Office hours:
Discord

formal languages (just languages): a set of strings over some alphabet Σ.

Σ^*: all strings over Σ

language is a subset of Σ^*

$\emptyset$ empty

$\{\varepsilon\}$

$\{0,1\}^*$

{BABA, KIKI, FOFO, JIJI}

binary strings from $\{0,1\}^*$ with an odd number 1s

valid Python programs

$A \cup B$

$A \cap B$

$A \setminus B$

$A \oplus B$

$\overline{A} = \Sigma^* \setminus A$ complement of A

concatenation $A \circ B$ or AB

$$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$$

Ex: $\{SUPER, SPIDER, BAT\} \circ \{MAN, WOMAN\}$

$$\emptyset \circ A = A \circ \emptyset = \emptyset$$

$$\{\epsilon\} \circ A = A \circ \{\epsilon\} = A$$

<u>Kleene closure</u> or <u>Kleene star</u> of L

is L^*

$$L^* = \{\varepsilon\} \cup L \cup L \bullet L \cup L \bullet L \bullet L \cup \ldots$$

or

smallest solution to

$$L^* = \{\varepsilon\} \cup L \bullet L^*$$

$w \in L^*$ iff $w = \varepsilon$ or $w = xy$ $\quad x \in L$, $y \in L^*$

Ex: $\{0, 11\}^* = \{\epsilon, 0, 11, 00, 011, 110, 1111, 000, 0011, \ldots\}$

Σ^* $\qquad \emptyset^* = \{\epsilon\} = \{\epsilon\}^*$

If L has even one non-empty string, L^* is infinite.

Language L is <u>regular</u> iff L satisfies one of...

- L is empty
- L contains exactly one string
- L is the union of two regular languages
- L is the concatenation of two regular languages
- $L = K^*$ for regularr language K

regular expressions:

- $\emptyset$
- singletons $\{w\}$ as w
- $A + B$ ← union of A & B's language
- AB ← concatenation of A & B's languages
- A^* ← kleene closure of A's language

$$0 + 10^* = \{0\} \cup (\{1\} \cdot \{0\}^*)$$
$$= \{0, 1, 10, 100, 1000, \dots\}$$

non-regular: $\{0^n 1^n \mid n \in \mathbb{Z}^+\}$
pos. integers

regular expression

$L(R)$: language represented by R

w <u>matches</u> R if $w \in L(R)$

regular: output by a program with no functions or unbounded variables

union: if / else
concatenation: sequences of instructions

kleene closures: while loop

$0+10^{*}$:

```
if ____:
    print 0
else:
    print 1
    while ____:
        print 0
```

even length binary strings:

$$((0+1)(0+1))^*$$

alternate 0's & 1's (no 00 or 11 anywhere)

$$(\varepsilon+1)(01)^*(\varepsilon+0)$$

non-negative binary numerals divisible by 4 with no redundant leading 0s.

$$0+1(0+1)^*00$$

Building regular expressions

case analysis

identify simple subpatterns

$$(0+1)^*$$

$$(00^*)$$

$$(01)^*$$

break output into small chunks, work on these separately

Lemma 2.1. *The following identities hold for all languages A, B, and C:*

(a) $A \cup B = B \cup A$.

(b) $(A \cup B) \cup C = A \cup (B \cup C)$.

(c) $\varnothing \bullet A = A \bullet \varnothing = \varnothing$.

(d) $\{\varepsilon\} \bullet A = A \bullet \{\varepsilon\} = A$.

(e) $(A \bullet B) \bullet C = A \bullet (B \bullet C)$.

(f) $A \bullet (B \cup C) = (A \bullet B) \cup (A \bullet C)$.

(g) $(A \cup B) \bullet C = (A \bullet C) \cup (B \bullet C)$.

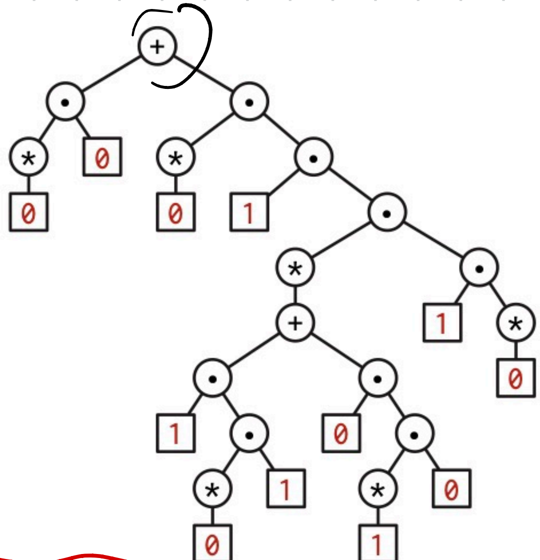
Lemma 2.2. *The following identities hold for every language L:*

(a) $L^* = \{\varepsilon\} \cup L^+ = L^* \bullet L^* = (L \cup \{\varepsilon\})^* = (L \setminus \{\varepsilon\})^* = \{\varepsilon\} \cup L \cup (L \bullet L^+)$.

(b) $L^+ = L \bullet L^* = L^* \bullet L = L^+ \bullet L^* = L^* \bullet L^+ = L \cup (L \bullet L^+) = L \cup (L^+ \bullet L^+)$.

(c) $L^+ = L^*$ *if and only if* $\varepsilon \in L$.

Lemma 2.3 (Arden's Rule). *For any languages A, B, and L such that $L = A \bullet L \cup B$, we have $A^* \bullet B \subseteq L$. Moreover, if A does not contain the empty string, then $L = A \bullet L \cup B$ if and only if $L = A^* \bullet B$.*



A regular expression tree for $0^*0 + 0^*1(10^*1 + 01^*0)^*10^*$

Proof: Let R be an arbitrary regular expression.
Assume that **every regular expression smaller than R** is perfectly cromulent.
There are five cases to consider.

- Suppose $R = \varnothing$.

  Therefore, R is perfectly cromulent.

- Suppose R is a single string.

  Therefore, R is perfectly cromulent.

- Suppose $R = S + T$ for some regular expressions S and T.
  The induction hypothesis implies that S and T are perfectly cromulent.

  Therefore, R is perfectly cromulent.

- Suppose $R = S \bullet T$ for some regular expressions S and T.
  The induction hypothesis implies that S and T are perfectly cromulent.

  Therefore, R is perfectly cromulent.

- Suppose $R = S^*$ for some regular expression. S.
  The induction hypothesis implies that S is perfectly cromulent.

  Therefore, R is perfectly cromulent.

In all cases, we conclude that ~~R~~ is perfectly cromulent. □

R