Regular Expressions

— A regular expression (RE) represents a set of strings.
— All the strings share a pattern specified by the RE.
— Think of a (RE) as a way to express a pattern.

Example:

`ab(ab)*(a+b)` is "ab followed by any number of ab's followed by a or b".

Patterns: concatenation

When thinking of patterns concatenation means sequence.

RE: `abc` matches a followed by b followed by c

RE: $R_1 R_2$ matches R_1 string followed by R_2 string

Examples:
`(a*)(b*)` = `a*b*` = (any a's) followed by (any b's)
`(hat)+(hot)` = `hat+hot` matches "hat" or "hot"

Patterns: +

When thinking of patterns + means "or"

RE: `a+b` matches "a" or "b"

RE: $R_1 + R_2$ matches string from R_1 or string from R_2

Examples:
`(a*)+(b*)` = `a*+b*` matches string of all a's or all b's
`h(a+o)t` matches "hat" or "hot" (parens required here)

Patterns: *

When thinking of patterns * means "any number of"

RE: `a*` matches any number of a's (incl 0)

RE: R^* matches any sequence of strings from R

Examples:
`(ab)*` matches ab repeated any number of times
λ, ab, abab, ababab, ...
`(ab*)*` matches ab* repeated any number of times
λ, a, ab, abb, abb|ab, a|ab|ab|ab, a|a|a|a|a, ...

Common Patterns: (a+b)*

`(a+b)*` = every possible string over {a,b}

`a(a+b)*` = is every possible string starting with a

Can be used with bigger building blocks...

`((a+b)(a+b))*` matches (a or b)(a or b) any number.
λ, aa, ba, ab|ab, aa|bb|aa, ...
All the strings of length a multiple of 2

Common Patterns: R + λ

To say something is optional use or with empty string.

`R +` λ matches string from R or an empty string.

Example:
`(s+`λ`)pot` matches "s" or empty followed by "pot"
spot, pot

Common Patterns: R1 + R2

Break big problem into smaller ones.

If $L = A \cup B$ then make REs for A and B instead.

Strings beginning and ending with same character {a,b}
(beginning/ending a) ∪ (beginning/ending b)
`(a (a+b)* a) + (b (a+b)* b)`

Use + to add missing elements
`(a (a+b)* a) + (b (a+b)* b) + a + b`

Design method

1. Think in patterns
   (any char)(any string)(same char)
   a(any string)a + b(any string)b
   `a(a+b)*a + b(a+b)*b`
2. Try to "break" your RE
   Find a string it produces that it shouldn't
   Find a string it doesn't produce that it should
   This is how I grade quizzes

Example: $\{w \in \{0,1\}^* \mid w \text{ has exactly one } 1\}$

Pattern: (any number of 0) 1 (any number of 0)

RE: `0* 1 0*`

What does it generate?
{0}* {1} {0}*
{λ, 0, 00, 000, ...} {1} {λ, 0, 00, 000, ...}
{1, 01, 10, 001, 010, 001, 0001, 0010, 0100, 1000, ...}

Fits description!

Example: $\{w \in \{0,1\}^* \mid w \text{ has at least one } 1\}$

Pattern: (any string) 1 (any string)

RE: `(0+1)* 1 (0+1)*`

What does it generate?
{0,1}* {1} {0,1}*
{λ, 0, 1, 00, 01, 10, 11, ...} {1} {λ, 0, 1, 00, 01, 10, 11, ...}
{1, 01, 10, 11, 001, 010, 011, 100, 101, 111, ...}
Useful to think in length order.
Write all length 1, then length 2, then length 3, ...

Example: $\{w \in \{0,1\}^* \mid w \text{ has even length}\}$

Even definition: n even iff $n = 2k$ for some integer k.

Pattern: (2 characters) any number of times

RE: `(00+01+10+11)* = ((0+1)(0+1))*`

Example: $\{w \in \{0,1\}^* \mid w \text{ is not length } 3\}$

It's harder to express what's missing in a pattern.

Break into subproblems.

Pattern:
(len 0) + (len 1) + (len 2) + (len 4) + (len 5) + ...
(len 0) + (len 1) + (len 2) + (len $\geq$ 4)

RE:
λ + (0+1) + (0+1)(0+1) + (0+1)(0+1)(0+1)(0+1)(0+1)*