regular expression syntax
from regular sets to regular expressions

- A regular language is exactly those languages that are defined by regular expressions
- Common feature in many languages; but the basics of regular expressions are much simpler than what you see in languages like Perl or Python

- Basic regular expression syntax corresponds to operations on regular sets:
  - A string of characters is a regular set: $r_1 = \text{hello}$
  - The **choice** operator unions together two regular sets: $r_3 = r_1 | r_2$
  - The **star** operator repeats a regular set 0 or more times: $r_2 = r_1^*$
  - Can use parentheses for operator precedence: $r = (\text{hello})^*(\text{world} | \text{class})$
• For convenience, regular expression engines provide a lot of syntax that makes it easier to define regular sets (but do not make regular expressions more expressive)

• Can make a sub-expression optional: $r_2 = r_1? = (r_1|\varepsilon)$

• Can repeat a sub-expression one or more times: $r_2 = r_1^+ = r_1r_1^*$

• Can match a range of characters: $r = [a-c] = (a|b|c)$

• Can match any character: $r = . = (a|b|c| \ldots)$
A digit: [0-9]

An integer: -?[1-9][0-9]*

A floating point literal: -?[0-9]+.[0-9]*

An identifier: (_|[A-Z]|a-z)+

- Question: does this match keywords like ‘if’ and ‘while’?
next: how do we match a regex?

Or: Time for some automata theory