Lecture 2: Regular Expressions

Dr Kieran T. Herley

Department of Computer Science
University College Cork

2018/19

Summary

Regular expressions and operators $\cdot, |$ *and* $*$. *Regular expressions and lexical analysis. Limitations of expressive power regular expressions.*

Regular Expressions

Regular Expression A *regular expression* is a notational device used to certain characterize languages more compactly than with the standard set notation.

Example

$$\begin{aligned} \{111\} \cdot (\{0\} \cup \{1\})* & \quad \text{set expression} \\ 111(0|1)* & \quad \text{regular expression} \end{aligned}$$

Unix Many Unix utilities (awk, grep, text editors) use form of regular expression to characterize "patterns" of some kind. [1]

[1] Typically use slightly different notation from that presented here.

Definition of Regular Expression (RE)

- Let lower-case a, b, c etc. denote alphabet symbols.
- Let upper-case R, S etc. denote regular expressions.
- Let $\mathcal{S}(X)$ denote the language (set) represented by regular expression X.

RE	**Set**	Note
$\emptyset$	$\emptyset$	
ϵ	$\{\epsilon\}$	
a	$\{a\}$	omit $\{, \}$
$(R)\|(S)$	$\mathcal{S}(R) \cup \mathcal{S}(S)$	use $\|$ in place of $\cup$
$(R)(S)$	$\mathcal{S}(R) \cdot \mathcal{S}(S)$	omit $\cdot$
$(R)*$	$(\mathcal{S}(R))*$	
(R)	$\mathcal{S}(R)$	

Regular Expression Basics

- Let lower-case a, b, c etc. denote alphabet symbols.
- ϵ denotes the empty string
- abc denotes a followed by b followed c

Regular Expression Basics

- Let lower-case a, b, c etc. denote alphabet symbols.
- ϵ denotes the empty string
- *abc* denotes *a* followed by *b* followed *c*
- Vertical bar symbol $|$ denotes "or":
  - $0|1$ either 0 or 1 i.e any one-bit string bit

Regular Expression Basics

- Let lower-case a, b, c etc. denote alphabet symbols.
- ϵ denotes the empty string
- abc denotes a followed by b followed c
- Vertical bar symbol $|$ denotes "or":
  - $0|1$ either 0 or 1 i.e any one-bit string bit
  - $(0|1)(0|1)(0|1)$ any three-bit string

Regular Expression Basics

- Let lower-case a, b, c etc. denote alphabet symbols.
- ϵ denotes the empty string
- abc denotes a followed by b followed c
- Vertical bar symbol $|$ denotes "or":
  - $0|1$ either 0 or 1 i.e any one-bit string bit
  - $(0|1)(0|1)(0|1)$ any three-bit string
  - $\epsilon|1011$ empty string or 1011

Star Operator

- The $*$ operator is used to denote repetition: [2]

Definition

$\alpha*$ indicates the concatenation zero of more strings matching α

[2] The star operator is *not* the same as Unix command line $*$.

Star Operator

- The $*$ operator is used to denote repetition: [2]

Definition

$\alpha*$ indicates the concatenation zero of more strings matching α

-

Example

- $1*$ strings consisting entirely of 1s

[2] The star operator is *not* the same as Unix command line $*$.

Star Operator

- The $*$ operator is used to denote repetition: [2]

Definition

$\alpha*$ indicates the concatenation zero of more strings matching α

-

Example

- $1*$ strings consisting entirely of 1s
- $(0|1)*$ all binary strings

[2] The star operator is *not* the same as Unix command line $*$.

Star Operator

- The $*$ operator is used to denote repetition: [2]

Definition

$\alpha*$ indicates the concatenation zero of more strings matching α

-

Example

- $1*$ strings consisting entirely of 1s
- $(0|1)*$ all binary strings
- $1(0|1)*$ binary strings beginning with 1

[2] The star operator is *not* the same as Unix command line $*$.

Star Operator

- The $*$ operator is used to denote repetition: [2]

Definition

$\alpha*$ indicates the concatenation zero of more strings matching α

-

Example

- $1*$ strings consisting entirely of 1s
- $(0|1)*$ all binary strings
- $1(0|1)*$ binary strings beginning with 1
- $(0|\epsilon)1*$ strings of 1s, optionally preceeded by a 0

[2] The star operator is *not* the same as Unix command line $*$.

Note on Parentheses

Just as with ordinary arithmetic expressions, regular expressions can be "de-cluttered" by adopting following conventions.

Operator Precedence

	highest		lowest
Arithmetic Exp.	$\uparrow$	$*, /$	$+, -$
Regular Exp.	$*$	$\cdot$	$\mid$

Left-to-right Association

Arithmetic Exp.	$*, /, +, -$
Regular Exp.	$\mid, \cdot$

Omitting $\cdot$ Concatenation operator often omitted, where meaning is clear.

Some Examples

Assume binary alphabet $\Sigma = \{0, 1\}$.

$$\begin{array}{rcl} 0|1* & \rightarrow & (0|(1*)) \\ 0 \cdot 1|1 \cdot 1 & \rightarrow & ((0 \cdot 1)|(1 \cdot 1)) \\ 0 \cdot 1 \cdot 0 & \rightarrow & ((0 \cdot 1) \cdot 0) \\ 0|1 \cdot 0|0 \cdot 1 & \rightarrow & ((0|(1 \cdot 0))|(0 \cdot 1)) \end{array}$$

- (1) $*$ binds more tightly than $|$, so single zero or a string consisting entirely of 1s
- (4) $\cdot$ binds more tightly than $|$, so 0 or 10 or 01

Example 1

Example

Strings with exactly one b over $\Sigma = \{a, b, c\}$:

$$(a|c) * b(a|c)*$$

Example 1

Example

Strings with exactly one b over $\Sigma = \{a, b, c\}$:

$$(a|c) * b(a|c)*$$

$$\underbrace{(a|c)*}_{1}\underbrace{b}_{2}\underbrace{(a|c)*}_{3}$$

1. Zero or more *a*s or *c*s *i.e.* non-*b*s
2. Single *b*
3. Zero or more *a*s or *c*s

Example 2

Example

Strings with at least one b:

$$(a|c) * b(a|b|c)*$$

Example 2

Example

Strings with at least one b:

$$(a|c)*b(a|b|c)*$$

$$\underbrace{(a|c)*}_{1}\underbrace{b}_{2}\underbrace{(a|b|c)*}_{3}$$

1. Zero or more *a*s or *c*s *i.e.* non-*b*s
2. Single b
3. Zero or more *a*s, *b*s or *c*s

Example 3

Example

Strings with at most one b:

$$(a|c)*|(a|c)*b(a|c)* \quad \text{or} \quad (a|c)*(b|\epsilon)(a|c)*$$

Example 3

Example

Strings with at most one b:

$$(a|c)*|(a|c)*b(a|c)* \quad \text{or} \quad (a|c)*(b|\epsilon)(a|c)*$$

$$\underbrace{(a|c)*}_{1}\,|\,\underbrace{(a|c)*b(a|c)*}_{2}$$

1. Zero or more *a*s or *c*s
2. Strings with exactly one b

Example 3

Example

Strings with at most one b:

$$(a|c)*|(a|c)*b(a|c)* \quad \text{or} \quad (a|c)*(b|\epsilon)(a|c)*$$

$$\underbrace{(a|c)*}_{3}\underbrace{(b|\epsilon)}_{4}\underbrace{(a|c)*}_{5}$$

3. Zero or more *a*s or *c*s
4. Optional *b*s
5. Zero or more *a*s or *c*s

Example 4

- Goal: Strings with no two consecutive *b*s

Example 4

- Goal: Strings with no two consecutive bs
- Idea: Each b must
  - *Either* be followed by a non-b (i.e. a or c)
  - *Or* be the very last symbol
- Strings where every b is followed by an non-b:

$$(a|c|ba|bc)*$$

Example 4

- Goal: Strings with no two consecutive bs
- Idea: Each b must
  - *Either* be followed by a non-b (i.e. a or c)
  - *Or* be the very last symbol
- Strings where every b is followed by an non-b:

$$(a|c|ba|bc)*$$

- Excludes all strings containing double-b, but also excludes strings ending in b *e.g.* $abcbab$ which we want to capture

Example 4 cont'd

Example

Strings with no two consecutive bs

$$(a|c|ba|bc)*(b|\epsilon)$$

$$\underbrace{(a|c|ba|bc)*}_{1}\underbrace{(b|\epsilon)}_{2}$$

1. Strings where each b is immediately followed by either a or c.
2. Optional b at the end

Yet More Examples

Example

Binary numbers with no leading zeros.

$$1(0|1)*|0$$

Example

Binary strings not containing two adjacent zeros.

$$(0|\epsilon)(11*0)*1*$$

Yet More Examples cont'd

Example

Binary strings containing exactly one 00 substring.

$$(0|\epsilon)(11*0)*11*00(11*0)*1*$$

Note similarity with above.

$$\underbrace{(0|\epsilon)(11*0)*11*}_{1}\underbrace{00}_{2}\underbrace{(11*0)*1*}_{3}$$

1. no two conseq. zeros but must end with a one
2. the 00
3. no two conseq. zeros but cannot begin with a zero

Useful Extensions

Repetitions

R^*	zero or more repetitions of R
R^+	one or more repetitions of R

Sets (lex)
[3]

$[aeiou]$	shorthand for $(a\|e\|i\|o\|u)$

Complements (lex)

$[\wedge aeiou]$	single-char. strings except for a, e, i, o, u

Ranges (lex)

$[0-9]$	shorthand for $(0\|1\|\cdots\|9)$ (hyphen indicates range)
$[a-zA-Z]$	shorthand for $(a\|b\cdots z\|A\|B\cdots Z)$

[3] lex is a classic C-based scanner-generator tool; we will use jflex, Java-based cousin, that uses a similar notation.

REs and Token Recognition

Let

$$\begin{aligned} \mathcal{L} &= [a - zA - Z] \\ \mathcal{D} &= [0 - 9] \end{aligned}$$

Pascal identifiers (one or more letters of digits, beginning with a letter) :

$$\overbrace{\mathcal{L}}^{1}\overbrace{(\mathcal{L}|\mathcal{D})*}^{2}$$

- 1: A single letter
- 2: (followed by) zero or more letters or digits

Example (Pascal Real Constants)

Examples

$$17, \quad 3.14159, \quad 2.99E8$$

RE

$$\overbrace{\mathcal{D}+}^{1}\overbrace{(\epsilon|\odot\mathcal{D}+)}^{2}\overbrace{(\epsilon|E(\epsilon|\oplus|\ominus)\mathcal{D}+)}^{3}$$

Explanation

-
  - 1: One or more digits
  - 2:(followed by) optional mantissa (decimal point $\odot$ followed by one or more digits)
  - 3:(followed by) optional exponent (capital E followed by optional sign ($\oplus/\ominus$) followed by one or more digits)

REs and Lexical Analysis

- Lexical structure of most programming langauages can be captured by regular expressions
- From REs we can generate software to recognize lexical elements of source programs
- Greatly simplifies creation of compiler component that decomposes source into constituent lexical elements (tokens)

Limitations of REs

- Many "patterns" cannot be expressed as regular expressions.
- Examples:
  - $L = \{a^n b^n : n \geq 0\}$
  - palindromes
  - balanced parentheses
  - binary representations of prime numbers

Some RE-Based Unix Tools

awk Searches file for specified pattern(s) and performs actions each time it finds a match.

sed Batch (i.e. non-interactive editor); performs series of edit commands (taken from a script file) on a file.

grep Searches a file(s) for specified pattern(s) and flags every line that contains a match.

lex Lexical scanner generator; language tokens (ids, nums, symbols etc.) specified by REs; lex generates C function that reads source and "recognises" tokens. More on this later.

note Unix RE notation may differ slightly from that employed here.

grep

- `grep 'and' gettysburg.txt`
- Scans file `gettysburg.txt` for of pattern "and" and outputs those lines that contain a match.
- ```
  Four score and seven years ago our fathers brought forth
  and dedicated to the proposition that all men are
  and so dedicated, can long endure. We are met on a
  nation might live. It is altogether fitting and proper
  hallow - this ground. The brave men, living and dead,
  God, shall have a new birth of freedom - and that
  ```
- Many options; Fancier features allow for more complex patterns (regular expressions), searches though directories etc.
  ```