaves {}

- Simboliza a quantidade de repetições do caractere anterior
- Exemplo:

```
egrep '^.{5}$'
   /etc/passwd
```

- Exemplo:

```
egrep '^.{5,10}$'
   /etc/passwd
egrep '^.{3,}$'
   /etc/passwd
```

Asterisco *

- Simboliza "qualquer coisa", inclusive nada.
- Exemplo:

```
egrep '^[aeiou].*bash$'
   /etc/passwd
```

Obs: Procura por uma linha que comece com vogal e termine com "bash", não importando o que há no meio.

Metacaracteres

- Ou |

- Para fazer um OU lógico, onde buscamos uma coisa OU outra, deve-se usar o | e delimitar as opções entre parênteses.
- Exemplo:

egrep ‘(root|aluno):’ /etc/passwd

Metacaracteres

- Lista Negada [^]
- Pesquisa para retornar os usuários cujo login começam com consoantes

```
egrep '^[bcdfghjklmnpqrstvwxyz]' /etc/passwd
```

- Negando a lista

```
egrep '^[^aeiou]' /etc/passwd
```

Metacaracteres

- Intervalo em Listas [-]
- Busca por números de três dígitos ou mais

```
egrep '[0123456789]{3,}' /etc/passwd
```

- Pode ser reescrito

```
egrep '[0-9]{3,}' /etc/passwd
```

Metacaracteres

Outros repetidores ? * +

- Definem quantidades e funcionam como as chaves.

Meta	Nome	Equivalente	Descrição
?	opcional	{0,1}	Pode aparecer ou não (opcional)
*	asterisco	{0,}	Pode aparecer em qualquer quantidade de vezes
+	mais	{1,}	Deve aparecer no mínimo 1 vez

Construção de uma ER

- Concatenação
  - A seguido de B
  - L(AB) = { st | s $\in$ L(A) AND t $\in$ L(B) }

- Exemplo
  - a
    - {“a”}
  - ab
    - {“ab”}

Construção de uma ER

- União
  - A or B
  - L(A | B) = L(A) union L(B)
    = { s | s $\in$ L(A) OR s $\in$ L(B) }

- Exemplo
  - a | b
    - {“a”, “b”}

Construção de uma ER

- Fechamento
  - Zero ou mais A
  - $L(A^*) = \{ s \mid s = \varepsilon \text{ OR } s \in L(A)L(A^*) \}$
    $= = \{ s \mid s = \varepsilon \text{ OR } s \in L(A) \text{ OR } s \in L(A)L(A) \text{ OR } ... \}$

- Example
  - a*
    - {ε, “a”, “aa”, “aaa”, “aaaa” …}
  - (ab)*c
    - {“c”, “abc”, “ababc”, “abababc”…}

Expressões Regulares em Java

- Java suporta expressões Regulares
  - Através da biblioteca java.util.regex.*
  - Aplicada a classe String a partir da versão Java 1.4
- Introduz novos metodos de especificação
  - Simplificada
  - Não incrementa o poder das expressões regulares
  - Pode simular concatenação e fechamento

Regular Expressions in Java

- Concatenação
  - ab “ab”
  - (ab)c “abc”
- União (barra | ou [ ])
  - a | b “a”, “b”
  - [abc] “a”, “b”, “c”
- Fechamento (star *)
  - (ab)* ε, “ab”, “abab”, “ababab” …
  - [ab]* ε, “a”, “b”, “aa”, “ab”, “ba”, “bb” …

Expressões Regulares em Java

- Um ou mais (+)
  - a+ 1 ou mais “a”s
- Intervalo (–)
  - [a–z] qualquer letra minuscula
  - [0–9] qualquer digitos
- Complementos (^ no começo da reunião)
  - [^a] Qual coisa menos “a”
  - [^a–z] Qualquer coisa menos letras minusculas

Expressões Regulares em Java

- Precedencias
  - Respeita-se a ordem de precedencia
- Ordem de Precedencia
  - Parentesis (…)
  - Fechamento a* b+
  - Concatenação ab
  - União a | b
  - Intervalo […]

Expressões Regulares em Java

- Exemplos
  - ab+ “ab”, “abb”, “abbb”, “abbbb”…
  - (ab)+ “ab”, “abab”, “ababab”, …
  - ab | cd “ab”, “cd”
  - a(b | c)d “abd”, “acd”
  - [abc]d “ad”, “bd”, “cd”

Expressões Regulares em Java

- Classes de carecteres pre-definidas em java
  - [.] Qualquer caractere excepto fim da linha
  - [\d] Digito: [0-9]
  - [\D] Não digito : [^0-9]
  - [\s] Caracteres de espaçamento: [\t\n\x0B\f\r]
  - [\S] não caracteres de espaçamento: [^\s]
  - [\w] caracteres de palavras: [a-zA-Z_0-9]
  - [\W] Não caracteres de palavras: [^\w]

FIM!!!

Duvidas e Questões?