Regular expressions over Σ

ϕ is an r.e.

ε . . . . r.e.

a . . . . . for each $a \in \Sigma$

if R_1 & R_2 are r.e.s,
then so are

$(R_1 \cup R_2)$

$(R_1 \circ R_2)$

$(R_1{}^*)$

---

the language denoted by R, $L(R)$ is:

$L(\phi) = \phi$

$L(\varepsilon) = \{\varepsilon\}$

$\vdots$

$L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$

$$L(\underbrace{(\phi^*)}_{R.E.}) = L(\phi)^*$$

$$= \phi^*$$

$$= \{\varepsilon\}$$

Short hands

$$\Sigma = \{a, b, c\}$$

$$L(((a \cup b) \cup c)) = \Sigma$$

$$(\underline{\Sigma}^* \cup \varepsilon) \cdot a$$

$$((\underline{((a \cup b) \cup c)^*} \cup \varepsilon) \cdot a)$$

precedence & associativity

$$(a \cup b \cup c)$$

$$a \cup b \cdot c^*$$

"words ending with ".TXT" "

$\Sigma^* .TXT$

$(a \cup b \cup \dots \cup Z) \cdot (a \cup \dots \cup Z \cup 0 \dots 9)^*$

$l \cdot (l \cup d)^*$

Shorthand

$(\Sigma\Sigma)^*$

$\left.\begin{array}{l}\Sigma\Sigma^* \\ aa^*\end{array}\right\}\begin{array}{l}\Sigma^+ \\ a^+\end{array}$

$0^* 1 0^*$

$(\varepsilon \cup \Sigma)(\varepsilon \cup \Sigma)$

$\Sigma \Sigma$

$00 \in \quad 0^*(10^*10^*)^*$

$00 \notin \quad (0^* 10^* 10^*)^*$

$(0^* 10^* 1)^* 0^*$

$(d^* . d^+ \cup d^+ . d^*)(\varepsilon \cup E(\varepsilon \cup + \cup -)d^+)$

Theorem:

$\forall$ regular expression R $\exists$ an NFA M_R s.t. $L(R) = L(M_R)$

Proof:

By induction on K, the # of $\cup, \circ, *$ operators in R

Base cases ($K=0$):

Then R is "ϕ", "ε", or "a" for $a \in \Sigma$,

Explicitly give simple NFA's recognizing ϕ, $\{\varepsilon\}$, and $\{a\}$ for each $a \in \Sigma$, (details omitted)

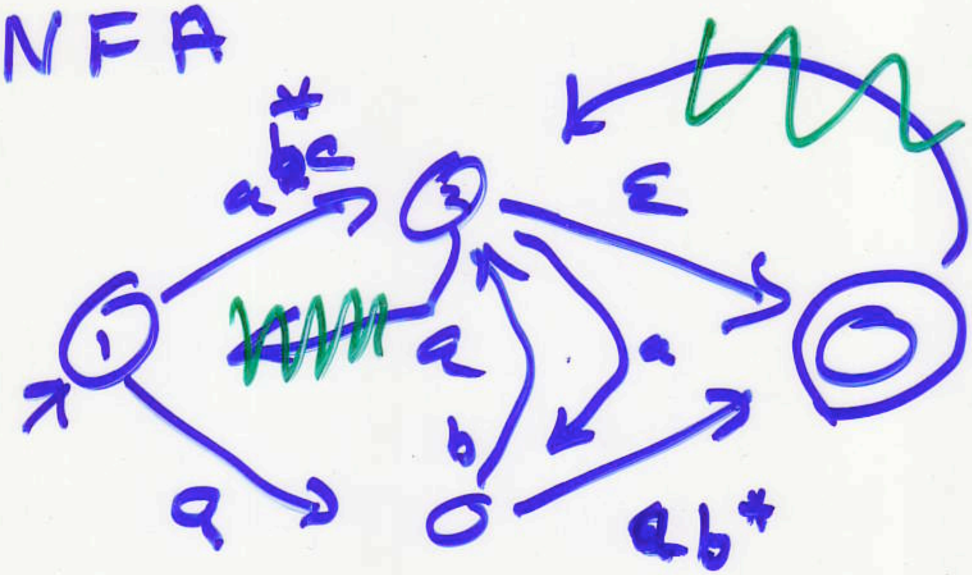
Induction Step (R has $K > 0$ operators)

> I.H.: assume that for all regular expressions R' with $< K$ operators, $\exists$ NFA $M_{R'}$ recognizing $L(R')$

R has $K > 0$ operators. So R is $(R_1 \cup R_2)$ or $(R_1 \circ R_2)$ or $(R_1)^*$ where R_1 (& R_2 if any) have $\leq K-1$ operators. By I.H., $\exists$ M_{R_1} (& M_{R_2}) s.t. $L(R_i) = L(M_{R_i})$, $i=1,2$. Modify/join it/them as in previous proofs of closure under $\cup, \circ, *$ to get M_R s.t. $L(R) = L(M_R)$.

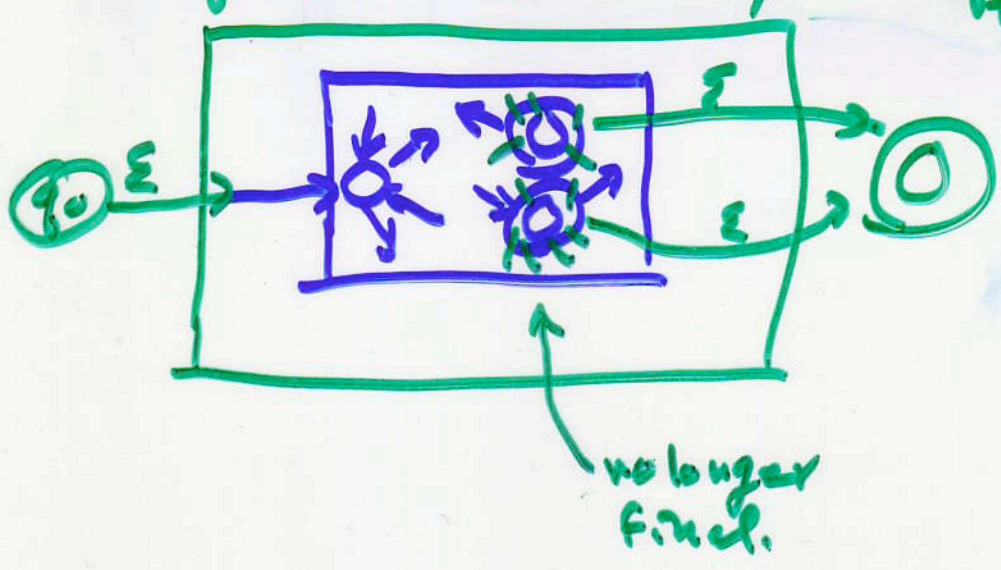
Every ~~FA~~ Regular language can be described by a regular expression.

GNFA



abbbc

Note: No loss in assuming no edges into q_0 / out of F / only one $q_f \in F$



GNFA

$G = (Q, \Sigma, \delta, q_0, q_f)$

$Q, \Sigma, q_0, q_f \in Q$ as usual

Regular expressions over Σ

$\delta : (Q - \{q_f\}) \times (Q - \{q_0\}) \to \mathcal{R}_\Sigma$

<u>Defn</u>

- G <u>can be in state</u> $q \in Q$ after reading $x \in \Sigma^*$ if $\exists k \geq 0$,

  $\exists\, r_0, r_1, \dots, r_k \in Q$

  $\exists\, x_1, \dots, x_k \in \Sigma^*$

  such that

  (i) $x = x_1 \cdot x_2 \cdot \dots \cdot x_k$

  (ii) $r_0 = q_0$

  (ii) $r_k = q$

  (iii) $\forall\, 1 \leq i \leq k,\ x_i \in L(\delta(r_{i-1}, r_i))$

- $L(G) = \{x \mid G$ can be in state $q_f \dots\}$

Note: δ syntax a little different; maps state pair to label (reg. exp.) rather than state $\times$ symbol $\to$ new state.