Regular Expressions

CSC207 Fall 2017

Computer Science
UNIVERSITY OF TORONTO

Describing a set of characters

- Eclipse describes the Java naming conventions for variables this way:

```
^[a-z][a-zA-Z0-9]*$
```

- This is a *regular expression* (or *regex*)
- This describes a pattern that appears in a set of strings. We say that any such string *matches* or *satisfies* the regular expression.

`^[a-z][a-zA-Z0-9]*$`

- The `^` character means that the pattern must start at the beginning of the string. This is called an *anchor*.
- Square brackets `[]` tell you to choose one of the characters listed inside. In the leftmost set of brackets, we are given all lowercase English letters to choose from. The second character will come from the second set of square brackets. It can be any lowercase letter, uppercase letter, or digit.
- The `*` means zero or more of whatever immediately precedes it.
- The `$` signifies the end of the string. This is another anchor.

`^[a-z][a-zA-Z0-9]*$` continued…

- So our entire string must be made out of letters and numbers, with the first character being a lower case letter.
- Here are some examples:

```
x, numStudents, obj1
```

- These do not satisfy the regular expression. (Why not?)

```
Alphabet, 2ab, next_value
```

- Do any of these strings match?

```
z3333, aBcB041, 78a
```

What if there are no anchors?

Here is another regular expression:

```
[abc]C[a-e][24680]*[A-Z]
```

When this regex is applied to a string, it will find the substrings that match.

- `cCaA` matches the entire expression
- `ABCcCcCa1A23` contains substrings that match: `cCcC` but not `cCa1A`

Other symbols

- Space characters:
  - `\s` is a single space
  - `\t` is a tab character
  - `\n` is a new line character
- Inside square brackets, `^` has another meaning: it matches any character *except* the contents of the square brackets.
  - For example, `[^aeiouAEIOU]` matches anything that isn't a vowel.

Quantifiers

- `*` means zero or more, `+` means one or more, and `?` means zero or one.
- We append `{2}` to a pattern for exactly two copies of the same pattern, `{2,}` for two or more copies of the same pattern, and `{2,4}` for two, three, or four copies of the same pattern.

Pattern	Matches	Explanation
a*	'' 'a' 'aa'	zero or more
b+	'b' 'bb'	one or more
ab?c	'ac' 'abc'	zero or one
[abc]	'a' 'b' 'c'	one from a set
[a-c]	'a' 'b' 'c'	one from a range
[abc]*	'' 'acbccb'	combination

Escaping a Symbol

- Sometimes we want symbols to show up in the String that otherwise have meanings in regular expressions. To "escape" the meaning of the symbol, we write a backslash `\` in front of it.
- A period `.` means any character. To have a period show up in the string, we write `\.`

one or more "."

- Ex 1: `abc123` matches the regex `[a-e][a-e].+`
- Ex 2: `1.4` matches the regex `[0-9]\.[0-9]`

Character Classes

- You can make your own character classes by using square brackets like `[q-z]`, `[AEIOU]`, and `[^1-3a-c]`, or you can use a predefined class.

Construct	Description
.	any character
\d	a digit [0-9]
\D	a non-digit [^0-9]
\s	a whitespace char [\t\n\x0B\f\r]
\S	a non-whitespace char [^\s]
\w	a word char [a-zA-Z_0-9]
\W	a non-word char [^\w]

Repetition of a pattern vs. a specific choice of character

- Here is a pattern that describes all phone numbers on the same continent:

```
\(\d\d\d\) \d\d\d - \d\d\d\d
```

- We could also write this as

```
\( \d{3} \) \d{3} - \d{4}
```

- Here we want to repeat the pattern, but not necessarily the digit. So `(123) 456-7890` matches the pattern.

Repetition of exact characters

- To repeat the same character twice, we use groups which are denoted by round brackets. Then we escape the number of the group we want to repeat:
- The string `124124a124` matches the regular expression:
- `(\d\d\d)\1a\1`
- Groups are assigned the number of open brackets that precede them.
- For example, `^(([ab])c)\2\1$` will repeat both groups. The Strings that match are: `acaac` and `bcbbc`

Logical operators

- | means "or"
- `&&` means the intersection of the range before the ampersands and the range that appears after. For example `[a-t&&[r-z]]` would only include the letters `r`, `s`, and `t`.
- If A is true X, else Y is written as: `(?(A)X|Y)`
  - "If the regex A appears in the string, the X must follow; otherwise, if it doesn't, then Y must appear."

Construct	Description
[abc]	a, b, or c (simple class)
[^abc]	any char except a, b, or c (negation)
[a-zA-Z]	a through z or A through Z inclusive (range)
[a-d[m-p]]	a through d or m through p (union)
[a-z&&[def]]	d, e, or f (intersection)
[a-z&&[^bc]]	a through z except for b and c (subtraction)
[a-z&&[^m-p]]	a through z and not m through p (subtraction)

Pattern	Text	Result
b+	abbc	Matches
^b+	abbc	Fails (no b at start)
^a*$	aabaa	Fails (not all a's)

For an excellent quick reference, take a look at:

http://www.rexegg.com/regex-quickstart.html