Day 3 - 1/22/2024

Regular Expressions - *Regex*

Synatx is about correct structure.

Semantics is about what something means

Arithematic Expressions

First, the pieces:

- A numberic literal: 26.5, 1, 100000
- An identifier (variable): x, x, z

Then the slides went too fast.

Examples

What language does this regex generate: $(a|x)^* cb$

$$\underline{\{cb, a^n x^m cb\}}$$

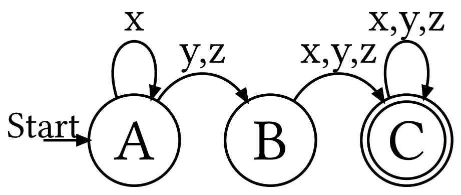
What language does this regex generate: $(xy)^* (|t)$

$$\{\varepsilon, (xy)^n \mid t\}$$

Write a regex to generate: $\{y, xy, xxy, xxxy, ..., z, xz, xxz, xxxz\}$

$$x^* (y|z)$$

Draw the DFA diagram for the previous language:



Imagine B is an accepting state.

In Practice

We end up with cumbersome regex statements ((1|2|3|4|5|6|7|8|9|0)).

Regex libraries are available and offer many QOL shortcuts.

First, "." matches any character.

Second, brackets with a list of chars between them will match any one of the chars in the list: $[qwerty] \Leftrightarrow [q|w|e|r|t|y]$.

Using a "-" specifies a range of consecutive characters.

The following will match any single ASCII letter: $[A - Za - z]$.

"+" means one or more, the same how "*" means one or more: $abc+ \Leftrightarrow (abc)^*$

"?" means zero or one - optional. $x(abc)? \Leftrightarrow x \mid xabc$

"\" escapes a special character. "\\" escapes backslash, "\." specifies just ".".