Lecture 2

Thursday, 28 January, 2021 10:47

Brief Recap of Strings

- finite set Σ "alphabet"
- strings

$$\left|\begin{array}{ll} \varepsilon & \text{(empty string)} \\ ax & a \in \Sigma,\ x \text{ is a string} \end{array}\right. \qquad w \cdot z \left|\begin{array}{ll} z & \text{if } w = \varepsilon \\ a(x \cdot z) & \text{if } w = ax \end{array}\right.$$

- language set of strings
- language concatenation

$$L \cdot M = \{ x \cdot y \mid x \in L, y \in M \}$$

e.g. $L = \{\text{'pumpkin Spice'}, \text{'Caramel'}\}$

$M = \{\text{'Latte'}, \text{'Chai'}\}$

$$L \cdot M = \left\{ \begin{array}{ll} \text{'PumpkinSpiceLatte'}, & \text{'PumpkinSpiceChai'}, \\ \text{'CaramelLatte'}, & \text{'CaramelChai'} \end{array} \right\}$$

$$M \cdot L = \{\text{'Latte PumpkinSpice'}, \text{'Chai PumpkinSpice'}\}$$

!! elements of $L \cdot M$ are strings
elements of $L \times M = \{ (x, y) \mid x \in L, y \in M \}$
are pairs of strings

$\varepsilon \neq \phi \rightarrow \{\varepsilon\} \neq \{\phi\}$

string set

e.g. - $\{0\} \cdot \{1\} = \{01\}$

- $\{0\} \cdot \{\varepsilon\} = \{0\} = \{\varepsilon\} \cdot \{0\}$
- $\phi \cdot \{0\} = \phi = \{0\} \cdot \phi$

Language concat is an easy way to build languages from other languages!

What other ways do we have of building languages?

What other ways do we have of building languages?

- union: $\mathcal{L} \cup \mathcal{M}$ is a language.
- Kleene star: $\mathcal{L}^*$

  - $\mathcal{L}^n = \begin{cases} \{\varepsilon\} & \text{if } n = 0 \\ \mathcal{L} \cdot \mathcal{L}^{n-1} & \text{if } n > 0 \end{cases}$

  e.g. Σ^n = set of all length n strings

  $$\mathcal{L}^* = \mathcal{L}^0 \cup \mathcal{L}^1 \cup \mathcal{L}^2 \cup \dots$$
  $$= \bigcup_{n \geq 0} \mathcal{L}^n$$

Kleene studied class of langs called

Regular Languages

- $\emptyset$ is a regular lang
- $\{x\}$ is a regular lang for any string x
- $\mathcal{L} \cup \mathcal{M}$ is a regular lang if $\mathcal{L}$ & $\mathcal{M}$ are regular
- $\mathcal{L} \cdot \mathcal{M}$ " "
- $\mathcal{L}^*$ is regular if $\mathcal{L}$ is regular

Kleene used these to classify/study seq's of op's.

- $\emptyset$ set containing no operations at all
- $\{x\}$ one instruction
- $\mathcal{L} \cup \mathcal{M}$ if/else
- $\mathcal{L} \cdot \mathcal{M}$ do one op then another
- $\mathcal{L}^*$ loops

while x
  if y
    a
    b
  else
    c
    d

$$\left((\{a\} \cdot \{b\}) \cup (\{c\} \cdot \{d\}) \right)^*$$

$\left(\left(\{a\}\cdot\{b\}\right)\cup\left(\{c\}\cdot\{d\}\right)\right)$

what languages are regular?

$0^2 = 00$

$0^3 = 000$

$L_1 = \{0^i \mid i \in \mathbb{N}\}$ ✓ $L_1 = \{0\}^*$

$L_2 = \{0^i \mid i \text{ is even}\}$ ✓ $L_2 = \{00\}^*$

$L_3 = \{0^i \mid i \text{ is divisible by } 3\}$

$= \{\varepsilon, 000, 000000, \dots\}$ ✓ $L_3 = \{000\}^*$

$L_4 = \{0^i \mid i \text{ is even or } i \text{ is divisible by } 3\}$ ✓ $L_4 = L_2 \cup L_3$

$L_5 = \{0^i 1^j \mid i \text{ is even}, j \text{ is divisible by } 3\}$ ✓ $L_5 = \{00\}^* \cdot \{111\}^*$

$L_6 = \{0^i 1^i \mid i \in \mathbb{N}\}$ ✗ why? later...

$\neq \underbrace{\{0\}^*\{1\}^*}_{\{\varepsilon,\, 0,\, 1,\, \dots\}}$ $0^1 1^0$ $0^0 1^1$

Notation: regular expressions.

regular language	regular expression
ϕ	ϕ
$\{x\}$	x
$R \cup S$	$r + s$
$R \cdot S$	$r \cdot s$
R^*	r^*

order of precedence: $*$, $\cdot$, $+$ (same as arithmetic)

skill: build a regular expression for a regular lang by splitting into cases / recursion.

- Language of binary strings where all 0s come before all 1s.

- Language of binary strings where all 0s come before all 1s.

  ex. 00, 0111, 11

  $= $(Language of seq's of 0's) $\bullet$ (Lang of seq's of 1s)

  $= \cancel{\{0\}^* \bullet \{1\}^*} \qquad 0^*1^*$

- strings that contain 00.

  ex. 00, 000, 1001

  bad: 010, 1

  (anything) 00 (anything)

  $$\underbrace{(0+1)^*}_{\text{regex for lang of all binary strings}} 00 \, (0+1)^*$$

- all strings other than ε $\qquad \{0,1\}^* \setminus \{\varepsilon\}$

  set diff is not a regular op.

  $$\underbrace{(0+1)}_{\text{at least one symbol}}\underbrace{(0+1)^*}_{\text{the rest of the string.}}$$

- all strings other than 0 $\qquad \{0,1\}^* \setminus \{0\}$

  ε ✓ $\quad$ 1 ✓ $\quad$ 0 ✗

  strings of length ≥ 2 ✓

  $\varepsilon + 1 + $ (reg ex for strings of length ≥ 2)

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

other possibilities?

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

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

if string isn't ε,
it begins w/ a 1.
00 cannot be generated.

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

string contains a 1.

00

language of alternating 0s & 1s.

e.g. 01, 0, 010, 101, 10101 0

bad. 00, 11, 0110

possibilities:

- $\varepsilon + (01)^* + (10)^*$

all strings generated are alternating.
but not all alternating strings are generated.
e.g. 0.

- $\varepsilon + (01)^* + (10)^* + 0 + 1$

10101 ?

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

regex for $(\{0\} \cup \{1\})^* = (\{1\} \cup \{0\})^*$

$(0+1)^*$ & $(1+0)^*$ are equivalent

↳ simplify to $((0+1)^*)^* \longrightarrow$ eq to $(0+1)^*$

in general $(L^*)^* = L^*$

this is bad because $(0+1)^*$ gives **all** binary strings all binary strings

incl. 00

possible approach: casework on the first character.

$$\varepsilon + (01)^*(0+\varepsilon) + (10)^*(1+\varepsilon)$$

start w/ this since it works. maybe simplify/optimize.

↳ simplify to $(01)^*(0+\varepsilon) + (10)^*(1+\varepsilon)$

↳ could simplify $(0+\varepsilon)(10)^*(1+\varepsilon)$

can it generate 0? yes:

$$\varepsilon + \underbrace{(01)^*(0+\varepsilon)}_{} + (10)^*(1+\varepsilon)$$

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

$(01)^* \to \{\varepsilon\} \cup \{01\} \cup \{0101\} \cup \ldots$

$$\varepsilon \cdot 0 = 0$$

can it generate 101?

$$(10)^*(1+\varepsilon) \to 10 \cdot 1$$